École Polytechnique Fédérale de Lausanne

Investigating Clang Static Analyzer's False-positives and False-negatives

by Cosme Jordan

Master Thesis

Approved by the Examining Committee:

Prof. Dr. sc. ETH Mathias Payer
Thesis Advisor

Francesco Berla
External Expert

Gwangmu Lee
Thesis Supervisor

EPFL IC IINFCOM HEXHIVE
BC 160 (Bâtiment BC)
Station 14
CH-1015 Lausanne

June 22, 2023

Live in the present, enjoy the moments that are now, because really that’s all you have.

— Jeff Shapiro[1]

Dedicated to my family.

[1] From Advanced Paragliding written by Gavin McClurg

Acknowledgments

I want to thank my advisor, Mathias Payer, for directly challenging us during the first meeting and for providing us with a template which really helped during the redaction of this thesis[2]. I also want to thank Francesco Berla who agreed to be the external expert. Finally, I also want to thank my supervisor, Gwangmu Lee, for his support, guidance and for all the feedback I received during this project. I really enjoyed working on this project. I wish all of them the best in their careers.

I want to thank my family for all their support during my studies and for believing in me. I wouldn't be here without them.

Last but not least, I especially want to thank all the people I've met during my paragliding journey. Flying allowed me to think about something else once in a while and keep my sanity during diffcult times in the last few years.
Happy flights, people!

Lausanne, June 22, 2023 Cosme Jordan

[2]The template is available at https://github.com/HexHive/thesis_template

Abstract

Static analyzers can be used to find bugs during the compilation of a program. The Clang Static Analyzer (CSA) is such a tool that is based on a popular approach of static analysis, abstract interpretation, but CSA is known to suffer from many false positives (FPs) and false negatives (FNs). Many papers have been published to benchmark and improve static analyzers, but they never implemented an automatic method to run static analyzers against benchmarks and gather statistics, while some of the results are outdated as CSA itself has evolved since then. In this work, we investigate the causes for the FPs and FNs with three different test suites: the Juliet Test Suite, the Cyber Grand Challenge and Magma. The first benchmark is a synthetic benchmark (SY) while the other two represent real-world (RW) code. Our results show that the causes for the FPs and FNs are different between the SY and RW benchmarks. For the RW, we found out that the most common causes were the experimental checkers and the complicated structures for the environment model (i.d., around 50% of FPs and FNs for each library), while for the SY: ignoring failures of function and wrong checker locations are the most prevalent. Furthermore, some causes are more noticeable with the RW benchmarks (e.g, experimental checkers, library functions who change and invalidate parts of the environment).

Contents

Chapter 1

Introduction

In a world where the number of software keeps increasing every day, the number of bugs also keeps on increasing. Research states that we can find on average about 25 bugs per 1,000 line of code [1, 2]. The news about security flaws are also a common recurrence. Ill-intentioned groups of people attempt to exploit vulnerabilities for their own personal gains [3–7]. As a result, it is important to find flaws and bugs before someone uses it to damages users or other property.

Software tools have been developed to help programmers test and find bugs automatically during the runtime of programs, called dynamic analysis, such as sanitizers, for example with ASan and TSan. We also have fuzzers, which can find bugs by providing random data to a program. Many papers have been published about fuzzing over the years and many different fuzzers have been developed, such as AFL, AFL++, and HonggFuzz [8–11].

Meanwhile, detecting bugs at compile time still has benefits as programmers and companies can speed up the development process by early-detecting software flaws. For example, compiler warnings can discover and find bugs, mistakes and code smells early in the development cycle of a software. Warnings such as using the result of an assignment as a condition without parentheses, shift count is negative, or comparison between signed and unsigned integer expressions can give programmers notice that potential bugs are present in their code.

However, compiler warnings are limited in their use and effects. Indeed, compiler warnings do not account for the control or data flows of a program as they usually only operate on the basic abstract syntax tree, and are often limited to a specific compilation unit. Static analyzers have been developed to improve this and can find bugs that compiler warnings would miss at the cost of overhead and performance impact.

Despite the fact that a lot of research has been done on fuzzing, static analyzers are still relevant nowadays. Fuzzing is inherently incomplete, we cannot know if there are undiscovered bugs with fuzzing and furthermore, fuzzing a program can be extremely slow, while static analysis can tackle both problems. Some research have also been done to reduce the overhead of dynamic analysis using static analysis [12].

Static analyzers (SA) can find bugs triggered by certain execution paths, and they also account for the expected behavior of well-known functions or APIs (e.g., knowing that using a value provided to `free` is prohibited allows for detecting a *use-after-free*-triggering path). SA can also find bugs which are costly or inaccurate to detect, and would not be implemented in a standard execution of a compiler because of the overhead and slow compile execution.

Clang Static Analyzer (CSA) is a widely-used static analyzer [13], that can find the same kind of bugs as other popular commercial static analyzers such as SonarQube [14] and Coverity [15]. Furthermore, according to Balázs Benics, an LLVM contributor and Sonar employee, SonarSource, the SonarQube founding company, *"make heavy use of the clang-frontend as a library. The same applies to the static analyzer and its internals like the ExplodedGraph."* [16]. According to its documentation [17], *"The Clang Static Analyzer (CSA) is a source code analysis tool that finds bugs in C, C++, and Objective-C programs. It implements path-sensitive, inter-procedural analysis based on symbolic execution technique"*. This means that CSA has a highly modular architecture, whose core part consists of tracking values, paths, environments, etc. It also incorporates a set of plugins called *checkers*, each of which is tuned to detect a specific bug (e.g., use-after-free or divide-by-zero). CSA implements the popular approach of static analysis known as *abstract interpretation*, which shares a similar methodology to another precise analysis called *symbolic execution* but allows for approximations and assumptions if needed.

However, due to the nature of abstract interpretation, CSA is known to suffer from many *false reports*, namely false positives (FPs), which CSA marked as bugs but they are not in practice, and false negatives (FNs), which should have been marked as bugs but CSA does not. In this thesis, we aim at identifying the major causes of why CSA produces false reports, so that we can better understand the major source of false reports in static analyzers. Identifying such causes could give us an insight about them, which allow researchers to further tackle them to improve the bug detection capability of static analyzers.

However, identifying the causes of false bug reports have a few challenges. First, we need reproducible data and finding with certainty FPs and FNs. This in turn means that we need a list of bugs and where we can find them, or in other words we need a ground-truth set of bugs. We also need to analyze them to find out what are the causes of the false reports. This would allow us to answer questions such as why does the analyzer misses a bug? or why the analyzer misclassify a bug? Furthermore, to correctly classify and say with certainty that a part of the analyzer is problematic we need sufficient data.

Having only a couple of FPs and FNs will not cut it and will not give us a necessary confidence interval that we actually have found something significantly wrong with the static analyzer because the causes we find could actually be a minor problem, that programmers may not need to concentrate their energy and time to improve the static analyzer, and can work on more important projects.

Several research has been done on static analyzers, some comparing the efficiency of such tools [18], others directly to improve CSA and to compare the before and after improvements against one test suite [19, 20], and other papers were written only how to use such tools [21]. And some other papers to benchmark static analyzers.

However, none of the previous works aimed at identifying the causes of FPs and FNs, as well as the results are likely outdated as the LLVM community has been working on CSA to improve the bug detection capability since then. Many papers do not attempt to find the causes of FPs and FNs, let alone benchmarking static analyzers automatically [18, 19].

In this thesis, we created a tool to automatically analyze three different test suites and benchmark them, and we investigated the most important causes that are responsible for false reports. Specifically, we used three benchmark suites: the Juliet Test Suite, an automatic and synthetic suite made of simple 64,099 test cases [22–24], the Cyber Grand Challenge, a challenge created by DARPA made of 131 complicated software that actually represents real world software (e.g., a chess program and a virtual machine) [25], and Magma, a ground-truth fuzzing benchmark suite with nine real-world software with 138 bugs [26–28]. The tool in turn gives us a list of FPs and FNs, which we can analyze and identify the roots causes of them.

We analyzed 801 FPs and 911 FNs to identify some of the most important causes. We found out that many of them are caused by limited memory modeling, lack of environment modeling and incomplete experimental checkers. We also found minor interesting causes such as the standard library modeling (e.g., `malloc`) and nondeterminism in Z3, the path constraint solver incorporated by CSA. We found the results between the real-world (RW) and synthetic (SY) benchmarks were different. For example, the problems with the experimental checkers were more noticeable with RW than SY benchmarks. There were no particular big differences between the causes for the FPs and FNs.

In this thesis, we present the key contributions as follows:

- We present an automated tool for false report analysis, tested with the Juliet Test Suite (JTS), the Grand Cyber Challenge (CGC), and Magma.
- We analyze 801 FPs and 911 FNs in three benchmarks (JTS, CGC and Magma) to identify major causes of false reports from static analyzer.
- We find that for RW suites many FPs and FNs were detected by the experimental checkers. For the SY suite, this was less noticeable, indeed only 20% of them are due to the experimental checkers. The RW and SY suites gave different types of statistics. Some causes were more noticeable for the SY than the RW (e.g., Memory Modeling for static and external variables). Unsurprisingly, the bigger the test suites, the more FPs we found.
- We open-source the automated tool to facilitate the investigation in this research and help researchers and static analyzer developers to regularly inspect their analyzers as they change over the course of development.

Chapter 2

Background

In this chapter, we introduce the necessary background to understand our work.

2.1 Clang Static Analyzer

The Clang Static Analyzer (CSA) is a static analyzer based on abstract interpretation. CSA keeps track of program states in terms of abstract values called *symbols*, which are initially undecided as they are introduced through an input. CSA then splits the execution into multiple program states with corresponding environments and stores, and then computes the transitions from one state to another to track the different paths that can be taken during execution. To keep each program state, expression and value of statements, CSA uses an immutable class called `ProgramState`. More specifically, a state contains constraints and symbolic equations of expressions and memory regions.

A `ProgramState` and each point of the possible paths of execution of a program are united into what is called an *exploded graph*. This means, at a certain point in the execution, that each node of the graph corresponds to a point of the program and contains the `ProgramState`. For debugging purposes, it can also be displayed cleanly.

CSA starts at the entry point of a program and simulates each line of the program (e.g., calls, pre-statement, and post-statement) using the visitor pattern by using the `visit` function of the *ExprEngine* class. If the analysis of the current point changes a `ProgramState`, then a new node is created.

And a bug is found when a pattern is matched, defined by *checkers*.

2.1.1 Memory model

Modeling memory is the trickiest part of analyzing C programs. It requires keeping track of possibles values that variables and memory regions can carry. In order to model all possible ways that the memory can take on, CSA uses a highly modularized set of classes as well as immutable maps to store values.

Several resources can be helpful to understand how the memory model of CSA works. One such document is an explanation on how CSA works by one of the author of the analyzer. In the document, the author explains how to create a checker, what kind of checker we can create (e.g., AST-based checkers, AST matchers), what analysis method CSA implements, program states, symbolic values, and the memory model of CSA [29]. Two presentations are also useful to understand CSA and its inner working [30, 31]. In the first presentation, the author explains what static analysis is, how it works and finally how some of the ideas have been implemented. In the second one, the author explains the same thing as the previous paper, but also adds concrete explanations on how to debug CSA (e.g., using the CFG or debug checkers).

2.1.2 Constraint Solver

A SAT solver is a software which aims at solving the Boolean satisfiability problem [32]. It asks the question: can we replace the variables in the formula with either true or false such that the formula evaluates to true? The obvious disadvantage of SAT solvers is that they only work with boolean logic, while a lot of systems are often at a higher level of abstraction (e.g., mathematical equations) than boolean logic. To resolve that problem, we can use Satisfiability Modulo Theories (SMT) [32].

Z3 is a SMT solver [33]. Z3 can be used to solve equations and constraints. CSA has their own simplified constraint manager version, but it is incomplete and would mark more trivial false positives than Z3. Indeed, the CSA constraint manager does not handle multiplication, division and neither modulo operations, while Z3 can compensate and handle such cases.

While the default constraint solver from CSA has a speed advantage over Z3, we chose Z3 over the default solver as this thesis aims at identifying the causes of false reports and using the default solver may cause most false reports attributed to the solver itself.

2.1.3 Cross Translation Unit Analysis

CodeChecker is a tool to run static analysis built on top of the CSA toolchain. CodeChecker replaces a tool called `scan-build` provided by CSA. The tool runs the static analyzer for a full library. The main disadvantage of `scan-build` is that it does not uses Cross Translation Unit (CTU) analysis, while we can ask CodeChecker to do it for ourselves. Static analysis usually works in the boundary of only one translation unit (TU). However, we can make it work for more than one TU, also known as CTU. CodeChecker fully supports automated CTU with Clang, using `CodeChecker analyze --ctu` [34]. Using CTU allows us to remove well-known trivial sources of false reports.

CodeChecker needs a compilation database before being able to run the static analyzer. To create the compilation database, we can either use the build in `log` option of CodeChecker or an external library such as compiledb, which is a tool to generate Clang JSON Compilation Database files [35].

The CodeChecker pipeline works as follows. First, we create the compilation database, then we forward this database to CodeChecker to do the analysis and then we can create the necessary reports. We can create reports in different formats such as JSON and HTML.

CodeChecker provides a semi-automated way to distinguish whether a generated report is a true positive or not when a special tag is provided one line above the expected buggy line. If the report is made on the line, then the report will confirm it with the tag.

Using CodeChecker, we can actually decide weather to enable or disable a checker. Also, CodeChecker can enable different analyzer options such as the Z3 backend solver and loop unrolling or loop widening. Both later options help CSA perform better analysis with loops as it increases their coverage [36, 37]. The standard loop method only covers 4 iterations, but with those special optimizations the coverage is increased, and we can also remove trivial false reports [19].

2.2 Benchmarks

2.2.1 Juliet Test Suite

The Juliet Test Suite is a collection of test cases for the C/C++ language. It contains 64,099 cases and 119 different CWEs [22–24]. The test cases were created by the National Security Agency's (NSA) Center for Assured Software specifically designed to test static analyzers.

In the Juliet Test Suite, each test cases have a *GOOD* and a *BAD* function, where only the *BAD* function contains one flaw per test case. Each test case contains the same library folder, called *lib*, which consists of common code through all test cases. Each case contains a src folder which contains the necessary flaws. Note some test case may contain more than one flaw, but all of the same CWE-type. The files are then named using their CWEs and flaws such as `CWE476_NULL_Pointer_Dereference__char_54a.c`. Windows specific test cases contain `w32` in their file name.

While the main advantage of the Juliet Test Suite is the ease of calculating *true negatives*, which is implausible with real code, there are several limitations with the Juliet Test Suite. First, the test cases have been either generated automatically or synthetically, thus do not represent real use of programs. For example, some test cases are too simple consisting of only a few lines. Furthermore, the origin of each flaws is unique in the test case. This eventually led us to incorporate more than one test suite.

2.2.2 Cyber Grand Challenge (CGC)

The Cyber Grand Challenge is a test suite developed by DARPA in 2016, consisting of 131 programs. The test suite was intended for a challenge to develop automatic defense systems that can discover, prove and correct software flaws in real time [25, 38].

The binaries of the challenge ran on a 32-bit Intel X86 machine, with a simplified ABI. This in result simplified the external interaction to its base components such as dynamic memory allocation and system calls. However, the programs themselves can be extremely complex such as a chess program, or scripting languages, virtual machines, imitating real-world code.

The code of the challenge used in this report can be found at [25].

Each test case contains multiple vulnerabilities with corresponding CWEs and a plain English description of the flaws. Each of the cases contains the same standard `lib` folder.

2.2.3 Magma

Magma is a ground-truth fuzzing benchmark suite based on 9 different real programs, containing 138 real bugs in total [26–28]. Magma provides all source code of the targets repository, as well as all patches and build files.

Each patch has the following specification. Each one of them are in the patches and bugs repository. Each patch contains compiler macros to enable the fixes or disable it using `MAGMA_ENABLE_FIXES`, and it also contains canaries which allows Magma to know if a fuzzer actually reached the bug, using `MAGMA_ENABLE_CANARIES`.

2.3 False-positive and False-negative Calculation

The false positive rate (FPR) is

$$FPR = \frac{FP}{\text{Total number of ground-truth negatives}} = \frac{FP}{FP + TN}$$

where TN are the true negatives. And the false negative rate (FNR) is

$$FNR = \frac{FN}{\text{Total number of ground-truth positives}} = \frac{FN}{FN + TP}$$

where TP are the true positives. The true positive rate (TPR), also called sensitivity,

$$TPR = \frac{TN}{\text{Total number of ground-truth positives}} = \frac{TP}{TP + FN}$$

and the true negative rate (TNR), also called specificity, is

$$TNR = \frac{TN}{\text{Total number of ground-truth negatives}} = \frac{TN}{TN + FP}$$

This means that we have

$$TPR + FNR = 1$$

and

$$TNR + FPR = 1$$

and we denote % of FPs to be the percentage of FPs:

$$\text{\% of FPs} = \frac{\text{Number of FPs}}{\text{Total Number of Reports}} = \frac{FP}{TP + TN + FP + FN}$$

when we do not have the TN number, we set it to zero in the previous equation.

Chapter 3

Design

Our goal is to develop an automatic tool[1] to analyze three different test suites: the Juliet Test Suite (JTS), the Cyber Grand Challenge (CGC) and Magma, and to gather the FP and FN rates of the static analysis performed with CSA.

In this section, we explain how to automatically identify the reports in each error category: false positives, false negatives, true positives and true negatives. False positives are the bugs that should not have been marked by the static analyzer, while false negatives are the bugs that should have been reported by the analyzer, as they are known to exist in the test suite (i.e., in the ground-truth). True positives are the bugs from the ground-truth set that have been found. True negatives are a little more complicated. The JTS is the only suite test where we can gather true negatives because we have a definitive list of known bugs, but for CGC and Magma we do not have a such a list of TNs. This means we can not find the FP and TN rate for both CGC and Magma, then in this case we use the percentage of FPs. .

3.1 Overview

The pipeline (c.f., Figure 3.1) to run a library is as follows: First during the pre-processing phase, we gather the bugs and necessary checkers associated with a test suite. After this, we add the CodeChecker flag to the necessary files during the processing phase. And then we create the compilation databases for each test case for the given suite (point 1), then we run the CSA analysis (point 2), this allows us to create all the necessary files for the analysis and we dump them in a report folder. Finally, with those previous files, we can generate the JSON reports for the statistics and if needed we can also generate the HTML reports (point 3).

