ATTACK DEFENSE

by PentesterAcademy

Name	Flawfinder: Statically Scanning C code
URL	https://www.attackdefense.com/challengedetails?cid=2045
Type	DevSecOps: Static Code Analysis

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

Flawfinder performs static code analysis on the C/C++ based applications and reports the vulnerabilities found.

A Kali CLI machine (kali-cli) is provided to the user with the flawfinder installed on it. The source code for three sample web applications is provided in the home directory of the root user.

Objective: Scan the source code using FlawFinder utility and find the security issues!

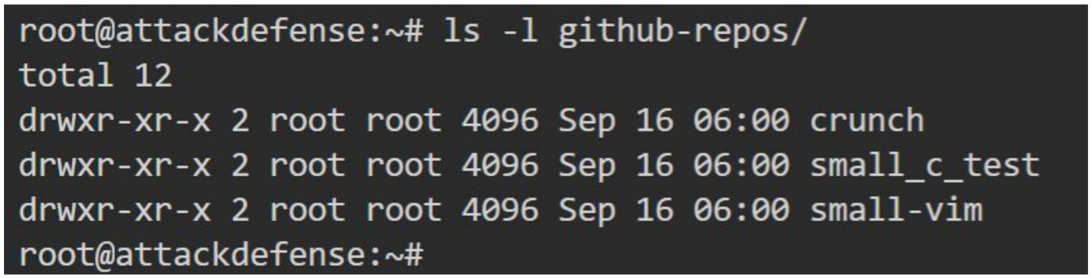
Instructions:

- The source code of web applications is provided at /root/github-repos

Solution

Step 1: Check the provided web applications.

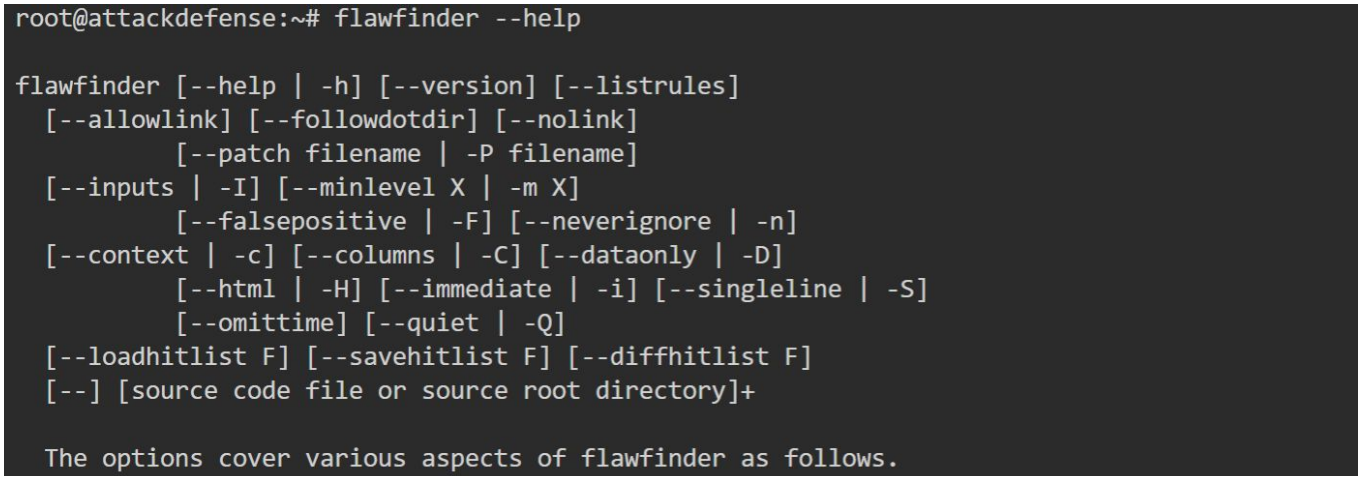
Command: ls -l github-repos/



```
root@attackdefense:~# ls -l github-repos/
total 12
drwxr-xr-x 2 root root 4096 Sep 16 06:00 crunch
drwxr-xr-x 2 root root 4096 Sep 16 06:00 small_c_test
drwxr-xr-x 2 root root 4096 Sep 16 06:00 small-vim
root@attackdefense:~#
```

