Secure Programming Experiment Note (1)
Code Review Tools

The followings are some open source tools

1. flawfinder (running in Linux system, see http://www.dwheeler.com/flawfinder/)
2. Splint (see http://www.splint.org/)
3. CPPCheck
   (http://sourceforge.net/apps/mediawiki/cppcheck/index.php?title=Main_Page)
4. Findbugs (http://findbugs.sourceforge.net/) for Java
   See the website below
   http://samate.nist.gov/index.php/Source_Code_Security_Analyzers.html

- Flawfinder


Flawfinder Home Page

Flawfinder

This is the main web site for *flawfinder*, a program that examines source code and reports possible security weaknesses (``flaws'') sorted by risk level. It's very useful for quickly potential security problems *before* a program is widely released to the public. See ``how does Flawfinder work?'', below, for more information on how it works.

Flawfinder is specifically designed to be easy to install and use. After installing it, at a command line just type:

```
flawfinder directory_with_source_code
```

Flawfinder works on Unix-like systems today (it's been tested on GNU/Linux), and it should be easy to port to Windows systems. It requires Python 1.5 or greater to run (Pytho

Please take a look at other static analysis tools for security, too. One reason I wrote flawfinder was to encourage using static analysis tools to find security vulnerabilities.

Sample Output

If you're curious what the results look like, here are some sample outputs:

1. The actual text output (when allowing all potential vulnerabilities to be displayed)

Test the following commands:

flawfinder bfsucc.c > bfsucc.ff

```
Flawfinder version 1.27, (C) 2001-2004 David A. Wheeler.
Number of dangerous functions in C/C++ ruleset: 160
Examining bfsucc.c
bfsucc.c:25:  [5] (buffer) gets:
  Does not check for buffer overflows. Use fgets() instead.
bfsucc.c:17:  [4] (shell) system:
  This causes a new program to execute and is difficult to use safely.
  try using a library call that implements the same functionality if
  available.
bfsucc.c:23:  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger than
  the maximum possible length.

Hits = 3
Lines analyzed = 29 in 0.51 seconds (2247 lines/second)
Physical Source Lines of Code (SLOC) = 29
Hits@level = [0]   0 [1]   0 [2]   1 [3]   0 [4]   1 [5]   1
Hits@level+ = [0+]   3 [1+]   3 [2+]   3 [3+]   2 [4+]   2 [5+]   1
Hits/KSLOC@level+ = [0+] 103.448 [1+] 103.448 [2+] 103.448 [3+] 68.9655 [4+] 68.
9655 [5+] 34.4828
Minimum risk level = 1
Not every hit is necessarily a security vulnerability.
--更多--(94%)
```

- Splint (Secure Programming Lint)

Test the following commands:

splint bfsucc.c > bfsucc.sp

documentation: http://www.splint.org/manual/manual.pdf

Practice: Please test qwik-smtpd.c by using the above two tools.

Connect to: hacknew@120.113.173.21

On-line tool

Checkmarx: http://www.checkmarx.com/

Kiuwan - SaaS Software Quality & Security Analysis https://www.kiuwan.com/

Others

1. Owasp source code analysis tools
   https://www.owasp.org/index.php/Source_Code_Analysis_Tools

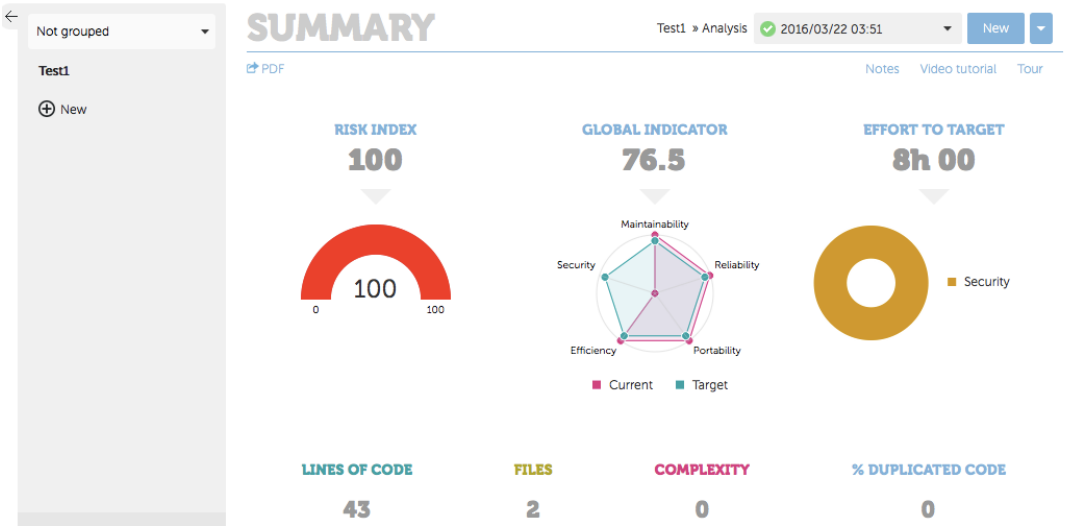
2. VCG - Scans C/C++, Java, C# and PL/SQL for security issues and for comments which may indicate defective code. The config files can be used to carry out additional checks for banned functions or functions which commonly cause security issues.
3. Fortify 與資訊安全
   http://www.gss.com.tw/index.php/focus/eis/44-eis59/122-fortify

