COMP490 - MARFCAT/GIPSY

Joseph-Antoine Martineau-Gagné
Concordia University
Montreal, Quebec, Canada
jos_m@encs.concordia.ca

Abstract

MARFCAT, a static code analyzer using a machine learning approach has been previously converted into a distributed application implementing the GIPSY multi-tier framework.[5] It will be benchmarked in order to appreciate the potential performance gain over the standalone version. A Web Front End and the GIPSY implementation of MARFCAT will be integrated if time permits.

Contents

1 Introduction

This document contains documentation about the development accomplished on MARFCAT and GIPSY as well as the summarized results of my experiments in measuring the runtime of MARFCAT standalone and MARFCAT GIPSY. More specifically, the Development 3 section contains information about how to build and run MARFCAT standalone and MARFCAT GIPSY. It also contains documentation about the GMT Console refactoring 3.3, which improves usability of the GIPSY system by enabling interaction with the GMT via a shell, and the Gradle integration. The results are presented in the Benchmarking 4 section.

2 Overview

This section contains a high level description the GISPSY framework and the MARFCAT application.

2.1 MARFCAT "MARF-based Code Analysis Tool"

MARFCAT has been developed on top of MARF, the *Modular A* Recognition Framework* which is itself an extension of the original *Modular Audio Recognition Framework* designed to do pattern recognition of audio samples. [5, 5.2]

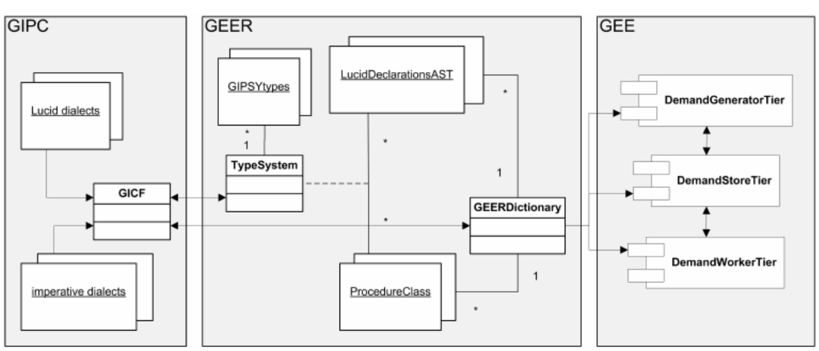
The main idea behind MARFCAT is to use a machine learning approach to perform static source code analysis. This approach has the advantage of being language independant and to work with binary as well since the source code is treated and processed as a signal. It has been shown that this approach is efficient in term of speed and precision compared to other approaches (semantic analysis). It could be used in conjunction with those other tools to prioritize their initial targets. [4] [6]

In order to detect vulnerabilities in some set of source files, we need to have access to the source code of a previous version of the software along with CVE ("Common Vulnerability Exporsure") annotations. CVE is a publicly available dictionary of known vulnerabilities and exposures [1].

Given the source code of a program along with some knowledge files (CVE and CWE [1]) that contains metadata about known vulnerabilities in that source code, MARFCAT first learns to recognize the signal associated with each one of those vulnerabilities using multiple techniques such as Signal Pipeline, NLP pipeline and others. [5, 5.4.5] It then enters the testing phase in which it will try to detect the already known vulnerabilities in the annotated version (the one it just learned from) to see if it is able to accurately detect them. After that MARFCAT is ready to test non-annotated source code and report vulnerabilities.

2.2 GIPSY

GIPSY is a demand-driven distributed multi-tier system built to "investigate properties of the Lucid family of intensional programming languages and beyond". [5, 6] GIPSY is divided into three subsystems (see [5, 6.2] for details about the architecture):



(from [5, 6.2])

GIPC "General Intensional Programming Compiler" Compiler.

GEE "General Eductive Engine" This is the demand-driven execution system that MARFCAT has been integrated too.[3] It is language independent and is organized as a multi-tiers architectures. Later in this document we will mention the GMT (General Manager Tier), the DST (Data Store Tier), the DGT (Demand Generator Tier) and the DWT (Demand Worker Tier). They are all part of this demand-driven execution system.

RIPE "Runtime Interactive Programming Environment" Graphical User Interface for interacting with the runtime-system.

3 Development

In this section I document how to run MARFCAT and MARFCAT GIPSY as well as the development done over the semester.

3.1 MARFCAT

The working directory of the project is located under the `src/marf/apps/MARFCAT` directory.

3.1.1 Build

To build the project, under the working directory 3.1 run one of the following :

```
# to trigger a Gradle build
make gradle-build
make gradle-install        # to have a run-script generated with all dependencies setup
# or alternatively :
make jar
```

N.B. For convenience those make targets are now available from the root directory of the project.

3.1.2 Run

Under the `src/marf/apps/MARFCAT` directory, some preparation needs to be done before running MARFCAT and a couple of options are available.

1. Using the marfcat TCSH wrapper

There is a TCSH script under the working directory which acts as a wrapper to the Java based MARFCAT application. It is used to run many instances with different configuration at the same time.

- A `symlink` to the source code under test needs to be created.
- The CVE annotation file need to be present also.

2. Using the Java based MARFCAT directly

The easiest way to do that is to first trigger a Gradle build using the `installDist` Gradle task from the root directory of the project and then use the `run.sh` script from the same directory. All arguments are passed directly to the Java application.

```
./gradlew installDist       # or 'make gradle-install'
./run.sh
```

3.1.3 Project Organization and Gradle

Why integrating Gradle? Firstly I felt that we should move toward a cleaner project organization for MARFCAT. When I got started working on MARFCAT I have had to spend a lot of efforts on finding how to build and run the project. This time would have been better spent doing actual development therefore I consider that integrating Gradle is one step toward a cleaner project organization. Some of the advantages we already see are :

- No more `class` files scattered in the `scr/` directory
- No need to maintain a hand written `makefile`
- Automatic Eclipse project generation
- Easy dependency management
- Standardized OS independent build process
- Easy to enforce and automated testing

Further improvments suggested to the project organization :
I beleive the following items should also be addressed in the futur since they all contributed in making it harder to get up and running using MARFCAT

- create an output directory so we don't end up with all the MARFCAT generated files under the `src/` directory.
- create a `data/` directory to which the knowledge files should belong.
- we shoudn't have to move deeply in the directory tree in order to run the program, therefore all run and build targets should be available from the root folder directory.
- rename the Git repository from 'marfcat-src' to 'marfcat'
- add Gradle plugin for IntelliJ project generation
- Keep the input data outside git (now the repository is 2 GB of size and I couldn't successfully clone it). Somewhere accessible via `curl` or `wget` so we can automatically have them installed to a `data/` folder.
- Create unit tests and make their successful execution mandatory to build the project (this is functionality build-in Gradle). This would prevent people working on the project to accidentally break stuff that others are using.

3.2 MARFCAT-GIPSY

3.2.1 Build

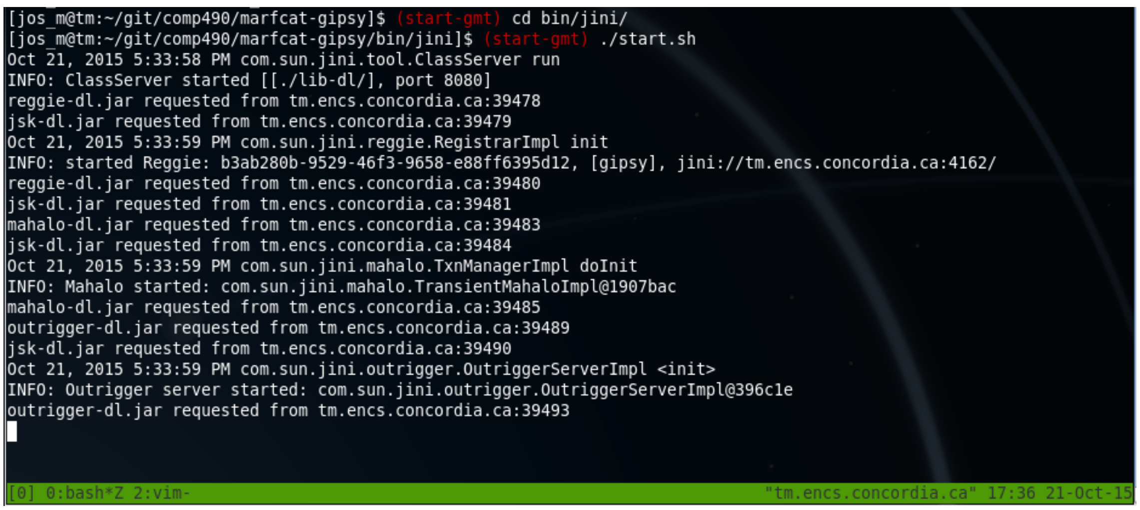
Under the marfcat-gipsy root directory execute : `make`. The following jars will be assembled : `ripe.jar`, `gee.jar`, `gipc.jar` and one that contains all three previous named `gipsy.jar`.

3.2.2 Run

Step 1 : start Jini
(note - you should be able to move directly to step 2)

Under `marfcat-gipsy/bin/jini` use the `start.sh` script.

If successful the output should be similar to :



useCodebaseOnly The JVM option `java.rmi.server.useCodebaseOnly` needs to be set to false if you are using a Java version equal or more recent than version 6u45. [2] The symptom of this is the following ClassNotFoundException's :

```
java.lang.ClassNotFoundException:
          com.sun.jini.reggie.RegistrarProxy
          java.lang.ClassNotFoundException: com.sun.jini.mahalo.TxnMgrProxy
          java.lang.ClassNotFoundException:
          com.sun.jini.outrigger.SpaceProxy2
```

Step 2 : start the first GMT ("General Manager Tier") node
(alternatively use the `gmtd/gmtc` to be able to interact with the GMT from a shell: 3.3)

Under `bin/multitier` use the `startMARFCATGMTNode.sh` script.

