Flawfinder – A Tool Analysis for Determining Security Weakness

Software Testing

Presentation for Exercise

Brandenburg
University of Technology
Cottbus - Senftenberg

Siddique Reza Khan
MSc in Cyber Security
Matrikel-Nr.:3846259

21.01.2019

Supervisor: Prof. Monika Heiner

Introduction

- A simple tool
  - Examines C/C++ source code
  - Reports possible security weaknesses (“flaws”) sorted by risk level
- Very useful
  - Quickly finding and
  - Removing at least some potential security problems
- Open source software(OSS)
  - Publically available to use
- Flawfinder works on
  - GNU/Linux systems and
  - Windows by using Cygwin.
  - It requires Python 2.7 or Python 3 to run

Tool Comparison

- Flawfinder:
  - Scans C/C++ code
  - Database of 128 C/C++ vulnerabilities.
  - Reporting warnings of 6 levels, form level 0 to 5,
- ITS4: “It's The Software Stupid! Security Scanner”.
  - Scans C/C++ code
  - Database has 144 vulnerabilities.
  - 6 levels of reporting warnings, level 0 to 5
- RATS: “Rough Automatic Tool for Security”.
  - Scans C/C++, Python, PHP and Perl source code.
  - Rats has 3 levels of reporting warnings, Low, Medium and High

Downloading and Installing

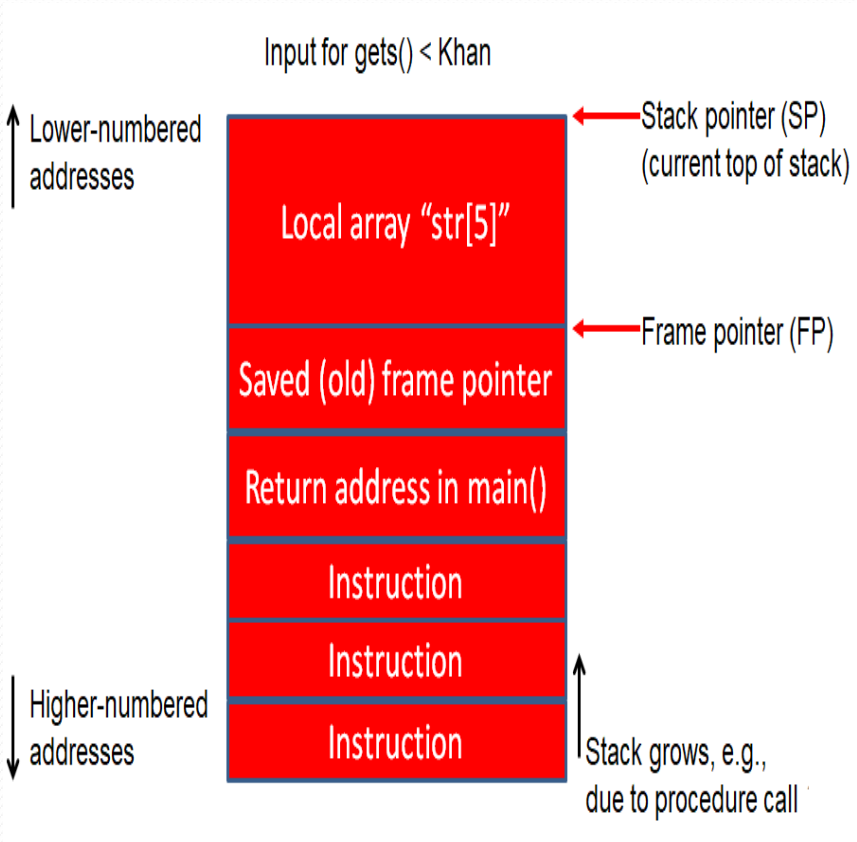
- The current version - 2.0.7
- Linux-based system
- First, download flawfinder in .tar.gz (tarball) format
  - https://dwheeler.com/flawfinder/flawfinder-2.0.7.tar.gz
- Install
  - First, uncompress the file
  - Become root to install:
    - tar xvzf flawfinder-*.tar.gz
    - cd flawfinder-*
  - Then install we can do this (omit "sudo" if you are root)
    - 1. sudo make prefix=/usr install
- If you use Windows, the recommended way is to install Cygwin first, and install it on top of Cygwin.

Speed

- Flawfinder is written in Python
- Python code is not as fast as C code
  1. Averaged analysis speed of 45,126 lines/second
     - OS - Linux kernel
     - CPU - Intel Core 2 Duo CPU E8400
     - Speed - 3.00GHz (each CPU running at 2GHz)
  2. Averaged 24,475 lines/second
     - OS – Windows
     - Environment - 2.8GHz laptop and Cygwin
     - Cygwin on Windows tends to be much slower than Linux