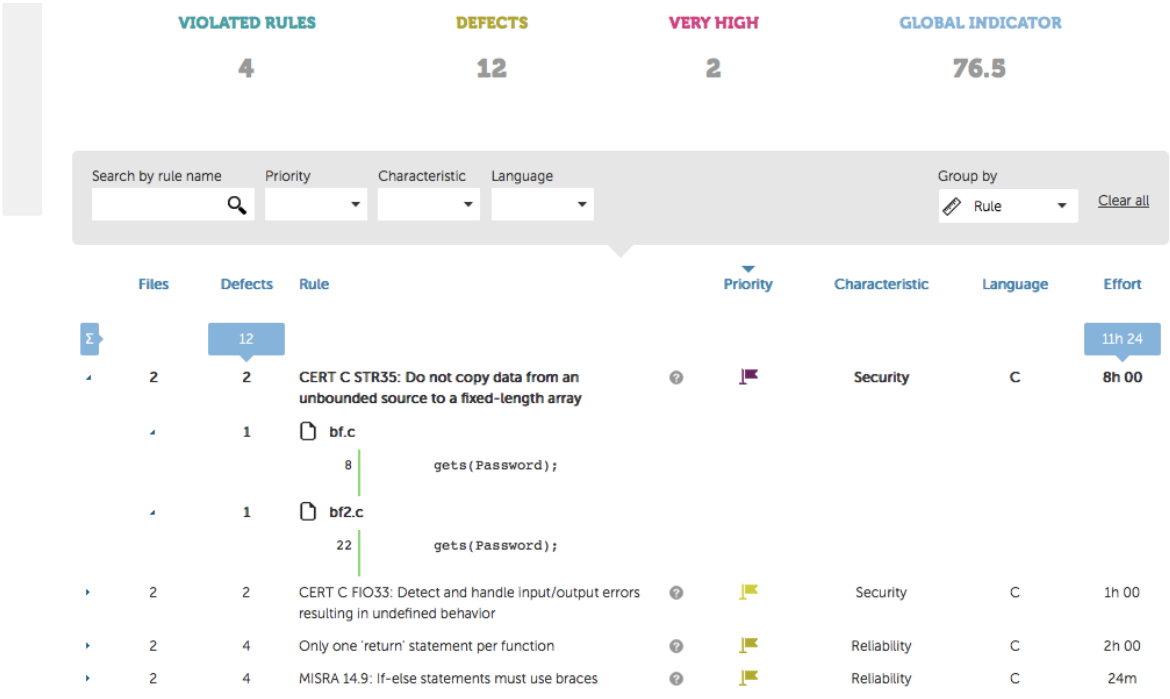
Kiuwan

https://www.kiuwan.com/

Software Analytics in the Cloud

Know the risks, the technical debt, the security flaws, the performance issues… create automatic action plans and analyze their progress. Make better decisions, faster, with Kiuwan Software Analytics!