[1] The tool can be found at https://github.com/cjordan7/CSA-Testing-Tool/tree/8d8306c2a128ef26e6b0e52b4436959d8b65702d

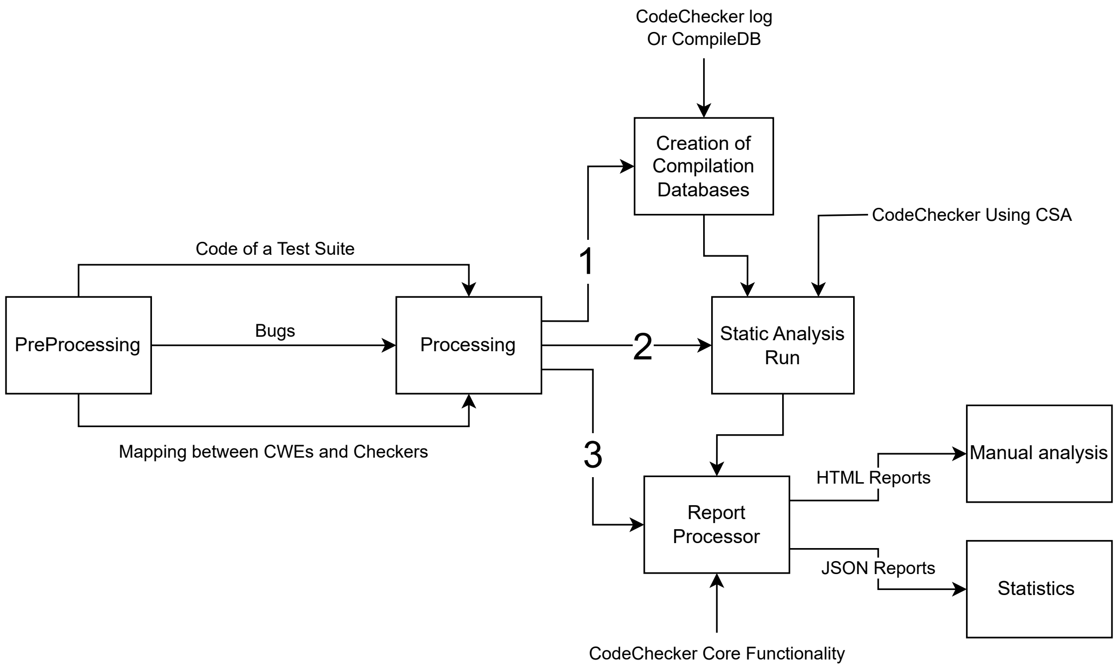


Figure 3.1 – Architecture of the tool

3.2 Static Analyzer

We use CodeChecker to be able to run CTU. We use CTU so that we can simulate real program, otherwise the static analysis could give us more false positives and false negatives than necessary. By using CTU, we can remove some of them and only keep the ones that are important.

We also use the CodeChecker flags to be able to easily find if the lines we want are actually buggy. This in turns simplify the calculations of false positives and false negatives.

The static analyzer run phase runs the static analyzer to analyze each test case of the modified test suite which has been provided by the processing phase. During its run the SA dumps CTU specific files and result of the analysis in the form of plist files in the folder of the same as the test case in the report folder of the test suite.

3.3 Report Processor

In this section, we explain in more detail how the analysis is run for each test suites.

3.3.1 Juliet Test Suite

In the case of the Juliet Test Suite, we can calculate all rates because we can find the number of false positives, false negatives, true positives and true negatives. We can then use what we explained in Section section 2.3 to calculate the rates.

During the run of the Juliet Test Suite, we actually run the analysis twice, once with only the bad, buggy functions and on the next run we run only the good, non buggy functions. By running only the buggy functions, we can find how many true positives we can have. Indeed, if CSA finds the correct bug in the buggy functions, then we have a false positive. Otherwise, we have a false negative. By using only the good functions, we can find the true negatives and subsequently false positives.

To find a bug, set the correct flag to the correct line, to do so, we read the `sarifs.json` that contains the flawed lines for each of the elements of the test suite. From the `sarifs.json`, we can also gather which test case are Windows specific test case. Note that the test suite also as designed potential flaws which are also related to the flaw of the test case, but are not present in the JSON file. In this case, we also flag the line of the potential flaw. Note each test case may have incidental flaws, in this case we simply ignore them.

3.3.2 Cyber Grand Challenge

For the Cyber Grand Challenge (CGC), we cannot find true negatives because we do not have a definite list of known bugs in the suite, unlike the Juliet Test Suite. This means that we cannot find the false positive rate. In this case, we use the percentage of FPs (% of FPs) to evaluate the FP number.

CGC is also a specific case, which was actually really difficult to automate. Each of the challenge only have their bugs explained in plain English, but the advantage of each one of them is that there were macros which defined the necessary patches. Of course, if one wanted to enable the patches he would have to specifically define the `patch` variable. This allowed us to add the CodeChecker flag very close to the patch and then move them manually to the correct emplacements. To be able to rerun the analysis automatically, we created a patch with the necessary flags and reapplied it during the processing phase when the user run the analysis.

3.3.3 Magma

Same as CGC, we cannot find true negatives for Magma. So once again, we use the percentage of FPs (% of FPs). Each of the bugs in Magma were pre-classified by the original authors.[2] Each bug

[2] https://hexhive.epfl.ch/magma/docs/bugs.html

has its associated CVE, which allowed us to gather the CWEs by following the links through the cve.mitre website and then to the NIST website [39, 40].

Magma is a special case compared to JTS and CGC because we only run one bug at a time. This is done to avoid potential problems between different bugs. Indeed, it could be possible that two bugs actually cancel each other which would make the analysis wrong. To find the rates of Magma, we calculate the rates of each analysis for each bug. We then proceeded to use the geometric mean to find the general rates for the library. We can then simply repeat the process for each Magma libraries.

3.4 CWEs

For many bugs, we only have their CWEs. However, most CSA checkers are not dedicated to a specific type of CWEs and cover many different types of CWEs at the same time. This motivated us to create a table for the found CWEs that are currently supported by checkers. It is also possible that we may need to take into account several checkers at the same time to find a bug. However during the analysis, we only took into account the correct checker and we ignored other reports.

Both tables Table 3.1 and Table 3.2 shows the full list of CWEs associated with a checker. Notice that we omitted the full path of the checker and only put the last name so that the table stays on the page. Also, the default core checkers are always enabled by default as they may be used by other checkers. The taint checker is also always enabled as some bugs actually depends on command inputs. It is only later in the generation of the reports that we ignore the unnecessary checkers.

Unfortunately, this means that some CWEs cannot be found by CSA but could only be found by other parts of the compiler. For example, the followings are two CWEs that concerns potential problems and bugs in a switch statements: *CWE-478: Missing Default Case in Multiple Condition Expression*, *CWE-484: Omitted Break Statement in Switch*. There are also CWEs in the different test suites which concerns cryptographic and hash functions which neither the compiler nor the static analyzer can handle those specific cases. (e.g., *CWE-916: Use of Password Hash With Insufficient Computational Effort* and *CWE-256: Plaintext Storage of a Password*)

CWEs	Checker
CWE-457, CWE-823, CWE-822,CWE-688, CWE-628, CWE-457	CallAndMessage
CWE-369	DivideZero
CWE-233	NonNullParamChecker
CWE-476, CWE-690	NullDereference
CWE-562	StackAddressEscape
CWE-129	ArraySubscript
CWE-908	Assign
CWE-665	Branch
CWE-908	UndefReturn
CWE-401, CWE-416, CWE-763, CWE-416, CWE-415	NewDelete
CWE-401, CWE-416	NewDeleteLeaks
CWE-1077	FloatLoopCounter
CWE-242, CWE-477, CWE-252	UncheckedReturn
CWE-242, CWE-477, CWE-676	bcmp
CWE-242, CWE-477, CWE-676	bcopy
CWE-242, CWE-477, CWE-676	bzero
CWE-242, CWE-477, CWE-676	getpw
CWE-242, CWE-676, CWE-242	gets
CWE-242, CWE-676	mkstemp
CWE-242, CWE-377, CWE-676	mktemp
CWE-242, CWE-338, CWE-676	rand
CWE-242, CWE-119, CWE-676	strcpy
CWE-242, CWE-477, CWE-676	vfork

Table 3.1 – Mapping between CWEs and Checkers. Part 1

CWEs	Checker
CWE-787	DeprecatedOrUnsafeBufferHandling
CWE-131, CWE-242	API
CWE-125, CWE-120, CWE-763, CWE-672, CWE-244, CWE-416, CWE-401, CWE-590, CWE-761, CWE-770, CWE-415	Malloc
CWE-119, CWE-190	BadSizeArg
CWE-667	C11Lock
CWE-457, CWE-825, CWE-824, CWE-825, CWE-457	CallAndMessageUnInitRefArg
CWE-588	CastToStruct
CWE-843, CWE-681, CWE-194, CWE-196, CWE-195	Conversion
CWE-704, CWE-843	DynamicTypeChecker
CWE-587	FixedAddr
CWE-823	PointerArithm
CWE-823, CWE-469	PointerSub
CWE-467	SizeofPtr
CWE-561	deadcode
CWE-122, CWE-121	DeprecatedOrUnsafeBufferHandling
CWE-129, CWE-122, CWE-119, CWE-193, CWE-170, CWE-805, CWE-824, CWE-125, CWE-120, CWE-123, CWE-788, CWE-787, CWE-190, CWE-126, CWE-124, CWE-680, CWE-127	ArrayBoundV2
CWE-190, CWE-122	MallocOverflow
CWE-466	ReturnPtrRange
CWE-242, CWE-477, CWE-252, CWE-201, CWE-20, CWE-78, CWE-606, CWE-134, CWE-377, CWE-114, CWE-789	taint
CWE-201, CWE-20, CWE-78, CWE-606, CWE-134, CWE-377, CWE-114, CWE-789	TaintPropagation
CWE-667	PthreadLock
CWE-122	OutOfBounds

Table 3.2 – Mapping between CWEs and Checkers. Part 2

Chapter 4

Implementation

Our tool allows has several different options so that we do not have to rerun all the analysis each time, but, in our case, we ran the full analysis.

Each test suite has been implemented in their corresponding files. The specific implementation for each test suites took on average about 500 lines and was implemented using Python. The pre-processing part and the report processor are common among all test suites.

The checkers and their corresponding CWEs have been defined in an external checker declaration document provided by CSA defined in our data folder which is later parsed to gather the corresponding mapping between CWEs and checkers. All checkers have been added to the document, but if there are no associated CWEs, we simply ignore the checker.

For the Magma patches, their corresponding CWEs have been added in another external checker declaration document where each line contains the CWE, the name of the patch and the CVE. The CVE is only kept for completeness.

CodeChecker can forward an external file with supplementary commands to CSA, that we will call a command file. We used one to run the exploded graph. Note: the only way to run the exploded graph is to have a version of the static analyzer which have enabled the assertions. By default the tool does not build it as it would be too slow.

Another pre-processing file is used to create the individual Makefiles for each of the CGC test suite as the original build builds all the test cases at once, while during our installation, we want to create the compilation database for each of the test case individually, otherwise, when we use CTU, CSA would create dependencies between each of the test case which is not what we want because each of the test case are independent from each other.

CGC also has its own pre-processing file, a patch, which is used to add the correct CodeChecker flags to the correct lines. The patch is applied during the run time of the tool.

Note at each time we start an analysis of CGC we checkout and remove all the modified files of the GitHub repository and then we apply the patch this allows us to avoid any complications.

For both the Juliet Test Suite and CGC, we ignore the library folder as we do not know for sure if there are any suplementary bugs in it. This can be done by forwarding another command to CSA using our command file. CGC however may have unknown bugs that have not been corrected by the creator of the challenge, thus have not been documented. Unfortunately the only way to know which of the unknown CGC bugs have been found will be to analyze our reports.

The pre-processing phase install all our test suites in a workdir folder. When the analysis is performed the reports are stored in a report folder.

Special variables are created when all the necessary dependencies are installed. Those special variables defines the different paths for the reports, workdir directories and necessary patches.

4.1 Juliet Test Suite

During the processing phase and to run a test case, we modify the Makefile of a test case by adding the possibility to define the variable to enable or disable the good and bad code.

We read the sarifs.json to know exactly where the bugs are and furthermore we check for potential flaws and other flaws using the comment in the test case.

During the analysis and later the report processor phase, for each test case of JTS we generate both GOOD and BAD reports independently. The result of the analysis is stored in the GOOD and BAD folder, while the HTML or JSON are generated in the *GOOD_report, BAD_report* and *GOOD.json* and *BAD.json* folders and files respectively. The generated files for both CGC and Magma are similar, but in their case we actually use either the name of the suite, for CGC, and the name of the bug for Magma.

4.2 CGC

During the processing phase and to run a test case, we had to automatically create the necessary Makefiles. Indeed, the original CGC installation is more abstract than this and unfortunately the standard way to create the compilation databases did not handle those cases. To do so, for each test case, we created a file to automatically capture the commands run by the CGC installation and forward them into a Makefile.

In the same phase, we also applied the special file for CodeChecker bugs we generated earlier during the pre-processing phase.

4.3 Magma

The pre-processing phase, processing phase and creation of compilation databases were not so different as for the other test suites. The only difference came by when we wanted to actually gather statistics and find out the classification (i.d. FP, FN or TP) of a bug report.

How do we actually find bugs with Magma? During the processing phase, we actually apply the different patches of the bug. Of course, the special Magma's macros are never enabled. As Magma defines canaries that are extremly close to the bug, we can use this fact to find if the static analyzer was capable to find a bug.

During the report processor phase, we read the CSA report to gather the graph of the report. We then traverse the graph of the report generated by CSA using a depth-first search algorithm, and we check in the corresponding file if the special macro `MAGMA_ENABLE_CANARIES` was ever reached. If it was, then we also check if the correct checker has been able to find the current bug.

Chapter 5

Result

In this section, we show the tool works by analysing hundreds of false positives and false negatives from the Juliet Test Suite, about 90 test cases of CGC and most false positives and false negatives from Magma after this we discuss the results of our analysis.

5.1 Hardware Setup

We ran the tool using CodeChecker version v6.21.0 and Clang version 15.0.7. The Juliet Test Suite's version is Juliet C/C++ 1.3[1]. CGC is compiled from commit number `797bd722ea54`[2]. And Magma is the version v1.2[3]. We also ran all test suites using Ubuntu Desktop: 22.04.2 LTS, which is the reason why we do not analyze the specific Windows test cases of the Juliet Test Suite. We ran the test suites using on Intel Core i7-8750H 2.2GHz and 16 GB DDR4 Memory RAM. Running the analysis for the Juliet Test Suite took about 29 hours. For CGC, it took about 10 hours and for Magma the full analysis took about 18 hours.

5.2 Ensuring Analyzable Set of Bugs

As discussed in section 3.4, CSA cannot analyze all types of CWEs. However, the benchmarks taken in this thesis specify bugs that are not analyzable or unsupported by CSA.

To bypass this issue, we identified the bugs analyzable with CSA and only considered them in this evaluation. Specifically in the Juliet Test Suite, we analyzed 39,852 out of the 64,099 test cases. In CGC, we analyzed 249 out of the 259 bugs. And finally in Magma, we analyzed 112 out

[1]Available at https://samate.nist.gov/SARD/test-suites/112
[2]Available at https://github.com/GrammaTech/cgc-cbs
[3]Available at https://github.com/HexHive/magma

of the 138 that can be found. Table 5.1 shows the number of CSA-analyzable bugs per library in Magma.

Library of Magma	Number of bugs
Libpng	6
Libtiff	13
Libxml2	14
Poppler	20
OpenSSL	14
Sqlite3	14
PHP	15
Lua	1
Libsndfile	15

Table 5.1 – Number of bugs CSA-analyzable per library of Magma

5.3 Sampling Produced Reports

	FPs	FNs
Juliet Test Suite	406	567
Cyber Grand Challenge	63	249
Magma	332	95

Table 5.2 – FPs and FNs analyzed for all test suites

As the number of reports were too many, we could not analyze all FPs and FNs. Instead, we decided to sample enough of them to be statistically significant. Specifically, the intervals of confidence are for the Juliet Test Suite: 95% confidence and about 5% error for the false negatives, and 95% confidence and about 6% error for the false positives. For CGC, we have a confidence for the false positives and the false negatives of 95% and about 5% error. Finally for Magma, we have a confidence for the false positives and the false negatives of 95% and about 5% error. Table 5.2 shows the specific number of analyzed false positives and false negatives for each test suite.

5.4 Causes of False Reports

In this section, we explain what each of the causes that we have found means.

- Loop Pattern (LP): The cause is due to a loop pattern that could not be taken into account by either the loop widening or loop unrolling techniques. To identify this cause, we looked at both the reports and the exploded graph and realized the loop was not executed as many times as it should have.

- Wrong checker location (WCL): The checker pointed the bug in the wrong location.
- Environement Modeling: (static variable) (EM: SV): The cause is due to a static variable that CSA said changes, but in fact never changed. To identify this cause, we looked at the reports and realized the static analyzer supposed the static variable could change. We also checked if the variable ever changed in the code, but it never did.
- Environement Modeling: (external variable) (EM: EV): The cause is due to an external variable that CSA said changes, but in fact was set. To identify this cause, we looked at the reports and realized the static analyzer supposed the external variable could change. We also checked if the variable ever changed in the code, but it never did.
- Environment Modeling: random variables (EM: RV): The cause is due to random variables. To identify this cause, we looked at the reports and at the exploded graph, and we realized the static analyzer made wrong assumptions about the range of the random variable.
- Lib function (LF): The call to a library function reset part of the environment variables. To identify this cause, we looked at the exploded graph and realized the static analyzer changed values after a call to a library function.
- Bad Modeling of Memory Function (BMMF): A library function was wrongly modelled, in this case we do not take into account its return value (c.f., Lib function bad return). We used the same method as above, but we did not see any changes in the values after a call to a library function.
- Lib function bad return (LF: BR): CSA did not take into account a library function could fail (e.g., malloc). To identify this cause, we looked at the reports and realized the static analyzer never made the assumption the value could fail.
- Z3: Undeterminism (Z3: Undet): The cause was due to some undecidable or non-linear mathematical equations that Z3 approximate. To identify this cause, we looked at the reports and saw that CSA made wrong supposition about the value generated after a mathematical expression.
- Environement Modeling: Complicated data structure (EM: CDS): The cause is due to the more complicated data structure and chaining of pointers.
- Experimental Checker (EC): The cause is due to the incompleteness or other causes of the checker (e.g., alpha checkers). To identify this cause, we first have to make sure that no other causes can be applied. If it is the case, then we make sure that the exploded graph and the environment states make sense. If it does not, then we identify it has EC. Otherwise, we may have found a new cause.

5.5 Results

After running the initial analysis, we get for all test suites Table 5.3, Figure 5.1 and Figure 5.2. In Table 5.3, "-" denotes values we could not produce. We also decided not to calculate the average when we only had one value (c.f., chapter 3). For a reminder on % of FPs, c.f., section 2.3.