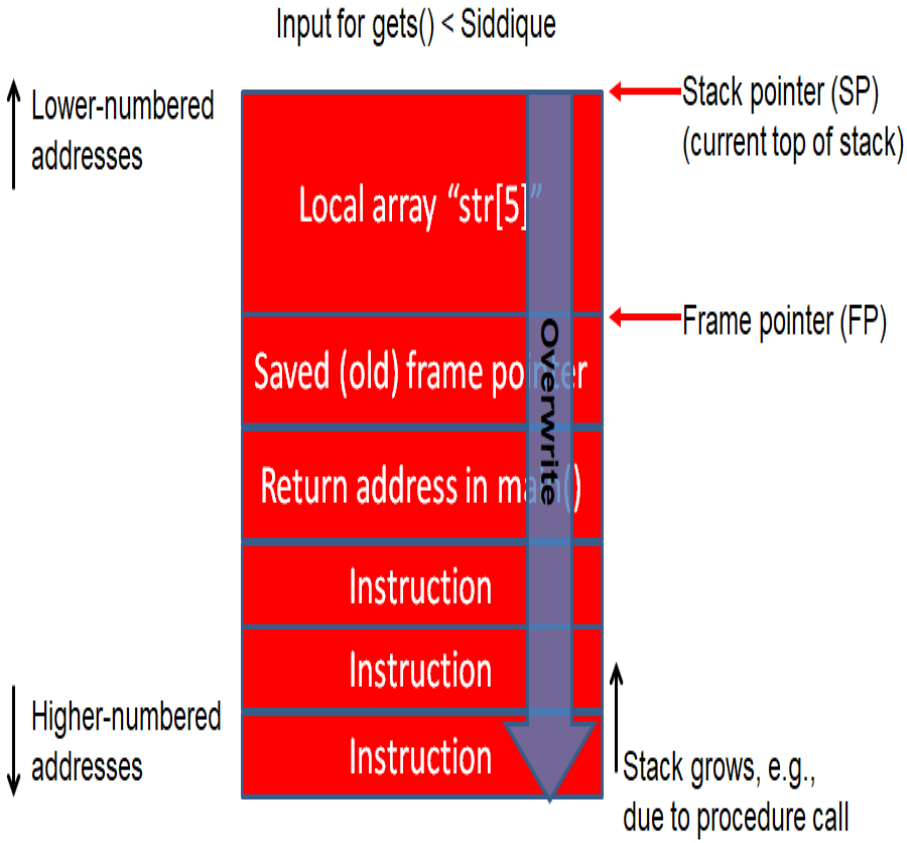
Buffer Overflow Risk

Programmer



Correct output:
Enter a value : Khan
You entered: Khan

Attacker



Program crash:
Enter a value: Siddique
Segmentation Fault

How does Flawfinder Work?(Cont. 1/2)

Programmer

Flawfinder Tool

```
/** Database*/
/** C library functions considered Harmful
*/
strcpy() // Buffer overflow risks
strcat() // Buffer overflow risks
gets() // Buffer overflow risks
sprintf() // Buffer overflow risks
scanf() // Buffer overflow risks
printf() // Format string problems
snprintf() // Format string problems
system() // shell metacharacter dangers
popen()// shell metacharacter dangers

/** Database of 128 C/C++ vulnerabilities
*/
```

Match found by Lexical Analysis, so, 1 hit

Attacker

```
/** Vulnerable Program */
#include <stdio.h>
//statements
void getName()
{   char str[5];
   printf( "Enter your name :");
   gets( str );
   printf( "\nYou entered: ");
   puts( str );
}

int main( )
{
   //Statements
   getName() /** Function Call */
   //Statements
    return 0;
}
```

Program crash:
Enter a value: Siddique
Segmentation Fault

Correct output:
Enter a value : Khan
You entered: Khan

How does Flawfinder Work?(cont. 2/2)

- Analyze software C/C++ filename extensions
  - Needn't build in some cases and even locally compile
- Physical source lines of code (SLOC) analyzed
- Built-in database of C/C++ functions
- Matches the source code against names of library function
- Ignoring text inside comments except for flawfinder directives
  - // Flawfinder: ignore or /* Flawfinder: ignore */
- Produces a list of “hits” sorted by risk,
  - Example "[0] 0[1] 9" means, at level 0 total 0 hits reported, and at level 1 total 9 hits reported; level 3+ has the sum of the number of hits at level 3, 4, and 5; KSLOC is each "level or higher" values multiplied by 1000 and divided by the physical SLOC

Built-in database

- Built-in database of C/C++ functions
  - Buffer overflow risks (e.g., strcpy(), strcat(), gets(), sprintf(), and the scanf() family),
  - Format string problems (printf(), snprintf(), and syslog()),
  - Race conditions (such as access(), chown(), chgrp(), chmod(), tmpfile(), tmpnam(), tempnam(), and mktemp()),
  - Potential shell metacharacter dangers (most of the exec() family, system(), popen()), and
  - Poor random number acquisition (such as random())

False Positive

- During audited a program
  - Can mark source code lines that are actually fine but cause false warnings
- Fundamentally a naive program
  - Doesn't know about the data types of function parameters
  - Doesn't do control flow or data flow analysis
- Important reminders:
  - Not every hit is necessarily a security vulnerability
  - There may be other security vulnerabilities not reported by the tool.