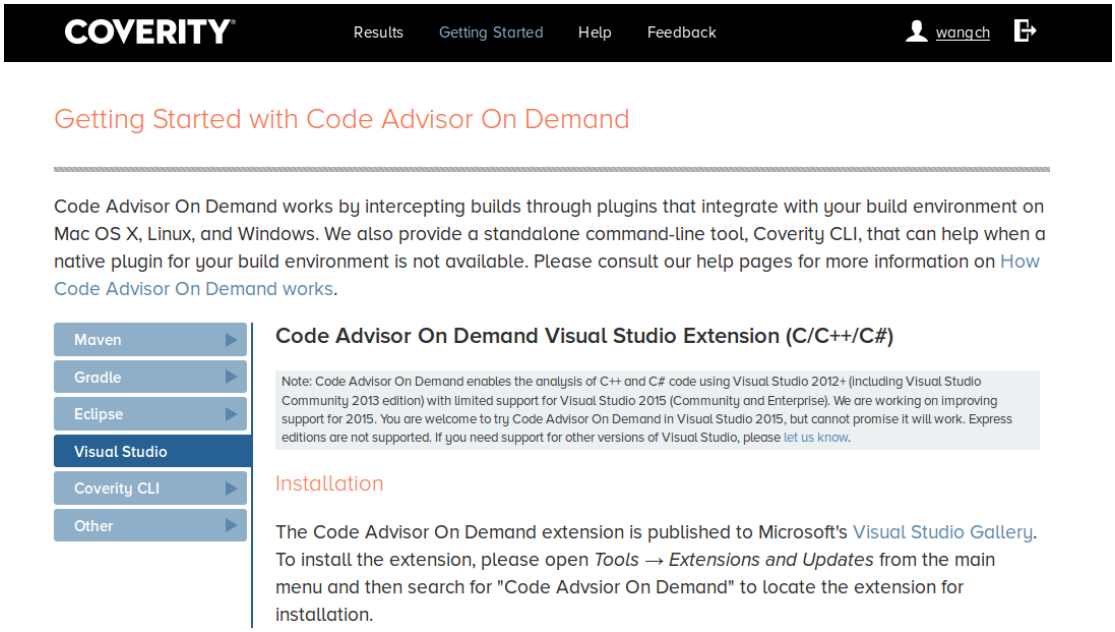


Coverity - Code Advisor On Demand

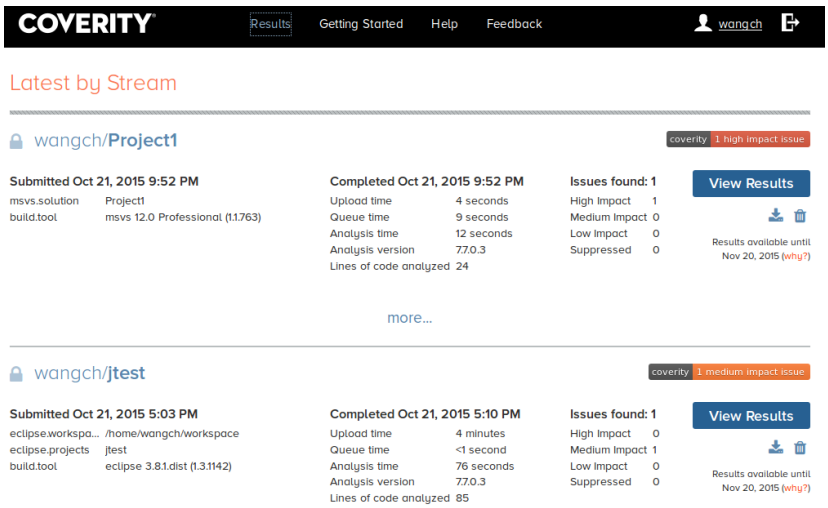
Free Trial (15 days)

Code Advisor On Demand works by intercepting builds through plugins that integrate with your build environment on Mac OS X, Linux, and Windows. We also provide a standalone command-line tool, Coverity CLI, that can help when a native plugin for your build environment is not available. Please consult our help pages for more information on How Code Advisor On Demand works.

