SolidityCheck使用手册

肖锋*

November 29, 2019

1 项目说明

*SolidityCheck*是一个基于正则表达式和代码插桩的静态代码问题检测工具，支持以太坊智能合约。 *SolidityCheck*接收智能合约的源代码文件。首先，对源代码进行格式化，以便可以正则表达式匹配。 然后用特定的正则表达式匹配问题语句以定位问题语句的位置。 对于重入漏洞和整数溢出问题，*SolidityCheck*结合代码插桩来预防这两个问题的出现。 实验表明，*SolidityCheck*是一种非常有效的智能合约问题检测工具，具有很高的查全率、查准率和检测效率。

2 如何获取

我在Github(https://github.com/xf97/SolidityCheck)上开源了源代码，*SolidityCheck*使用C++开发，遵循的语言标准是 C++ 11，开放的源代码包含全部的头文件、源代码文件。

由于最初是在windows 10环境下开发*SolidityCheck*，使用的开发工具是Visual Studio 2017社区版，所以获取*SolidityCheck*最合适的方式(在windows 10系统下)是:

1. 从Github上下载我所有的代码。在保证你的电脑已经安装了Git的情况下，在Git中使用以下命令克隆项目到本地:
   mkdir SolidityCheck
   cd SolidityCheck
   git clone https://github.com/xf97/SolidityCheck
2. 使用Visual Studio 2017(当然其他版本的Visual Studio或者是任何一个可以创建项目的ide也是可以的)创建一个新的项目，将下载获得的所有源代码添加进入新创建的项目中。

*肖锋的邮箱：1150264019@qq.com

3. 编译项目，生成可执行程序。

4. 将生成的可执行程序加入到环境变量或者系统变量*Path*中，现在你就能够通过命令行来使用*SolidityCheck*啦。

由于我在开发*SolidityCheck*时并没有依赖任何操作系统的功能，所以理论上*SolidityCheck*可以在任何一种操作系统上运行。事实上是，我在ubuntu 16.04(和18.04)系统中使用*SolidityCheck*是没有问题的。

在ubuntu中，我使用Clion编译*SolidityCheck*获得了可以运行的程序。最基础的情况，当你使用的电脑上有g++编译器，你就可以使用*SolidityCheck*。使用如下命令编译*SolidityCheck*(在linux环境下):

mkdir SolidityCheck
cd SolidityCheck
git clone https://github.com/xf97/SolidityCheck cd SolidityCheck
g++ –std=c++11 *.cpp *.h -o SolidityCheckV1

使用以上命令生成的可执行程序名为*SolidityCheckV1*，现在你可以在linux系统上通过终端使用*SolidityCheck*了。至于mac os系统，额，我确实没有一台运行mac os的电脑，所以，你可以试着自己编译一下源代码，应该也是可行的(如果不行，请发邮件告诉我)。

为了保证在任何一个目录下你都能使用*SolidityCheck*，你可以将包含*SolidityCheck*的路径加入到环境变量PATH中。

PS: 由于我使用的C++ 11标准中，没有异常捕获机制(在我看的C++ primer plus第五版里说的没有)，所以。。。额。。。当你使用*SolidityCheck*时可能会发生一些看起来很不友好的错误提示，不要慌，很大可能是没有给定正确的文件路径，可以输入在当前系统中被检测合约正确路径。如果真的不是文件打开错误，那么可能是格式化代码的时候有错误，没事，我会在以后把它修好的。（如果你不介意的话，遇到这种情况，可以把运行错误截图和被检测合约发给我，通过这些我可以知道为什么错了）

3 *SolidityCheck*的功能

通过命令行(CLI,command-line interface)终端调用*SolidityCheck*的功能，*SolidityCheck*一共提供了以下功能:

1. 获取帮助信息 —>SolidityCheck –help

2. 生成预防重入漏洞的智能合约 —>SolidityCheck –r

3. 生成预防整数溢出问题的智能合约 —>SolidityCheck –o

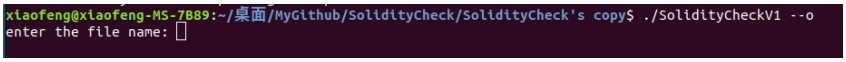
4. 检测18种智能合约问题 —>SolidityCheck –d
5. 调整昂贵的循环判定标准 —>SolidityCheck –s
6. 获取当前的昂贵的循环判定标准(会有初始值) —>SolidityCheck –g
7. 批量检测，用于一次性获取多份智能合约的检测结果 —>SolidityCheck –f