Flawfinder Options

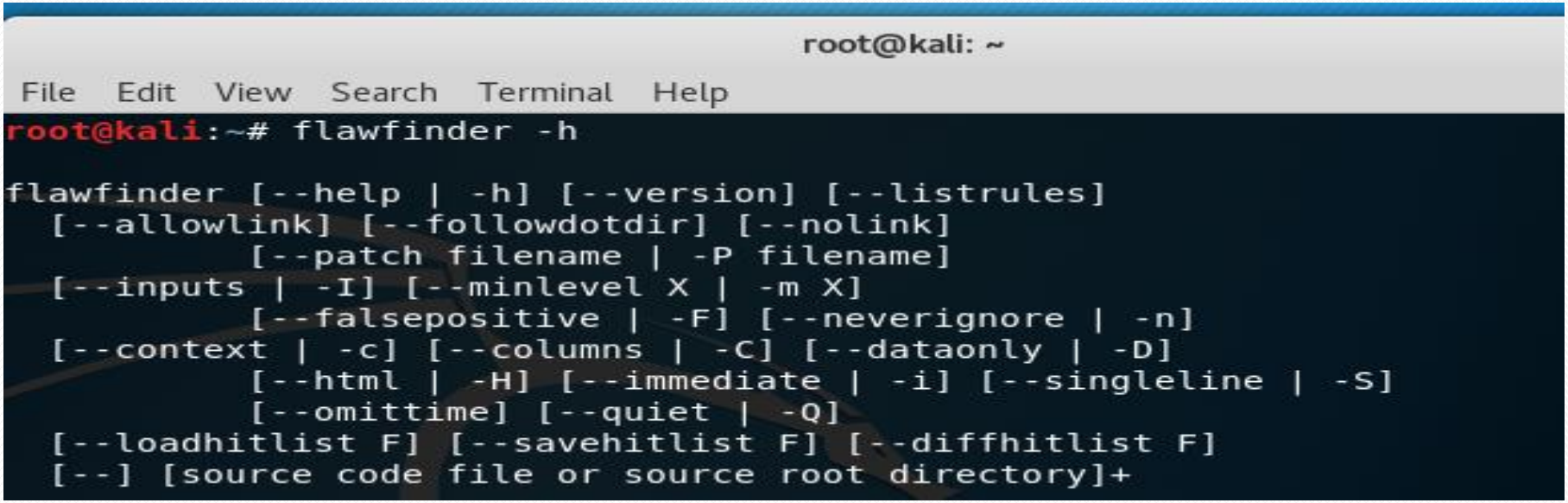
- Number of command line options
- Grouped into options
  - Control its own documentation
  - Select which hits to display
  - Select the output format, and
  - Perform hit list management
- Flawfinder can identify a program input in command line
  - flawfinder TestProgram.c|less
- Long option arguments
  - Provided as “--name=value” or “-name value”.

Options : Documentation (Cont. 1/4)

- ––help or –h
  - Show usage (help) information.
- ––version
  - Shows (just) the version number and exits.
- ––listrules
  - List the terms (tokens) that trigger further examination
  - Default risk level, and
  - Default warning (including the CWE identifier(s), if applicable), all tab-separated.
  - Called potentially-dangerous functions.

Options : Documentation (Cont. 2/4)

- Show usage (help) information
  - root@kali:~# flawfinder -h



Options : Documentation (Cont. 3/4)

- Shows (just) the version number and exits.
  - root@kali:~# flawfinder --version

Options : Documentation (Cont. 4/4)

- List the terms (tokens) that trigger further examination
  - root@kali:~# flawfinder --listrules

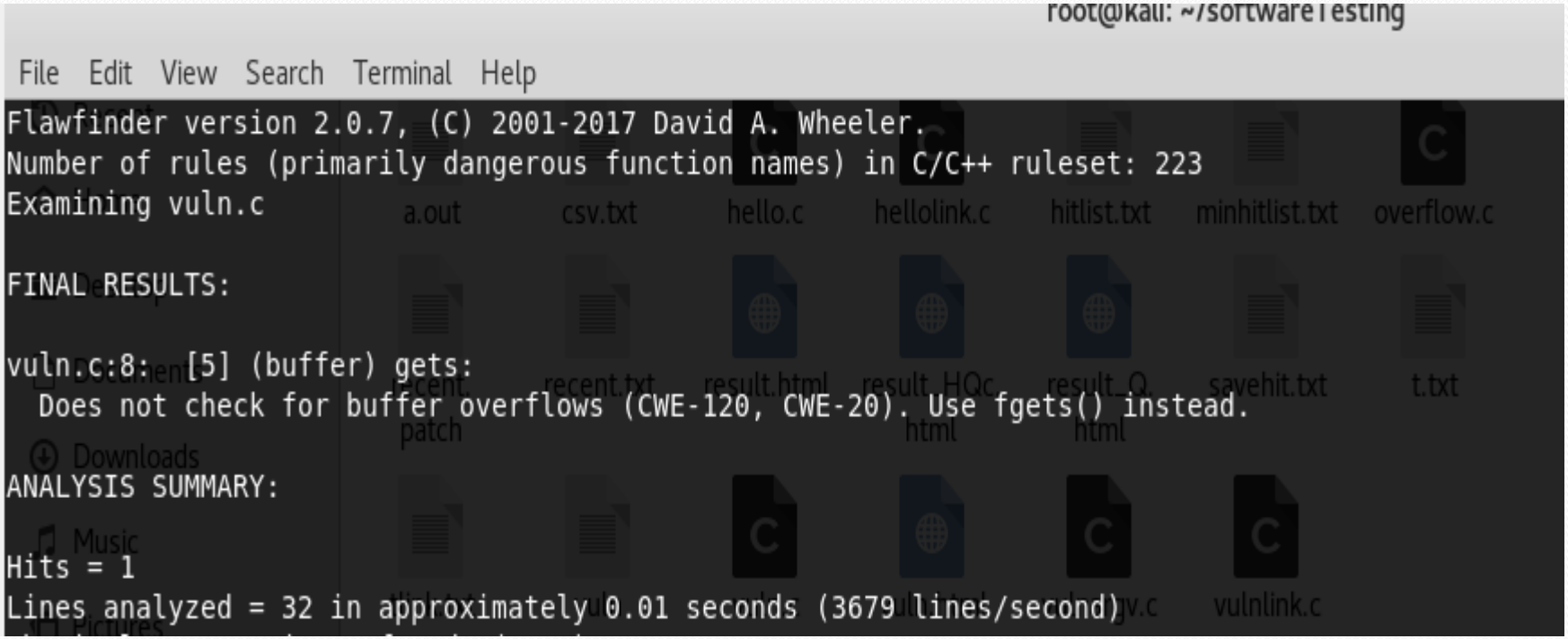
```
root@kali:~# flawfinder --listrules
Flawfinder version 2.0.7, (C) 2001-2017 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
```