Note for the Juliet Test Suite, the number of FPs actually can come from the GOOD and BAD code. The TNs were the GOOD code which has never been reported by the static analyzer, while the FNs were the BAD code which should have been reported by CSA.

Benchmark	**Number of Reports**				**Rate**				
	TP	**TN**	**FP**	**FN**	**TP**	**TN**	**FP**	**FN**	**% of FPs**
Juliet Test Suite	4866	40580	3933	36320	11.8%	91.2%	8.8%	88.2%	6.02%
CGC	0	-	63	249	0%	-	-	100%	20.2%
Magma: Libpng	0	-	64	6	0%	-	-	100%	91.43%
Magma: Libtiff	2	-	49	11	15.38%	-	-	84.62%	79.03%
Magma: Libxml2	3	-	51	11	21.42%	-	-	78.58%	78.46%
Magma: Poppler	1	-	79	19	5%	-	-	95%	79.79%
Magma: OpenSSL	3	-	1	11	21.4%	-	-	78.6%	6.67%
Magma: Sqlite3	1	-	65	13	7.2%	-	-	92.8%	82.28%
Magma: PHP	0	-	55	15	0%	-	-	100%	78.57%
Magma: Lua	0	-	14	1	0%	-	-	100%	93.34%
Magma: Libsndfile	0	-	21	15	0%	-	-	100%	58.34%
Average	443.2	-	399.5	3333.7	7.47%	-	-	92.53%	61.28%

Table 5.3 – Results for all test suites. "-" denotes values we could not produce (c.f., chapter 3). For a reminder on the percentage of FPs (% of FPs), (c.f., section 2.3)

The results (c.f., Table 5.3) actually show that for real code, the static analyzer gives a very high percentage rate of FNs for all test suites. The percentage of FPs is more than 80% for all Magma test suites. The only exception is openSSL. For four Magma libraries, CSA did not find any TPs and for the other one the TP rate is less than 22%. The analyzer perform much better for simple test suites as we do not have a lot of FPs (e.g., for JTS, we have a percentage of FPs of 6%), while it still does outperform for many test cases and prefers to give false negatives than FPs, while it actually performs well in the case of non buggy codes. Indeed, for the most of them the static analyzer is correct with the 91% of TNs for the Juliet Test Suite.

For CGC we have on average one FP per test case. And the rate of TPs is 0%. It means CSA tends to consider the bugs as non buggy.

We have a similar story for the Magma libraries, but in this case the percentage of FPs are bigger than with JTS and CGC even as we have at least 50 of them for most libraries, while the bugs are almost never found. The fact that the rate of FPs is higher for the Magma libraries could be explained with the number of lines compared to the CGC code. Indeed, by using the `cloc` command, for both Magma and CGC, we use `cloc--include-lang=C,C++,"C/C++ Header" .` and for Juliet Test Suite, we use `cloc --match-f=CWE --include-lang=C,C++,"C/C++ Header"`

. . We find that we have about 2,795 lines per CGC's test case and 1,358 files total, while for Magma we have about 49,045 lines of code per library and 695 files total. And finally, for the Juliet Test Suite, we have about 76 lines of code per file for a total of 105,183 files for the full test suite, we did not count the library files.

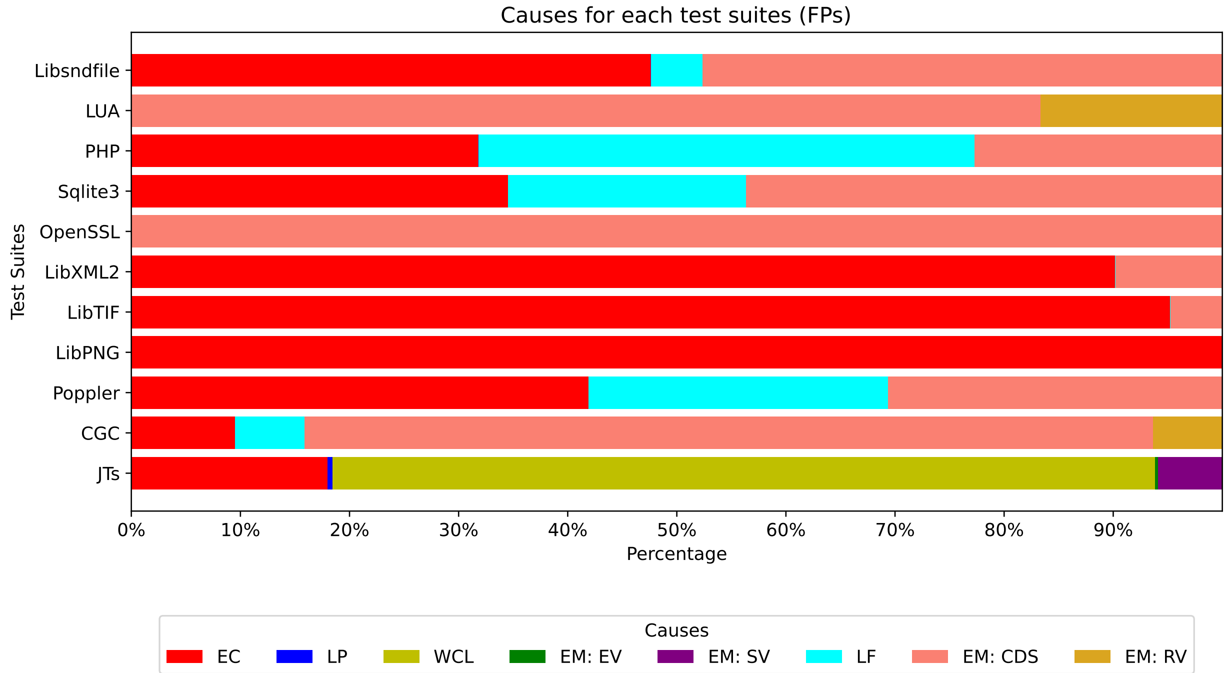


Figure 5.1 – FP causes for all test suites. For the abbreviations, c.f., section 5.4

We also found that it is very likely that many true negatives in the Juliet Test Suite may not actually be due to the CSA's "cleverness", but on occasion those actually comes from the same problem as with finding the actual bug in the BAD code. Note, 32,867 test cases have FNs and true negatives, but no true positives and FPs. For example, if we take the test case 111,866, the static analyzer totally ignores the potential failure of `malloc` and thus never considers it can return NULL, such that the if condition in the GOOD code is ignored by the static analyzer. This is of course a fair assumption as the failure of `malloc` can actually be extremely rare because failure would mean the system can not allocate such memory. With a powerful system, this is actually a fair assumption, but when talking about embedded systems this may be a more common recurrence as the memory availability can be more constrained than using a modern computer. For example ATtiny, a family of 8-bit micro-controller, has only a few KiBs of available memory RAM for their different version, or even STM32, a family of 32-bit micro-controllers, has several hundreds of KiBs of memory [41, 42].

We also have the same story with other functions that can return a value when it fails, which the static analyzer does not take into account.

Remark in our analysis, we did not investigate further if the true negatives were actually due to the understanding of the code or the limitations of the static analyzer as we wanted to

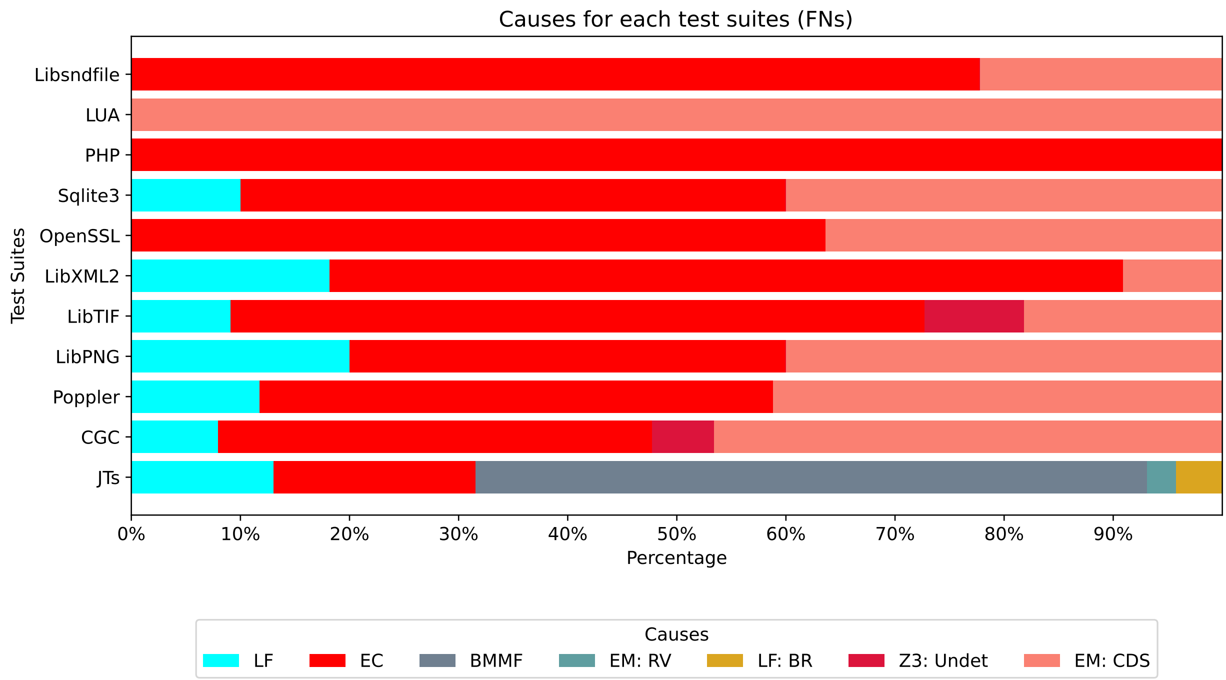


Figure 5.2 – FN causes for all test suites. For the abbreviations, c.f., section 5.4

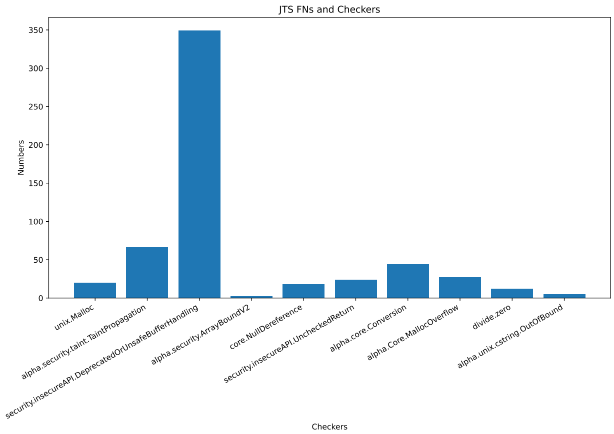
concentrate ourselves on the causes for the FPs and FNs, but we believe that if the static analyzer is further improved, then the number for the FNs, FPs and true negatives will very likely change instead of only seeing changes for the FPs and FNs.

Figure 5.1 and Figure 5.2 show that experimental checkers are more problematic for CGC and Magma, while it has less impact for JTS. This is because JTS has simpler code and does not cover more complicated cases that would actually trigger more FPs due to the experimental checkers, while CGC and Magma covers more complicated case which in fact shows the limitations of those checkers. We also found that some checkers also give FPs because they make use of tainted variables.

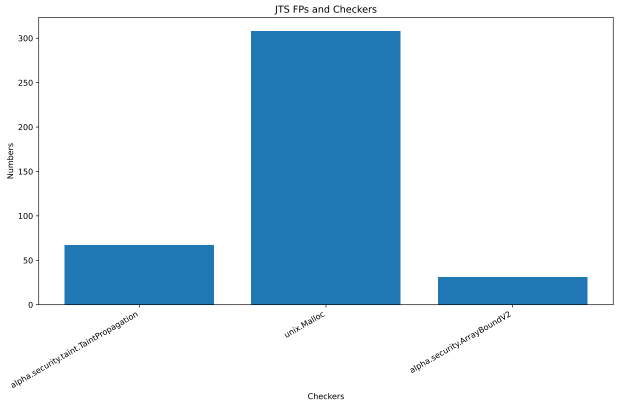
More complicated code also shows the limitations of the different checkers (e.g., alpha checkers and core checkers), which in turn also shows the limitations of some CSA techniques and optimizations such as the use Z3. Indeed, once again, JTS never used complicated logical expressions while some Magma libraries actually uses them.

The limitations with the memory modeling and environment modeling are more prevalent than with JTS, once again this is probably due to the fact that those libraries are more complicated and use more complicated data structure and pointer chaining more often than JTS. Indeed, we can actually see that fact as some FPs are due to pointer chaining, and CSA creates new symbolic values for those.

If the static analyzer has trouble with simpler code, logic and algorithm, then it will clearly

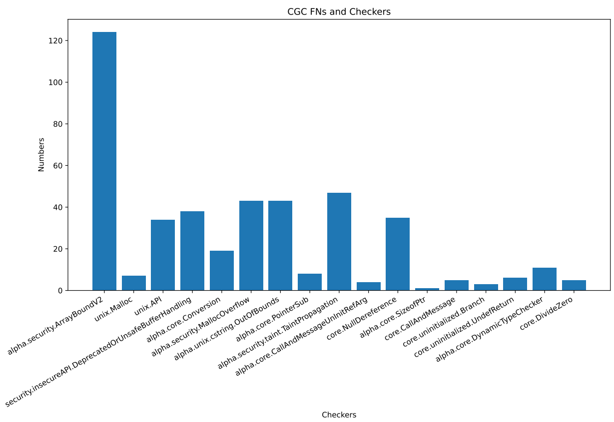


(a) Checkers for each FN of JTS

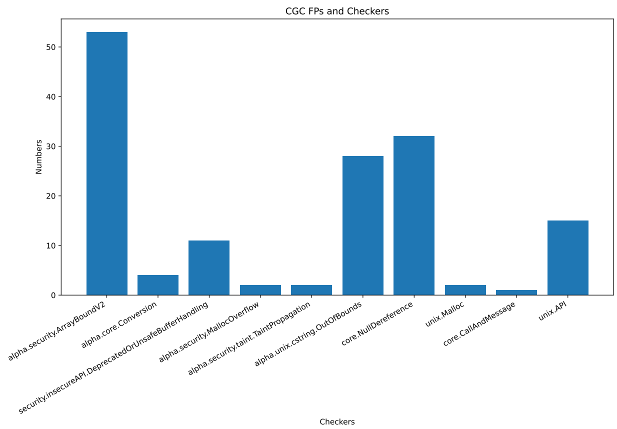


(b) Checkers for each FP of JTS

Figure 5.3 – Number of Checkers for JTS

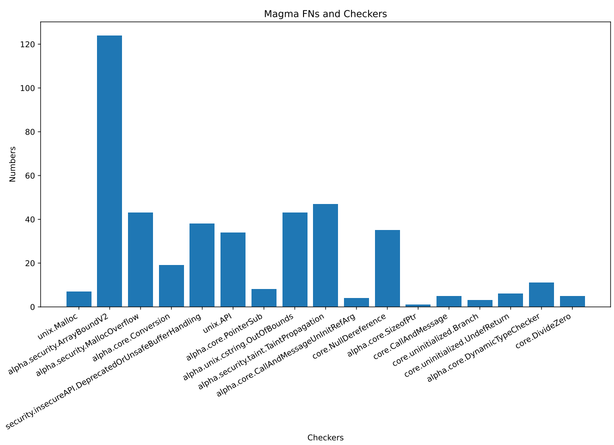


(a) Checkers for each FN of CGC

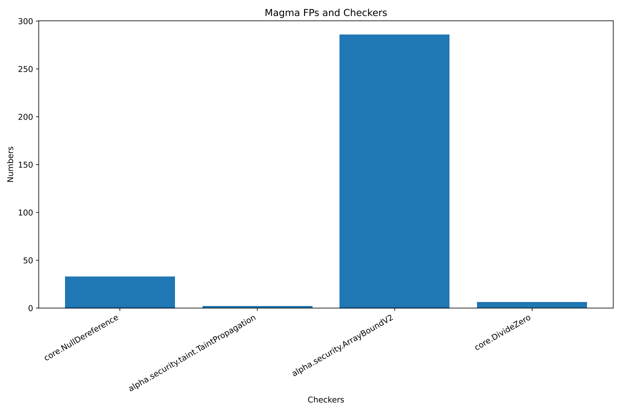


(b) Checkers for each FP of CGC

Figure 5.4 – Number of Checkers for CGC



(a) Checkers for each FN of Magma



(b) Checkers for each FP of Magma

Figure 5.5 – Number of Checkers for Magma

have more trouble with more complicated structures and real code which actually makes the analysis of simpler code still important.

Chapter 6

Related Work

In this chapter, we speak about a few papers and related works.

6.1 A Comparison of Static Analysis Tools for Vulnerability Detection in C/C++ Code

In this paper, the goal of the authors was to develop a customized static analysis tools for detecting potential vulnerabilites in C/C++ code [43].

They have benchmarked several C/C++ static analysis tools against the Toyota ITC test suite.

The results of the paper for CSA are

Checkers	TPs
Clang (core)	125/1915
Clang (alpha)	326/1915

They did not try to improve the CSA. They briefly talked about the false positives, but they do not give any FP numbers.

In this paper, the authors make no attemps in trying to find out what the problems and causes of the FPs and FNs they have. They also do not give any numbers for the FNs.

Our work gave numbers for the FPs and FNs, we also found causes for those FPs and FNs.

The paper also does not specify what they did to caclculate the false positives and false negatives, so we assumed they did it manually.

6.2 A Static Analysis Methods For Memory Leak Detection: A Survey

In this paper the authors discussed static analysis methods for memory leak detection and benchmarked different static analyzers, including CSA, using the Juliet test suite [44].

Here's a summary of their results for CSA:

Language	TP	TN	FP	FN	FPR	FNR
C	447	4509	98	421	2.12%	48.5%
C++	32	4516	0	838	0%	96.3%

Their results implies a high number of false negatives for CSA. In this paper, the authors only took into account memory leaks bugs, while we in our research we take into account every bug CSA could actually find.

They also did not try to improve CSA. They briefly talked about the false positives. No attempts in trying to analyze the causes of the false positives and false negatives.

In this paper, the authors make no attemps in trying to find out what the problems and causes of the FPs and FNs they have. They also don't give any numbers for the FPs, while our work gave numbers for the FPs and FNs, we also found causes for those FPs and FNs.

The paper also does not specify what they did to calculate the false positives and false negatives. And they did not specify which checkers they have enabled. Did they enable only unix.Malloc or some other checkers. Note, in the test suite, we have C++ code and unix.Malloc does not take into account C++ `new` and `delete` calls.