Once executed you should see a prompt. Enter the following command :

```
start GMT GMTJini.config
```

You should now have one `xterm` labeled JiniDST and one GMT Console as follow :

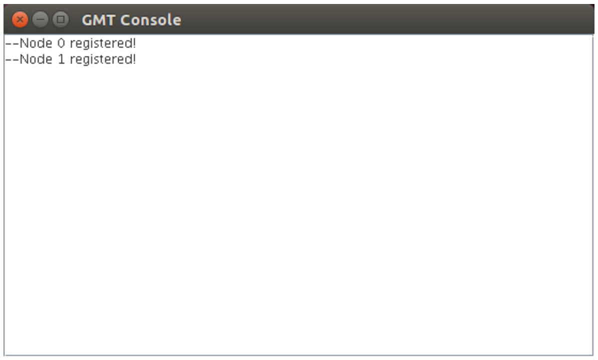
N.B. Again, the `useCodebaseOnly` needs to be set to false. (3.2.2)

Step 3 : start a second Gipsy Node
(note - optional, you may run on a single node)

Under `bin/multitier` use the `startMARFCATRegularNode.sh` script.

In the prompt enter : `register`

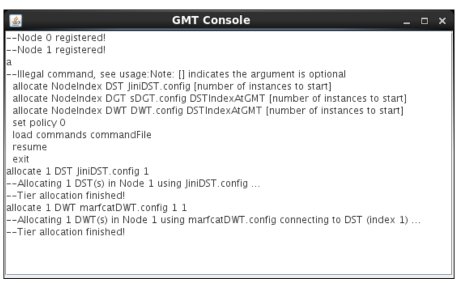
In the GMT window you should see that the node has been successfuly registered.



Step 4 : start DST, DWT and DGT

Tiers are allocated by typing commands in the GMT console. [3, section 2.4]
Valid GMT commands :

```
allocate NodeIndex DST JiniDST.config           [number of instances to start]
allocate NodeIndex DGT DGT.config DSTIndexAtGMT [number of instances to start]
allocate NodeIndex DWT DWT.config DSTIndexAtGMT [number of instances to start]
set policy 0
load commands commandFile
resume
exit
```



Start a regular DST on Node 1 :
`allocate 1 DST JiniDST.config 1`

Start a marfcat DWT on Node 1 which will use the DST just started :
`allocate 1 DWT marfcatDWT.config 1 1`

Start a marfcat DGT on Node 1 which will use the DST just started :
`allocate 1 DGT marfcatDGT.config 1 1`

N.B.

- under `bin/mulitier` you need to have a symlink to the data set with proper naming (the symlink should have the same name as the directory where the source code is located).

- make sure you clean the `bin/multitier` directory from any `*.stats` files before running the DGT.
- if you want to skip the training phase, you need to place your *gzbin in the `bin/multitier` directory.
- make sure the property marfcat.meta.basedir in the `marfcatDGT.config` and `marfcatDWT.config` files is set correctly.

3.2.3 Conclusion

The above process for starting `marfcat-gipsy` is tedious, repetitive and time consuming. In order to improve the usability of the system and to facilitate my testing task, I created an alternative to the above `GMTTestConsole` 3.2.2 which enables interaction with the GMT from regular shell scripts. (see 3.3)

3.3 GMT Refactoring

In order to efficiently test `marfcat-gipsy` we need to be able to easily interact with the General Manager Tier, ideally from a shell script so no manual intervention is needed to run our tests. We want to be able to programmatically configure a GIPSY network, perform some tests, then reconfigure the network to perform other tests.

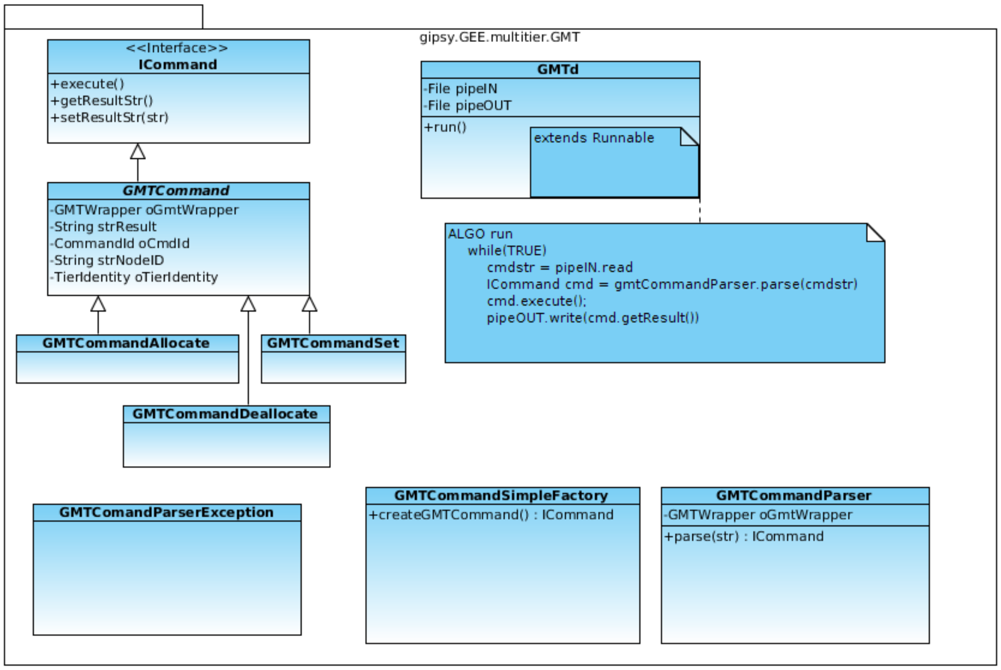
Because the deployment of a GIPSY system using the `GMTTestConsole` requires many manual interventions (see 3.2.2) and that some of its features are broken (the command history used to make the GMT crash because of a race condition bug) or incomplete, it does not not meet our above criteria of usability/testability. A more flexible and more efficient solution is needed. We can either imporove the `GMTTestConsole` or consider other options.

I mentioned that being able to interact with the GMT from a shell script would be a great improvement in usability, therefore I propose a solution that goes it that direction.

3.3.1 GMTd

First, I want to get rid of the JFrame that is popping up when starting the GMT. As shown in 3.2.2, starting a GMT along with one DGT, one DWT and one DST opens up 3 windows that are of no practical use as we cannot even copy the content to the clipboard. By eliminating the `GMTTestConsole` JFrame we will have one less window without loosing anything. Therefore my idea is to have a GMT daemon, the `GMTd`, that does its work silently. We will interact with the `GMTd` via 2 named pipes : `/tmp/gmt_cmd_in` and `/tmp/gmtc_cmd_out`.

To implement the GMTd, I had to extract all the parsing logic which was embedded within the `GMTTestConsole` JFrame in a proper `GMTCommandParser`. I followed the Command design pattern and created a class hierarchy for the commands so the command logic is decoupled from the parsing logic. Commands are created using the `GMTCommandSimpleFactory`. Note that the original `GMTTeStConsole` has not been affected by those changes, but it is now using the `Icommand` class hierarchy. It should also be modified to use the newly created `GMTCommandParser` so there would be no more parsing logic in it.



3.3.2 GMTc

In order to interact with the GMTd conveniently, we need some client to write commands to the input pipe and read output from the output pipe. I implemented a simple GMTc (as a BASH script) to do that. The best way to use it is to add it to your PATH environment variable and use it as a shell command.

```
$ gmtc -u
usage: gmtc [OTPS] -- GMTCommand
            -c              :: create the in and out FIFO
            -u              :: display usage
```

Examples, from a shell script or directly in a terminal :

```
gmtc allocate 0 DST JiniDST.config 1
gmtc allocate 0 DWT marfcatDWT.config 1 1
gmtc allocate 0 DGT marfcatDGT.config 1 1
```

3.3.3 Run the GMTd

To start the GMT with the GMTd, use the `startMARFCATGMTNode.sh` script :

```
$./startMARFCATGMTNode.sh  -u
usage: ./startMARFCATGMTNode.sh [OTPS]
    -n [gmtd|jini]  :: pipe command to STDIN :
                    ::                   gmtd => 'start GMT GMTJini.config gmtd'
                    ::                   jini => 'start GMT GMTJini.config'
    -u              :: display usage
```

example :
```
./startMARFCATGMTNode.sh -n gmtd
```

Then use the GMTc client to interact with the GMT.

3.4 MARFCAT configuration in MARFCAT GIPSY

The configuration of MARFCAT in the MAFCATDGT and MARFCATDWT classes was hard coded. This was making the use of MARFCAT GIPSY tedious since recompilation was needed every time a different MARFCAT configuration was used. This issue is twofold:

- we want to read the MARFCAT configuration from the `marfcatDGT.config` and `marfcatDWT.config` files.
- we want to move toward a *per demand* configuration.

3.4.1 I - Parametrized configuration

The properties **`marfcat.meta.basedir`**, **`marfcat.meta.filename`** and **`marfcat.args`** have been added to the `marfcatDGT.config` and `marfcatDWT.config` files. MARFCATDGT and MARFCATDWT are now reading the configuration from those files, therefore eliminating the need for recompilation when a change in the MARFCAT configuration is wanted.