Options : Hits to Display (Cont. 1/6)

- ––inputs or –I
  - Show only functions that obtain data from outside the program
- ––minlevel=X or -m X
  - Set minimum risk level to X for inclusion in hit list
- ––falsepositive or –F
  - Do not include hits that are likely to be false positives.
- ––neverignore or -n
  - Never ignore security issues, even if they have an “ignore” directive in a comment
- ––regexp=PATTERN or -e PATTERN
  - Only report hits with text that matches the regular expression pattern PATTERN

Options : Hits to Display (Cont. 2/6)

- Show only functions that obtain data from outside the program
  - root@kali:~# flawfinder -I vuln.c| less



Options : Hits to Display (Cont. 3/6)

- Set minimum risk level to X for inclusion in hit list
  - root@kali:~# flawfinder -m 4 vuln.c | less

```
Flawfinder version 2.0.7, (C) 2001-2017 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
Examining vuln.c
FINAL RESULTS:

vuln.c:8:  [5] (buffer) gets:
  Does not check for buffer overflows (CWE-120, CWE-20). Use fgets() instead.
vuln.c:18:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
  easily misused).

ANALYSIS SUMMARY:

Hits = 2
Lines analyzed = 32 in approximately 0.01 seconds (3412 lines/second)
Physical Source Lines of Code (SLOC) = 18
Hits@level = [0]   3 [1]   0 [2]   2 [3]   0 [4]   1 [5]   1
Hits@level+ = [0+]   7 [1+]   4 [2+]   4 [3+]   2 [4+]   2 [5+]   1
Hits/KSLOC@level+ = [0+] 388.889 [1+] 222.222 [2+] 222.222 [3+] 111.111 [4+] 111.111 [5+] 55.5556
Minimum risk level = 4
```

Options : Hits to Display (Cont. 4/6)

- Do not include hits that are likely to be false positives
  - root@kali:~# flawfinder -F vuln.c | less

```
Flawfinder version 2.0.7, (C) 2001-2017 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
Examining vuln.c

FINAL RESULTS:

vuln.c:8:  [5] (buffer) gets:
  Does not check for buffer overflows (CWE-120, CWE-20). Use fgets() instead.
vuln.c:18:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
  easily misused).

ANALYSIS SUMMARY:

Hits = 2
Lines analyzed = 32 in approximately 0.01 seconds (4690 lines/second)
Physical Source Lines of Code (SLOC) = 18
Hits@level = [0]   3 [1]   0 [2]   0 [3]   0 [4]   1 [5]   1
Hits@level+ = [0+]   5 [1+]   2 [2+]   2 [3+]   2 [4+]   2 [5+]   1
Hits/KSLOC@level+ = [0+] 277.778 [1+] 111.111 [2+] 111.111 [3+] 111.111 [4+] 111.111 [5+] 55.5556
Minimum risk level = 1
```

Options : Hits to Display (Cont. 5/6)

- Never ignore security issues, even if they have an “ignore” directive in a comment
  - root@kali:~# flawfinder -n vuln.c | less

```
FINAL RESULTS:

vuln.c:8:  [5] (buffer) gets:
  Does not check for buffer overflows (CWE-120, CWE-20). Use fgets() instead.
vuln.c:18:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
  easily misused).
vuln.c:6:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
vuln.c:17:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.

ANALYSIS SUMMARY:

Hits = 4
Lines analyzed = 32 in approximately 0.01 seconds (5109 lines/second)
Physical Source Lines of Code (SLOC) = 18
Hits@level = [0]   3 [1]   0 [2]   2 [3]   0 [4]   1 [5]   1
Hits@level+ = [0+]   7 [1+]   4 [2+]   4 [3+]   2 [4+]   2 [5+]   1
Hits/KSLOC@level+ = [0+] 388.889 [1+] 222.222 [2+] 222.222 [3+] 111.111 [4+] 111.111 [5+] 55.5556
Minimum risk level = 1
```

Options : Hits to Display (Cont. 6/6)

- Only report hits with text that matches the regular expression pattern PATTERN
  - root@kali:~/softwareTesting# flawfinder -e CWE-20 vuln.c | less

```
FINAL RESULTS:

vuln.c:8:  [5] (buffer) gets:
  Does not check for buffer overflows (CWE-120, CWE-20). Use fgets() instead.

ANALYSIS SUMMARY:

Hits = 1
Hits limited to regular expression CWE-20
Lines analyzed = 33 in approximately 0.01 seconds (4901 lines/second)
Physical Source Lines of Code (SLOC) = 18
Hits@level = [0]   0 [1]   0 [2]   0 [3]   0 [4]   0 [5]   1
Hits@level+ = [0+]   1 [1+]   1 [2+]   1 [3+]   1 [4+]   1 [5+]   1
Hits/KSLOC@level+ = [0+] 55.5556 [1+] 55.5556 [2+] 55.5556 [3+] 55.5556 [4+] 55.5556 [5+] 55.5556
Minimum risk level = 1
```

Options : Output Format (Cont. 1/4)

- ––columns or –C
  - Show the column number (as well as the file name and line number) of each hit;
- ––context or –c
  - Show context, i.e., the line having the "hit"/potential flaw. By default the line is shown immediately after the warning.
- ––csv
  - Generate output in comma-separated-value (CSV) format.
  - This is the recommended format for sending to other tools for processing

Options : Output Format (Cont. 2/4)

- Show the column number (as well as the file name and line number) of each hit
  - root@kali:~/softwareTesting# flawfinder -C vuln.c|less

```
Flawfinder version 2.0.7, (C) 2001-2017 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
Examining vuln.c

FINAL RESULTS:

vuln.c:8:2:  [5] (buffer) gets:
  Does not check for buffer overflows (CWE-120, CWE-20). Use fgets() instead.
vuln.c:17:2:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
  easily misused).
vuln.c:6:2:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
vuln.c:16:2:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
```