Furthermore, there is no mention of what the author actually did with the Windows only specific test cases.

There is also no mention on the number of CWEs and which one they enabled.

And did they use CTU analysis or did they only use the scan-build tool?

6.3 An Empirical Study on the Effectiveness of Static C Code Analyzers for Vulnerability Detection

The authors benchmarked different static analyzers (including CSA) on Magma. The results of the paper is between 0% and 20% true positives.

The paper also states:

"Our empirical evaluation shows that state-of-the-art static C code analyzers overlook a large number of real-world vulnerabilities" [18].

In this paper, they did not try to improve CSA. They talked about the false positives, but they do not give any FPs and FNs numbers. They did not take into consideration alpha checkers, which means that many Magma bugs can not be found. The authors also made no attempts in trying to analyze the causes of false positives and false negatives [18].

An other weakness of all the previous papers is that they do not specify how they run their analysis. It seems the author acquired their numbers manually, while our research can acquire them automatically.

6.4 Work on CSA

CSA has been constantly worked on and improved. Furthermore some more work are currently being made on CSA. For example several work have been done to improve the static analyzer such as the work of Csaba Dabis during his time at Google Summer of Code, called **Apply the Clang Static Analyzer to LLVM-based projects** in **2019** [45]. And the one of Dániel Domján with the work **C++17 structured bindings in the Clang Static Analyzer**, during his Google Summer of Code in **2022** in which he states: "Both the coverage and the accuracy of the analysis have been improved. On WebKit 81 false positives have been replaced by 22 true positives" [46]. The last work did not have any effects on our analysis and benchmark as none of the analyzed code actually uses C++ bindings.

6.5 Caveat

Another thing to keep in mind with most of the previous research is that the static analyzer is still being worked on and the results of the paper are very likely no longer up to date which means further work would have to be done. And the same analysis and benchmark would have to be remade manually in their case, with our research the benchmark can be rerun automatically for CSA.

Chapter 7

Discussion

In this thesis, we also aimed at covering another test suite: Syzbot. Syzbot is a system which continuously fuzzes the Linux kernel using, a fuzzer called Syzkaller, and saves the found bug and its report on the Syzbot dashboard located at [47]. Each time Syzkaller finds a bug, the system also runs *git bissect* which allows the system to find the first appearance of the bug.

However, during our work, we found out that fully working and finishing benchmarking Syzbot would be too time consuming for a few reasons. First of all to run the static analyzer and find the bugs, we would need to collect all of the bugs, which was time consuming, it lasted more than a week using a crawler. The second problem was that each bug has a list of commits which the fuzzer found where the bug also exist. So it would mean we are only sure we can find the bug on only a few commits.

Each bug possesses what Syzbot calls a crash report. This crash reports is actually useful to know where to find the bug which we could use in a similar way as shown in our previous test suites. Of course for each commits that are buggy, the report is actually different. Indeed, the line numbers are different.

Each buggy commit have their own configuration, called .config, which allows us to know which Linux options we can enable or not. This means that the number of bugs we can run at the same time is limited and thus we would need to run CSA on Linux as many times as we have bugs. It could actually be possible to run several bugs at the same time, but someone would need to have more knowledge about the Linux kernel and which of the 7000 configuration options can safely be disabled to still trigger the necessary bug, but as with Magma we may have the same problem that if we enable two different bugs, one can stop the triggering condition of the second one [47, 48].

Chapter 8

Conclusion

In this work, we created a tool to automatically run the static analyzer on three different test suites: the Juliet Test Suite, the Cyber Grand Challenge and Magma. We also gathered benchmark results. We discovered a high number of false negatives for JTS, CGC and Magma. The two latter test suites also have a high number of false positives.

During our work, we analyzed hundreds of reports to find some primary causes of FPs and FNs. We found out that many of them are determined by the memory model, the environment model and experimental checkers. We also found some minors causes such as the modeling of standard library functions such as malloc. Or even some unpredictability with the Z3 backend solver that we decided to use with CodeChecker.

At the end, we advanced the status quo with our tool because we can rerun the analysis without any problem on three different test suites. Furthermore, the tool can also be easily completed with more test suites if needs be.

Bibliography

[1] Les Hatton. "Programming Languages and Safety-Related Systems". In: *Proceedings of 3rd. Safety-Critical Systems Symposium.* SCSC. 1995.

[2] Steve McConnell. *Code Complete.* Cisco Press, 2004.

[3] Lauren von Beust. *Depuis dimanche, il est impossible d'acheter un billet en ligne pour aller voir un film dans les cinémas Pathé. Selon nos sources, le système informatique aurait été piraté.* [Online; accessed 20-June-2023]. 2023. URL: `https://www.lematin.ch/story/les-cinemas-pathe-victimes-dun-monstre-bug-informatique-363384201929`.

[4] SwissInfo. *Hackers steal Swiss police and customs data.* [Online; accessed 20-June-2023]. 2023. URL: `https://www.swissinfo.ch/eng/politics/hackers-steal-swiss-police-and-customs-data/48563830`.

[5] SwissInfo. *Swiss government and Federal Railways hit by cyberattacks.* [Online; accessed 20-June-2023]. 2023. URL: `https://www.swissinfo.ch/eng/politics/swiss-government-and-federal-railways-hit-by-cyberattacks/48583086`.

[6] Alex Hern. *WannaCry, Petya, NotPetya: how ransomware hit the big time in 2017.* [Online; accessed 20-June-2023]. 2017. URL: `https://www.theguardian.com/technology/2017/dec/30/wannacry-petya-notpetya-ransomware`.

[7] The Guardian. *Apple security flaw 'actively exploited' by hackers to fully control devices.* [Online; accessed 20-June-2023]. 2022. URL: `https://www.theguardian.com/technology/2022/aug/18/apple-security-flaw-hack-iphone-ipad-macs`.

[8] Michał Zalewski. *American Fuzz Lop.* [Online; accessed 20-June-2023]. 2018. URL: `http://lcamtuf.coredump.cx/afl/`.

[9] Andrea Fioraldi, Dominik Maier, Heiko Eißfeldt, and Marc Heuse. "AFL++ : Combining Incremental Steps of Fuzzing Research". In: *14th USENIX Workshop on Offensive Technologies (WOOT 20).* USENIX Association, Aug. 2020. URL: `https://www.usenix.org/conference/woot20/presentation/fioraldi`.

[10] Google Inc. *honggfuzz.* [Online; accessed 20-June-2023]. URL: `https://honggfuzz.dev`.

[11] Sushant Dinesh, Nathan Burow, Dongyan Xu, and Mathias Payer. "RetroWrite: Statically Instrumenting COTS Binaries for Fuzzing and Sanitization". In: *IEEE International Symposium on Security and Privacy.* 2020.

[12] Intel. *Dynamic Analysis vs. Static Analysis.* [Online; accessed 20-June-2023]. URL: https://www.intel.com/content/www/us/en/docs/inspector/user-guide-windows/2022/dynamic-analysis-vs-static-analysis.html#:~:text=Dynamic%5C%20analysis%5C%20is%5C%20the%5C%20testing,detected%5C%20both%5C%20dynamically%5C%20and%5C%20statically.

[13] LLVM Discourse. *Who is using Static Analyzer? Where are our users?* [Online; accessed 20-June-2023]. URL: https://discourse.llvm.org/t/who-is-using-static-analyzer-where-are-our-users/67985/2.

[14] Sonar. *the latest news and updates from SonarQube's product team.* [Online; accessed 20-June-2023]. URL: https://www.sonarsource.com/products/sonarqube/whats-new/.

[15] Synopsys. *Coverity.* [Online; accessed 20-June-2023]. URL: https://scan.coverity.com/faq.

[16] LLVM Discourse. *Who is using Static Analyzer? Where are our users?* [Online; accessed 20-June-2023]. URL: https://discourse.llvm.org/t/who-is-using-static-analyzer-where-are-our-users/67985/8.

[17] The Clang Team. *Clang Static Analyzer.* [Online; accessed 20-June-2023]. 2022. URL: https://clang.llvm.org/docs/ClangStaticAnalyzer.html.

[18] Stephan Lipp, Sebastian Banescu, and Alexander Pretschner. "An Empirical Study on the Effectiveness of Static C Code Analyzers for Vulnerability Detection". In: *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis.* ISSTA 2022. Virtual, South Korea: Association for Computing Machinery, 2022, pp. 544–555. ISBN: 9781450393799. DOI: 10.1145/3533767.3534380. URL: https://doi.org/10.1145/3533767.3534380.

[19] Péter György Szécsi, Gábor Horváth, and Zoltán Porkoláb. "Improved Loop Execution Modeling in the Clang Static Analyzer". In: *Acta Cybernetica* (2020).

[20] Marcelo Arroyo, Francisco Chiotta, and Francisco Bavera. "An user configurable clang static analyzer taint checker". In: *2016 35th International Conference of the Chilean Computer Science Society (SCCC).* IEEE. 2016, pp. 1–12.

[21] Kristóf Umann and Zoltán Porkoláb. "Detecting uninitialized variables in C++ with the Clang Static Analyzer". In: *Acta Cybernetica* 25.4 (2022), pp. 923–940.

[22] Center for Assured Software. National Security Agency. *Juliet C/C++ 1.3.* [Online; accessed 20-June-2023]. 2017. URL: https://samate.nist.gov/SARD/test-suites/112.

[23] Center for Assured Software. National Security Agency. *Juliet Test Suite v1.2 for C/C++ User Guide.* [Online; accessed 20-June-2023]. December 2012. URL: https://samate.nist.gov/SARD/downloads/documents/Juliet_Test_Suite_v1.2_for_C_Cpp_-_User_Guide.pdf.

[24] Paul E. Black. *Juliet 1.3 Test Suite: Changes From 1.2.* [Online; accessed 20-June-2023]. 2018. URL: https://www.govinfo.gov/content/pkg/GOVPUB-C13-1db3eda81a8a6d45e7fb42ad1cd8194c/pdf/GOVPUB-C13-1db3eda81a8a6d45e7fb42ad1cd8194c.pdf.

[25] GrammaTech. *CGC Challenge Binaries.* [Online; accessed 20-June-2023]. URL: https://github.com/GrammaTech/cgc-cbs.

[26] Ahmad Hazimeh, Adrian Herrera, and Mathias Payer. *HexHive/magma.* [Online; accessed 20-June-2023]. 2020. URL: https://github.com/HexHive/magma.

[27] Ahmad Hazimeh, Adrian Herrera, and Mathias Payer. "Magma: A Ground-Truth Fuzzing Benchmark". In: *Proc. ACM Meas. Anal. Comput. Syst.* 4.3 (Dec. 2020). DOI: 10.1145/3428334. URL: https://doi.org/10.1145/3428334.

[28] Ahmad Hazimeh, Adrian Herrera, and Mathias Payer. *Magma: A Ground-Truth Fuzzing Benchmark.* [Online; accessed 20-June-2023]. 2020. URL: https://hexhive.epfl.ch/magma/.

[29] haoNoQ. *clang-analyzer-guide.* [Online; accessed 20-June-2023]. 2016. URL: https://github.com/haoNoQ/clang-analyzer-guide/.

[30] A. Dergachev. *2019 LLVM Developers' Meeting: A. Dergachev "Developing the Clang Static Analyzer".* [Online; accessed 20-June-2023]. 2019. URL: https://www.youtube.com/watch?v=g0Mqx1niUi0.

[31] Ted Kremenek. "Finding software bugs with the clang static analyzer". In: *Apple Inc* (2008).

[32] Hans van Maaren Armin Biere Marijn Heule and Toby Walsch. *Handbook of Satisfiability.* IOS Press, 2008.

[33] Leonardo De Moura and Nikolaj Bjørner. "Z3: an efficient SMT solver". In: *Proceedings of the Theory and practice of software* (2008). DOI: 10.5555/1792734.1792766. URL: https://doi.org/10.5555/1792734.1792766.

[34] LLVM Foundation. *2.1. Cross Translation Unit (CTU) Analysis.* [Online; accessed 20-June-2023]. URL: https://clang.llvm.org/docs/analyzer/user-docs/CrossTranslationUnit.html.

[35] Nick Yamane. *Compilation Database Generator.* [Online; accessed 20-June-2023]. URL: https://github.com/nickdiego/compiledb.

[36] Ericsson. *CodeChecker.* [Online; accessed 20-June-2023]. URL: https://codechecker.readthedocs.io/en/latest/.

[37] Ericsson. *codechecker.* [Online; accessed 20-June-2023]. URL: https://github.com/Ericsson/codechecker.

[38] DARPA. *Cyber Grand Challenge (CGC) (Archived).* [Online; accessed 20-June-2023]. URL: https://www.darpa.mil/program/cyber-grand-challenge.

[39] The MITRE Corporation. *CVE Mitre.* [Online; accessed 20-June-2023]. URL: https://cve.mitre.org/index.html.

[40] NVD. *NVD - Home.* [Online; accessed 20-June-2023]. URL: https://nvd.nist.gov.

[41] Microship Technology. *AtTiny | Microship.* [Online; accessed 20-June-2023]. URL: https://www.microchip.com/en-us/product/.

[42] STMicroelectronics. *STM32 32-bit Arm Cortex MCUs.* [Online; accessed 20-June-2023]. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html.

[43] Stefan Ciobaca et al. Andrei Arusoaie. "A Comparison of Open-Source Static Analysis Tools for Vulnerability Detection in C/C++ Code". In: *19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)* (2017).

[44] et al. Hayk Aslanyan Zhora Gevorgyan. "Static Analysis Methods For Memory Leak Detection: A Survey". In: *2022 Ivannikov Memorial Workshop (IVMEM)* (2022).

[45] Csaba Dabis. *Apply the Clang Static Analyzer to LLVM-based projects.* [Online; accessed 20-June-2023]. URL: https://summerofcode.withgoogle.com/archive/2019/projects/4876350878384128.

[46] Dániel Domján. *Implement support for C++17 structured bindings in the Clang Static Analyzer.* [Online; accessed 20-June-2023]. URL: https://summerofcode.withgoogle.com/archive/2022/projects/dsXyCnNA.

[47] Google Inc. *syzkaller appspot.* [Online; accessed 20-June-2023]. URL: https://syzkaller.appspot.com/upstream/fixed.

[48] Google Inc. *syzbot.* [Online; accessed 20-June-2023]. URL: https://github.com/google/syzkaller/blob/master/docs/syzbot.md.

Appendix A

List of analyzed test cases: JTS

Index of Bug	Cause	Checkers
243845	Experimental Checker	alpha.security.taint.TaintPropagation
243813	Experimental Checker	alpha.security.taint.TaintPropagation
243607	Experimental Checker	alpha.security.taint.TaintPropagation
117125	Experimental Checker	alpha.security.taint.TaintPropagation
117126	Experimental Checker	alpha.security.taint.TaintPropagation
117365	Experimental Checker	alpha.security.taint.TaintPropagation
117366	Experimental Checker	alpha.security.taint.TaintPropagation
238756	Experimental Checker	alpha.security.taint.TaintPropagation
238764	Experimental Checker	alpha.security.taint.TaintPropagation
238774	Experimental Checker	alpha.security.taint.TaintPropagation
238784	Experimental Checker	alpha.security.taint.TaintPropagation
238792	Experimental Checker	alpha.security.taint.TaintPropagation
238757	Experimental Checker	alpha.security.taint.TaintPropagation
238765	Experimental Checker	alpha.security.taint.TaintPropagation
238776	Experimental Checker	alpha.security.taint.TaintPropagation
238785	Experimental Checker	alpha.security.taint.TaintPropagation
238793	Experimental Checker	alpha.security.taint.TaintPropagation
238758	Experimental Checker	alpha.security.taint.TaintPropagation
238766	Experimental Checker	alpha.security.taint.TaintPropagation
238777	Experimental Checker	alpha.security.taint.TaintPropagation
238786	Experimental Checker	alpha.security.taint.TaintPropagation
238801	Experimental Checker	alpha.security.taint.TaintPropagation
238759	Experimental Checker	alpha.security.taint.TaintPropagation
238768	Experimental Checker	alpha.security.taint.TaintPropagation
238779	Experimental Checker	alpha.security.taint.TaintPropagation
238787	Experimental Checker	alpha.security.taint.TaintPropagation

238803	Experimental Checker	alpha.security.taint.TaintPropagation
238760	Experimental Checker	alpha.security.taint.TaintPropagation
238769	Experimental Checker	alpha.security.taint.TaintPropagation
238780	Experimental Checker	alpha.security.taint.TaintPropagation
238788	Experimental Checker	alpha.security.taint.TaintPropagation
238806	Experimental Checker	alpha.security.taint.TaintPropagation
238761	Experimental Checker	alpha.security.taint.TaintPropagation
238770	Experimental Checker	alpha.security.taint.TaintPropagation
238781	Experimental Checker	alpha.security.taint.TaintPropagation
238789	Experimental Checker	alpha.security.taint.TaintPropagation
238810	Experimental Checker	alpha.security.taint.TaintPropagation
238762	Experimental Checker	alpha.security.taint.TaintPropagation
238772	Experimental Checker	alpha.security.taint.TaintPropagation
238782	Experimental Checker	alpha.security.taint.TaintPropagation
238790	Experimental Checker	alpha.security.taint.TaintPropagation
238763	Experimental Checker	alpha.security.taint.TaintPropagation
238773	Experimental Checker	alpha.security.taint.TaintPropagation
238783	Experimental Checker	alpha.security.taint.TaintPropagation
238791	Experimental Checker	alpha.security.taint.TaintPropagation
243843	Experimental Checker	alpha.security.taint.TaintPropagation
243822	Experimental Checker	alpha.security.taint.TaintPropagation
243820	Experimental Checker	alpha.security.taint.TaintPropagation
243635	Experimental Checker	alpha.security.taint.TaintPropagation
243830	Experimental Checker	alpha.security.taint.TaintPropagation
243625	Experimental Checker	alpha.security.taint.TaintPropagation
86497	Experimental Checker	alpha.security.taint.TaintPropagation
86511	Experimental Checker	alpha.security.taint.TaintPropagation
86512	Experimental Checker	alpha.security.taint.TaintPropagation
86524	Experimental Checker	alpha.security.taint.TaintPropagation
87687	Experimental Checker	alpha.security.taint.TaintPropagation
87691	Experimental Checker	alpha.security.taint.TaintPropagation
87735	Experimental Checker	alpha.security.taint.TaintPropagation
87739	Experimental Checker	alpha.security.taint.TaintPropagation
87783	Experimental Checker	alpha.security.taint.TaintPropagation
87787	Experimental Checker	alpha.security.taint.TaintPropagation
87879	Experimental Checker	alpha.security.taint.TaintPropagation
87883	Experimental Checker	alpha.security.taint.TaintPropagation
87927	Experimental Checker	alpha.security.taint.TaintPropagation
87931	Experimental Checker	alpha.security.taint.TaintPropagation
87975	Experimental Checker	alpha.security.taint.TaintPropagation