Step 2: Check the available options for the flawfinder tool

Command: flawfinder --help



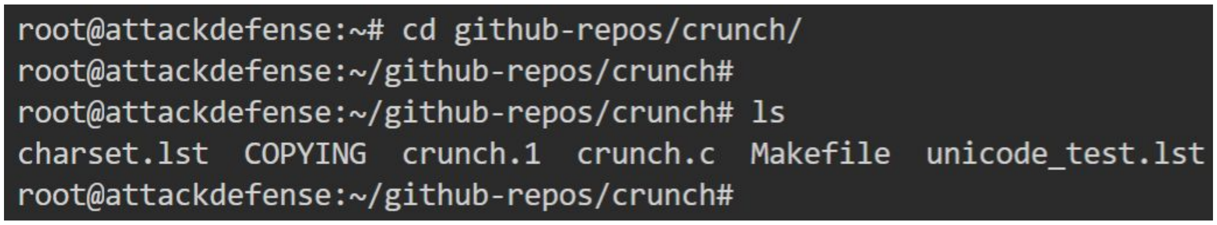
```
root@attackdefense:~# flawfinder --help

flawfinder [--help | -h] [--version] [--listrules]
  [--allowlink] [--followdotdir] [--nolink]
           [--patch filename | -P filename]
  [--inputs | -I] [--minlevel X | -m X]
           [--falsepositive | -F] [--neverignore | -n]
  [--context | -c] [--columns | -C] [--dataonly | -D]
           [--html | -H] [--immediate | -i] [--singleline | -S]
           [--omittime] [--quiet | -Q]
  [--loadhitlist F] [--savehitlist F] [--diffhitlist F]
  [--] [source code file or source root directory]+

  The options cover various aspects of flawfinder as follows.
```

We will take one example at a time and run the tool on that.