Options : Output Format (Cont. 3/4)

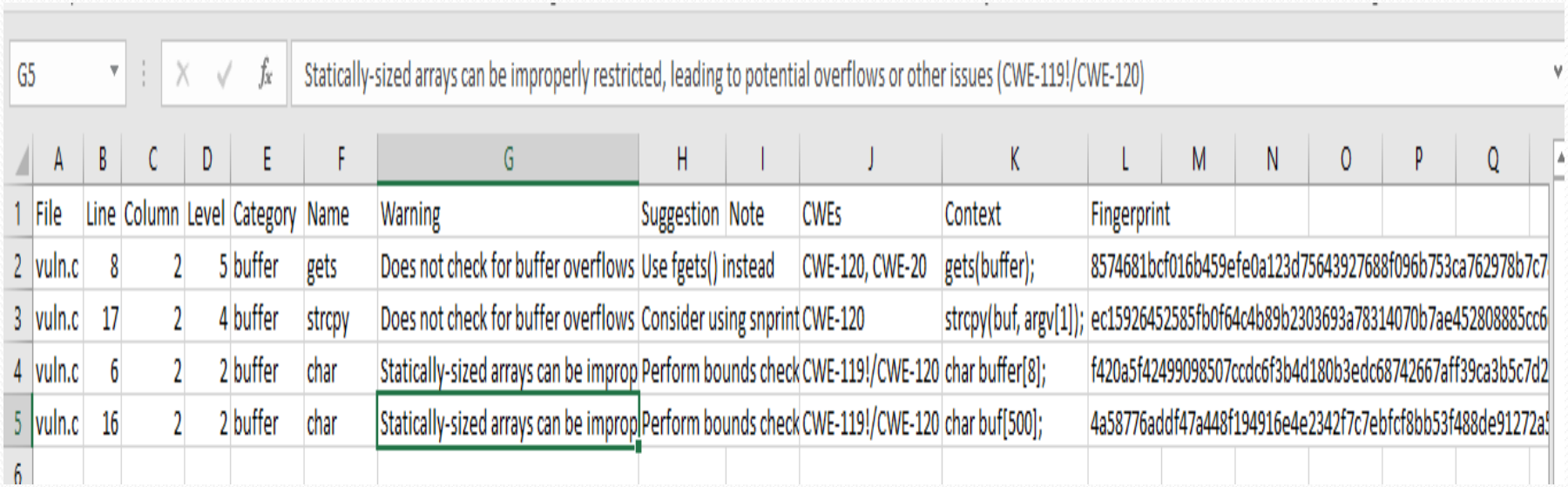
- Show context, i.e., the line having the "hit"/potential flaw. By default the line is shown immediately after the warning.
  - root@kali:~/softwareTesting# flawfinder -c vuln.c|less

```
FINAL RESULTS:

vuln.c:8:  [5] (buffer) gets:
  Does not check for buffer overflows (CWE-120, CWE-20). Use fgets() instead.
        gets(buffer);
vuln.c:17:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
  easily misused).
        strcpy(buf, argv[1]);
vuln.c:6:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
        char buffer[8];
vuln.c:16:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
        char buf[500];
```

Options : Output Format (Cont. 4/4)

- Generate output in comma-separated-value (CSV) format.
  - flawfinder --csv vuln.c >data.csv



G5 | Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119!/CWE-120)