87979	Experimental Checker	alpha.security.taint.TaintPropagation
240336	Loop Pattern	unix.Malloc
108157	Wrong checker location	unix.Malloc
101368	Loop Pattern	unix.Malloc
77651	Experimental Checker	alpha.security.ArrayBoundV2
77699	Environment modeling : (external variable)	alpha.security.ArrayBoundV2
77203	Experimental Checker	alpha.security.ArrayBoundV2
77198	Experimental Checker	alpha.security.ArrayBoundV2
77155	Experimental Checker	alpha.security.ArrayBoundV2
77107	Experimental Checker	alpha.security.ArrayBoundV2
77059	Experimental Checker	alpha.security.ArrayBoundV2
110604	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110654	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110705	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110757	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110844	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110606	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110657	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110709	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110796	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110846	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110609	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110661	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110748	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110798	Environment modeling : (static variable)	alpha.security.ArrayBoundV2

110849	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110613	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110700	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110750	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110801	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
112597	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110652	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110702	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110753	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
110805	Environment modeling : (static variable)	alpha.security.ArrayBoundV2
102225	Wrong checker location	unix.Malloc
102269	Wrong checker location	unix.Malloc
102292	Wrong checker location	unix.Malloc
102335	Wrong checker location	unix.Malloc
102247	Wrong checker location	unix.Malloc
102270	Wrong checker location	unix.Malloc
102313	Wrong checker location	unix.Malloc
102336	Wrong checker location	unix.Malloc
102248	Wrong checker location	unix.Malloc
102291	Wrong checker location	unix.Malloc
102314	Wrong checker location	unix.Malloc
101262	Wrong checker location	unix.Malloc
101320	Wrong checker location	unix.Malloc
101369	Wrong checker location	unix.Malloc
101418	Wrong checker location	unix.Malloc
101472	Wrong checker location	unix.Malloc
101272	Wrong checker location	unix.Malloc
101321	Wrong checker location	unix.Malloc
101370	Wrong checker location	unix.Malloc
101419	Wrong checker location	unix.Malloc

101502	Wrong checker location	unix.Malloc
101273	Wrong checker location	unix.Malloc
101322	Wrong checker location	unix.Malloc
101371	Wrong checker location	unix.Malloc
101424	Wrong checker location	unix.Malloc
101512	Wrong checker location	unix.Malloc
101274	Wrong checker location	unix.Malloc
101323	Wrong checker location	unix.Malloc
101376	Wrong checker location	unix.Malloc
101464	Wrong checker location	unix.Malloc
101513	Wrong checker location	unix.Malloc
101275	Wrong checker location	unix.Malloc
101328	Wrong checker location	unix.Malloc
101406	Wrong checker location	unix.Malloc
101465	Wrong checker location	unix.Malloc
101514	Wrong checker location	unix.Malloc
101280	Wrong checker location	unix.Malloc
101358	Wrong checker location	unix.Malloc
101416	Wrong checker location	unix.Malloc
101466	Wrong checker location	unix.Malloc
101515	Wrong checker location	unix.Malloc
101310	Wrong checker location	unix.Malloc
101417	Wrong checker location	unix.Malloc
101467	Wrong checker location	unix.Malloc
101520	Wrong checker location	unix.Malloc
100339	Wrong checker location	unix.Malloc
100340	Wrong checker location	unix.Malloc
100331	Wrong checker location	unix.Malloc
100379	Wrong checker location	unix.Malloc
100424	Wrong checker location	unix.Malloc
100568	Wrong checker location	unix.Malloc
100712	Wrong checker location	unix.Malloc
100332	Wrong checker location	unix.Malloc
100382	Wrong checker location	unix.Malloc
100445	Wrong checker location	unix.Malloc
100589	Wrong checker location	unix.Malloc
100734	Wrong checker location	unix.Malloc
100334	Wrong checker location	unix.Malloc
100384	Wrong checker location	unix.Malloc
100455	Wrong checker location	unix.Malloc

100599	Wrong checker location	unix.Malloc
100744	Wrong checker location	unix.Malloc
100335	Wrong checker location	unix.Malloc
100385	Wrong checker location	unix.Malloc
100456	Wrong checker location	unix.Malloc
100600	Wrong checker location	unix.Malloc
100745	Wrong checker location	unix.Malloc
100337	Wrong checker location	unix.Malloc
100388	Wrong checker location	unix.Malloc
100457	Wrong checker location	unix.Malloc
100601	Wrong checker location	unix.Malloc
100746	Wrong checker location	unix.Malloc
100338	Wrong checker location	unix.Malloc
100389	Wrong checker location	unix.Malloc
100458	Wrong checker location	unix.Malloc
100602	Wrong checker location	unix.Malloc
100747	Wrong checker location	unix.Malloc
100341	Wrong checker location	unix.Malloc
100390	Wrong checker location	unix.Malloc
100461	Wrong checker location	unix.Malloc
100605	Wrong checker location	unix.Malloc
100750	Wrong checker location	unix.Malloc
100342	Wrong checker location	unix.Malloc
100391	Wrong checker location	unix.Malloc
100462	Wrong checker location	unix.Malloc
100607	Wrong checker location	unix.Malloc
100751	Wrong checker location	unix.Malloc
100343	Wrong checker location	unix.Malloc
100392	Wrong checker location	unix.Malloc
100463	Wrong checker location	unix.Malloc
100608	Wrong checker location	unix.Malloc
100752	Wrong checker location	unix.Malloc
100344	Wrong checker location	unix.Malloc
100393	Wrong checker location	unix.Malloc
100464	Wrong checker location	unix.Malloc
100609	Wrong checker location	unix.Malloc
100753	Wrong checker location	unix.Malloc
100345	Wrong checker location	unix.Malloc
100394	Wrong checker location	unix.Malloc
100465	Wrong checker location	unix.Malloc

100616	Wrong checker location	unix.Malloc
100754	Wrong checker location	unix.Malloc
100346	Wrong checker location	unix.Malloc
100395	Wrong checker location	unix.Malloc
100472	Wrong checker location	unix.Malloc
100637	Wrong checker location	unix.Malloc
100792	Wrong checker location	unix.Malloc
100347	Wrong checker location	unix.Malloc
100396	Wrong checker location	unix.Malloc
100493	Wrong checker location	unix.Malloc
100647	Wrong checker location	unix.Malloc
100793	Wrong checker location	unix.Malloc
100348	Wrong checker location	unix.Malloc
100397	Wrong checker location	unix.Malloc
100503	Wrong checker location	unix.Malloc
100648	Wrong checker location	unix.Malloc
100794	Wrong checker location	unix.Malloc
100350	Wrong checker location	unix.Malloc
100398	Wrong checker location	unix.Malloc
100504	Wrong checker location	unix.Malloc
100649	Wrong checker location	unix.Malloc
100795	Wrong checker location	unix.Malloc
100351	Wrong checker location	unix.Malloc
100399	Wrong checker location	unix.Malloc
100505	Wrong checker location	unix.Malloc
100650	Wrong checker location	unix.Malloc
100798	Wrong checker location	unix.Malloc
100352	Wrong checker location	unix.Malloc
100400	Wrong checker location	unix.Malloc
100506	Wrong checker location	unix.Malloc
100653	Wrong checker location	unix.Malloc
100799	Wrong checker location	unix.Malloc
100353	Wrong checker location	unix.Malloc
100401	Wrong checker location	unix.Malloc
100509	Wrong checker location	unix.Malloc
100654	Wrong checker location	unix.Malloc
100800	Wrong checker location	unix.Malloc
100355	Wrong checker location	unix.Malloc
100402	Wrong checker location	unix.Malloc
100510	Wrong checker location	unix.Malloc

100655	Wrong checker location	unix.Malloc
100802	Wrong checker location	unix.Malloc
100357	Wrong checker location	unix.Malloc
100404	Wrong checker location	unix.Malloc
100511	Wrong checker location	unix.Malloc
100656	Wrong checker location	unix.Malloc
100809	Wrong checker location	unix.Malloc
100360	Wrong checker location	unix.Malloc
100405	Wrong checker location	unix.Malloc
100512	Wrong checker location	unix.Malloc
100657	Wrong checker location	unix.Malloc
100830	Wrong checker location	unix.Malloc
100361	Wrong checker location	unix.Malloc
100407	Wrong checker location	unix.Malloc
100513	Wrong checker location	unix.Malloc
100664	Wrong checker location	unix.Malloc
100840	Wrong checker location	unix.Malloc
100362	Wrong checker location	unix.Malloc
100408	Wrong checker location	unix.Malloc
100541	Wrong checker location	unix.Malloc
100685	Wrong checker location	unix.Malloc
100841	Wrong checker location	unix.Malloc
100364	Wrong checker location	unix.Malloc
100409	Wrong checker location	unix.Malloc
100551	Wrong checker location	unix.Malloc
100695	Wrong checker location	unix.Malloc
100842	Wrong checker location	unix.Malloc
100365	Wrong checker location	unix.Malloc
100410	Wrong checker location	unix.Malloc
100553	Wrong checker location	unix.Malloc
100696	Wrong checker location	unix.Malloc
100843	Wrong checker location	unix.Malloc
100366	Wrong checker location	unix.Malloc
100411	Wrong checker location	unix.Malloc
100554	Wrong checker location	unix.Malloc
100697	Wrong checker location	unix.Malloc
100846	Wrong checker location	unix.Malloc
100367	Wrong checker location	unix.Malloc
100412	Wrong checker location	unix.Malloc
100557	Wrong checker location	unix.Malloc

100698	Wrong checker location	unix.Malloc
100847	Wrong checker location	unix.Malloc
100368	Wrong checker location	unix.Malloc
100413	Wrong checker location	unix.Malloc
100558	Wrong checker location	unix.Malloc
100701	Wrong checker location	unix.Malloc
100850	Wrong checker location	unix.Malloc
100369	Wrong checker location	unix.Malloc
100414	Wrong checker location	unix.Malloc
100559	Wrong checker location	unix.Malloc
100703	Wrong checker location	unix.Malloc
100857	Wrong checker location	unix.Malloc
100376	Wrong checker location	unix.Malloc
100415	Wrong checker location	unix.Malloc
100560	Wrong checker location	unix.Malloc
100704	Wrong checker location	unix.Malloc
100378	Wrong checker location	unix.Malloc
100416	Wrong checker location	unix.Malloc
100561	Wrong checker location	unix.Malloc
100705	Wrong checker location	unix.Malloc
240026	Wrong checker location	unix.Malloc
240028	Wrong checker location	unix.Malloc
240045	Wrong checker location	unix.Malloc
240055	Wrong checker location	unix.Malloc
240057	Wrong checker location	unix.Malloc
240126	Wrong checker location	unix.Malloc
240262	Wrong checker location	unix.Malloc
240360	Wrong checker location	unix.Malloc
240362	Wrong checker location	unix.Malloc
240381	Wrong checker location	unix.Malloc
240784	Wrong checker location	unix.Malloc
241333	Wrong checker location	unix.Malloc
241340	Wrong checker location	unix.Malloc
99964	Wrong checker location	unix.Malloc
108357	Wrong checker location	unix.Malloc
108517	Wrong checker location	unix.Malloc
108677	Wrong checker location	unix.Malloc
108197	Wrong checker location	unix.Malloc
108397	Wrong checker location	unix.Malloc
108557	Wrong checker location	unix.Malloc

108717	Wrong checker location	unix.Malloc
108277	Wrong checker location	unix.Malloc
108437	Wrong checker location	unix.Malloc
108597	Wrong checker location	unix.Malloc
108757	Wrong checker location	unix.Malloc
108317	Wrong checker location	unix.Malloc
108477	Wrong checker location	unix.Malloc
108637	Wrong checker location	unix.Malloc
108797	Wrong checker location	unix.Malloc
239400	Wrong checker location	unix.Malloc
239431	Wrong checker location	unix.Malloc
239448	Wrong checker location	unix.Malloc
239465	Wrong checker location	unix.Malloc
239486	Wrong checker location	unix.Malloc
239401	Wrong checker location	unix.Malloc
239432	Wrong checker location	unix.Malloc
239451	Wrong checker location	unix.Malloc
239466	Wrong checker location	unix.Malloc
239488	Wrong checker location	unix.Malloc
239402	Wrong checker location	unix.Malloc
239433	Wrong checker location	unix.Malloc
239452	Wrong checker location	unix.Malloc
239467	Wrong checker location	unix.Malloc
239489	Wrong checker location	unix.Malloc
239403	Wrong checker location	unix.Malloc
239434	Wrong checker location	unix.Malloc
239453	Wrong checker location	unix.Malloc
239468	Wrong checker location	unix.Malloc
239490	Wrong checker location	unix.Malloc
239409	Wrong checker location	unix.Malloc
239435	Wrong checker location	unix.Malloc
239454	Wrong checker location	unix.Malloc
239470	Wrong checker location	unix.Malloc
239491	Wrong checker location	unix.Malloc
239420	Wrong checker location	unix.Malloc
239436	Wrong checker location	unix.Malloc
239455	Wrong checker location	unix.Malloc
239472	Wrong checker location	unix.Malloc
239492	Wrong checker location	unix.Malloc
239421	Wrong checker location	unix.Malloc

239437	Wrong checker location	unix.Malloc
239456	Wrong checker location	unix.Malloc
239473	Wrong checker location	unix.Malloc
239493	Wrong checker location	unix.Malloc
239422	Wrong checker location	unix.Malloc
239438	Wrong checker location	unix.Malloc
239457	Wrong checker location	unix.Malloc
239474	Wrong checker location	unix.Malloc
239494	Wrong checker location	unix.Malloc
239423	Wrong checker location	unix.Malloc
239439	Wrong checker location	unix.Malloc
239458	Wrong checker location	unix.Malloc
239475	Wrong checker location	unix.Malloc
239495	Wrong checker location	unix.Malloc
239424	Wrong checker location	unix.Malloc
239440	Wrong checker location	unix.Malloc
239459	Wrong checker location	unix.Malloc
239476	Wrong checker location	unix.Malloc
239496	Wrong checker location	unix.Malloc
239425	Wrong checker location	unix.Malloc
239441	Wrong checker location	unix.Malloc
239460	Wrong checker location	unix.Malloc
239477	Wrong checker location	unix.Malloc
239497	Wrong checker location	unix.Malloc
239427	Wrong checker location	unix.Malloc
239442	Wrong checker location	unix.Malloc
239461	Wrong checker location	unix.Malloc
239479	Wrong checker location	unix.Malloc
239498	Wrong checker location	unix.Malloc
239428	Wrong checker location	unix.Malloc
239444	Wrong checker location	unix.Malloc
239462	Wrong checker location	unix.Malloc
239480	Wrong checker location	unix.Malloc
239499	Wrong checker location	unix.Malloc
239429	Wrong checker location	unix.Malloc
239445	Wrong checker location	unix.Malloc
239463	Wrong checker location	unix.Malloc
239483	Wrong checker location	unix.Malloc
239430	Wrong checker location	unix.Malloc
239447	Wrong checker location	unix.Malloc

239464	Wrong checker location	unix.Malloc
239484	Wrong checker location	unix.Malloc
99836	Wrong checker location	unix.Malloc
100094	Wrong checker location	unix.Malloc
100124	Wrong checker location	unix.Malloc

Table A.1 – FPs for JTS

Index of Bug	Cause	Checkers
104465	Lib function	unix.Malloc
103823	Lib function	unix.Malloc
103987	Lib function	unix.Malloc
104255	Lib function	unix.Malloc
106076	Lib function	unix.Malloc
106107	Lib function	unix.Malloc
106190	Lib function	unix.Malloc
112597	Experimental Checker	alpha.security.taint.TaintPropagation
80260	Experimental Checker	alpha.security.taint.TaintPropagation
80312	Experimental Checker	alpha.security.taint.TaintPropagation
80353	Experimental Checker	alpha.security.taint.TaintPropagation
80420	Experimental Checker	alpha.security.taint.TaintPropagation
80564	Experimental Checker	alpha.security.taint.TaintPropagation
80587	Experimental Checker	alpha.security.taint.TaintPropagation
80595	Experimental Checker	alpha.security.taint.TaintPropagation
80597	Experimental Checker	alpha.security.taint.TaintPropagation
80682	Experimental Checker	alpha.security.taint.TaintPropagation
80689	Experimental Checker	alpha.security.taint.TaintPropagation
80691	Experimental Checker	alpha.security.taint.TaintPropagation
80975	Experimental Checker	alpha.security.taint.TaintPropagation
81414	Experimental Checker	alpha.security.taint.TaintPropagation
81478	Experimental Checker	alpha.security.taint.TaintPropagation
81563	Experimental Checker	alpha.security.taint.TaintPropagation
81908	Experimental Checker	alpha.security.taint.TaintPropagation
81428	Experimental Checker	alpha.security.taint.TaintPropagation
109245	Experimental Checker	alpha.security.taint.TaintPropagation
232212	Experimental Checker	alpha.security.taint.TaintPropagation
108998	Experimental Checker	alpha.security.taint.TaintPropagation
109043	Experimental Checker	alpha.security.taint.TaintPropagation
109234	Experimental Checker	alpha.security.taint.TaintPropagation

232593	Experimental Checker	alpha.security.taint.TaintPropagation
232669	Experimental Checker	alpha.security.taint.TaintPropagation
232747	Experimental Checker	alpha.security.taint.TaintPropagation
248370	Experimental Checker	alpha.security.taint.TaintPropagation
248312	Experimental Checker	alpha.security.taint.TaintPropagation
248305	Experimental Checker	alpha.security.taint.TaintPropagation
248277	Experimental Checker	alpha.security.taint.TaintPropagation
247921	Experimental Checker	alpha.security.taint.TaintPropagation
247896	Experimental Checker	alpha.security.taint.TaintPropagation
247827	Experimental Checker	alpha.security.taint.TaintPropagation
247794	Experimental Checker	alpha.security.taint.TaintPropagation
247464	Experimental Checker	alpha.security.taint.TaintPropagation
247460	Experimental Checker	alpha.security.taint.TaintPropagation
247447	Experimental Checker	alpha.security.taint.TaintPropagation
246376	Experimental Checker	alpha.security.taint.TaintPropagation
245940	Experimental Checker	alpha.security.taint.TaintPropagation
245580	Experimental Checker	alpha.security.taint.TaintPropagation
245577	Experimental Checker	alpha.security.taint.TaintPropagation
245539	Experimental Checker	alpha.security.taint.TaintPropagation
245494	Experimental Checker	alpha.security.taint.TaintPropagation
239385	Experimental Checker	alpha.security.taint.TaintPropagation
239335	Experimental Checker	alpha.security.taint.TaintPropagation
246474	Experimental Checker	alpha.security.taint.TaintPropagation
248413	Experimental Checker	alpha.security.taint.TaintPropagation
244482	Experimental Checker	alpha.security.taint.TaintPropagation
244486	Experimental Checker	alpha.security.taint.TaintPropagation
244549	Experimental Checker	alpha.security.taint.TaintPropagation
244581	Experimental Checker	alpha.security.taint.TaintPropagation
244584	Experimental Checker	alpha.security.taint.TaintPropagation
244635	Experimental Checker	alpha.security.taint.TaintPropagation
244961	Experimental Checker	alpha.security.taint.TaintPropagation
245090	Experimental Checker	alpha.security.taint.TaintPropagation
241061	Experimental Checker	alpha.security.taint.TaintPropagation
243738	Experimental Checker	alpha.security.taint.TaintPropagation
243904	Experimental Checker	alpha.security.taint.TaintPropagation
243924	Experimental Checker	alpha.security.taint.TaintPropagation
243939	Experimental Checker	alpha.security.taint.TaintPropagation
243946	Experimental Checker	alpha.security.taint.TaintPropagation
243972	Experimental Checker	alpha.security.taint.TaintPropagation
244078	Experimental Checker	alpha.security.taint.TaintPropagation

