

lab 3.1

overview

The learning objective of this lab is for students to gain the first-hand experience on using static code analysis tools to check c program for security vulnerabilities and coding mistakes.

your goal is to achieve the followings:

- Install splint;
- Finish code samples with 2 different kinds of problems which can be detected by Splint. You can choose any 2 of 11 problems as above.
- Use splint to detect the 2 kinds of problems. Describe your observations in your report.

install splint

```
(base) thy@DESKTOP-VNN9TOF:/mnt/f/桌面/一些文件/通识/ZJU通识/安全编程技术/作业/lab3.1/tmp$ sudo mkdir /usr/local/splint
[sudo] password for thy:
(base) thy@DESKTOP-VNN9TOF:/mnt/f/桌面/一些文件/通识/ZJU通识/安全编程技术/作业/lab3.1/tmp$ cd splint-3.1.2
(base) thy@DESKTOP-VNN9TOF:/mnt/f/桌面/一些文件/通识/ZJU通识/安全编程技术/作业/lab3.1/tmp/splint-3.1.2$ ls
Makefile.am      Makefile.binary.in  README            aclocal.m4        configure          configure.binary    fixBinaryDist.sh  install.html
Makefile.binary.am  Makefile.in         acinclude.m4      config.hin         configure.ac        configure.binary.ac  fixBinaryDist.sh  install.html
(base) thy@DESKTOP-VNN9TOF:/mnt/f/桌面/一些文件/通识/ZJU通识/安全编程技术/作业/lab3.1/tmp/splint-3.1.2$ ./configure --prefix=/usr/local/splint
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking target system type... x86_64-unknown-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ANSI C... none needed
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
checking how to run the C preprocessor... gcc -E
checking for flex... no
checking for lex... no
checking for yywrap in -lfl... no
checking for yywrap in -ll... no
checking for a BSD-compatible install... /usr/bin/install -c
checking whether make sets $(MAKE)... (cached) yes
checking whether ln -s works... yes
checking for bison... no
checking for grep... grep
checking for diff... diff
checking for cat... cat
checking for rm... rm
checking for mv... mv
checking for cp... cp
checking for sed... sed
checking whether we need _ALL_SOURCE to expose mode_t... no
checking whether to include support for LCL files... yes
```

```
(base) thy@DESKTOP-VN9STOF:/mnt/f/桌面/一些文件/通识/ZJU通识/安全编程技术/作业/lab3.1/tmp/splint-3.1.2$ make
make all-recursive
make[1]: Entering directory '/mnt/f/桌面/一些文件/通识/ZJU通识/安全编程技术/作业/lab3.1/tmp/splint-3.1.2'
Making all in src
make[2]: Entering directory '/mnt/f/桌面/一些文件/通识/ZJU通识/安全编程技术/作业/lab3.1/tmp/splint-3.1.2/src'
grep "FLG_" flags.def > Headers/flag_codes.gen
make
make[3]: Entering directory '/mnt/f/桌面/一些文件/通识/ZJU通识/安全编程技术/作业/lab3.1/tmp/splint-3.1.2/src'
Compiling cgrammar.c...
Compiling cscanner.c...
Compiling mtscanner.c...
Compiling mtgrammar.c...
Compiling llgrammar.c...
Compiling signature.c...
Compiling cppmain.c...
Compiling cpplib.c...
Compiling cppexp.c...
Compiling cpphash.c...
Compiling cpperror.c...
Compiling context.c...
Compiling uentry.c...
Compiling cprim.c...
Compiling macrocache.c...
Compiling qual.c...
Compiling qtype.c...
Compiling stateClause.c...
Compiling stateClauseList.c...
Compiling ctype.c...
Compiling cvar.c...
Compiling clabstract.c...
Compiling idDecl.c...
Compiling clause.c...
Compiling globalsClause.c...
Compiling modifiesClause.c...
Compiling warnClause.c...
Compiling functionClause.c...
Compiling functionClauseList.c...
Compiling metaStateConstraint.c...
Compiling metaStateConstraintList.c...
Compiling metaStateExpression.c...
```

Finish code samples with 2 different kinds of problems

Problem code 1

使用如下代码进行漏洞分析

```
#include<stdio.h>
#include<string.h>

int main() {
    char test[100];
    scanf("%s", test);
    printf(test);
    return 1;
}
```

Problem code 2

使用如下代码进行漏洞分析

```
int main() {
    int a = 0;
    int b = 1;
    if(a = b) {
        printf("wrong");
    }
}
```

对于test1.c,分析得到如下结果,可以看到,splint检测到了两个错误,一个是程序没有注意scanf函数的返回值,另一个是典型的格式化字符串漏洞

对于test2.c,分析得到如下结果,splint检测到了三个错误,一个是使用了if(a = b),一个是if()内的表达式既不是Boolean也不是int,还有一个是main函数没有写返回值

[illegible]