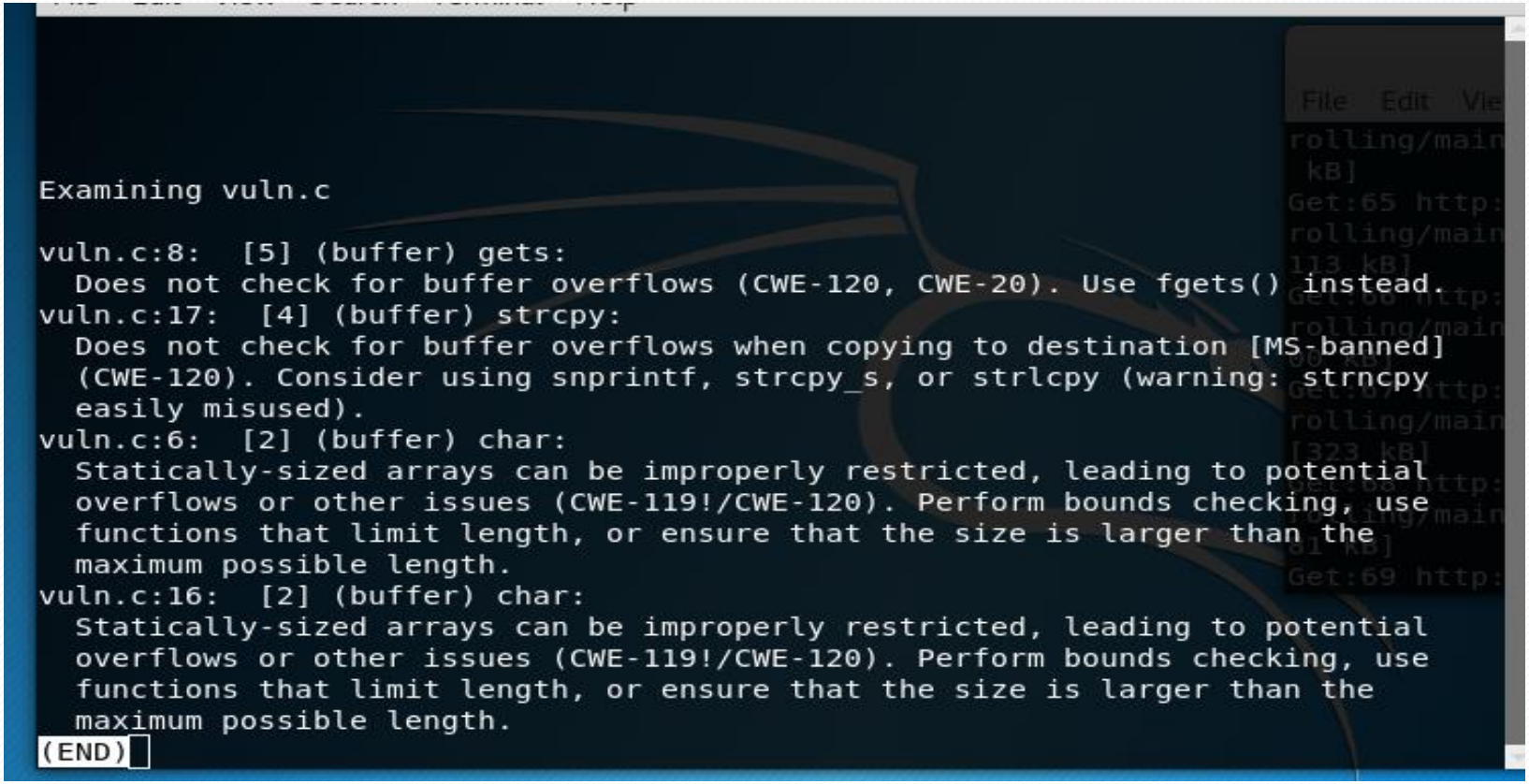
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	File	Line	Column	Level	Category	Name	Warning	Suggestion	Note	CWEs	Context	Fingerprint					
2	vuln.c	8	2	5	buffer	gets	Does not check for buffer overflows	Use fgets() instead		CWE-120, CWE-20	gets(buffer);	8574681bcf016b459efe0a123d75643927688f096b753ca762978b7c7					
3	vuln.c	17	2	4	buffer	strcpy	Does not check for buffer overflows	Consider using snprint		CWE-120	strcpy(buf, argv[1]);	ec15926452585fb0f64c4b89b2303693a78314070b7ae452808885cc6					
4	vuln.c	6	2	2	buffer	char	Statically-sized arrays can be improp	Perform bounds check		CWE-119!/CWE-120	char buffer[8];	f420a5f42499098507ccdc6f3b4d180b3edc68742667aff39ca3b5c7d2					
5	vuln.c	16	2	2	buffer	char	Statically-sized arrays can be improp	Perform bounds check		CWE-119!/CWE-120	char buf[500];	4a58776addf47a448f194916e4e2342f7c7ebfcf8bb53f488de91272a5					
6																	

Options : Output Format (Cont. 1/7)

- –D
  - Don't display the header and footer.
- ––html or –H
  - Format the output as HTML instead of as simple text.
- ––immediate or -i
  - Immediately display hits (don't just wait until the end).
- ––singleline or -S
  - Display as single line of text output for each hit.
- ––omittime
  - Omit timing information.
- ––quiet or –Q
  - Don't display which files are being examined

Options : Output Format (Cont. 2/7)

- Don't display the header and footer.
  - root@kali:~/softwareTesting# flawfinder -D vuln.c|less



Options : Output Format (Cont. 3/7)

- Format the output as HTML instead of as simple text.
  - root@kali:~/softwareTesting# flawfinder -H vuln.c>vuln1.html

Flawfinder Results

Here are the security scan results from Flawfinder version 2.0.7, (C) 2001-2017 David A. Wheeler. Number of rules (primarily dangerous function names) in C/C++ ruleset: 223

- vuln.c:8: **[5]** (buffer) *gets: Does not check for buffer overflows (CWE-120, CWE-20). Use fgets() instead.*

```
gets(buffer);
```

- vuln.c:17: **[4]** (buffer) *strcpy: Does not check for buffer overflows when copying to destination [MS-banned] (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy easily misused).*

```
strcpy(buf, argv[1]);
```

- vuln.c:6: **[2]** (buffer) *char: Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.*

```
char buffer[8];
```

- vuln.c:16: **[2]** (buffer) *char: Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.*

```
char buf[500];
```