Example 1: crunch

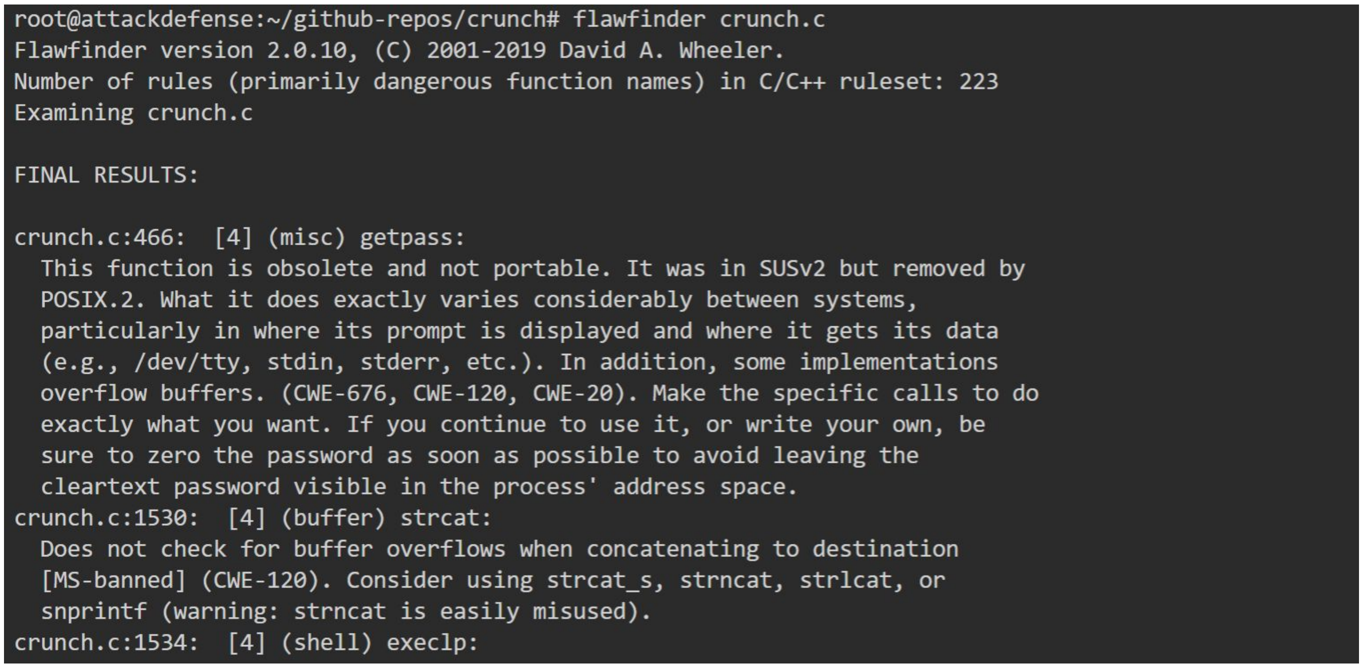
Step A: Change to the crunch repository

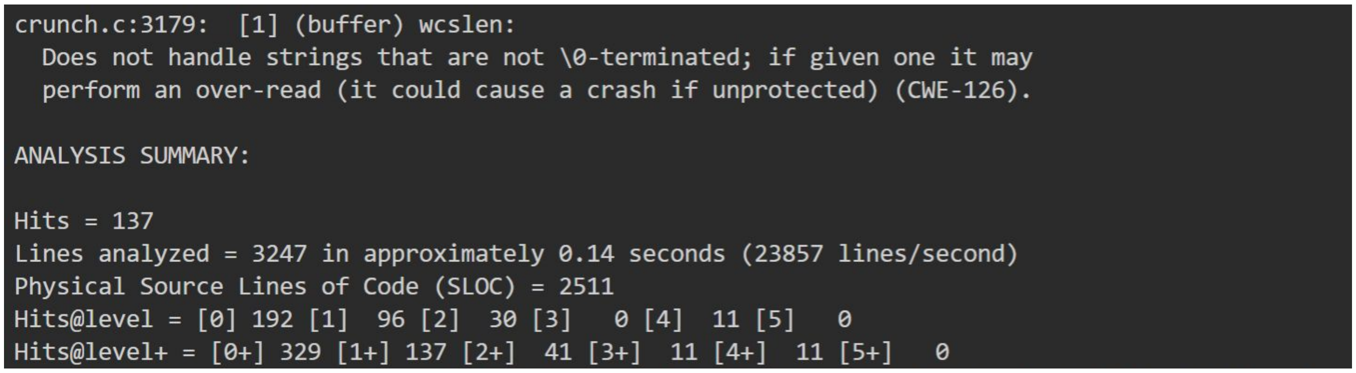
Commands:
cd github-repos/crunch
ls

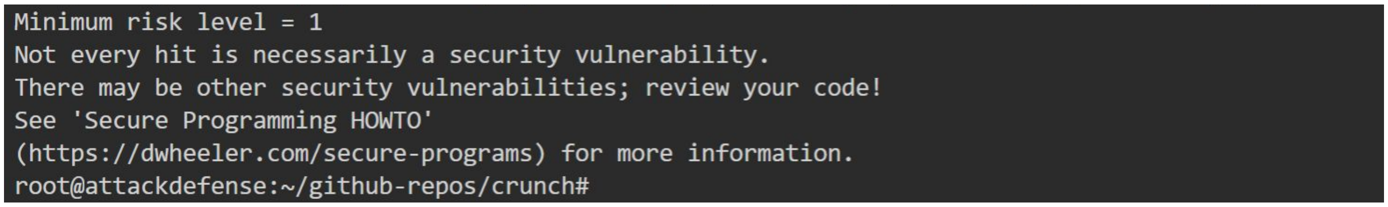

```
root@attackdefense:~# cd github-repos/crunch/
root@attackdefense:~/github-repos/crunch#
root@attackdefense:~/github-repos/crunch# ls
charset.lst  COPYING  crunch.1  crunch.c  Makefile  unicode_test.lst
root@attackdefense:~/github-repos/crunch#
```

Step B: Run the flawfinder tool to find possible flaws in the explicitly specified C code.