244109	Experimental Checker	alpha.security.taint.TaintPropagation
244116	Experimental Checker	alpha.security.taint.TaintPropagation
244145	Experimental Checker	alpha.security.taint.TaintPropagation
65436	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65354	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65330	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65219	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65208	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65134	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65100	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64181	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64161	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64113	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64026	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64021	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64016	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63899	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63879	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63874	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64000	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64022	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64128	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64157	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64178	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64246	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64001	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64023	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64129	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64158	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64179	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64247	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64002	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64024	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64132	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64159	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64180	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64248	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64003	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64025	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64134	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling

64160	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64249	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64004	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64135	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64182	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64250	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64005	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64112	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64136	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64162	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64184	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64251	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64006	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64137	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64163	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64185	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64252	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64007	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64114	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64138	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64164	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64186	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64254	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64008	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64115	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64139	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64165	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64233	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64255	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64009	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64116	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64140	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64166	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64234	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64256	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64010	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64117	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64141	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64167	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64235	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling

64257	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64012	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64118	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64142	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64168	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64236	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64258	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64014	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64119	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64143	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64169	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64237	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64259	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64015	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64120	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64144	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64170	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64238	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64261	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64122	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64145	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64171	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64240	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64262	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64017	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64123	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64152	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64172	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64241	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64264	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64018	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64124	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64153	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64174	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64242	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64265	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64019	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64125	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64154	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64175	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling

64243	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64020	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64126	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64155	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64176	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64244	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64127	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64156	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64177	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64245	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
86305	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
86319	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
86320	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
86332	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
86333	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
86881	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
86895	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
86896	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
86909	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87073	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87087	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87088	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87100	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87101	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87457	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87471	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87472	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87484	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87485	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87495	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87499	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87543	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87547	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87591	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87595	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87649	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87663	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87676	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
87677	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88033	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling

88047	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88048	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88060	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88061	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88119	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88123	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88167	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88171	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88225	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88239	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88240	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88253	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88263	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88267	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88311	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88359	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88363	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88455	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88459	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88503	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88551	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
88555	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
62524	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
62904	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63005	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63011	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63088	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63094	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63158	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63161	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63212	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63314	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63361	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63440	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63443	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63517	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63553	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63638	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63640	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63682	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling

63734	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63836	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63934	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
63957	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64041	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64045	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64055	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64096	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64200	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64291	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64362	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64366	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64654	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
64724	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65126	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65198	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65314	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65446	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65453	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65561	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65566	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65567	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65676	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65768	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65907	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
65916	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66012	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66014	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66143	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66206	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66236	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66245	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66260	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66490	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66499	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66530	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
79822	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66532	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66573	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66593	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling

66756	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66764	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66767	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66867	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66869	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66918	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66950	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66954	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66963	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
66966	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
67064	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
67079	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
67118	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
67187	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
67216	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
67311	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
67421	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
67573	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
67738	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
67910	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
68068	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
68073	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
68242	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
68394	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
68438	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
68441	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
68626	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
68648	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
68914	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
68928	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
69065	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
69067	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
69076	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
69081	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
69731	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
69746	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
69585	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
69704	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
69892	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
69897	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling

70039	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
70043	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
70129	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
70266	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
73503	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
73522	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
73647	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
73863	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
73905	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
73964	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
74029	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
74105	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
74720	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
74725	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
74820	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
74837	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
74871	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
74972	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
74980	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75072	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75104	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75109	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75170	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75209	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75279	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75360	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75370	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75522	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75610	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75656	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75839	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75925	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75957	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75973	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75988	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
75994	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
76029	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
76040	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
76065	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
76080	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling

76184	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
76188	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
76522	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
76729	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
76782	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
77019	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
77160	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
77242	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
77446	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
77539	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
77541	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
78234	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
78419	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
78420	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
78432	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
78480	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
78575	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
78581	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
78656	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
78721	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
78722	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
78984	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
79101	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
79149	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
79171	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
79253	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
79557	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
79595	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
79745	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
79827	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
79843	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
79874	Bad Modeling of Memory Function	DeprecatedOrUnsafeBufferHandling
74199	Environement Modeling: random variables	alpha.security.ArrayBoundV2
62766	Environement Modeling: random variables	alpha.security.ArrayBoundV2
99969	Lib function	unix.Malloc
92295	Lib function	unix.Malloc
99916	Lib function	unix.Malloc
100054	Lib function	unix.Malloc
99747	Lib function	unix.Malloc
99836	Lib function	unix.Malloc

100007	Lib function	unix.Malloc
100012	Lib function	unix.Malloc
99869	Lib function	unix.Malloc
100141	Lib function	unix.Malloc
108994	Lib function	unix.Malloc
108980	Lib function	unix.Malloc
110754	Lib function	unix.Malloc
111866	Lib function	core.NullDereference
111195	Lib function	core.NullDereference
111088	Lib function	core.NullDereference
111143	Lib function	core.NullDereference
111198	Lib function	core.NullDereference
111386	Lib function	core.NullDereference
111419	Lib function	core.NullDereference
110990	Lib function	core.NullDereference
111422	Lib function	core.NullDereference
111851	Lib function	core.NullDereference
117527	Lib function	core.NullDereference
111890	Lib function	core.NullDereference
111867	Lib function	core.NullDereference
111664	Lib function	core.NullDereference
111428	Lib function	core.NullDereference
108948	Lib function	core.NullDereference
108947	Lib function	core.NullDereference
233247	Lib function	core.NullDereference
86645	Lib function bad return	security.insecureAPI.UncheckedReturn
86786	Lib function	alpha.core.Conversion
86532	Lib function	alpha.core.Conversion
81422	Lib function	alpha.core.Conversion
81281	Lib function	alpha.core.Conversion
81766	Lib function	alpha.core.Conversion
86733	Lib function	alpha.core.Conversion
86591	Lib function	alpha.core.Conversion
86821	Lib function	alpha.core.Conversion
88456	Lib function	alpha.core.Conversion
87356	Lib function	alpha.core.Conversion
87029	Lib function	alpha.core.Conversion
88502	Lib function	alpha.core.Conversion
232750	Lib function	alpha.core.Conversion
233148	Lib function	alpha.core.Conversion

233394	Lib function	alpha.core.Conversion
233494	Lib function	alpha.core.Conversion
233520	Lib function	alpha.core.Conversion
233556	Lib function	alpha.core.Conversion
233680	Lib function	alpha.core.Conversion
233689	Lib function	alpha.core.Conversion
233731	Lib function	alpha.core.Conversion
233736	Lib function	alpha.core.Conversion
233755	Lib function	alpha.core.Conversion
233883	Lib function	alpha.core.Conversion
233965	Lib function	alpha.core.Conversion
234282	Lib function	alpha.core.Conversion
234300	Lib function	alpha.core.Conversion
234647	Lib function	alpha.core.Conversion
234694	Lib function	alpha.core.Conversion
234759	Lib function	alpha.core.Conversion
235843	Lib function	alpha.core.Conversion
235977	Lib function	alpha.core.Conversion
236093	Lib function	alpha.core.Conversion
236414	Lib function	alpha.core.Conversion
236546	Lib function	alpha.core.Conversion
236617	Lib function	alpha.core.Conversion
110919	Experimental Checker	alpha.core.Conversion
110306	Experimental Checker	alpha.core.Conversion
239065	Experimental Checker	alpha.core.Conversion
238807	Experimental Checker	alpha.core.Conversion
110924	Experimental Checker	alpha.core.Conversion
87888	Experimental Checker	alpha.core.Conversion
87400	Environement Modeling: random variables	alpha.core.Conversion
238924	Environement Modeling: random variables	alpha.core.Conversion
84343	Experimental Checker	alpha.Core.MallocOverflow
83960	Experimental Checker	alpha.Core.MallocOverflow
83269	Experimental Checker	alpha.Core.MallocOverflow
83276	Experimental Checker	alpha.Core.MallocOverflow
236223	Experimental Checker	alpha.Core.MallocOverflow
235605	Experimental Checker	alpha.Core.MallocOverflow
236270	Experimental Checker	alpha.Core.MallocOverflow
236381	Experimental Checker	alpha.Core.MallocOverflow
236545	Experimental Checker	alpha.Core.MallocOverflow
235594	Experimental Checker	alpha.Core.MallocOverflow

236090	Experimental Checker	alpha.Core.MallocOverflow
236373	Experimental Checker	alpha.Core.MallocOverflow
235807	Experimental Checker	alpha.Core.MallocOverflow
236240	Experimental Checker	alpha.Core.MallocOverflow
234441	Experimental Checker	alpha.Core.MallocOverflow
236834	Experimental Checker	alpha.Core.MallocOverflow
233791	Experimental Checker	alpha.Core.MallocOverflow
233585	Experimental Checker	alpha.Core.MallocOverflow
234782	Experimental Checker	alpha.Core.MallocOverflow
232948	Experimental Checker	alpha.Core.MallocOverflow
233796	Experimental Checker	alpha.Core.MallocOverflow
233812	Experimental Checker	alpha.Core.MallocOverflow
235774	Environement Modeling: random variables	alpha.Core.MallocOverflow
236604	Environement Modeling: random variables	alpha.Core.MallocOverflow
235757	Environement Modeling: random variables	alpha.Core.MallocOverflow
236687	Environement Modeling: random variables	alpha.Core.MallocOverflow
236348	Environement Modeling: random variables	alpha.Core.MallocOverflow
95513	Environement Modeling: random variables	divide.zero
95369	Environement Modeling: random variables	divide.zero
95415	Environement Modeling: random variables	divide.zero
95352	Environement Modeling: random variables	divide.zero
232376	Environement Modeling: random variables	divide.zero
237018	Environement Modeling: random variables	divide.zero
95170	Experimental Checker	divide.zero
95180	Experimental Checker	divide.zero
95251	Experimental Checker	divide.zero
95268	Experimental Checker	divide.zero
95306	Experimental Checker	divide.zero
95071	Experimental Checker	divide.zero
232267	Experimental Checker	alpha.unix.cstring.OutOfBound
232359	Experimental Checker	alpha.unix.cstring.OutOfBound
236837	Experimental Checker	alpha.unix.cstring.OutOfBound
237177	Experimental Checker	alpha.unix.cstring.OutOfBound
237169	Experimental Checker	alpha.unix.cstring.OutOfBound
92485	Lib function bad return	security.insecureAPI.UncheckedReturn
92300	Lib function bad return	security.insecureAPI.UncheckedReturn
92666	Lib function bad return	security.insecureAPI.UncheckedReturn
92565	Lib function bad return	security.insecureAPI.UncheckedReturn
239290	Lib function bad return	security.insecureAPI.UncheckedReturn
92608	Lib function bad return	security.insecureAPI.UncheckedReturn

92102	Lib function bad return	security.insecureAPI.UncheckedReturn
92425	Lib function bad return	security.insecureAPI.UncheckedReturn
92252	Lib function bad return	security.insecureAPI.UncheckedReturn
92540	Lib function bad return	security.insecureAPI.UncheckedReturn
92557	Lib function bad return	security.insecureAPI.UncheckedReturn
87890	Lib function bad return	security.insecureAPI.UncheckedReturn
92088	Lib function bad return	security.insecureAPI.UncheckedReturn
92205	Lib function bad return	security.insecureAPI.UncheckedReturn
92256	Lib function bad return	security.insecureAPI.UncheckedReturn
92583	Lib function bad return	security.insecureAPI.UncheckedReturn
92322	Lib function bad return	security.insecureAPI.UncheckedReturn
92113	Lib function bad return	security.insecureAPI.UncheckedReturn
95226	Lib function bad return	security.insecureAPI.UncheckedReturn
92675	Lib function bad return	security.insecureAPI.UncheckedReturn
109853	Lib function bad return	security.insecureAPI.UncheckedReturn
238799	Lib function bad return	security.insecureAPI.UncheckedReturn
239057	Lib function bad return	security.insecureAPI.UncheckedReturn

Table A.2 – FNs for JTS

Appendix B

List of analyzed test cases: CGC

Index of Bug	Cause	Checkers
NRFIN_00016_1	Lib function	alpha.security.ArrayBoundV2, alpha.core.Conversion, DeprecatedOrUnsafeBufferHandling
NRFIN_00016_2	Lib function	alpha.security.ArrayBoundV2, alpha.core.Conversion, DeprecatedOrUnsafeBufferHandling
NRFIN_00022_1	Lib function	alpha.security.taint.TaintPropagation, alpha.security.ArrayBoundV2, alpha.core.Conversion, alpha.security.MallocOverflow
NRFIN_00022_2	Lib function	alpha.security.taint.TaintPropagation, alpha.security.ArrayBoundV2, alpha.core.Conversion, alpha.security.MallocOverflow
YANI_01_0007_1	Experimental Checker	alpha.security.ArrayBoundV2
YANI_01_0007_2	Experimental Checker	alpha.security.ArrayBoundV2
YANI_01_0007_3	Experimental Checker	alpha.security.ArrayBoundV2
CROMU_00031	Experimental Checker	alpha.security.ArrayBoundV2
NRFIN_00041_1	Experimental Checker	alpha.security.ArrayBoundV2
NRFIN_00041_2	Experimental Checker	alpha.security.ArrayBoundV2
CROMU_00015_1	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference
CROMU_00015_2	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference

CROMU_00015_3	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference
CROMU_00015_4	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference
CROMU_00015_5	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference
CROMU_00015_6	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference
CROMU_00015_7	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference
CROMU_00016_1	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference
CROMU_00016_2	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference
CROMU_00030	Environement Modeling: Complicated data structure	DeprecatedOrUnsafeBufferHandling
CROMU_00036_1	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
CROMU_00036_2	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
CROMU_00036_3	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
KPRCA_00049	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds
NRFIN_00015_1	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, DeprecatedOrUnsafeBufferHandling
NRFIN_00015_2	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, DeprecatedOrUnsafeBufferHandling
NRFIN_00014_1	Environement Modeling: Complicated data structure	core.NullDereference
NRFIN_00014_2	Environement Modeling: Complicated data structure	core.NullDereference

KPRCA_00054	Environement Modeling: Complicated data structure	unix.Malloc
KPRCA_00050	Environement Modeling: Complicated data structure	core.CallAndMessage
KPRCA_00048_cb2_1	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API
KPRCA_00048_cb2_2	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API
KPRCA_00048_cb2_3	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API
KPRCA_00048_cb2_4	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API
KPRCA_00048_cb2_5	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API
KPRCA_00048_cb2_6	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API
KPRCA_00048_cb2_7	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API
KPRCA_00048_cb2_8	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API
KPRCA_00048_cb2_9	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API
KPRCA_00048_cb2_10	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API
KPRCA_00048_cb2_11	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API
KPRCA_00048_cb2_12	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API

KPRCA_00048_cb2_13	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API
KPRCA_00048_cb2_14	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API
KPRCA_00048_cb2_15	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2, core.NullDereference, unix.API
KPRCA_00045	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds
KPRCA_00044_4	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
KPRCA_00043	Environement Modeling: Complicated data structure	DeprecatedOrUnsafeBufferHandling
KPRCA_00011	Environement Modeling: Complicated data structure	unix.Malloc, DeprecatedOrUnsafeBufferHandling
CROMU_00041	Environement Modeling: Complicated data structure	DeprecatedOrUnsafeBufferHandling
CROMU_00040_1	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2, core.NullDereference
CROMU_00040_2	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2, core.NullDereference
CROMU_00040_3	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2, core.NullDereference
CROMU_00040_4	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2, core.NullDereference
CROMU_00040_5	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2, core.NullDereference
CROMU_00040_6	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2, core.NullDereference
KPRCA_00044_1	Environement Modeling: random variables	alpha.security.ArrayBoundV2
KPRCA_00044_2	Environement Modeling: random variables	alpha.security.ArrayBoundV2
KPRCA_00044_3	Environement Modeling: random variables	alpha.security.ArrayBoundV2
KPRCA_00008	Environement Modeling: random variables	alpha.security.ArrayBoundV2
KPRCA_00027	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2, DeprecatedOrUnsafeBufferHandling

KPRCA_00029	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2, DeprecatedOrUnsafeBufferHandling
KPRCA_00032	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2, DeprecatedOrUnsafeBufferHandling

Table B.1 – FPs for CGC

Index of Bug	Cause	Checkers
CROMU_00025_1	Z3: Undeterminism	unix.Malloc, alpha.security.ArrayBoundV2
CROMU_00025_2	Z3: Undeterminism	unix.Malloc, alpha.security.ArrayBoundV2
CROMU_00042_1	Z3: Undeterminism	DeprecatedOrUnsafeBufferHandling, alpha.security.MallocOverflow, unix.API, alpha.core.Conversion
CROMU_00042_2	Z3: Undeterminism	DeprecatedOrUnsafeBufferHandling, alpha.security.MallocOverflow, unix.API, alpha.core.Conversion
CROMU_00042_3	Z3: Undeterminism	DeprecatedOrUnsafeBufferHandling, alpha.security.MallocOverflow, unix.API, alpha.core.Conversion
CROMU_00042_4	Z3: Undeterminism	DeprecatedOrUnsafeBufferHandling, alpha.security.MallocOverflow, unix.API, alpha.core.Conversion
CROMU_00042_5	Z3: Undeterminism	DeprecatedOrUnsafeBufferHandling, alpha.security.MallocOverflow, unix.API, alpha.core.Conversion
KPRCA_00007_1	Z3: Undeterminism	alpha.security.ArrayBoundV2
KPRCA_00046_1	Z3: Undeterminism	alpha.core.PointerSub, alpha.unix.cstring.OutOfBounds
KPRCA_00046_2	Z3: Undeterminism	alpha.core.PointerSub, alpha.unix.cstring.OutOfBounds
CROMU_00043_1	Environement Modeling: Complicated data structure	alpha.security.taint.TaintPropagation, alpha.security.ArrayBoundV2
CROMU_00043_2	Environement Modeling: Complicated data structure	alpha.security.taint.TaintPropagation, alpha.security.ArrayBoundV2
KPRCA_00013_1	Environement Modeling: Complicated data structure	alpha.core.CallAndMessageUnInitRefArg, alpha.security.ArrayBoundV2
KPRCA_00013_2	Environement Modeling: Complicated data structure	alpha.core.CallAndMessageUnInitRefArg, alpha.security.ArrayBoundV2