Analysis Summary

Hits = 4

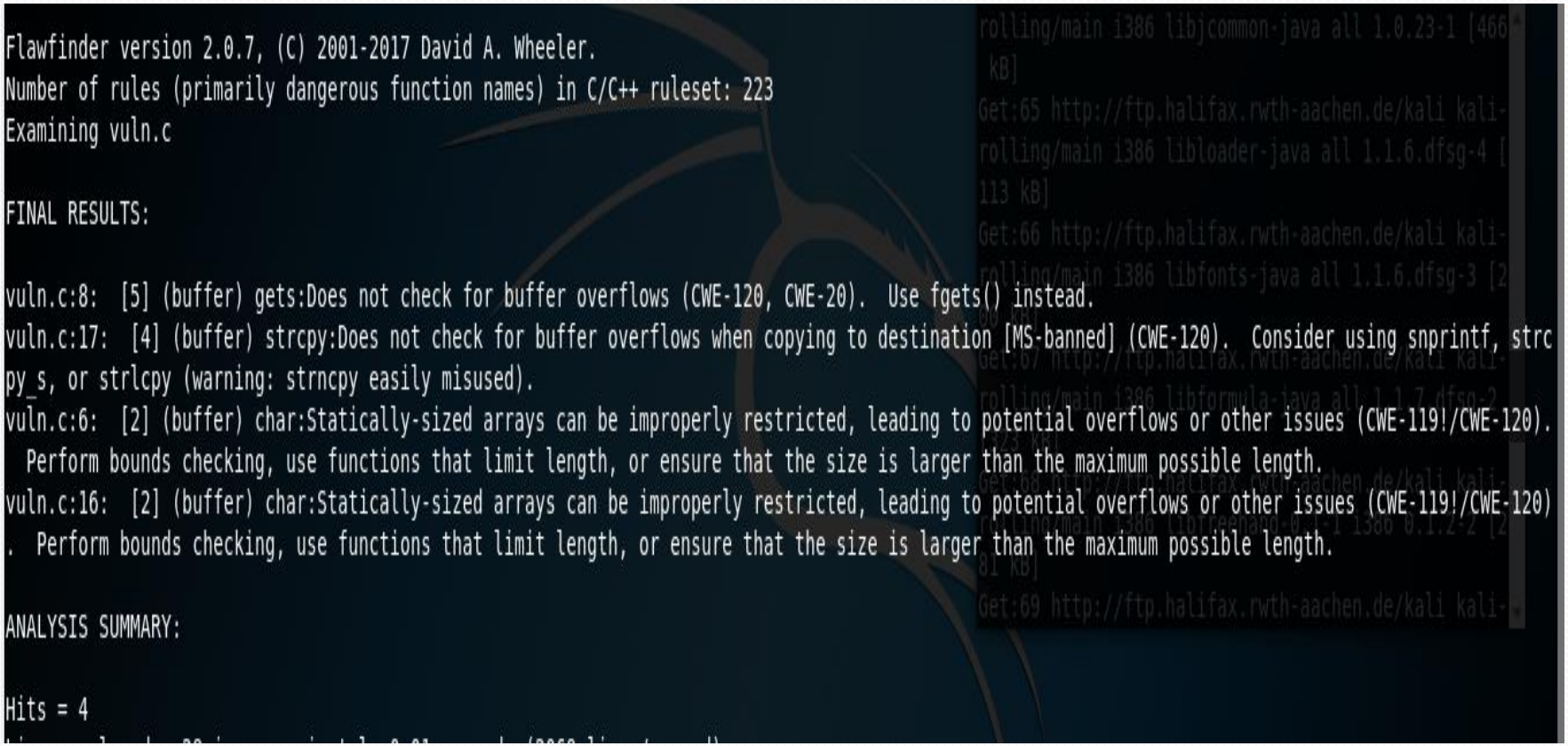
Options : Output Format (Cont. 4/7)

- Immediately display hits
  - root@kali:~/softwareTesting# flawfinder -i vuln.c|less

```
Flawfinder version 2.0.7, (C) 2001-2017 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
Examining vuln.c
vuln.c:6:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
vuln.c:7:  [0] (format) printf:
  If format strings can be influenced by an attacker, they can be exploited
  (CWE-134). Use a constant for the format specification. Constant format
  string, so not considered risky.
vuln.c:8:  [5] (buffer) gets:
  Does not check for buffer overflows (CWE-120, CWE-20). Use fgets() instead.
vuln.c:9:  [0] (format) printf:
  If format strings can be influenced by an attacker, they can be exploited
  (CWE-134). Use a constant for the format specification. Constant format
  string, so not considered risky.
vuln.c:16:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
vuln.c:17:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
  easily misused).
vuln.c:19:  [0] (format) printf:
  If format strings can be influenced by an attacker, they can be exploited
  (CWE-134). Use a constant for the format specification. Constant format
  string, so not considered risky.
```

Options : Output Format (Cont. 5/7)

- Display as single line of text output for each hit.
  - root@kali:~/softwareTesting# flawfinder -S vuln.c|less



Options : Output Format (Cont. 6/7)

- Omit timing information. This is useful for regression tests of flawfinder itself, so that the output doesn't vary depending on how long the analysis takes.
  - root@kali:~/softwareTesting# flawfinder --omittime vuln.c|less

```
ANALYSIS SUMMARY:

Hits = 4
Lines analyzed = 28 in approximately 0.01 seconds (2839 lines/second)
```

```
ANALYSIS SUMMARY:

Hits = 4
Lines analyzed = 28
Physical Source Lines of Code (SLOC) = 18
Hits@level = [0]   3 [1]   0 [2]   2 [3]   0 [4]   1 [5]   1
Hits@level+ = [0+]   7 [1+]   4 [2+]   4 [3+]   2 [4+]   2 [5+]   1
Hits/KSLOC@level+ = [0+] 388.889 [1+] 222.222 [2+] 222.222 [3+] 111.111 [4+] 111.111 [5+] 55.5556
Minimum risk level = 1
Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!
```

Options : Output Format (Cont. 7/7)

- Don't display status information (i.e., which files are being examined) while the analysis is going on.
  - root@kali:~/softwareTesting# flawfinder -Q vuln.c|less

```
Flawfinder version 2.0.7, (C) 2001-2017 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
vuln.c:8:  [5] (buffer) gets:
  Does not check for buffer overflows (CWE-120, CWE-20). Use fgets() instead.
vuln.c:17:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
  easily misused).
vuln.c:6:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
vuln.c:16:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
```

Options : Hitlist Management (Cont. 1/3)

- ––savehitlist=F
  - Save all resulting hits (the "hitlist") to F.
- ––loadhitlist=F
  - Load the hitlist from F instead of analyzing source programs.
    - Warning: Do not load hitlists from untrusted sources (for security reasons).