Example (from `marfcatDGT.config`)

```
marfcat.meta.basedir=.
marfcat.meta.filename=wireshark-1.2.0_train.xml
marfcat.args=--batch-ident -nopreprep -raw -fft -cheb --dgt
```

3.4.2 II - Per demand configuration

A per-demand configuration scheme has been implemented to address the need to be able to run MARFCAT GIPSY for several configurations at the same time.

To do so a `MARFCATProceduralDemand` class has been created. It extends `ProceduralDemand` and has a configuration field added to its members. The MARFCATDGT has been modified to create one such demand per configuration/FileItem pair. Similarly, the MARFCATDWT has been modified to extract and use this demand specific configuration upon receiving a demand. (see 4.2.2 for multi-configuration run results).

Further Improvments
For now the MARFCATDGT reads the multiple configuration strings from the `marfcatDGT.config` file. This allows a fine grained selection of the configurations to use. It would be useful also to have the option to run several configurations in a more systematic manner. Practically this could mean to programmatically loop over a set of configuration strings instead of reading them from file.

4 Benchmarking

The goal is to compare the execution time of MARFCAT standalone and MARFCAT GIPSY, its distributed counterpart. To do so, I wanted to be able to automate the benchmarking tests as much as possible so they can be re-run easily with different parameters or different middlewares. For the standalone version, a little bit of shell scripting did the trick, but for the GIPSY version, I had to find a way to script the interaction with the GMT (see the `GMTTestConsole` refactoring here : 3.3). Before, the GIPSY system needed many manual interventions to be started, the `GMTTestConsole` was involved and it was tedious to use.

At this point the automation goal is mostly achieved, in particular, we can trigger the tests for MARFCAT GIPSY on a single machine with only 2 commands :

```
# 1. start the GIPSY system
# under 'marfcat-gipsy/bin/multitier'
 ./startMARFCATGMTNode.sh -n gmtd

# 2. Allocate the tiers and start computation
# under 'comp490-f15/' (root directory of the 'comp490-f15' repository)
./time-gipsy.sh
```

I said *mostly* achieved because that to run the benchmarking tests on more than one machine using the above approach we were still required to do the following on every additional machine :

```
./startMARFCATRegularNode.sh  -r        # -r for register right away with the GMT
```

To make the experimentation process on multiple hosts fast, easy and reproductible, I have used tmux + shell scripting to automatically ssh to each host and run the start scripts appropriately (ex. 6). This approach works fine when running the experiments only a couple of times each, but if we want to be able to run the experiments say 100 times in a row to make sure that the results are consistent, this is still tedious and requires manual intervention. To address this issue the `multi-host-time.sh` and `multi-run.sh` scripts have been written. To run an experiment multiple times on up to 3 machines with an arbitrary number of DWT (one per gipsy node), use the `multi-run.sh` script as follow :

```
./multi-run.sh  -u
usage: ./gipsy.sh
            -c              :: clear the result files before
            -m nb_of_runs
            -n nb_nodes     :: number of gipsy nodes to be allocated (1 DWT per gipsy node)
            -u              :: display usage
```

N.B. You need the redirect X to your display, GIPSY is creating xterm's for the DST's.

The command `./multi-run.sh -n 9 -m 3 -c` will repeat the same computations 3 times (`-m`) on 3 machines (`willpower, awareness and orwell`) using 9 DWT's and save the results in `results/multi-run.res`.

When running the experiments on a single machine, I have used `samsara-laptop` and `willpower` and when running the experiments on multiple computers, I have used the duo `willpower, awareness` or the trio `willpower, awareness and orwell`.

4.1 time-standalone.sh

Using the `time-standalone.sh` script, the computation time per configuration as well as the total running time for a subset of the possible configuration parameter permutations is shown here. Note that we'll be mostly using the TEST:0 results in our comparisons since the per-demand execution of MARFCAT GIPSY still needs further development 3.4.2, but this doesn't affect the validity of the results, since if we get speedup when running a particular configuration in parallel, then we should logically get speedup when running any configuration in parallel.

4.1.1 Wireshark 1.2.0

```
Host: willpower.encs.concordia.ca
CPU:  Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
------------------------------------------------
[TEST:0] {--batch-ident -nopreprep -raw -fft -cheb -flucid
    ↪ wireshark-1.2.0_train.xml}
real     14.91
user     17.99
sys      1.01
max-mem 235476 KB
Running total execution time (in seconds) : 14.91
------------------------------------------------
[TEST:1] {--batch-ident -nopreprep -raw -fft -diff -flucid
    ↪ wireshark-1.2.0_train.xml}
real     13.90
user     17.99
sys      0.88
max-mem 266896 KB
Running total execution time (in seconds) : 28.81
------------------------------------------------
[TEST:2] {--batch-ident -nopreprep -raw -fft -eucl -flucid
    ↪ wireshark-1.2.0_train.xml}
real     14.12
user     17.92
sys      0.94
max-mem 403932 KB
Running total execution time (in seconds) : 42.93
------------------------------------------------
[TEST:3] {--batch-ident -nopreprep -raw -fft -hamming -flucid
    ↪ wireshark-1.2.0_train.xml}
real     13.50
user     18.15
sys      0.91
max-mem 240652 KB
Running total execution time (in seconds) : 56.43
------------------------------------------------
[TEST:4] {--batch-ident -nopreprep -raw -fft -mink -flucid
    ↪ wireshark-1.2.0_train.xml}
real     15.92
```

```
user     19.68
sys      0.92
max-mem 335600 KB
Running total execution time (in seconds) : 72.35
-------------------------------------------------
[TEST:5] {--batch-ident -noprep -raw -fft -cos  -flucid
    ↪ wireshark-1.2.0_train.xml}
real     17.37
user     20.40
sys      1.43
max-mem 452024 KB
Running total execution time (in seconds) : 89.72
-------------------------------------------------
Total Execution Time (seconds) : 89.72


Host: samsara-laptop
CPU:   Intel(R) Core(TM) i3 CPU M 370 @ 2.40GHz
-------------------------------------------------
[TEST:0] {--batch-ident -noprep -raw -fft -cheb -flucid
    ↪ wireshark-1.2.0_train.xml}
real     16.86
user     23.68
sys      0.85
max-mem 286428 KB
Running total execution time (in seconds) : 16.86
-------------------------------------------------
[TEST:1] {--batch-ident -noprep -raw -fft -diff -flucid
    ↪ wireshark-1.2.0_train.xml}
real     17.57
user     26.24
sys      0.88
max-mem 381240 KB
Running total execution time (in seconds) : 34.43
-------------------------------------------------
[TEST:2] {--batch-ident -noprep -raw -fft -eucl -flucid
    ↪ wireshark-1.2.0_train.xml}
real     17.33
user     25.04
sys      0.83
max-mem 466780 KB
Running total execution time (in seconds) : 51.76
-------------------------------------------------
[TEST:3] {--batch-ident -noprep -raw -fft -hamming -flucid
    ↪ wireshark-1.2.0_train.xml}
real     18.43
user     30.75
sys      0.97
```

```
max-mem 492936 KB
Running total execution time (in seconds) : 70.19
-----------------------------------------------
[TEST:4] {--batch-ident -noprep -raw -fft -mink -flucid
    ↪ wireshark-1.2.0_train.xml}
real     21.10
user     34.42
sys      0.87
max-mem 296844 KB
Running total execution time (in seconds) : 91.29
-----------------------------------------------
[TEST:5] {--batch-ident -noprep -raw -fft -cos  -flucid
    ↪ wireshark-1.2.0_train.xml}
real     21.14
user     32.19
sys      1.26
max-mem 482216 KB
Running total execution time (in seconds) : 112.43
-----------------------------------------------
Total Execution Time (seconds) : 112.43
```

From the above we see that running a single configuration using MARFCAT standalone takes 14 to 17 seconds on `willpower` and between 17 and 21 seconds on `samsara-laptop`.