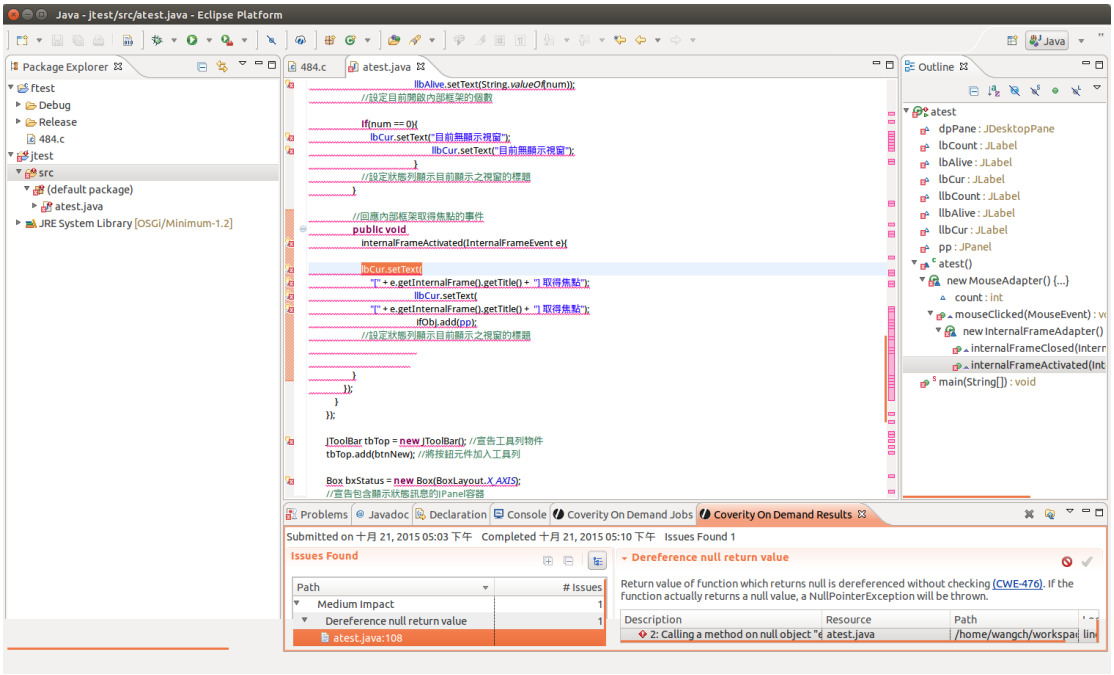
Also can be used in eclipse (Java) and visual studio (C/C++/C#)



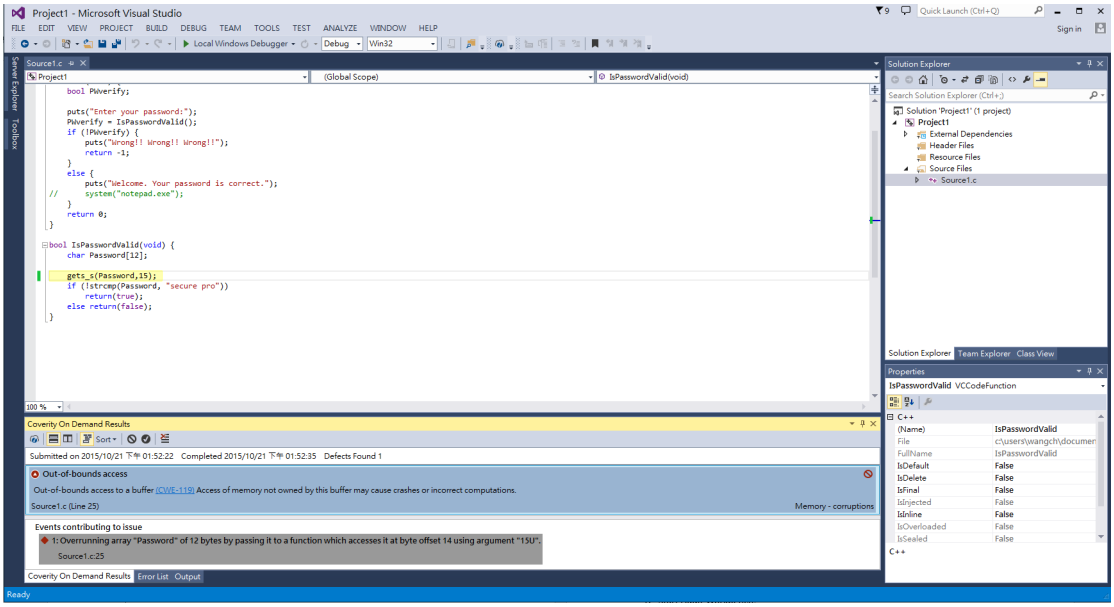
Results



Used for eclipse



Used for visual studio 2013



CWE (Common Weakness Enumeration) https://cwe.mitre.org/

CWE™ International in scope and free for public use, CWE provides a unified, measurable set of software weaknesses that is enabling more effective discussion, description, selection, and use of software security tools and services that can find these weaknesses in source code and operational systems as well as better understanding and management of software weaknesses related to architecture and design.