Command: flawfinder crunch.c


```
root@attackdefense:~/github-repos/crunch# flawfinder crunch.c
Flawfinder version 2.0.10, (C) 2001-2019 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
Examining crunch.c

FINAL RESULTS:

crunch.c:466:  [4] (misc) getpass:
  This function is obsolete and not portable. It was in SUSv2 but removed by
  POSIX.2. What it does exactly varies considerably between systems,
  particularly in where its prompt is displayed and where it gets its data
  (e.g., /dev/tty, stdin, stderr, etc.). In addition, some implementations
  overflow buffers. (CWE-676, CWE-120, CWE-20). Make the specific calls to do
  exactly what you want. If you continue to use it, or write your own, be
  sure to zero the password as soon as possible to avoid leaving the
  cleartext password visible in the process' address space.
crunch.c:1530:  [4] (buffer) strcat:
  Does not check for buffer overflows when concatenating to destination
  [MS-banned] (CWE-120). Consider using strcat_s, strncat, strlcat, or
  snprintf (warning: strncat is easily misused).
crunch.c:1534:  [4] (shell) execlp:
```


```
crunch.c:3179:  [1] (buffer) wcslen:
  Does not handle strings that are not \0-terminated; if given one it may
  perform an over-read (it could cause a crash if unprotected) (CWE-126).

ANALYSIS SUMMARY:

Hits = 137
Lines analyzed = 3247 in approximately 0.14 seconds (23857 lines/second)
Physical Source Lines of Code (SLOC) = 2511
Hits@level = [0] 192 [1]  96 [2]  30 [3]   0 [4]  11 [5]   0
Hits@level+ = [0+] 329 [1+] 137 [2+]  41 [3+]  11 [4+]  11 [5+]   0
```


```
Minimum risk level = 1
Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://dwheeler.com/secure-programs) for more information.
root@attackdefense:~/github-repos/crunch#
```

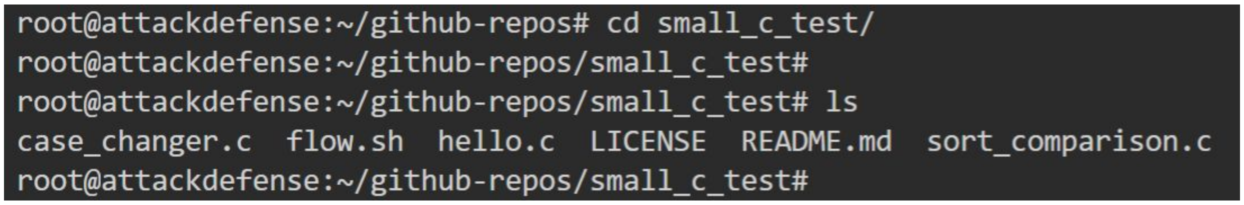
Issues Detected

- Buffer Overflows

Example 2: small c test

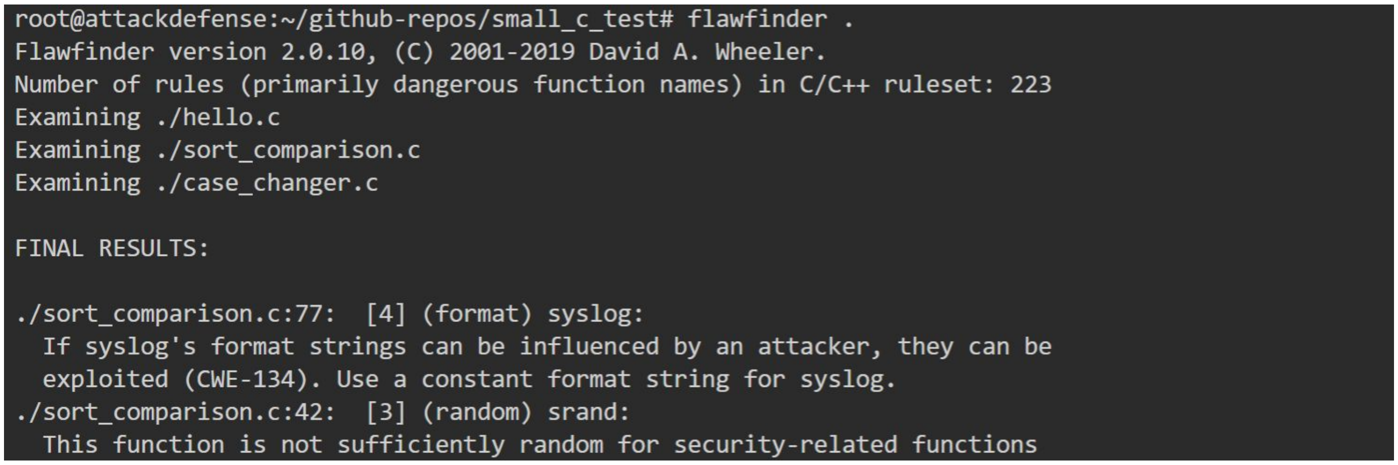
Step A: Change to the small_c_test repository.