4.2 time-gipsy.sh

4.2.1 Wireshark 1.2.0 - Single MARFCAT configuration

```
=========================================================================
Host: samsara-laptop
CPU:  Intel(R) Core(TM) i3 CPU M 370 @ 2.40GHz
-----------------------------------------------
--Allocating 1 DST(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 0
--Tier allocation finished!
--Allocating 1 DGT(s) in Node 0
--Tier allocation finished!
[INFO] now wait for DGT to complete...DONE
-----------------------------------------------
Total Execution Time (seconds) : 60.47

=========================================================================
Host: samsara-laptop
CPU:  Intel(R) Core(TM) i3 CPU M 370 @ 2.40GHz
-----------------------------------------------
--Allocating 1 DST(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 0
```

```
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 1
--Tier allocation finished!
--Allocating 1 DGT(s) in Node 0
--Tier allocation finished!
[INFO] now wait for DGT to complete...
----------------------------------------
Total Execution Time (seconds) : 41.80
================================================================
Host: willpower.encs.concordia.ca
CPU:  Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
----------------------------------------
--Allocating 1 DST(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 0
--Tier allocation finished!
--Allocating 1 DGT(s) in Node 0
--Tier allocation finished!
[INFO] now wait for DGT to complete...DONE
----------------------------------------
Total Execution Time (seconds) : 25.22

================================================================
Host: willpower.encs.concordia.ca
CPU:  Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
----------------------------------------
--Allocating 1 DST(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 1
--Tier allocation finished!
--Allocating 1 DGT(s) in Node 0
--Tier allocation finished!
[INFO] now wait for DGT to complete...DONE
----------------------------------------
Total Execution Time (seconds) : 14.52

================================================================
Host: willpower.encs.concordia.ca
CPU:  Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
----------------------------------------
--Allocating 1 DST(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 1
--Tier allocation finished!
```

```
--Allocating 1 DWT(s) in Node 2
--Tier allocation finished!
--Allocating 1 DGT(s) in Node 0
--Tier allocation finished!
[INFO] now wait for DGT to complete...DONE
------------------------------------------------
Total Execution Time (seconds) : 10.65

================================================================================
Host: willpower.encs.concordia.ca
CPU:  Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
------------------------------------------------
--Allocating 1 DST(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 1
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 2
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 3
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 4
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 5
--Tier allocation finished!
--Allocating 1 DGT(s) in Node 0
--Tier allocation finished!
[INFO] now wait for DGT to complete...DONE
------------------------------------------------
Total Execution Time (seconds) : 7.56

================================================================================
Host: willpower.encs.concordia.ca
      awareness.encs.concordia.ca
CPU:  Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
      Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
------------------------------------------------
--Allocating 1 DST(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 1
--Tier allocation finished!
--Allocating 1 DGT(s) in Node 0
--Tier allocation finished!
[INFO] now wait for DGT to complete...DONE
------------------------------------------------
```

```
Total Execution Time (seconds) : 20.67

==================================================================
Host: willpower.encs.concordia.ca
      awareness.encs.concordia.ca
      orwell.encs.concordia.ca
CPU:  Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
      Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
      AMD Opteron(tm) Processor 6180 SE
-------------------------------------------
--Allocating 1 DST(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 1
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 2
--Tier allocation finished!
--Allocating 1 DGT(s) in Node 0
--Tier allocation finished!
[INFO] now wait for DGT to complete...DONE
-------------------------------------------
Total Execution Time (seconds) : 14.92

==================================================================
Host: willpower.encs.concordia.ca
      awareness.encs.concordia.ca
      orwell.encs.concordia.ca
CPU:  Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
      Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
      AMD Opteron(tm) Processor 6180 SE
-------------------------------------------
--Allocating 1 DST(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 1
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 2
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 3
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 4
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 5
--Tier allocation finished!
--Allocating 1 DGT(s) in Node 0
--Tier allocation finished!
```

```
[INFO] now wait for DGT to complete...DONE
----------------------------------------
Total Execution Time (seconds) : 6.99
```

4.2.2 Wireshark 1.2.0 - Multiple MARFCAT configurations

see 3.4.2

```
============================================================
Host:  willpower.encs.concordia.ca
CPU:   Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
----------------------------------------
--Allocating 1 DST(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 0
--Tier allocation finished!
--Allocating 1 DGT(s) in Node 0
--Tier allocation finished!
[INFO] now wait for DGT to complete...DONE
----------------------------------------
Total Execution Time (seconds) : 134.44

============================================================
Host:  willpower.encs.concordia.ca
CPU:   Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
----------------------------------------
--Allocating 1 DST(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 1
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 2
--Tier allocation finished!
--Allocating 1 DGT(s) in Node 0
--Tier allocation finished!
[INFO] now wait for DGT to complete...DONE
----------------------------------------
Total Execution Time (seconds) : 52.24

============================================================
Host:  willpower.encs.concordia.ca
       awareness.encs.concordia.ca
       orwell.encs.concordia.ca
CPU:   Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
       Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
      AMD Opteron(tm) Processor 6180 SE
----------------------------------------
--Allocating 1 DST(s) in Node 0
```

```
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 1
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 2
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 3
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 4
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 5
--Tier allocation finished!
--Allocating 1 DGT(s) in Node 0
--Tier allocation finished!
[INFO] now wait for DGT to complete...DONE
-----------------------------------------
 Total Execution Time (seconds) : 33.30

==============================================================
 Host: willpower.encs.concordia.ca
       awareness.encs.concordia.ca
       orwell.encs.concordia.ca
CPU:   Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
       Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
      AMD Opteron(tm) Processor 6180 SE
-----------------------------------------
--Allocating 1 DST(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 0
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 1
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 2
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 3
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 4
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 5
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 6
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 7
--Tier allocation finished!
--Allocating 1 DWT(s) in Node 8
--Tier allocation finished!
--Allocating 1 DGT(s) in Node 0
```

```
--Tier allocation finished!
[INFO] now wait for DGT to complete...DONE
-------------------------------------------------
Total Execution Time (seconds) : 29.61
```

4.3 Observations

4.3.1 Single Configuration

Be aware that the amount of work to perform for a single marfcat configuration is relatively small (1 minute of work on a laptop with an i3 processor) and therefore we should expect a speedup greater than the one observed when running MARFCAT GIPSY for multiple configurations in one shot. Thus our observations should be considered to be a lower bound on the expected speedup because the proportion of overhead compared to the total workload is high in the tests performed.

By running the MARFCAT GIPSY using the same configuration as in MARFCAT TEST:0 on a single node using a single DWT, we can get an approximation of the amount of overheads incurred by running MARFCAT in parallel. From the above results, we see that the overheads account for about 44% of the total running time (11 seconds out of 25). Remember that this proportion will go down as the total workload increases in size.

Now we increase the number of DWT's on a single machine (`willpower`) to see if we get any improvement. Running the same experiment with 2 DWT's reduces the runtime to 14.5 seconds, which is roughly equivalent to the standalone run (4.1.1). Adding a third DWT on the same host further reduces the runtime to 10.6 seconds. We already see a huge potential benefit of running in parallel since by using a single machine with 3 DWT's we get roughly 30% speedup compare to a standalone run on the same machine. Note that `willpower` is a very powerfull multiprocessor machine (48x Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz) and that running all DWT's on a single host limit the communication cost to a minimum.

But what if we distribute those 3 DWT's on 3 separate machines? Will we still get the same kind of improvement? Running the same experiment on `willpower, awareness and orwell` gives us a running time similar to the running time of the standalone version, which means that for this workload size, the communication cost relative to the total cost is relatively high.

Using `willpower, awareness and orwell` and now running 2 DWT's on each one of them leads to **50% improvement in runtime** and is our best runtime so far (this is faster than using 6 DWT's on `willpower`). The startup cost for this configuration accounts for a large portion of the total running time which means that by increasing the workload, we would get a speedup above 50% using that configuration (6 DWT's equally distributed on `willpower, awareness and orwell`) since the weight of the startup cost would go down.

4.3.2 Multiple Configurations

A per-demand configuration scheme has just been implemented (3.4.2). It's been said above that by increasing the workload we expect to get a greater speedup than what's already been desmonstrated. Here 4.2.2 are some results that show exactly that. From 4.1.1 we see that running a subset of 6 configurations using the standalone version takes roughly 90 seconds (on `willpower`). Running the same experiment using MARFCAT GIPSY on three machines (`willpower, awareness and orwell`) with 2 DWT's per machine, we can perform the same computations in 33 seconds, which means a **63% speedup**.

5 Conclusion

Over the term, development work related to usability has been done as well as maintenance work (configuration, bug fixing). Mainly, MARFCAT GIPSY has been benchmarked. From our experiments, we see that the MARFCAT application gains greatly from running over the GIPSY system in parallel even on a single midrange laptop (see `samsara-laptop` results 4.2.1). Therefore our goal is acheived since we experimentally demonstrated that MARFCAT can successfully run with significant performance improvment over the GIPSY framework.

To conlude, I believe that usability of MARFCAT and MARFCAT GIPSY should be further improved and that the code should go through a clean-up and documentation phase since modifying the code has proven to be difficult (specially see 3.4.2) and that alot of efforts were put in troubleshooting the system startup which didn't work out-of-the-box using a trial-and-error approach (by lack of a better one). I suggest that testing be enforced by the build process in an automated fashion. Gradle would be the perfect tool for this since it is a very flexible Groovy based build tool that can handle heterogeneous multi-projects build (this means for example that experimenting with Scala could be done easily) and that conveniently handles dependencies.