CROMU_00002_1	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
KPRCA_00010_1	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds
CROMU_00022_1	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
NRFIN_00014_1	Environement Modeling: Complicated data structure	core.NullDereference
NRFIN_00016_1	Environement Modeling: Complicated data structure	alpha.core.Conversion, DeprecatedOrUnsafeBufferHandling, alpha.security.ArrayBoundV2
NRFIN_00016_2	Environement Modeling: Complicated data structure	alpha.core.Conversion, DeprecatedOrUnsafeBufferHandling, alpha.security.ArrayBoundV2
NRFIN_00016_3	Environement Modeling: Complicated data structure	alpha.core.Conversion, DeprecatedOrUnsafeBufferHandling, alpha.security.ArrayBoundV2
CROMU_00010_1	Environement Modeling: Complicated data structure	alpha.core.SizeofPtr
CROMU_00017_1	Environement Modeling: Complicated data structure	DeprecatedOrUnsafeBufferHandling
CROMU_00019_1	Lib function	DeprecatedOrUnsafeBufferHandling
CROMU_00020_1	Environement Modeling: Complicated data structure	DeprecatedOrUnsafeBufferHandling
CROMU_00023_1	Experimental Checker	alpha.security.ArrayBoundV2
CROMU_00026_1	Environement Modeling: Complicated data structure	DeprecatedOrUnsafeBufferHandling, unix.API, alpha.security.ArrayBoundV2
CROMU_00026_2	Environement Modeling: Complicated data structure	DeprecatedOrUnsafeBufferHandling, unix.API, alpha.security.ArrayBoundV2
CROMU_00026_3	Environement Modeling: Complicated data structure	DeprecatedOrUnsafeBufferHandling, unix.API, alpha.security.ArrayBoundV2
CROMU_00029_1	Environement Modeling: Complicated data structure	alpha.security.MallocOverflow, unix.API, alpha.security.ArrayBoundV2
CROMU_00029_2	Environement Modeling: Complicated data structure	alpha.security.MallocOverflow, unix.API, alpha.security.ArrayBoundV2

CROMU_00029_3	Environement Modeling: Complicated data structure	alpha.security.MallocOverflow, unix.API, alpha.security.ArrayBoundV2
NRFIN_00017_1	Environement Modeling: Complicated data structure	core.NullDereference, alpha.security.taint.TaintPropagation, alpha.security.ArrayBoundV2
NRFIN_00017_2	Environement Modeling: Complicated data structure	core.NullDereference, alpha.security.taint.TaintPropagation, alpha.security.ArrayBoundV2
NRFIN_00017_3	Environement Modeling: Complicated data structure	core.NullDereference, alpha.security.taint.TaintPropagation, alpha.security.ArrayBoundV2
NRFIN_00017_4	Environement Modeling: Complicated data structure	core.NullDereference, alpha.security.taint.TaintPropagation, alpha.security.ArrayBoundV2
NRFIN_00004_1	Experimental Checker	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2
NRFIN_00004_2	Experimental Checker	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2
NRFIN_00004_3	Experimental Checker	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2
NRFIN_00001_1	Experimental Checker	alpha.security.taint.TaintPropagation
NRFIN_00005_1	Environement Modeling: Complicated data structure	core.NullDereference, alpha.security.ArrayBoundV2
NRFIN_00005_2	Environement Modeling: Complicated data structure	core.NullDereference, alpha.security.ArrayBoundV2
NRFIN_00009_1	Experimental Checker	alpha.security.taint.TaintPropagation
NRFIN_00009_2	Experimental Checker	alpha.security.taint.TaintPropagation
NRFIN_00009_3	Experimental Checker	alpha.security.taint.TaintPropagation
NRFIN_00009_4	Experimental Checker	alpha.security.taint.TaintPropagation
KPRCA_00011_1	Lib function	DeprecatedOrUnsafeBufferHandling, unix.Malloc
KPRCA_00011_2	Lib function	DeprecatedOrUnsafeBufferHandling, unix.Malloc
KPRCA_00034_1	Experimental Checker	alpha.security.taint.TaintPropagation
NRFIN_00008_1	Experimental Checker	alpha.security.ArrayBoundV2
NRFIN_00007_1	Experimental Checker	alpha.security.ArrayBoundV2
NRFIN_00007_2	Experimental Checker	alpha.security.ArrayBoundV2
NRFIN_00038_1	Environement Modeling: Complicated data structure	DeprecatedOrUnsafeBufferHandling, alpha.security.taint.TaintPropagation, core.CallAndMessage

NRFIN_00038_2	Environement Modeling: Complicated data structure	DeprecatedOrUnsafeBufferHandling, alpha.security.taint.TaintPropagation, core.CallAndMessage
NRFIN_00038_3	Environement Modeling: Complicated data structure	DeprecatedOrUnsafeBufferHandling, alpha.security.taint.TaintPropagation, core.CallAndMessage
NRFIN_00036_1	Environement Modeling: Complicated data structure	alpha.core.PointerSub, alpha.unix.cstring.OutOfBounds, unix.API, alpha.security.ArrayBoundV2
NRFIN_00036_2	Environement Modeling: Complicated data structure	alpha.core.PointerSub, alpha.unix.cstring.OutOfBounds, unix.API, alpha.security.ArrayBoundV2
NRFIN_00036_3	Environement Modeling: Complicated data structure	alpha.core.PointerSub, alpha.unix.cstring.OutOfBounds, unix.API, alpha.security.ArrayBoundV2
NRFIN_00036_4	Environement Modeling: Complicated data structure	alpha.core.PointerSub, alpha.unix.cstring.OutOfBounds, unix.API, alpha.security.ArrayBoundV2
NRFIN_00036_5	Environement Modeling: Complicated data structure	alpha.core.PointerSub, alpha.unix.cstring.OutOfBounds, unix.API, alpha.security.ArrayBoundV2
NRFIN_00036_6	Environement Modeling: Complicated data structure	alpha.core.PointerSub, alpha.unix.cstring.OutOfBounds, unix.API, alpha.security.ArrayBoundV2
KPRCA_00035_1	Environement Modeling: Complicated data structure	core.uninitialized.Branch, alpha.security.ArrayBoundV2
KPRCA_00035_2	Environement Modeling: Complicated data structure	core.uninitialized.Branch, alpha.security.ArrayBoundV2
KPRCA_00035_3	Environement Modeling: Complicated data structure	core.uninitialized.Branch, alpha.security.ArrayBoundV2
CROMU_00041_1	Lib function	DeprecatedOrUnsafeBufferHandling
CROMU_00041_2	Lib function	DeprecatedOrUnsafeBufferHandling
YAN01_00007_1	Environement Modeling: Complicated data structure	core.NullDereference, alpha.security.ArrayBoundV2

YAN01_00007_2	Environement Modeling: Complicated data structure	core.NullDereference, alpha.security.ArrayBoundV2
YAN01_00009_1	Experimental Checker	alpha.security.taint.TaintPropagation
YAN01_00012_1	Experimental Checker	alpha.unix.cstring.OutOfBounds
KPRCA_00043_1	Lib function	DeprecatedOrUnsafeBufferHandling
KPRCA_00026_1	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, core.NullDereference
KPRCA_00026_2	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, core.NullDereference
KPRCA_00021_1	Environement Modeling: Complicated data structure	core.uninitialized.UndefReturn, DeprecatedOrUnsafeBufferHandling, unix.Malloc
KPRCA_00021_2	Environement Modeling: Complicated data structure	core.uninitialized.UndefReturn, DeprecatedOrUnsafeBufferHandling, unix.Malloc
KPRCA_00021_3	Environement Modeling: Complicated data structure	core.uninitialized.UndefReturn, DeprecatedOrUnsafeBufferHandling, unix.Malloc
CROMU_00006_1	Experimental Checker	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2
CROMU_00006_2	Experimental Checker	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2
NRFIN_00022_1	Experimental Checker	alpha.core.Conversion, alpha.security.taint.TaintPropagation, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00022_2	Experimental Checker	alpha.core.Conversion, alpha.security.taint.TaintPropagation, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00022_3	Experimental Checker	alpha.core.Conversion, alpha.security.taint.TaintPropagation, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00022_4	Experimental Checker	alpha.core.Conversion, alpha.security.taint.TaintPropagation, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2

NRFIN_00022_5	Experimental Checker	alpha.core.Conversion, alpha.security.taint.TaintPropagation, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00022_6	Experimental Checker	alpha.core.Conversion, alpha.security.taint.TaintPropagation, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00041_1	Experimental Checker	alpha.security.taint.TaintPropagation, alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2
NRFIN_00041_2	Experimental Checker	alpha.security.taint.TaintPropagation, alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2
NRFIN_00041_3	Experimental Checker	alpha.security.taint.TaintPropagation, alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2
NRFIN_00041_4	Experimental Checker	alpha.security.taint.TaintPropagation, alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2
NRFIN_00041_5	Experimental Checker	alpha.security.taint.TaintPropagation, alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2
KPRCA_00042_1	Experimental Checker	alpha.security.taint.TaintPropagation
KPRCA_00020_1	Experimental Checker	alpha.security.ArrayBoundV2
CROMU_00021_1	Lib function	core.NullDereference, alpha.core.DynamicTypeChecker, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
CROMU_00021_2	Lib function	core.NullDereference, alpha.core.DynamicTypeChecker, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
CROMU_00021_3	Lib function	core.NullDereference, alpha.core.DynamicTypeChecker, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
CROMU_00021_4	Lib function	core.NullDereference, alpha.core.DynamicTypeChecker, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2

CROMU_00021_5	Lib function	core.NullDereference, alpha.core.DynamicTypeChecker, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
KPRCA_00039_1	Experimental Checker	alpha.security.taint.TaintPropagation
KPRCA_00038_1	Experimental Checker	alpha.security.taint.TaintPropagation, core.NullDereference
KPRCA_00038_2	Experimental Checker	alpha.security.taint.TaintPropagation, core.NullDereference
CROMU_00027_1	Experimental Checker	alpha.core.Conversion, core.NullDereference, alpha.security.ArrayBoundV2
CROMU_00027_2	Experimental Checker	alpha.core.Conversion, core.NullDereference, alpha.security.ArrayBoundV2
CROMU_00027_3	Experimental Checker	alpha.core.Conversion, core.NullDereference, alpha.security.ArrayBoundV2
CROMU_00018_1	Lib function	alpha.core.Conversion, alpha.security.ArrayBoundV2
CROMU_00018_2	Lib function	alpha.core.Conversion, alpha.security.ArrayBoundV2
NRFIN_00035_1	Experimental Checker	alpha.security.taint.TaintPropagation, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00035_2	Experimental Checker	alpha.security.taint.TaintPropagation, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00035_3	Experimental Checker	alpha.security.taint.TaintPropagation, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00035_4	Experimental Checker	alpha.security.taint.TaintPropagation, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00035_5	Experimental Checker	alpha.security.taint.TaintPropagation, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00035_6	Experimental Checker	alpha.security.taint.TaintPropagation, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
KPRCA_00009_1	Experimental Checker	alpha.security.ArrayBoundV2

KPRCA_00036_1	Experimental Checker	alpha.core.CallAndMessageUnInitRefArg, alpha.security.ArrayBoundV2
KPRCA_00036_2	Experimental Checker	alpha.core.CallAndMessageUnInitRefArg, alpha.security.ArrayBoundV2
NRFIN_00032_1	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, unix.API
NRFIN_00032_2	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, unix.API
CROMU_00016_1	Environement Modeling: Complicated data structure	unix.API, alpha.security.ArrayBoundV2
CROMU_00016_2	Environement Modeling: Complicated data structure	unix.API, alpha.security.ArrayBoundV2
YAN01_00010_1	Environement Modeling: Complicated data structure	core.NullDereference, alpha.security.ArrayBoundV2
YAN01_00010_2	Environement Modeling: Complicated data structure	core.NullDereference, alpha.security.ArrayBoundV2
CROMU_00034_1	Environement Modeling: Complicated data structure	core.NullDereference, DeprecatedOrUnsafeBufferHandling, alpha.security.ArrayBoundV2
CROMU_00034_2	Environement Modeling: Complicated data structure	core.NullDereference, DeprecatedOrUnsafeBufferHandling, alpha.security.ArrayBoundV2
CROMU_00034_3	Environement Modeling: Complicated data structure	core.NullDereference, DeprecatedOrUnsafeBufferHandling, alpha.security.ArrayBoundV2
CROMU_00034_4	Environement Modeling: Complicated data structure	core.NullDereference, DeprecatedOrUnsafeBufferHandling, alpha.security.ArrayBoundV2
KPRCA_00014_1	Experimental Checker	alpha.security.ArrayBoundV2
NRFIN_00021_1	Environement Modeling: Complicated data structure	core.NullDereference, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00021_2	Environement Modeling: Complicated data structure	core.NullDereference, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00021_3	Environement Modeling: Complicated data structure	core.NullDereference, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00021_4	Environement Modeling: Complicated data structure	core.NullDereference, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2

KPRCA_00025_1	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, core.uninitialized.UndefReturn, alpha.security.MallocOverflow
KPRCA_00025_2	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, core.uninitialized.UndefReturn, alpha.security.MallocOverflow
KPRCA_00025_3	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, core.uninitialized.UndefReturn, alpha.security.MallocOverflow
NRFIN_00026_1	Environement Modeling: Complicated data structure	DeprecatedOrUnsafeBufferHandling, unix.API
NRFIN_00026_2	Environement Modeling: Complicated data structure	DeprecatedOrUnsafeBufferHandling, unix.API
KPRCA_00040_1	Experimental Checker	DeprecatedOrUnsafeBufferHandling, alpha.security.ArrayBoundV2
KPRCA_00040_2	Experimental Checker	DeprecatedOrUnsafeBufferHandling, alpha.security.ArrayBoundV2
KPRCA_00040_3	Experimental Checker	DeprecatedOrUnsafeBufferHandling, alpha.security.ArrayBoundV2
KPRCA_00047_1	Experimental Checker	DeprecatedOrUnsafeBufferHandling, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
KPRCA_00047_2	Experimental Checker	DeprecatedOrUnsafeBufferHandling, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
KPRCA_00047_3	Experimental Checker	DeprecatedOrUnsafeBufferHandling, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00018_1	Experimental Checker	alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00018_2	Experimental Checker	alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00018_3	Experimental Checker	alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00027_1	Environement Modeling: Complicated data structure	core.DivideZero, alpha.core.DynamicTypeChecker, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2

NRFIN_00027_2	Environement Modeling: Complicated data structure	core.DivideZero, alpha.core.DynamicTypeChecker, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00027_3	Environement Modeling: Complicated data structure	core.DivideZero, alpha.core.DynamicTypeChecker, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00027_4	Environement Modeling: Complicated data structure	core.DivideZero, alpha.core.DynamicTypeChecker, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
NRFIN_00027_5	Environement Modeling: Complicated data structure	core.DivideZero, alpha.core.DynamicTypeChecker, alpha.security.MallocOverflow, alpha.security.ArrayBoundV2
KPRCA_00023_1	Experimental Checker	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2
KPRCA_00023_2	Experimental Checker	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2
NRFIN_00020_1	Experimental Checker	alpha.security.taint.TaintPropagation, alpha.security.ArrayBoundV2
NRFIN_00020_2	Experimental Checker	alpha.security.taint.TaintPropagation, alpha.security.ArrayBoundV2
NRFIN_00020_3	Experimental Checker	alpha.security.taint.TaintPropagation, alpha.security.ArrayBoundV2
NRFIN_00029_1	Experimental Checker	alpha.security.taint.TaintPropagation, core.CallAndMessage
NRFIN_00029_2	Experimental Checker	alpha.security.taint.TaintPropagation, core.CallAndMessage
CROMU_00032_1	Experimental Checker	alpha.security.ArrayBoundV2
NRFIN_00042_1	Environement Modeling: Complicated data structure	unix.API, alpha.security.ArrayBoundV2
NRFIN_00042_2	Environement Modeling: Complicated data structure	unix.API, alpha.security.ArrayBoundV2
KPRCA_00041_1	Lib function	DeprecatedOrUnsafeBufferHandling
KPRCA_00048_1	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2, alpha.unix.cstring.OutOfBounds, unix.API, core.NullDereference

KPRCA_00048_2	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2, alpha.unix.cstring.OutOfBounds, unix.API, core.NullDereference
KPRCA_00048_3	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2, alpha.unix.cstring.OutOfBounds, unix.API, core.NullDereference
KPRCA_00048_4	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2, alpha.unix.cstring.OutOfBounds, unix.API, core.NullDereference
NRFIN_00030_1	Environement Modeling: Complicated data structure	alpha.security.taint.TaintPropagation, alpha.unix.cstring.OutOfBounds, unix.API, alpha.security.ArrayBoundV2
NRFIN_00030_2	Environement Modeling: Complicated data structure	alpha.security.taint.TaintPropagation, alpha.unix.cstring.OutOfBounds, unix.API, alpha.security.ArrayBoundV2
NRFIN_00030_3	Environement Modeling: Complicated data structure	alpha.security.taint.TaintPropagation, alpha.unix.cstring.OutOfBounds, unix.API, alpha.security.ArrayBoundV2
NRFIN_00030_4	Environement Modeling: Complicated data structure	alpha.security.taint.TaintPropagation, alpha.unix.cstring.OutOfBounds, unix.API, alpha.security.ArrayBoundV2
NRFIN_00030_5	Environement Modeling: Complicated data structure	alpha.security.taint.TaintPropagation, alpha.unix.cstring.OutOfBounds, unix.API, alpha.security.ArrayBoundV2
CROMU_00014_1	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2
CROMU_00014_2	Environement Modeling: Complicated data structure	alpha.unix.cstring.OutOfBounds, alpha.security.ArrayBoundV2
KPRCA_00033_1	Experimental Checker	alpha.core.DynamicTypeChecker
KPRCA_00002_1	Experimental Checker	alpha.unix.cstring.OutOfBounds
KPRCA_00051_1	Experimental Checker	alpha.unix.cstring.OutOfBounds
CROMU_00008_1	Experimental Checker	alpha.security.ArrayBoundV2
KPRCA_00044_1	Experimental Checker	alpha.security.ArrayBoundV2
KPRCA_00032_1	Experimental Checker	alpha.unix.cstring.OutOfBounds
CROMU_00012_1	Experimental Checker	alpha.security.ArrayBoundV2