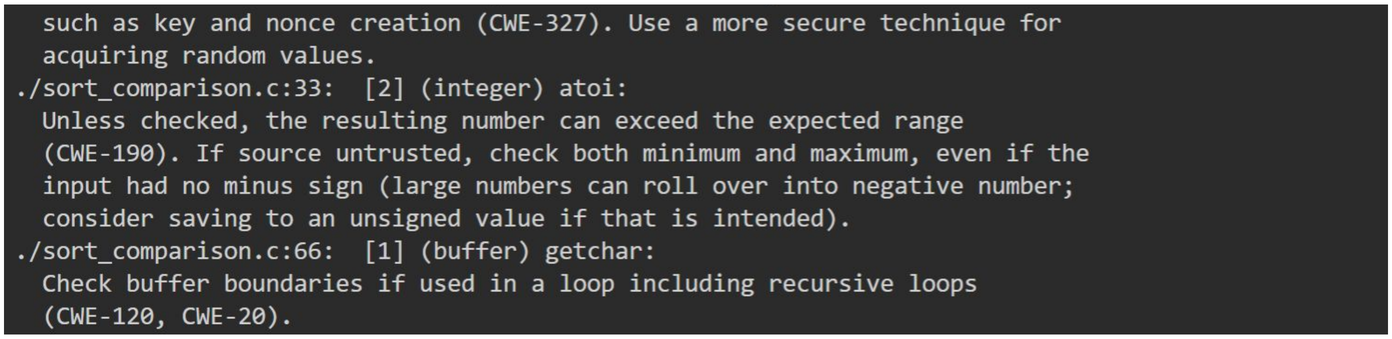
Commands:
cd ~/github-repos/small_c_test
ls


```
root@attackdefense:~/github-repos# cd small_c_test/
root@attackdefense:~/github-repos/small_c_test#
root@attackdefense:~/github-repos/small_c_test# ls
case_changer.c  flow.sh  hello.c  LICENSE  README.md  sort_comparison.c
root@attackdefense:~/github-repos/small_c_test#
```

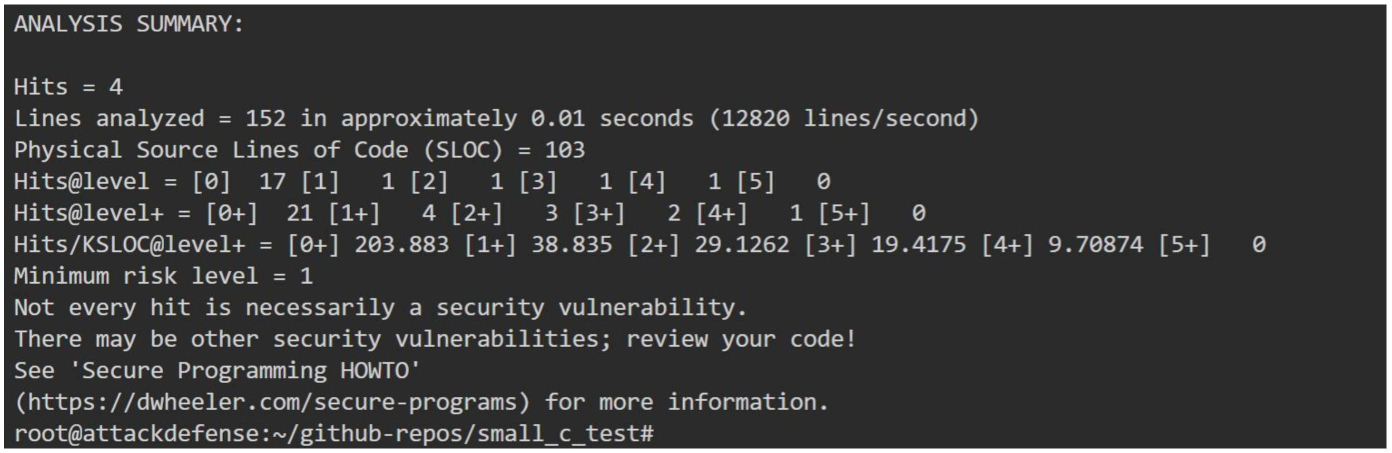
Step B: Run the flawfinder tool in the whole directory.