6 Annex

6.1 tmux-encs.sh

```
#!/bin/bash
#*****************************************************************************
#
# AUTHOR : Joseph-Antoine Martineau-Gagné
#
# PURPOSE : create a TMUX session with one window per host
# (willpower,awareness,orwell,grace) cd to the startMARFCAT*Node.sh directory
# in each pane, write the appropriate start command to the prompt, but wait
# for the user to press enter.
#
# HOW : ssh to willpower, then run the script, all is left is to press enter on
# each node to execute the start scripts, then in the comp490-f15 folder use the
# time-gipsy.sh to send the appropriate allocation commands to the GMTd using the
# GMTc (gmtc) command.
#
#
#*****************************************************************************
SESSION="490"

WILLPOWER="willpower"
AWARENESS="awareness"
ORWELL="orwell"
GRACE="grace"
TARGET_DIR_CMD='490gbm'
SSH_CMD="ssh -X "
BASH_CMD="exec bash"
DISPLAY_CMD="export DISPLAY=:0.0"


tmux has-session -t $SESSION

if [ $? != 0 ]
then
    tmux new-session -s $SESSION -d -n "$WILLPOWER"          # creates named session
    tmux send-keys -t $SESSION "490 && cd comp490-f15/" C-m        # moves to TARGET_DIR_CMD
    tmux send-keys     -t $SESSION "./time-gipsy.sh -u" C-m

    tmux split-window  -t $SESSION -v
    tmux select-layout -t $SESSION main-horizontal
    tmux send-keys     -t $SESSION "$TARGET_DIR_CMD" C-m
    tmux send-keys     -t $SESSION "./startMARFCATGMTNode.sh -n gmtd"

    tmux split-window  -t $SESSION -v
    tmux select-layout -t $SESSION main-horizontal
    tmux send-keys     -t $SESSION "$TARGET_DIR_CMD" C-m
    tmux send-keys     -t $SESSION "./startMARFCATRegularNode.sh -r "

    tmux new-window -a -t $SESSION  -n "$AWARENESS"
    tmux send-keys     -t $SESSION "$SSH_CMD $AWARENESS.encs.concordia.ca" C-m
    tmux send-keys     -t $SESSION "$BASH_CMD" C-m
    tmux send-keys     -t $SESSION "$TARGET_DIR_CMD" C-m
    tmux send-keys     -t $SESSION "./startMARFCATRegularNode.sh -r "

    tmux split-window  -t $SESSION -v
    tmux select-layout -t $SESSION main-horizontal
    tmux send-keys     -t $SESSION "$SSH_CMD $AWARENESS.encs.concordia.ca" C-m
    tmux send-keys     -t $SESSION "$BASH_CMD" C-m
    tmux send-keys     -t $SESSION "$TARGET_DIR_CMD" C-m
    tmux send-keys     -t $SESSION "./startMARFCATRegularNode.sh -r "

    tmux new-window -a -t $SESSION  -n "$ORWELL"
    tmux send-keys     -t $SESSION "$SSH_CMD $ORWELL.encs.concordia.ca" C-m
    tmux send-keys     -t $SESSION "$BASH_CMD" C-m
    tmux send-keys     -t $SESSION "$TARGET_DIR_CMD" C-m
    tmux send-keys     -t $SESSION "./startMARFCATRegularNode.sh -r "

    tmux split-window  -t $SESSION -v
    tmux select-layout -t $SESSION main-horizontal
    tmux send-keys     -t $SESSION "$SSH_CMD $ORWELL.encs.concordia.ca" C-m
    tmux send-keys     -t $SESSION "$BASH_CMD" C-m
    tmux send-keys     -t $SESSION "$TARGET_DIR_CMD" C-m
    tmux send-keys     -t $SESSION "./startMARFCATRegularNode.sh -r "

    tmux new-window -a -t $SESSION  -n "$GRACE"
    tmux send-keys     -t $SESSION "$SSH_CMD $GRACE.encs.concordia.ca" C-m
    tmux send-keys     -t $SESSION "$BASH_CMD" C-m
    tmux send-keys     -t $SESSION "$TARGET_DIR_CMD" C-m
    tmux send-keys     -t $SESSION "./startMARFCATRegularNode.sh -r "

    tmux select-window -t 0                                  # foncus to window 0
fi

tmux attach -t $SESSION
```

6.2 Sample STATS result file

Table 1: Consolidated results (), Part 1.

Run #	Guess	Configuration	GOOD	BAD	Precision,%
1	1st	-nopreprep -raw -fft -cheb -flucid	38	167	18.54
2	1st	CVE-2009-2559 (-nopreprep -raw -fft -cheb -flucid)	1	15	6.25
3	1st	CVE-2009-3242 (-nopreprep -raw -fft -cheb -flucid)	2	0	100.00
4	1st	CVE-2009-4376 (-nopreprep -raw -fft -cheb -flucid)	1	2	33.33
5	1st	CVE-2009-3243 (-nopreprep -raw -fft -cheb -flucid)	4	0	100.00
6	1st	CVE-2009-4378 (-nopreprep -raw -fft -cheb -flucid)	1	14	6.67
7	1st	CVE-2009-4377 (-nopreprep -raw -fft -cheb -flucid)	2	7	22.22
8	1st	CVE-2009-3241 (-nopreprep -raw -fft -cheb -flucid)	4	0	100.00
9	1st	CVE-2009-2560 (-nopreprep -raw -fft -cheb -flucid)	2	0	100.00
10	1st	CVE-2009-2561 (-nopreprep -raw -fft -cheb -flucid)	1	11	8.33
11	1st	CVE-2009-2562 (-nopreprep -raw -fft -cheb -flucid)	1	21	4.55
12	1st	CVE-2009-2563 (-nopreprep -raw -fft -cheb -flucid)	1	16	5.88
13	1st	CVE-2010-1455 (-nopreprep -raw -fft -cheb -flucid)	8	1	88.89
14	1st	CVE-2009-3829 (-nopreprep -raw -fft -cheb -flucid)	1	0	100.00
15	1st	CVE-2009-3549 (-nopreprep -raw -fft -cheb -flucid)	1	12	7.69
16	1st	CVE-2010-2287 (-nopreprep -raw -fft -cheb -flucid)	1	5	16.67
17	1st	CVE-2009-3550 (-nopreprep -raw -fft -cheb -flucid)	1	19	5.00
18	1st	CVE-2010-2285 (-nopreprep -raw -fft -cheb -flucid)	1	10	9.09
19	1st	CVE-2009-3551 (-nopreprep -raw -fft -cheb -flucid)	1	0	100.00
20	1st	CVE-2010-2286 (-nopreprep -raw -fft -cheb -flucid)	1	0	100.00
21	1st	CVE-2010-2283 (-nopreprep -raw -fft -cheb -flucid)	1	0	100.00
22	1st	CVE-2010-2284 (-nopreprep -raw -fft -cheb -flucid)	1	21	4.55
23	1st	CVE-2010-0304 (-nopreprep -raw -fft -cheb -flucid)	1	13	7.14
24	1st	-nopreprep -raw -fft -diff -flucid	38	167	18.54
25	1st	CVE-2009-2559 (-nopreprep -raw -fft -diff -flucid)	1	15	6.25
26	1st	CVE-2009-3242 (-nopreprep -raw -fft -diff -flucid)	2	0	100.00
27	1st	CVE-2009-4376 (-nopreprep -raw -fft -diff -flucid)	1	2	33.33
28	1st	CVE-2009-3243 (-nopreprep -raw -fft -diff -flucid)	4	0	100.00
29	1st	CVE-2009-4378 (-nopreprep -raw -fft -diff -flucid)	1	14	6.67
30	1st	CVE-2009-4377 (-nopreprep -raw -fft -diff -flucid)	2	7	22.22
31	1st	CVE-2009-3241 (-nopreprep -raw -fft -diff -flucid)	4	0	100.00
32	1st	CVE-2009-2560 (-nopreprep -raw -fft -diff -flucid)	2	0	100.00
33	1st	CVE-2009-2561 (-nopreprep -raw -fft -diff -flucid)	1	11	8.33
34	1st	CVE-2009-2562 (-nopreprep -raw -fft -diff -flucid)	1	21	4.55
35	1st	CVE-2009-2563 (-nopreprep -raw -fft -diff -flucid)	1	16	5.88
36	1st	CVE-2010-1455 (-nopreprep -raw -fft -diff -flucid)	8	1	88.89
37	1st	CVE-2009-3829 (-nopreprep -raw -fft -diff -flucid)	1	0	100.00
38	1st	CVE-2009-3549 (-nopreprep -raw -fft -diff -flucid)	1	12	7.69
39	1st	CVE-2010-2287 (-nopreprep -raw -fft -diff -flucid)	1	5	16.67
40	1st	CVE-2009-3550 (-nopreprep -raw -fft -diff -flucid)	1	19	5.00
41	1st	CVE-2010-2285 (-nopreprep -raw -fft -diff -flucid)	1	10	9.09
42	1st	CVE-2009-3551 (-nopreprep -raw -fft -diff -flucid)	1	0	100.00
43	1st	CVE-2010-2286 (-nopreprep -raw -fft -diff -flucid)	1	0	100.00
44	1st	CVE-2010-2283 (-nopreprep -raw -fft -diff -flucid)	1	0	100.00
45	1st	CVE-2010-2284 (-nopreprep -raw -fft -diff -flucid)	1	21	4.55
46	1st	CVE-2010-0304 (-nopreprep -raw -fft -diff -flucid)	1	13	7.14
47	1st	-nopreprep -raw -fft -eucl -flucid	29	176	14.15
48	1st	CVE-2009-2559 (-nopreprep -raw -fft -eucl -flucid)	1	15	6.25
49	1st	CVE-2009-3242 (-nopreprep -raw -fft -eucl -flucid)	0	2	0.00

Table 2: Consolidated results (), Part 2.