PVS-Studio

PVS-Studio is a static analyzer that detects errors in source code of C/C++ applications. There are sets of rules included into PVS-Studio:

- General-purpose diagnosis
- Detection of possible optimizations
- Diagnosis of 64-bit errors (Viva64)

http://www.viva64.com/

Text-based HEX Editor

hexedit

F2: save

F3: load file

Ctrl-X: save and exit

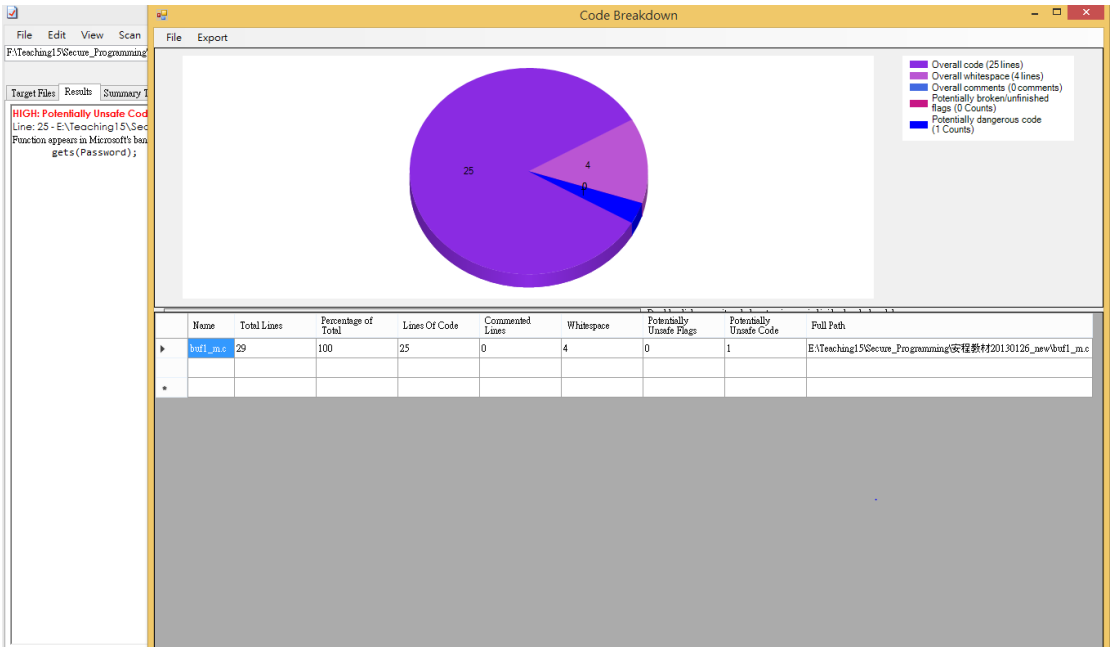
Ctrl-C: exit without saving

Esc+T - truncate the file at the current location

VCG

Visual Codegrepp

http://sourceforge.net/projects/visualcodegrepp/



Other useful links

- NIST Source code security analyzers

https://samate.nist.gov/index.php/Source_Code_Security_Analyzers.html

- SourceMeter https://www.sourcemeter.com/
- Static source code analysis tools for C http://spinroot.com/static/
- (WIKI) List of tools for static code analysis
  https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis
- CPPCHECK http://cppcheck.sourceforge.net/
- PC-lint for C/C++ http://www.gimpel.com/html/index.htm