Table B.2 – FNs for CGC

Appendix C

List of analyzed test cases: Magma

Index of Bug	Cause	Checkers
LUA0002_1	Environement Modeling: Complicated data structure	core.NullDereference
LUA0002_2	Environement Modeling: Complicated data structure	core.NullDereference
LUA0002_3	Environement Modeling: Complicated data structure	core.NullDereference
LUA0002_4	Environement Modeling: Complicated data structure	core.NullDereference
LUA0002_5	Environement Modeling: Complicated data structure	core.NullDereference
LUA0002_6	Environement Modeling: Complicated data structure	core.NullDereference
LUA0002_7	Environement Modeling: Complicated data structure	core.NullDereference
LUA0002_8	Environement Modeling: Complicated data structure	core.NullDereference
LUA0002_11	Environement Modeling: Complicated data structure	core.NullDereference
LUA0002_12	Environement Modeling: Complicated data structure	core.NullDereference
LUA0002_9	Environement Modeling: random variables	alpha.security.taint.TaintPropagation
LUA0002_10	Environement Modeling: random variables	alpha.security.taint.TaintPropagation
PNG003_1	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_2	Experimental Checker	alpha.security.ArrayBoundV2

PNG003_3	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_4	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_5	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_6	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_7	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_8	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_9	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_10	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_11	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_12	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_13	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_17	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_21	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_22	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_23	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_24	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_25	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_26	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_27	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_28	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_29	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_30	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_31	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_32	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_33	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_34	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_35	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_36	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_37	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_38	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_39	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_40	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_41	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_42	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_43	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_44	Experimental Checker	alpha.security.ArrayBoundV2
PNG003_45	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_1	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_2	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_3	Experimental Checker	alpha.security.ArrayBoundV2

TIF002_4	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_5	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_6	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_7	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_8	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_9	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_10	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_11	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_12	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_13	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_14	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_15	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_16	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_17	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_18	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_19	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_20	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_21	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_22	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_23	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_24	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_25	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_26	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_27	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_28	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_29	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_30	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_31	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_32	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_33	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_34	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_35	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_36	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_37	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_38	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_39	Experimental Checker	alpha.security.ArrayBoundV2
TIF002_40	Experimental Checker	alpha.security.ArrayBoundV2
TIF004_1	Environement Modeling: Complicated data structure	core.NullDereference

TIF004_2	Environement Modeling: Complicated data structure	core.NullDereference
XML001_1	Experimental Checker	alpha.security.ArrayBoundV2
XML001_2	Experimental Checker	alpha.security.ArrayBoundV2
XML001_3	Experimental Checker	alpha.security.ArrayBoundV2
XML001_4	Experimental Checker	alpha.security.ArrayBoundV2
XML001_5	Experimental Checker	alpha.security.ArrayBoundV2
XML001_6	Experimental Checker	alpha.security.ArrayBoundV2
XML001_7	Experimental Checker	alpha.security.ArrayBoundV2
XML001_8	Experimental Checker	alpha.security.ArrayBoundV2
XML001_9	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
XML001_10	Experimental Checker	alpha.security.ArrayBoundV2
XML001_11	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
XML001_12	Experimental Checker	alpha.security.ArrayBoundV2
XML001_13	Experimental Checker	alpha.security.ArrayBoundV2
XML001_14	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
XML001_15	Experimental Checker	alpha.security.ArrayBoundV2
XML001_16	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
XML001_17	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
XML001_18	Experimental Checker	alpha.security.ArrayBoundV2
XML001_19	Experimental Checker	alpha.security.ArrayBoundV2
XML001_20	Experimental Checker	alpha.security.ArrayBoundV2
XML001_21	Experimental Checker	alpha.security.ArrayBoundV2
XML001_22	Experimental Checker	alpha.security.ArrayBoundV2
XML001_23	Experimental Checker	alpha.security.ArrayBoundV2
XML001_24	Experimental Checker	alpha.security.ArrayBoundV2
XML001_25	Experimental Checker	alpha.security.ArrayBoundV2
XML001_26	Experimental Checker	alpha.security.ArrayBoundV2
XML001_27	Experimental Checker	alpha.security.ArrayBoundV2
XML001_28	Experimental Checker	alpha.security.ArrayBoundV2
XML001_29	Experimental Checker	alpha.security.ArrayBoundV2
XML001_30	Experimental Checker	alpha.security.ArrayBoundV2
XML001_31	Experimental Checker	alpha.security.ArrayBoundV2
XML001_32	Experimental Checker	alpha.security.ArrayBoundV2
XML001_33	Experimental Checker	alpha.security.ArrayBoundV2

XML001_34	Experimental Checker	alpha.security.ArrayBoundV2
XML001_35	Experimental Checker	alpha.security.ArrayBoundV2
XML001_36	Experimental Checker	alpha.security.ArrayBoundV2
XML001_37	Experimental Checker	alpha.security.ArrayBoundV2
XML001_38	Experimental Checker	alpha.security.ArrayBoundV2
XML001_39	Experimental Checker	alpha.security.ArrayBoundV2
XML001_40	Experimental Checker	alpha.security.ArrayBoundV2
XML001_41	Experimental Checker	alpha.security.ArrayBoundV2
XML001_42	Experimental Checker	alpha.security.ArrayBoundV2
XML001_43	Experimental Checker	alpha.security.ArrayBoundV2
XML001_44	Experimental Checker	alpha.security.ArrayBoundV2
XML001_45	Experimental Checker	alpha.security.ArrayBoundV2
XML001_46	Experimental Checker	alpha.security.ArrayBoundV2
XML001_47	Experimental Checker	alpha.security.ArrayBoundV2
XML001_48	Experimental Checker	alpha.security.ArrayBoundV2
XML001_49	Experimental Checker	alpha.security.ArrayBoundV2
XML001_50	Experimental Checker	alpha.security.ArrayBoundV2
XML001_51	Experimental Checker	alpha.security.ArrayBoundV2
SQL003_1	Environement Modeling: Complicated data structure	core.NullDereference
SQL003_2	Environement Modeling: Complicated data structure	core.NullDereference
SQL003_3	Environement Modeling: Complicated data structure	core.NullDereference
SQL003_4	Environement Modeling: Complicated data structure	core.NullDereference
SQL003_5	Environement Modeling: Complicated data structure	core.NullDereference
SQL003_6	Environement Modeling: Complicated data structure	core.NullDereference
SQL003_7	Environement Modeling: Complicated data structure	core.NullDereference
SQL003_8	Lib function	core.NullDereference
SQL003_9	Lib function	core.NullDereference
SQL003_10	Lib function	core.NullDereference
SQL009_1	Lib function	alpha.security.ArrayBoundV2
SQL009_2	Lib function	alpha.security.ArrayBoundV2
SQL009_3	Lib function	alpha.security.ArrayBoundV2
SQL009_4	Lib function	alpha.security.ArrayBoundV2
SQL009_5	Lib function	alpha.security.ArrayBoundV2

SQL009_6	Lib function	alpha.security.ArrayBoundV2
SQL009_7	Lib function	alpha.security.ArrayBoundV2
SQL009_8	Lib function	alpha.security.ArrayBoundV2
SQL009_9	Lib function	alpha.security.ArrayBoundV2
SQL009_10	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_11	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_12	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_13	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_14	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_15	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_16	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_17	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_18	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_19	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_20	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_21	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_22	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_23	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_24	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_25	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_26	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SQL009_27	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_28	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_29	Experimental Checker	alpha.security.ArrayBoundV2

SQL009_30	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_31	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_32	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_33	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_34	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_35	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_36	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_37	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_38	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_39	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_40	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_41	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_42	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_43	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_44	Experimental Checker	alpha.security.ArrayBoundV2
SQL009_45	Experimental Checker	alpha.security.ArrayBoundV2
SSL008	Environement Modeling: Complicated data structure	core.NullDereference
SND004	Lib function	core.DivideZero
SND001_1	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SND001_2	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SND001_3	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SND001_4	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SND001_5	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SND001_6	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SND001_7	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SND001_8	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SND001_9	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SND001_10	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
SND001_11	Experimental Checker	alpha.security.ArrayBoundV2
SND001_12	Experimental Checker	alpha.security.ArrayBoundV2

SND001_13	Experimental Checker	alpha.security.ArrayBoundV2
SND001_14	Experimental Checker	alpha.security.ArrayBoundV2
SND001_15	Experimental Checker	alpha.security.ArrayBoundV2
SND001_16	Experimental Checker	alpha.security.ArrayBoundV2
SND001_17	Experimental Checker	alpha.security.ArrayBoundV2
SND001_18	Experimental Checker	alpha.security.ArrayBoundV2
SND001_19	Experimental Checker	alpha.security.ArrayBoundV2
SND001_20	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_1	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_2	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_3	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_4	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_5	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_6	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_7	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_8	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_9	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_10	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_11	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_12	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_13	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_14	Experimental Checker	alpha.security.ArrayBoundV2
PHP001_23	Lib function	alpha.security.ArrayBoundV2
PHP001_24	Lib function	alpha.security.ArrayBoundV2
PHP001_25	Lib function	alpha.security.ArrayBoundV2
PHP001_26	Lib function	alpha.security.ArrayBoundV2
PHP001_27	Lib function	alpha.security.ArrayBoundV2
PHP001_28	Lib function	alpha.security.ArrayBoundV2
PHP001_29	Lib function	alpha.security.ArrayBoundV2
PHP001_30	Lib function	alpha.security.ArrayBoundV2
PHP001_31	Lib function	alpha.security.ArrayBoundV2
PHP001_32	Lib function	alpha.security.ArrayBoundV2
PHP001_33	Lib function	alpha.security.ArrayBoundV2
PHP001_34	Lib function	alpha.security.ArrayBoundV2
PHP001_35	Lib function	alpha.security.ArrayBoundV2
PHP001_36	Lib function	alpha.security.ArrayBoundV2
PHP001_37	Lib function	alpha.security.ArrayBoundV2
PHP001_38	Lib function	alpha.security.ArrayBoundV2
PHP001_39	Lib function	alpha.security.ArrayBoundV2
PHP001_40	Lib function	alpha.security.ArrayBoundV2

PHP001_41	Lib function	alpha.security.ArrayBoundV2
PHP001_42	Lib function	alpha.security.ArrayBoundV2
PHP001_43	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PHP001_44	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PHP001_45	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PHP001_46	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PHP001_47	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PHP001_48	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PHP001_49	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PHP001_50	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PHP001_51	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PHP001_52	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF001_1	Environement Modeling: Complicated data structure	core.DivideZero
PDF001_2	Environement Modeling: Complicated data structure	core.DivideZero
PDF001_3	Environement Modeling: Complicated data structure	core.DivideZero
PDF001_4	Lib function	core.DivideZero
PDF001_5	Lib function	core.DivideZero
PDF004_6	Environement Modeling: Complicated data structure	core.NullDereference
PDF004_7	Environement Modeling: Complicated data structure	core.NullDereference
PDF004_8	Environement Modeling: Complicated data structure	core.NullDereference
PDF004_9	Environement Modeling: Complicated data structure	core.NullDereference
PDF004_10	Lib function	core.NullDereference
PDF004_11	Lib function	core.NullDereference
PDF004_12	Lib function	core.NullDereference

PDF004_13	Lib function	core.NullDereference
PDF004_14	Lib function	core.NullDereference
PDF004_15	Lib function	core.NullDereference
PDF022_1	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_2	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_3	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_4	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_5	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_6	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_7	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_8	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_9	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_10	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF022_11	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF022_12	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF022_13	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF022_14	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF022_15	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF022_16	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF022_17	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF022_18	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF022_19	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF022_20	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF022_21	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF022_22	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_23	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_24	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_25	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_26	Experimental Checker	alpha.security.ArrayBoundV2

PDF022_27	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_28	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_29	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_30	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_31	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_32	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_33	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_34	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_35	Lib function	alpha.security.ArrayBoundV2
PDF022_36	Lib function	alpha.security.ArrayBoundV2
PDF022_37	Lib function	alpha.security.ArrayBoundV2
PDF022_38	Lib function	alpha.security.ArrayBoundV2
PDF022_39	Lib function	alpha.security.ArrayBoundV2
PDF022_40	Lib function	alpha.security.ArrayBoundV2
PDF022_41	Lib function	alpha.security.ArrayBoundV2
PDF022_42	Lib function	alpha.security.ArrayBoundV2
PDF022_43	Lib function	alpha.security.ArrayBoundV2
PDF022_44	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_45	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_46	Experimental Checker	alpha.security.ArrayBoundV2
PDF022_47	Experimental Checker	alpha.security.ArrayBoundV2

Table C.1 – FPs for Magma

Index of Bug	Cause	Checkers
LUA0002	Environement Modeling: Complicated data structure	core.NullDereference
PNG002	Environement Modeling: Complicated data structure	unix.Malloc
PNG003	Experimental Checker	alpha.security.ArrayBoundV2
PNG004	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PNG005	Experimental Checker	alpha.security.ArrayBoundV2
PNG007	Lib function	core.NullDereference
TIF001	Z3: Undeterminism	alpha.security.ArrayBoundV2
TIF002	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
TIF004	Environement Modeling: Complicated data structure	core.DivideZero
TIF005	Experimental Checker	alpha.security.ArrayBoundV2

TIF007	Experimental Checker	alpha.security.ArrayBoundV2
TIF008	Experimental Checker	alpha.security.ArrayBoundV2
TIF009	Experimental Checker	alpha.security.ArrayBoundV2
TIF010	Experimental Checker	alpha.security.ArrayBoundV2
TIF012	Lib function	alpha.security.ArrayBoundV2
TIF013	Experimental Checker	alpha.security.ArrayBoundV2
TIF014	Experimental Checker	alpha.security.taint.TaintPropagation
XML001	Experimental Checker	alpha.security.ArrayBoundV2
XML002	Experimental Checker	alpha.security.ArrayBoundV2
XML005	Experimental Checker	alpha.security.ArrayBoundV2
XML007	Experimental Checker	alpha.security.ArrayBoundV2
XML010	Experimental Checker	alpha.security.taint.TaintPropagation
XML011	Experimental Checker	alpha.security.ArrayBoundV2
XML012	Lib function	unix.Malloc
XML013	Lib function	unix.Malloc
XML014	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
XML016	Experimental Checker	alpha.security.ArrayBoundV2
XML017	Experimental Checker	alpha.security.ArrayBoundV2
PDF002	Experimental Checker	alpha.security.MallocOverflow
PDF003	Experimental Checker	alpha.security.ArrayBoundV2
PDF004	Environement Modeling: Complicated data structure	core.NullDereference
PDF005	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF006	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF007	Environement Modeling: Complicated data structure	alpha.security.ArrayBoundV2
PDF008	Environement Modeling: Complicated data structure	core.DivideZero
PDF009	Lib function	alpha.security.MallocOverflow
PDF010	Environement Modeling: Complicated data structure	core.NullDereference
PDF011	Experimental Checker	alpha.security.ArrayBoundV2
PDF012	Experimental Checker	alpha.security.MallocOverflow
PDF013	Experimental Checker	alpha.security.taint.TaintPropagation
PDF014	Lib function	core.NullDereference
PDF016	Experimental Checker	alpha.security.ArrayBoundV2

PDF018	Environement Modeling: Complicated data structure	core.NullDereference
PDF019	Experimental Checker	alpha.security.MallocOverflow
PDF021	Experimental Checker	alpha.security.taint.TaintPropagation
SSL001	Experimental Checker	alpha.security.ArrayBoundV2
SSL004	Experimental Checker	alpha.security.ArrayBoundV2
SSL007	Experimental Checker	alpha.security.ArrayBoundV2
SSL008	Environement Modeling: Complicated data structure	core.NullDereference
SSL009	Experimental Checker	alpha.security.ArrayBoundV2
SSL010	Experimental Checker	alpha.security.MallocOverflow
SSL013	Environement Modeling: Complicated data structure	core.NullDereference
SSL015	Environement Modeling: Complicated data structure	core.NullDereference
SSL017	Environement Modeling: Complicated data structure	core.NullDereference
SSL019	Experimental Checker	alpha.security.taint.TaintPropagation
SSL020	Experimental Checker	alpha.security.taint.TaintPropagation
SQL003	Environement Modeling: Complicated data structure	core.NullDereference
SQL010	Lib function	unix.Malloc
SQL014	Environement Modeling: Complicated data structure	core.NullDereference
SQL015	Experimental Checker	alpha.core.Conversion
SQL016	Experimental Checker	alpha.security.taint.TaintPropagation
SQL017	Environement Modeling: Complicated data structure	core.NullDereference
SQL018	Experimental Checker	alpha.security.taint.TaintPropagation
SQL019	Experimental Checker	alpha.security.ArrayBoundV2
SQL020	Environement Modeling: Complicated data structure	core.NullDereference
SQL001	Experimental Checker	alpha.security.ArrayBoundV2
PHP012	Experimental Checker	alpha.security.ArrayBoundV2
PHP001	Experimental Checker	alpha.security.ArrayBoundV2
PHP002	Experimental Checker	alpha.security.ArrayBoundV2
PHP003	Experimental Checker	alpha.security.ArrayBoundV2
PHP004	Experimental Checker	alpha.security.ArrayBoundV2
PHP005	Experimental Checker	alpha.security.ArrayBoundV2
PHP006	Experimental Checker	alpha.security.ArrayBoundV2

PHP007	Experimental Checker	alpha.security.ArrayBoundV2
PHP009	Experimental Checker	alpha.security.ArrayBoundV2
PHP015	Experimental Checker	alpha.security.MallocOverflow
PHP016	Experimental Checker	alpha.security.ArrayBoundV2
SND001	Experimental Checker	alpha.security.ArrayBoundV2
SND004	Environement Modeling: Complicated data structure	core.DivideZero
SND005	Experimental Checker	alpha.security.ArrayBoundV2
SND006	Experimental Checker	alpha.security.ArrayBoundV2
SND007	Experimental Checker	alpha.security.ArrayBoundV2
SND010	Experimental Checker	alpha.security.ArrayBoundV2
SND012	Experimental Checker	alpha.security.ArrayBoundV2
SND013	Experimental Checker	alpha.security.ArrayBoundV2
SND016	Environement Modeling: Complicated data structure	core.DivideZero

Table C.2 – FNs for Magma