Run #	Guess	Configuration	GOOD	BAD	Precision,%
50	1st	CVE-2009-4376 (-nopreprep -raw -fft -eucl -flucid)	1	2	33.33
51	1st	CVE-2009-3243 (-nopreprep -raw -fft -eucl -flucid)	3	1	75.00
52	1st	CVE-2009-4378 (-nopreprep -raw -fft -eucl -flucid)	1	14	6.67
53	1st	CVE-2009-4377 (-nopreprep -raw -fft -eucl -flucid)	2	7	22.22
54	1st	CVE-2009-3241 (-nopreprep -raw -fft -eucl -flucid)	2	2	50.00
55	1st	CVE-2009-2560 (-nopreprep -raw -fft -eucl -flucid)	2	0	100.00
56	1st	CVE-2009-2561 (-nopreprep -raw -fft -eucl -flucid)	1	11	8.33
57	1st	CVE-2009-2562 (-nopreprep -raw -fft -eucl -flucid)	1	21	4.55
58	1st	CVE-2009-2563 (-nopreprep -raw -fft -eucl -flucid)	1	16	5.88
59	1st	CVE-2010-1455 (-nopreprep -raw -fft -eucl -flucid)	4	5	44.44
60	1st	CVE-2009-3829 (-nopreprep -raw -fft -eucl -flucid)	1	0	100.00
61	1st	CVE-2009-3549 (-nopreprep -raw -fft -eucl -flucid)	1	12	7.69
62	1st	CVE-2010-2287 (-nopreprep -raw -fft -eucl -flucid)	1	5	16.67
63	1st	CVE-2009-3550 (-nopreprep -raw -fft -eucl -flucid)	1	19	5.00
64	1st	CVE-2010-2285 (-nopreprep -raw -fft -eucl -flucid)	1	10	9.09
65	1st	CVE-2009-3551 (-nopreprep -raw -fft -eucl -flucid)	1	0	100.00
66	1st	CVE-2010-2286 (-nopreprep -raw -fft -eucl -flucid)	1	0	100.00
67	1st	CVE-2010-2283 (-nopreprep -raw -fft -eucl -flucid)	1	0	100.00
68	1st	CVE-2010-2284 (-nopreprep -raw -fft -eucl -flucid)	1	21	4.55
69	1st	CVE-2010-0304 (-nopreprep -raw -fft -eucl -flucid)	1	13	7.14
70	1st	ALL	191	1207	13.66
71	1st	CVE-2009-2559 (ALL)	6	96	5.88
72	1st	CVE-2009-3242 (ALL)	7	46	13.21
73	1st	CVE-2009-4376 (ALL)	6	12	33.33
74	1st	CVE-2009-3243 (ALL)	18	45	28.57
75	1st	CVE-2009-4378 (ALL)	6	84	6.67
76	1st	CVE-2009-4377 (ALL)	12	63	16.00
77	1st	CVE-2009-3241 (ALL)	16	8	66.67
78	1st	CVE-2009-2560 (ALL)	8	44	15.38
79	1st	CVE-2009-2561 (ALL)	6	76	7.32
80	1st	CVE-2009-2562 (ALL)	6	126	4.55
81	1st	CVE-2009-2563 (ALL)	6	96	5.88
82	1st	CVE-2010-1455 (ALL)	34	20	62.96
83	1st	CVE-2009-3829 (ALL)	6	0	100.00
84	1st	CVE-2009-3549 (ALL)	6	72	7.69
85	1st	CVE-2010-2287 (ALL)	6	30	16.67
86	1st	CVE-2009-3550 (ALL)	6	114	5.00
87	1st	CVE-2010-2285 (ALL)	6	71	7.79
88	1st	CVE-2009-3551 (ALL)	6	0	100.00
89	1st	CVE-2010-2286 (ALL)	6	0	100.00
90	1st	CVE-2010-2283 (ALL)	6	0	100.00
91	1st	CVE-2010-2284 (ALL)	6	126	4.55
92	1st	CVE-2010-0304 (ALL)	6	78	7.14
93	1st	-nopreprep -raw -fft -hamming -flucid	26	179	12.68
94	1st	CVE-2009-2559 (-nopreprep -raw -fft -hamming -flucid)	1	15	6.25
95	1st	CVE-2009-3242 (-nopreprep -raw -fft -hamming -flucid)	1	1	50.00
96	1st	CVE-2009-4376 (-nopreprep -raw -fft -hamming -flucid)	1	2	33.33
97	1st	CVE-2009-3243 (-nopreprep -raw -fft -hamming -flucid)	3	1	75.00
98	1st	CVE-2009-4378 (-nopreprep -raw -fft -hamming -flucid)	1	14	6.67
99	1st	CVE-2009-4377 (-nopreprep -raw -fft -hamming -flucid)	2	7	22.22

Table 3: Consolidated results (), Part 3.

Run #	Guess	Configuration	GOOD	BAD	Precision,%
100	1st	CVE-2009-3241 (-nopreprep -raw -fft -hamming -flucid)	0	4	0.00
101	1st	CVE-2009-2560 (-nopreprep -raw -fft -hamming -flucid)	0	2	0.00
102	1st	CVE-2009-2561 (-nopreprep -raw -fft -hamming -flucid)	1	11	8.33
103	1st	CVE-2009-2562 (-nopreprep -raw -fft -hamming -flucid)	1	21	4.55
104	1st	CVE-2009-2563 (-nopreprep -raw -fft -hamming -flucid)	1	16	5.88
105	1st	CVE-2010-1455 (-nopreprep -raw -fft -hamming -flucid)	4	5	44.44
106	1st	CVE-2009-3829 (-nopreprep -raw -fft -hamming -flucid)	1	0	100.00
107	1st	CVE-2009-3549 (-nopreprep -raw -fft -hamming -flucid)	1	12	7.69
108	1st	CVE-2010-2287 (-nopreprep -raw -fft -hamming -flucid)	1	5	16.67
109	1st	CVE-2009-3550 (-nopreprep -raw -fft -hamming -flucid)	1	19	5.00
110	1st	CVE-2010-2285 (-nopreprep -raw -fft -hamming -flucid)	1	10	9.09
111	1st	CVE-2009-3551 (-nopreprep -raw -fft -hamming -flucid)	1	0	100.00
112	1st	CVE-2010-2286 (-nopreprep -raw -fft -hamming -flucid)	1	0	100.00
113	1st	CVE-2010-2283 (-nopreprep -raw -fft -hamming -flucid)	1	0	100.00
114	1st	CVE-2010-2284 (-nopreprep -raw -fft -hamming -flucid)	1	21	4.55
115	1st	CVE-2010-0304 (-nopreprep -raw -fft -hamming -flucid)	1	13	7.14
116	1st	-nopreprep -raw -fft -mink -flucid	23	182	11.22
117	1st	CVE-2009-2559 (-nopreprep -raw -fft -mink -flucid)	1	15	6.25
118	1st	CVE-2009-3242 (-nopreprep -raw -fft -mink -flucid)	0	2	0.00
119	1st	CVE-2009-4376 (-nopreprep -raw -fft -mink -flucid)	1	2	33.33
120	1st	CVE-2009-3243 (-nopreprep -raw -fft -mink -flucid)	1	3	25.00
121	1st	CVE-2009-4378 (-nopreprep -raw -fft -mink -flucid)	1	14	6.67
122	1st	CVE-2009-4377 (-nopreprep -raw -fft -mink -flucid)	2	7	22.22
123	1st	CVE-2009-3241 (-nopreprep -raw -fft -mink -flucid)	2	2	50.00
124	1st	CVE-2009-2560 (-nopreprep -raw -fft -mink -flucid)	0	2	0.00
125	1st	CVE-2009-2561 (-nopreprep -raw -fft -mink -flucid)	1	11	8.33
126	1st	CVE-2009-2562 (-nopreprep -raw -fft -mink -flucid)	1	21	4.55
127	1st	CVE-2009-2563 (-nopreprep -raw -fft -mink -flucid)	1	16	5.88
128	1st	CVE-2010-1455 (-nopreprep -raw -fft -mink -flucid)	2	7	22.22
129	1st	CVE-2009-3829 (-nopreprep -raw -fft -mink -flucid)	1	0	100.00
130	1st	CVE-2009-3549 (-nopreprep -raw -fft -mink -flucid)	1	12	7.69
131	1st	CVE-2010-2287 (-nopreprep -raw -fft -mink -flucid)	1	5	16.67
132	1st	CVE-2009-3550 (-nopreprep -raw -fft -mink -flucid)	1	19	5.00
133	1st	CVE-2010-2285 (-nopreprep -raw -fft -mink -flucid)	1	10	9.09
134	1st	CVE-2009-3551 (-nopreprep -raw -fft -mink -flucid)	1	0	100.00
135	1st	CVE-2010-2286 (-nopreprep -raw -fft -mink -flucid)	1	0	100.00
136	1st	CVE-2010-2283 (-nopreprep -raw -fft -mink -flucid)	1	0	100.00
137	1st	CVE-2010-2284 (-nopreprep -raw -fft -mink -flucid)	1	21	4.55
138	1st	CVE-2010-0304 (-nopreprep -raw -fft -mink -flucid)	1	13	7.14
139	1st	-nopreprep -raw -fft -cos -flucid	37	336	9.92
140	1st	CVE-2009-2559 (-nopreprep -raw -fft -cos -flucid)	1	21	4.55
141	1st	CVE-2009-3242 (-nopreprep -raw -fft -cos -flucid)	2	41	4.65
142	1st	CVE-2009-4376 (-nopreprep -raw -fft -cos -flucid)	1	2	33.33
143	1st	CVE-2009-3243 (-nopreprep -raw -fft -cos -flucid)	3	40	6.98
144	1st	CVE-2009-4378 (-nopreprep -raw -fft -cos -flucid)	1	14	6.67
145	1st	CVE-2009-4377 (-nopreprep -raw -fft -cos -flucid)	2	28	6.67
146	1st	CVE-2009-3241 (-nopreprep -raw -fft -cos -flucid)	4	0	100.00
147	1st	CVE-2009-2560 (-nopreprep -raw -fft -cos -flucid)	2	40	4.76
148	1st	CVE-2009-2561 (-nopreprep -raw -fft -cos -flucid)	1	21	4.55
149	1st	CVE-2009-2562 (-nopreprep -raw -fft -cos -flucid)	1	21	4.55

Table 4: Consolidated results (), Part 4.