Command: flawfinder .


```
root@attackdefense:~/github-repos/small_c_test# flawfinder .
Flawfinder version 2.0.10, (C) 2001-2019 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
Examining ./hello.c
Examining ./sort_comparison.c
Examining ./case_changer.c

FINAL RESULTS:

./sort_comparison.c:77:  [4] (format) syslog:
  If syslog's format strings can be influenced by an attacker, they can be
  exploited (CWE-134). Use a constant format string for syslog.
./sort_comparison.c:42:  [3] (random) srand:
  This function is not sufficiently random for security-related functions
```



```
  such as key and nonce creation (CWE-327). Use a more secure technique for
  acquiring random values.
./sort_comparison.c:33:  [2] (integer) atoi:
  Unless checked, the resulting number can exceed the expected range
  (CWE-190). If source untrusted, check both minimum and maximum, even if the
  input had no minus sign (large numbers can roll over into negative number;
  consider saving to an unsigned value if that is intended).
./sort_comparison.c:66:  [1] (buffer) getchar:
  Check buffer boundaries if used in a loop including recursive loops
  (CWE-120, CWE-20).
```



```
ANALYSIS SUMMARY:

Hits = 4
Lines analyzed = 152 in approximately 0.01 seconds (12820 lines/second)
Physical Source Lines of Code (SLOC) = 103
Hits@level = [0]  17 [1]   1 [2]   1 [3]   1 [4]   1 [5]   0
Hits@level+ = [0+]  21 [1+]   4 [2+]   3 [3+]   2 [4+]   1 [5+]   0
Hits/KSLOC@level+ = [0+] 203.883 [1+] 38.835 [2+] 29.1262 [3+] 19.4175 [4+] 9.70874 [5+]   0
Minimum risk level = 1
Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://dwheeler.com/secure-programs) for more information.
root@attackdefense:~/github-repos/small_c_test#
```