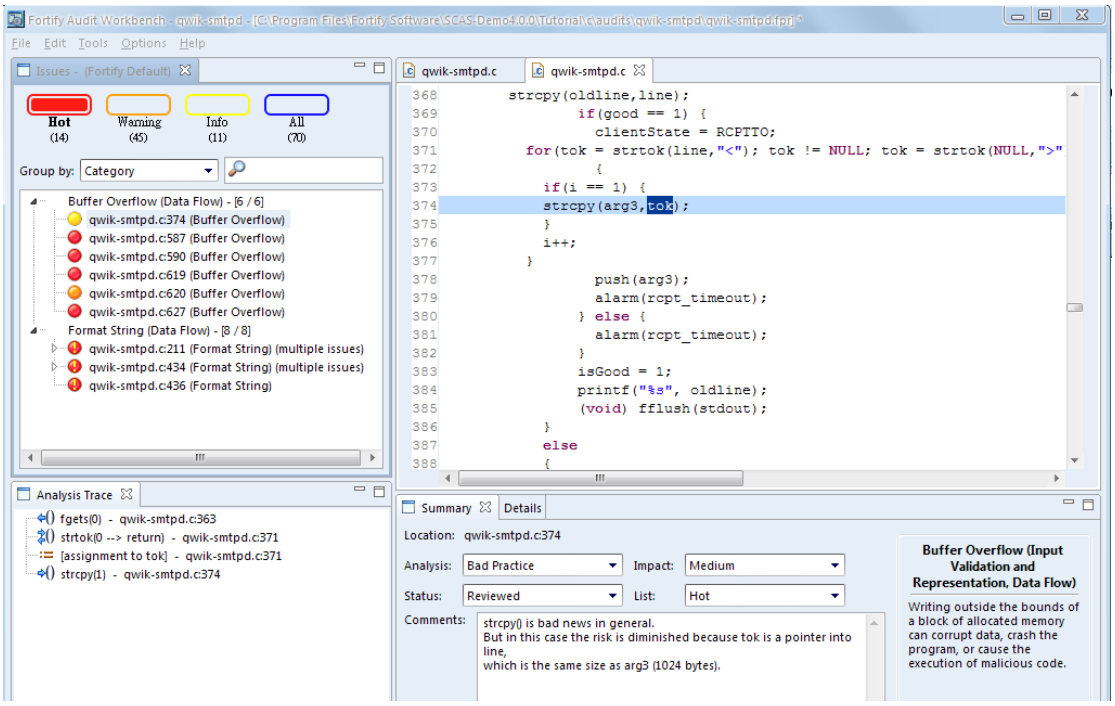
…….

Fortify Software

1. Start Audit Workbench

   Open the audit project:

   <install_dir>/Tutorial/c/audits/qwik-smtpd/qwik-smtpd.fpr



   Read all Hot issues (buffer overflow and format string)

   Generate Audit Report

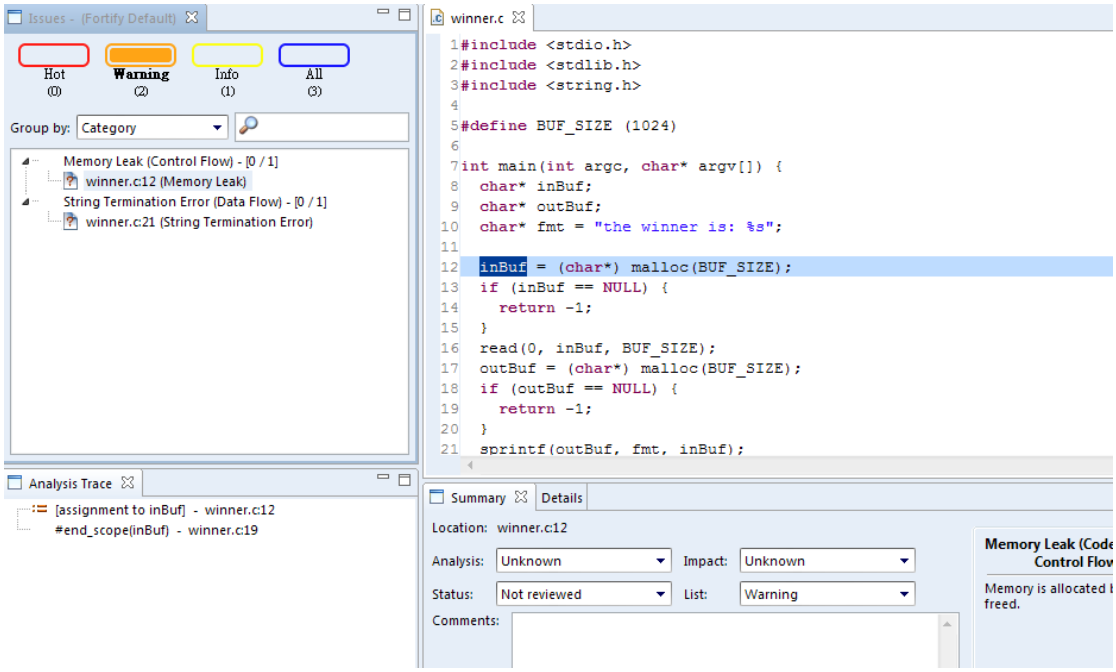
2. Consider the source code for winner.c