Run #	Guess	Configuration	GOOD	BAD	Precision,%
150	1st	CVE-2009-2563 (-nopreprep -raw -fft -cos -flucid)	1	16	5.88
151	1st	CVE-2010-1455 (-nopreprep -raw -fft -cos -flucid)	8	1	88.89
152	1st	CVE-2009-3829 (-nopreprep -raw -fft -cos -flucid)	1	0	100.00
153	1st	CVE-2009-3549 (-nopreprep -raw -fft -cos -flucid)	1	12	7.69
154	1st	CVE-2010-2287 (-nopreprep -raw -fft -cos -flucid)	1	5	16.67
155	1st	CVE-2009-3550 (-nopreprep -raw -fft -cos -flucid)	1	19	5.00
156	1st	CVE-2010-2285 (-nopreprep -raw -fft -cos -flucid)	1	21	4.55
157	1st	CVE-2009-3551 (-nopreprep -raw -fft -cos -flucid)	1	0	100.00
158	1st	CVE-2010-2286 (-nopreprep -raw -fft -cos -flucid)	1	0	100.00
159	1st	CVE-2010-2283 (-nopreprep -raw -fft -cos -flucid)	1	0	100.00
160	1st	CVE-2010-2284 (-nopreprep -raw -fft -cos -flucid)	1	21	4.55
161	1st	CVE-2010-0304 (-nopreprep -raw -fft -cos -flucid)	1	13	7.14
162	2nd	-nopreprep -raw -fft -cheb -flucid	43	162	20.98
163	2nd	CVE-2009-2559 (-nopreprep -raw -fft -cheb -flucid)	1	15	6.25
164	2nd	CVE-2009-3242 (-nopreprep -raw -fft -cheb -flucid)	2	0	100.00
165	2nd	CVE-2009-4376 (-nopreprep -raw -fft -cheb -flucid)	1	2	33.33
166	2nd	CVE-2009-3243 (-nopreprep -raw -fft -cheb -flucid)	4	0	100.00
167	2nd	CVE-2009-4378 (-nopreprep -raw -fft -cheb -flucid)	1	14	6.67
168	2nd	CVE-2009-4377 (-nopreprep -raw -fft -cheb -flucid)	2	7	22.22
169	2nd	CVE-2009-3241 (-nopreprep -raw -fft -cheb -flucid)	4	0	100.00
170	2nd	CVE-2009-2560 (-nopreprep -raw -fft -cheb -flucid)	2	0	100.00
171	2nd	CVE-2009-2561 (-nopreprep -raw -fft -cheb -flucid)	1	11	8.33
172	2nd	CVE-2009-2562 (-nopreprep -raw -fft -cheb -flucid)	1	21	4.55
173	2nd	CVE-2009-2563 (-nopreprep -raw -fft -cheb -flucid)	1	16	5.88
174	2nd	CVE-2010-1455 (-nopreprep -raw -fft -cheb -flucid)	8	1	88.89
175	2nd	CVE-2009-3829 (-nopreprep -raw -fft -cheb -flucid)	1	0	100.00
176	2nd	CVE-2009-3549 (-nopreprep -raw -fft -cheb -flucid)	1	12	7.69
177	2nd	CVE-2010-2287 (-nopreprep -raw -fft -cheb -flucid)	6	0	100.00
178	2nd	CVE-2009-3550 (-nopreprep -raw -fft -cheb -flucid)	1	19	5.00
179	2nd	CVE-2010-2285 (-nopreprep -raw -fft -cheb -flucid)	1	10	9.09
180	2nd	CVE-2009-3551 (-nopreprep -raw -fft -cheb -flucid)	1	0	100.00
181	2nd	CVE-2010-2286 (-nopreprep -raw -fft -cheb -flucid)	1	0	100.00
182	2nd	CVE-2010-2283 (-nopreprep -raw -fft -cheb -flucid)	1	0	100.00
183	2nd	CVE-2010-2284 (-nopreprep -raw -fft -cheb -flucid)	1	21	4.55
184	2nd	CVE-2010-0304 (-nopreprep -raw -fft -cheb -flucid)	1	13	7.14
185	2nd	-nopreprep -raw -fft -diff -flucid	43	162	20.98
186	2nd	CVE-2009-2559 (-nopreprep -raw -fft -diff -flucid)	1	15	6.25
187	2nd	CVE-2009-3242 (-nopreprep -raw -fft -diff -flucid)	2	0	100.00
188	2nd	CVE-2009-4376 (-nopreprep -raw -fft -diff -flucid)	1	2	33.33
189	2nd	CVE-2009-3243 (-nopreprep -raw -fft -diff -flucid)	4	0	100.00
190	2nd	CVE-2009-4378 (-nopreprep -raw -fft -diff -flucid)	1	14	6.67
191	2nd	CVE-2009-4377 (-nopreprep -raw -fft -diff -flucid)	2	7	22.22
192	2nd	CVE-2009-3241 (-nopreprep -raw -fft -diff -flucid)	4	0	100.00
193	2nd	CVE-2009-2560 (-nopreprep -raw -fft -diff -flucid)	2	0	100.00
194	2nd	CVE-2009-2561 (-nopreprep -raw -fft -diff -flucid)	1	11	8.33
195	2nd	CVE-2009-2562 (-nopreprep -raw -fft -diff -flucid)	1	21	4.55
196	2nd	CVE-2009-2563 (-nopreprep -raw -fft -diff -flucid)	1	16	5.88
197	2nd	CVE-2010-1455 (-nopreprep -raw -fft -diff -flucid)	8	1	88.89
198	2nd	CVE-2009-3829 (-nopreprep -raw -fft -diff -flucid)	1	0	100.00
199	2nd	CVE-2009-3549 (-nopreprep -raw -fft -diff -flucid)	1	12	7.69

Table 5: Consolidated results (), Part 5.

Run #	Guess	Configuration	GOOD	BAD	Precision,%
200	2nd	CVE-2010-2287 (-noprepreр -raw -fft -diff -flucid)	6	0	100.00
201	2nd	CVE-2009-3550 (-nopreprep -raw -fft -diff -flucid)	1	19	5.00
202	2nd	CVE-2010-2285 (-nopreprep -raw -fft -diff -flucid)	1	10	9.09
203	2nd	CVE-2009-3551 (-nopreprep -raw -fft -diff -flucid)	1	0	100.00
204	2nd	CVE-2010-2286 (-nopreprep -raw -fft -diff -flucid)	1	0	100.00
205	2nd	CVE-2010-2283 (-nopreprep -raw -fft -diff -flucid)	1	0	100.00
206	2nd	CVE-2010-2284 (-nopreprep -raw -fft -diff -flucid)	1	21	4.55
207	2nd	CVE-2010-0304 (-nopreprep -raw -fft -diff -flucid)	1	13	7.14
208	2nd	-nopreprep -raw -fft -eucl -flucid	34	171	16.59
209	2nd	CVE-2009-2559 (-nopreprep -raw -fft -eucl -flucid)	1	15	6.25
210	2nd	CVE-2009-3242 (-nopreprep -raw -fft -eucl -flucid)	0	2	0.00
211	2nd	CVE-2009-4376 (-nopreprep -raw -fft -eucl -flucid)	1	2	33.33
212	2nd	CVE-2009-3243 (-nopreprep -raw -fft -eucl -flucid)	3	1	75.00
213	2nd	CVE-2009-4378 (-nopreprep -raw -fft -eucl -flucid)	1	14	6.67
214	2nd	CVE-2009-4377 (-nopreprep -raw -fft -eucl -flucid)	2	7	22.22
215	2nd	CVE-2009-3241 (-nopreprep -raw -fft -eucl -flucid)	2	2	50.00
216	2nd	CVE-2009-2560 (-nopreprep -raw -fft -eucl -flucid)	2	0	100.00
217	2nd	CVE-2009-2561 (-nopreprep -raw -fft -eucl -flucid)	1	11	8.33
218	2nd	CVE-2009-2562 (-nopreprep -raw -fft -eucl -flucid)	1	21	4.55
219	2nd	CVE-2009-2563 (-nopreprep -raw -fft -eucl -flucid)	1	16	5.88
220	2nd	CVE-2010-1455 (-nopreprep -raw -fft -eucl -flucid)	4	5	44.44
221	2nd	CVE-2009-3829 (-nopreprep -raw -fft -eucl -flucid)	1	0	100.00
222	2nd	CVE-2009-3549 (-nopreprep -raw -fft -eucl -flucid)	1	12	7.69
223	2nd	CVE-2010-2287 (-nopreprep -raw -fft -eucl -flucid)	6	0	100.00
224	2nd	CVE-2009-3550 (-nopreprep -raw -fft -eucl -flucid)	1	19	5.00
225	2nd	CVE-2010-2285 (-nopreprep -raw -fft -eucl -flucid)	1	10	9.09
226	2nd	CVE-2009-3551 (-nopreprep -raw -fft -eucl -flucid)	1	0	100.00
227	2nd	CVE-2010-2286 (-nopreprep -raw -fft -eucl -flucid)	1	0	100.00
228	2nd	CVE-2010-2283 (-nopreprep -raw -fft -eucl -flucid)	1	0	100.00
229	2nd	CVE-2010-2284 (-nopreprep -raw -fft -eucl -flucid)	1	21	4.55
230	2nd	CVE-2010-0304 (-nopreprep -raw -fft -eucl -flucid)	1	13	7.14
231	2nd	ALL	221	1177	15.81
232	2nd	CVE-2009-2559 (ALL)	6	96	5.88
233	2nd	CVE-2009-3242 (ALL)	7	46	13.21
234	2nd	CVE-2009-4376 (ALL)	6	12	33.33
235	2nd	CVE-2009-3243 (ALL)	18	45	28.57
236	2nd	CVE-2009-4378 (ALL)	6	84	6.67
237	2nd	CVE-2009-4377 (ALL)	12	63	16.00
238	2nd	CVE-2009-3241 (ALL)	16	8	66.67
239	2nd	CVE-2009-2560 (ALL)	8	44	15.38
240	2nd	CVE-2009-2561 (ALL)	6	76	7.32
241	2nd	CVE-2009-2562 (ALL)	6	126	4.55
242	2nd	CVE-2009-2563 (ALL)	6	96	5.88
243	2nd	CVE-2010-1455 (ALL)	34	20	62.96
244	2nd	CVE-2009-3829 (ALL)	6	0	100.00
245	2nd	CVE-2009-3549 (ALL)	6	72	7.69
246	2nd	CVE-2010-2287 (ALL)	36	0	100.00
247	2nd	CVE-2009-3550 (ALL)	6	114	5.00
248	2nd	CVE-2010-2285 (ALL)	6	71	7.79
249	2nd	CVE-2009-3551 (ALL)	6	0	100.00

Table 6: Consolidated results (), Part 6.