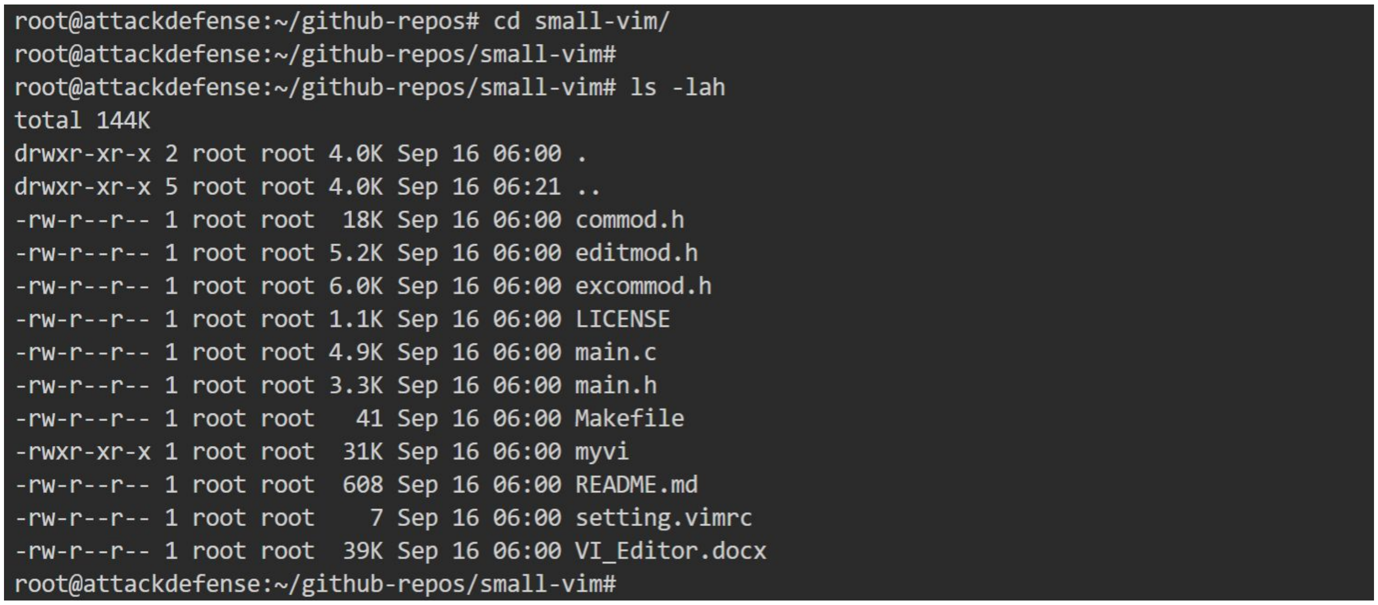
Issues Detected

- Format strings which can be controlled by an attacker
- The “srand” function is not random and should not be used for security-related functions
- Untrusted input should not be passed to function ‘atoi’
- Buffer boundaries should be checked for ‘getchar’ function.

Example 3: small_vim

Step A: Change to the small_vim directory

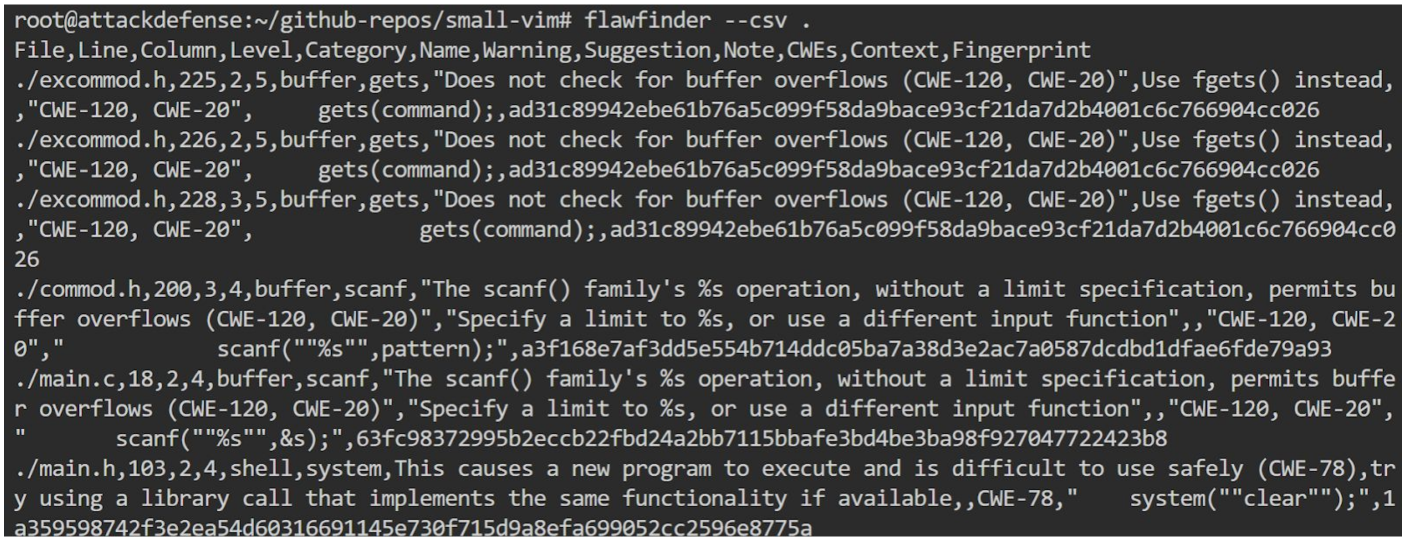
Commands:
cd ~/github-repos/small-vim
ls -lah