4 使用示例

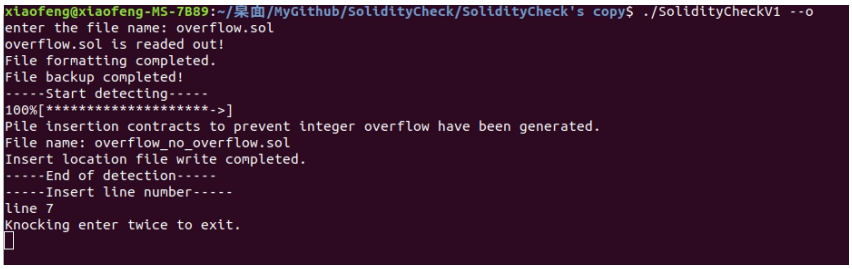
在这一章中，我详细介绍该如何使用*SolidityCheck*。

4.1 生成预防整数溢出问题的合约

第一步，你应该打开终端，输入命令”SolidityCheck –o”。
第二步，此时*SolidityCheck*会要求你输入文件名，注意这里的文件名应该是文件的绝对路径。



第三步，给定正确的文件路径后，敲击回车，*SolidityCheck*会自动为分析程序插入代码，当程序正确运行时，会出现如下截图(假设给定的合约与可执行程序在同一文件夹，名为overflow.sol):
通过查看*SolidityCheck*输出的信息，可以看到，生成的预防整数溢出问题



的合约名为 overflow_no_overflow.sol，插入代码的位置在第七行，这个第七行的参照系是与原合约同目录的 overflow_backup.txt。（注意，程序完成后敲击两次回车便可退出程序）
第四步，获取生成的预防合约，查看被检测合约存放的目录，生成的预防合约就存在于那个目录。一般程序会生成以下几个文件:

1. overflow_backup.txt，用于备份原合约，会在格式化后的每行代码前插入行号，便于用户对照插入代码的位置。

2. overflow_format.sol，用于保存格式化后的代码的文件。

3. overflow_insert_position.txt，记录着插入代码的位置和该位置的参照文件名。

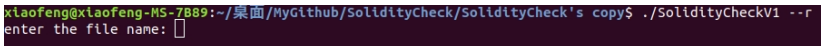
4. overflow_no_overflow.sol，根据原合约生成的预防整数溢出问题的合约。

PS: 生成的预防合约的格式可能比较差劲，额，这个是我偷懒了，我在之后会增加代码缩进的功能的。

4.2 生成预防重入漏洞的合约

第一步，你应该打开终端，输入命令”SolidityCheck –r”。

第二步，此时*SolidityCheck*会要求你输入文件名，注意这里的文件名应该是文件的绝对路径。



第三步，给定正确的文件路径后，敲击回车，*SolidityCheck*会自动为分析程序插入代码，一般情况下，如果包含”危险语句”的函数使用了参数的话，*SolidityCheck*会要求你给定参数值，这是为了在deposit_test函数中写入合适的函数调用语句。如下图所示，我们要求检测的合约名为: reentrancy.sol，其中包含“危险语句”的函数 withdrawBalance 使用了一个参数，类型为uint256，参数名为_money。此时，作为开发或者审计合约的人，你需要给定一个合适的值，在这里，我们给定值为1。
输入参数后敲击回车，程序自动完成了后续的部分。

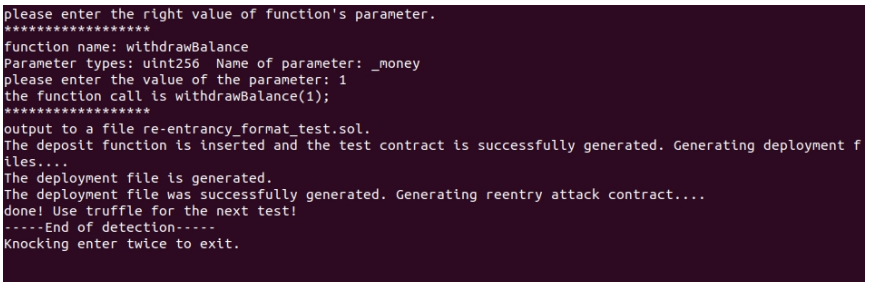
第四步，获取生成的预防合约，通过查看*SolidityCheck*的输出信息(见下图), 生成的预防合约名为”re-entrancy_format_test.sol”(原文件名+”_format_test.sol”)

除了预防合约之外，程序还会生成以下几个文件:

1. re-entrancy_format.sol，用于保存格式化后的代码的文件。

2. re-entrancy_format_chain.txt，这个文件用于记录被检测合约内函数之间的调用关系，根据这个文件，我们能够构造函数调用链，准确地插入代码

3. n_deploy_ReentranceTEST.js，这是根据被测试合约生成的truffle环境部署文件，方便用户将预防合约部署到私有链环境中测试合约。

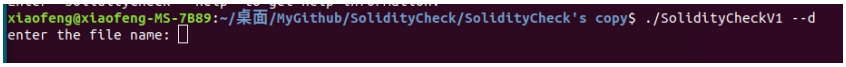


4.3 快速扫描智能合约

*SolidityCheck*提供的智能合约检测功能可以快速地扫描18种智能合约问题，扫描的问题种类请参看我们的论文。*SolidityCheck*的检测效率十分出众，一般而言，在数秒之内你就能获得检测结果。该功能的操作步骤如下：