Run #	Guess	Configuration	GOOD	BAD	Precision,%
250	2nd	CVE-2010-2286 (ALL)	6	0	100.00
251	2nd	CVE-2010-2283 (ALL)	6	0	100.00
252	2nd	CVE-2010-2284 (ALL)	6	126	4.55
253	2nd	CVE-2010-0304 (ALL)	6	78	7.14
254	2nd	-nopreprep -raw -fft -hamming -flucid	31	174	15.12
255	2nd	CVE-2009-2559 (-nopreprep -raw -fft -hamming -flucid)	1	15	6.25
256	2nd	CVE-2009-3242 (-nopreprep -raw -fft -hamming -flucid)	1	1	50.00
257	2nd	CVE-2009-4376 (-nopreprep -raw -fft -hamming -flucid)	1	2	33.33
258	2nd	CVE-2009-3243 (-nopreprep -raw -fft -hamming -flucid)	3	1	75.00
259	2nd	CVE-2009-4378 (-nopreprep -raw -fft -hamming -flucid)	1	14	6.67
260	2nd	CVE-2009-4377 (-nopreprep -raw -fft -hamming -flucid)	2	7	22.22
261	2nd	CVE-2009-3241 (-nopreprep -raw -fft -hamming -flucid)	0	4	0.00
262	2nd	CVE-2009-2560 (-nopreprep -raw -fft -hamming -flucid)	0	2	0.00
263	2nd	CVE-2009-2561 (-nopreprep -raw -fft -hamming -flucid)	1	11	8.33
264	2nd	CVE-2009-2562 (-nopreprep -raw -fft -hamming -flucid)	1	21	4.55
265	2nd	CVE-2009-2563 (-nopreprep -raw -fft -hamming -flucid)	1	16	5.88
266	2nd	CVE-2010-1455 (-nopreprep -raw -fft -hamming -flucid)	4	5	44.44
267	2nd	CVE-2009-3829 (-nopreprep -raw -fft -hamming -flucid)	1	0	100.00
268	2nd	CVE-2009-3549 (-nopreprep -raw -fft -hamming -flucid)	1	12	7.69
269	2nd	CVE-2010-2287 (-nopreprep -raw -fft -hamming -flucid)	6	0	100.00
270	2nd	CVE-2009-3550 (-nopreprep -raw -fft -hamming -flucid)	1	19	5.00
271	2nd	CVE-2010-2285 (-nopreprep -raw -fft -hamming -flucid)	1	10	9.09
272	2nd	CVE-2009-3551 (-nopreprep -raw -fft -hamming -flucid)	1	0	100.00
273	2nd	CVE-2010-2286 (-nopreprep -raw -fft -hamming -flucid)	1	0	100.00
274	2nd	CVE-2010-2283 (-nopreprep -raw -fft -hamming -flucid)	1	0	100.00
275	2nd	CVE-2010-2284 (-nopreprep -raw -fft -hamming -flucid)	1	21	4.55
276	2nd	CVE-2010-0304 (-nopreprep -raw -fft -hamming -flucid)	1	13	7.14
277	2nd	-nopreprep -raw -fft -mink -flucid	28	177	13.66
278	2nd	CVE-2009-2559 (-nopreprep -raw -fft -mink -flucid)	1	15	6.25
279	2nd	CVE-2009-3242 (-nopreprep -raw -fft -mink -flucid)	0	2	0.00
280	2nd	CVE-2009-4376 (-nopreprep -raw -fft -mink -flucid)	1	2	33.33
281	2nd	CVE-2009-3243 (-nopreprep -raw -fft -mink -flucid)	1	3	25.00
282	2nd	CVE-2009-4378 (-nopreprep -raw -fft -mink -flucid)	1	14	6.67
283	2nd	CVE-2009-4377 (-nopreprep -raw -fft -mink -flucid)	2	7	22.22
284	2nd	CVE-2009-3241 (-nopreprep -raw -fft -mink -flucid)	2	2	50.00
285	2nd	CVE-2009-2560 (-nopreprep -raw -fft -mink -flucid)	0	2	0.00
286	2nd	CVE-2009-2561 (-nopreprep -raw -fft -mink -flucid)	1	11	8.33
287	2nd	CVE-2009-2562 (-nopreprep -raw -fft -mink -flucid)	1	21	4.55
288	2nd	CVE-2009-2563 (-nopreprep -raw -fft -mink -flucid)	1	16	5.88
289	2nd	CVE-2010-1455 (-nopreprep -raw -fft -mink -flucid)	2	7	22.22
290	2nd	CVE-2009-3829 (-nopreprep -raw -fft -mink -flucid)	1	0	100.00
291	2nd	CVE-2009-3549 (-nopreprep -raw -fft -mink -flucid)	1	12	7.69
292	2nd	CVE-2010-2287 (-nopreprep -raw -fft -mink -flucid)	6	0	100.00
293	2nd	CVE-2009-3550 (-nopreprep -raw -fft -mink -flucid)	1	19	5.00
294	2nd	CVE-2010-2285 (-nopreprep -raw -fft -mink -flucid)	1	10	9.09
295	2nd	CVE-2009-3551 (-nopreprep -raw -fft -mink -flucid)	1	0	100.00
296	2nd	CVE-2010-2286 (-nopreprep -raw -fft -mink -flucid)	1	0	100.00
297	2nd	CVE-2010-2283 (-nopreprep -raw -fft -mink -flucid)	1	0	100.00
298	2nd	CVE-2010-2284 (-nopreprep -raw -fft -mink -flucid)	1	21	4.55
299	2nd	CVE-2010-0304 (-nopreprep -raw -fft -mink -flucid)	1	13	7.14

Table 7: Consolidated results (), Part 7.

Run #	Guess	Configuration	GOOD	BAD	Precision,%
300	2nd	-nopreprep -raw -fft -cos -flucid	42	331	11.26
301	2nd	CVE-2009-2559 (-nopreprep -raw -fft -cos -flucid)	1	21	4.55
302	2nd	CVE-2009-3242 (-nopreprep -raw -fft -cos -flucid)	2	41	4.65
303	2nd	CVE-2009-4376 (-nopreprep -raw -fft -cos -flucid)	1	2	33.33
304	2nd	CVE-2009-3243 (-nopreprep -raw -fft -cos -flucid)	3	40	6.98
305	2nd	CVE-2009-4378 (-nopreprep -raw -fft -cos -flucid)	1	14	6.67
306	2nd	CVE-2009-4377 (-nopreprep -raw -fft -cos -flucid)	2	28	6.67
307	2nd	CVE-2009-3241 (-nopreprep -raw -fft -cos -flucid)	4	0	100.00
308	2nd	CVE-2009-2560 (-nopreprep -raw -fft -cos -flucid)	2	40	4.76
309	2nd	CVE-2009-2561 (-nopreprep -raw -fft -cos -flucid)	1	21	4.55
310	2nd	CVE-2009-2562 (-nopreprep -raw -fft -cos -flucid)	1	21	4.55
311	2nd	CVE-2009-2563 (-nopreprep -raw -fft -cos -flucid)	1	16	5.88
312	2nd	CVE-2010-1455 (-nopreprep -raw -fft -cos -flucid)	8	1	88.89
313	2nd	CVE-2009-3829 (-nopreprep -raw -fft -cos -flucid)	1	0	100.00
314	2nd	CVE-2009-3549 (-nopreprep -raw -fft -cos -flucid)	1	12	7.69
315	2nd	CVE-2010-2287 (-nopreprep -raw -fft -cos -flucid)	6	0	100.00
316	2nd	CVE-2009-3550 (-nopreprep -raw -fft -cos -flucid)	1	19	5.00
317	2nd	CVE-2010-2285 (-nopreprep -raw -fft -cos -flucid)	1	21	4.55
318	2nd	CVE-2009-3551 (-nopreprep -raw -fft -cos -flucid)	1	0	100.00
319	2nd	CVE-2010-2286 (-nopreprep -raw -fft -cos -flucid)	1	0	100.00
320	2nd	CVE-2010-2283 (-nopreprep -raw -fft -cos -flucid)	1	0	100.00
321	2nd	CVE-2010-2284 (-nopreprep -raw -fft -cos -flucid)	1	21	4.55
322	2nd	CVE-2010-0304 (-nopreprep -raw -fft -cos -flucid)	1	13	7.14

7 References

References

[1] Common vulnerabilities and exposures. https://cve.mitre.org/.

[2] java 6u45 release notes. http://www.oracle.com/technetwork/java/javase/6u45-relnotes-1932876.html. Accessed: 2015-09.

[3] Yi Ji. Scalability evaluation of the GIPSY runtime system. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, March 2011. http://spectrum.library.concordia.ca/7152/.

[4] Serguei A. Mokhov. The use of machine learning with signal- and NLP processing of source code to fingerprint, detect, and classify vulnerabilities and weaknesses with MARFCAT. [online], October 2010. Online at http://arxiv.org/abs/1010.2511.

[5] Serguei A. Mokhov. *Intensional Cyberforensics*. PhD thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, September 2013. Online at http://arxiv.org/abs/1312.0466.

[6] Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. MARFCAT: Fast code analysis for defects and vulnerabilities. In Olga Baysal and Latifa Guerrouj, editors, *Proceedings of SWAN'15*, pages 35–38. IEEE, March 2015.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the definition; numbers in roman refer to the pages where the entry is used.