Options : Hitlist Management (Cont. 2/3)

- Save all resulting hits (the "hitlist") to F
  - root@kali:~/softwareTesting# flawfinder -Q --savehitlist savehit1.txt vuln.c|less

```
Flawfinder version 2.0.7, (C) 2001-2017 David A. Wheeler.
Saving hitlist to savehit1.txt
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
vuln.c:8:  [5] (buffer) gets:
  Does not check for buffer overflows (CWE-120, CWE-20). Use fgets() instead.
vuln.c:17:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
  easily misused).
vuln.c:6:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
vuln.c:16:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.

ANALYSIS SUMMARY:

Hits = 4
:
```

Options : Hitlist Management (Cont. 1/3)

- Load the hitlist from F instead of analyzing source programs
  - root@kali:~/softwareTesting# flawfinder -Q --loadhitlist savehit1.txt|less

```
File  Edit  View  Search  Terminal  Help
Flawfinder version 2.0.7, (C) 2001-2017 David A. Wheeler.
Loading hits from savehit1.txt
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
vuln.c:8:  [5] (buffer) gets:
  Does not check for buffer overflows (CWE-120, CWE-20). Use fgets() instead.
vuln.c:17:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
  easily misused).
vuln.c:6:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
vuln.c:16:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.

ANALYSIS SUMMARY:

Hits = 4
:
```

COMMON WEAKNESS ENUMERATION (CWE)

- It is a community-developed list of common software security weaknesses that can occur in software's architecture, design, code or implementation that can lead to exploitable security vulnerabilities
- Flawfinder is officially CWE-Compatible and can report on the CWEs
- Example:
  - CWE-20: Improper Input Validation
  - CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer (a parent of CWE-120*, so this is shown as CWE-119!/CWE-120)
  - CWE-120: Buffer Copywithout Checking Size of Input ("Classic Buffer Overflow")*
  - CWE-134: Uncontrolled Format String*
  - CWE-190: Integer Overflow or Wraparound*
  - CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ("Race Condition")
  - CWE-676: Use of Potentially Dangerous Function*

Conclusion

- Static analysis Source code scanning tool to find the most common security problems with C/C++ programs
- Easy to understand
- Looking for C function's token(simple lexical tokenization) matches with tool's database
- Searching for potential vulnerabilities or estimating the level of risk
- Report formats can customize
- The application has the ability to take input values as command line arguments
- Doesn't do data flow or control flow or data types analysis
  - Produce many false positives for vulnerabilities
  - Fail to report many vulnerabilities

Tool Demonstration

C/C++ Code : For Tool Demonstration

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

void GetInput()
{
    char buffer[8];
    printf("Give input for function \"gets() \" testing buffer: ");
    gets(buffer);
    printf("Function \"gets() \" testing buffer: ");
    puts(buffer);
}


int main (int argc, char** argv)
{
    char buf[500];
    strcpy(buf, argv[1]);
    GetInput();
    printf("Function \"strcpy() \" testing buffer:  %s\n", buf);
    return 0;
}

//Comment:
//----------------------------------------
    /*Flawfinder:ignore*/

    // Flawfinder: ignore
//----------------------------------------
```

Options : Demonstration (Cont. 1/2)

- flawfinder -m 4 vuln.c | less
  - Examine file only report vulnerabilities level 4 and up
- flawfinder -I vuln.c| less
  - Examine file to report the functions that take inputs
- flawfinder -n vuln.c | less
  - Examine file to report including even the hits marked for ignoring in the code comments
- flawfinder --csv vuln.c >csvData.csv
  - Examine file to report all hits in CSV format
- flawfinder -QD vuln.c | less
  - Examine file to report only the actual results (removing the header and footer of the output).

Options : Demonstration (Cont. 2/2)

- flawfinder -QDSC vuln.c|less
  - Examine file to report only the actual results
- flawfinder -QHc vuln.c>result.html
  - Examine the file and produce HTML formatted results
- flawfinder -Q --savehitlist savehit.txt vuln.c|less
  - Examine file save the resulting hitlist in savehit.txt file
- flawfinder --loadhitlist savehit.txt vuln.c|less
  - Examine file and show hits that were already in the file savehit.txt
- flawfinder --regex "CWE-119|CWE-120" vuln.c | less
  - Examine file, but only report hits where CWE-119 or CWE-120 apply.

Bibliography

- [1] David A. Wheeler, “Secure Programming HOWTO”, v3.72 Edition, 2015.
- [2]Online: https://dwheeler.com/flawfinder/, “Flawfinder”, accessed on: 02.11.2018
- [3]Online:https://www.debian.org/security/audit/examples/flawfinder, “Automated Audit Example: flawfinder”, accessed on: 02.11.2018
- [4]Online: https://dwheeler.com/secure-class/index.html, “Secure Software Design and Programming: Class Materials by David A. Wheeler”, accessed on: 02.11.2018
- [5]Online:http://manpages.ubuntu.com/manpages/bionic/man1/flawfinder.1.html, “flawfinder - lexically find potential security flaws ("hits") in source code ”, accessed on: 02.11.2018
- [6]Online: https://dwheeler.com/flawfinder/flawfinder.pdf, ”Documentation - Flawfinder ”, accessed on: 02.11.2018
- [7]Online: Daniel Persson, Dejan Baca, Software Security Analysis - Managing source code audit, https://www.diva-portal.org/smash/get/diva2:830925/FULLTEXT01.pdf, accessed on: 19.12.2018
- [8]Online: http://cwe.mitre.org/top25/ , “2011 CWE/SANS Top 25 Most Dangerous Software Errors” , accessed on: 07.01.2019
- [9] Online: https://www.sans.org/reading-room/whitepapers/securecode/secure-software-development-code-analysis-tools-389, “Secure Software Development and Code Analysis Tools”, accessed on: 19.01.2019

Questions?