第一步，你应该打开终端，输入命令”SolidityCheck –d”。

第二步，此时*SolidityCheck*会要求你输入文件名，注意这里的文件名应该是文件的绝对路径。



第三步，给定正确的文件路径后，敲击回车，*SolidityCheck*会全自动地完成合约的扫描工作。*SolidityCheck*在终端中的输出如下图所示:

第四步，获取输出的检测报告。观察*SolidityCheck*在终端的输出可以知道，*SolidityCheck*输出的检测报告名为 testCase1_detect.report，该检测报告的存放目录是被检测合约的目录。检测报告是普通的文本格式的文件，你可以使用任何一种能查看ascii码文本文档的文档查看器获取内容。例如，*SolidityCheck*为testCase1.sol生成的检测报告的一部分如下图所示:

```
xiaofeng@xiaofeng-MS-7B89:~/桌面/MyGithub/SolidityCheck/SolidityCheck's copy$ ./SolidityCheckV1 --d
enter the file name: testCase1.sol
testCase1.sol is readed out!
File formatting completed.
File backup completed!
-----Start detecting-----
100%[************************->]
Detection report has generated.
file name: testCase1_detect.report
-----End of detection-----
Knocking enter twice to exit.
```

```
file name: testCase1.sol
number of lines of code: 273
use time: 0.167562 s.
total number of vulnerabilities: 37

No private modifier.


No Costly Loop.


No balanace equality.


No mishandled exceptions.


No tx.origin for authentication .


No unsafe type inference.


[Vulnerability 7]
vulnerability name: Compiler version problem
number of vulnerabilities: 1
row number: 1
additional description: It is recommended to explicitly specify the compiler version used in the current
contract and version operators '^' be cancelled and new versions of security specifications be added(such
as " pragma experimental "v0.5.0"; "). Because future compilers may allow situations that developers
haven't met before.
Vulnerability level:warning
Attention: Add the following statement to indicate the version of the security specification used: pragma
experimental "vCompiler Version"

No time dependence.


No integer division.


No locked money.


No byte[].


No redundant refusal of payment.


[Vulnerability 17]
```

报告的前四行分别是:

1. 合约文件名
2. 合约代码行数
3. 检测用时
4. 该合约总共扫描出的问题数

然后，往下的内容分别是每个问题的:

1. 问题编号
2. 该种类问题的数量
3. 问题存在的行号(行号仍然是参考生成的备份文件)
4. 问题危害描述及修改建议
5. 问题严重级别(分warning 和 error)
6. 附加注意事项（可选）

除了检测报告外，*SolidityCheck*还会生成以下文件:

1. testCase1_format.sol，用来存储格式化后代码的文件
2. testCase1_backup.txt，原合约代码备份文件，在每行代码前插入了行号，检测报告中的行号是参照这个文件生成的
3. GasLimit.ini，这个文件是问题 昂贵的循环 的标准配置文件，昂贵的循环 具体信息和这个标准的初始值设定请参看我们的论文

4.4 其他功能

*SolidityCheck*除了以上介绍的功能外，还提供了其他三个功能，它们都是辅助功能，我在接下来的三小节中介绍它们:

4.4.1 昂贵的循环问题判定标准获取和修改

命令 SolidityCheck –>s 用于修改 GasLimit.ini文件的值，修改该值的效果是修改问题检测（扫描合约）功能中昂贵的循环问题的判定标准。为了保证程序的正常运行，请保证输入的新值是一串数字（防止错误输入值的功能会在之后添加）。

命令 SolidityCheck –>g 用于获取当前GasLimit.ini文件的值，该值是现在问题检测（扫描合约）功能中昂贵的循环问题的判定标准。

4.4.2 批量检测

命令 SolidityCheck –f 是批量检测功能，该功能实质上问题检测（扫描合约）功能的批量使用。如果要使用该功能，你应该打开终端并且在输入命令 SolidityCheck –f，此时*SolidityCheck*会要求你给定一个文件的绝对路径，这个文件中应该包含所有要被*SolidityCheck*检测的智能合约文件的绝对路径。

给定该文件路径后，敲击回车，程序会输出当前的检测进度。当检测完成时，程序会输出该批量检测的总共消耗时间。

5 学术论文

我们的论文可以在以下网址获取，它是免费且开放的（可能和最终发表的有细微的区别），网址是：https://arxiv.xilesou.top/abs/1911.09425。

6 声明

*SolidityCheck*是在麻省理工学院(MIT)许可证下发布的，这意味着：

6.1 权利

1. 任何人都有权利使用、复制、修改、合并、出版发行、散布、再授权以及售卖*SolidityCheck*及*SolidityCheck*的副本。
2. 任何人都可以根据程序的需要修改授权条款为适当的内容。

6.2 义务

在*SolidityCheck*和*SolidityCheck*的所有副本中都必须包含版权声明和许可声明。

7 注意

因为原来的代码有一些问题（主要是原来的代码在linux和mac os系统上运行会存在一些问题，并且生成的预防合约对用户而言不太友好，我们决定修复这些问题），所以我们暂时删除了之前的提交记录和下架了源代码，我保证更好的代码很快就会回来的。