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

#define BUF_SIZE (1024)

int main(int argc, char* argv[]) {
    char* inBuf;
    char* outBuf;
    char* fmt = "the winner is: %s";
```

```
    inBuf = (char*) malloc(BUF_SIZE);
    if (inBuf == NULL) {
        return -1;
    }
    read(0, inBuf, BUF_SIZE);
    outBuf = (char*) malloc(BUF_SIZE);
    if (outBuf == NULL) {
        return -1;
    }
    sprintf(outBuf, fmt, inBuf);
    fprintf(stdout, "%s\n", outBuf);
    fprintf(stderr, "%s\n", outBuf);
    free(inBuf);
    free(outBuf);
}
```

Answer the question: How is this code vulnerable to attack?

Open the file of winner.fpr



Read all security issues

Try to review and make comments!

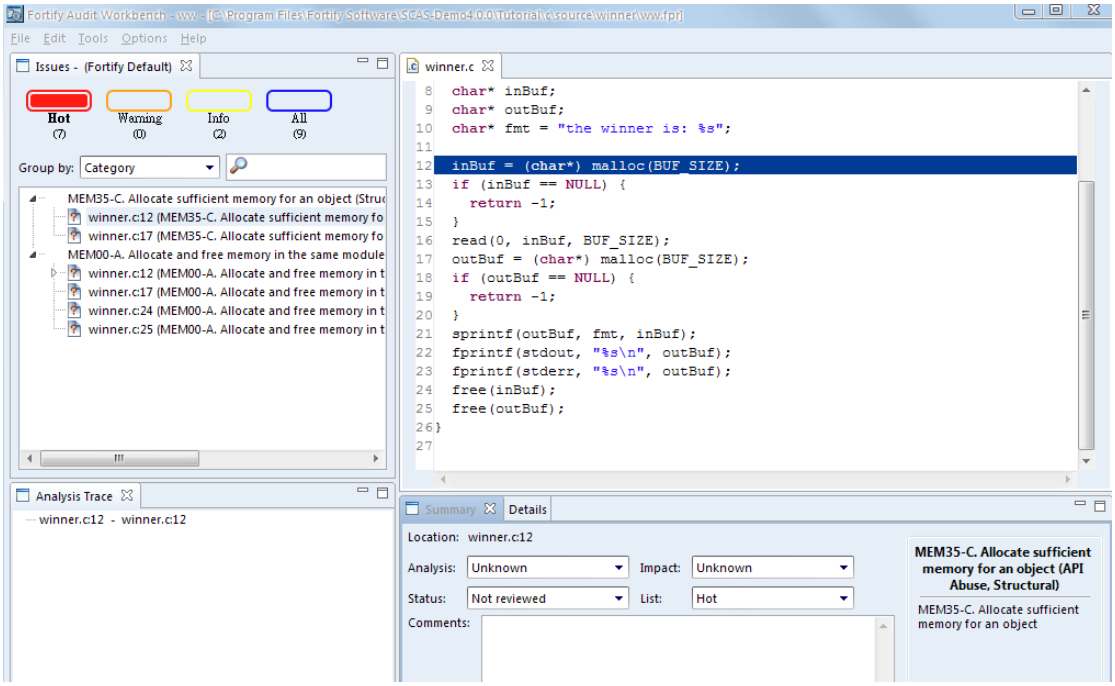
3. Running Fortify SCA

Using rules of CSCS-C.xml (CERT C Rule Pack)

sourceanalyzer –f ww.fpr –rules CSCS-C.xml gcc winner.c

Open ww.fpr with Audit Workbench

Note: You much install gcc (or cygwin)



4. Test other codes

(1) Test.c from flawfinder homepage

Use flawfinder and splint

(2) Sort.c (a textbook example)