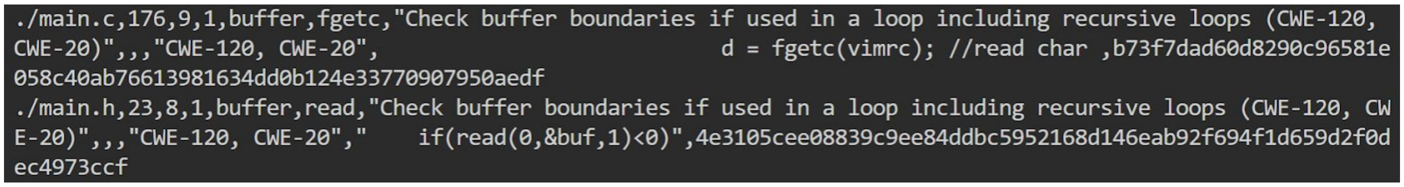
```
root@attackdefense:~/github-repos# cd small-vim/
root@attackdefense:~/github-repos/small-vim#
root@attackdefense:~/github-repos/small-vim# ls -lah
total 144K
drwxr-xr-x 2 root root 4.0K Sep 16 06:00 .
drwxr-xr-x 5 root root 4.0K Sep 16 06:21 ..
-rw-r--r-- 1 root root  18K Sep 16 06:00 commod.h
-rw-r--r-- 1 root root 5.2K Sep 16 06:00 editmod.h
-rw-r--r-- 1 root root 6.0K Sep 16 06:00 excommod.h
-rw-r--r-- 1 root root 1.1K Sep 16 06:00 LICENSE
-rw-r--r-- 1 root root 4.9K Sep 16 06:00 main.c
-rw-r--r-- 1 root root 3.3K Sep 16 06:00 main.h
-rw-r--r-- 1 root root   41 Sep 16 06:00 Makefile
-rwxr-xr-x 1 root root  31K Sep 16 06:00 myvi
-rw-r--r-- 1 root root  608 Sep 16 06:00 README.md
-rw-r--r-- 1 root root    7 Sep 16 06:00 setting.vimrc
-rw-r--r-- 1 root root  39K Sep 16 06:00 VI_Editor.docx
root@attackdefense:~/github-repos/small-vim#
```

Step 14: Run the flawfinder tool in the whole directory and return data in CSV format.

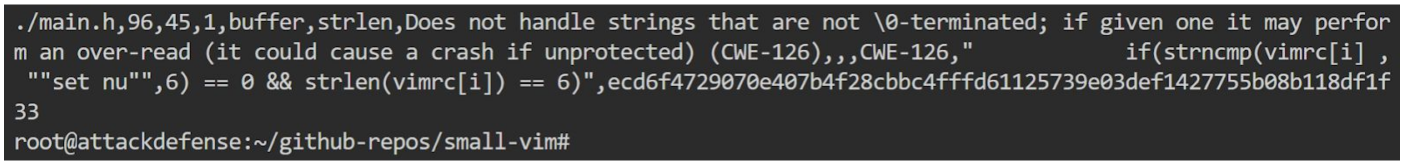
Command: flawfinder --csv .



```
root@attackdefense:~/github-repos/small-vim# flawfinder --csv .
File,Line,Column,Level,Category,Name,Warning,Suggestion,Note,CWEs,Context,Fingerprint
./excommod.h,225,2,5,buffer,gets,"Does not check for buffer overflows (CWE-120, CWE-20)",Use fgets() instead,
,"CWE-120, CWE-20",     gets(command);,ad31c89942ebe61b76a5c099f58da9bace93cf21da7d2b4001c6c766904cc026
./excommod.h,226,2,5,buffer,gets,"Does not check for buffer overflows (CWE-120, CWE-20)",Use fgets() instead,
,"CWE-120, CWE-20",     gets(command);,ad31c89942ebe61b76a5c099f58da9bace93cf21da7d2b4001c6c766904cc026
./excommod.h,228,3,5,buffer,gets,"Does not check for buffer overflows (CWE-120, CWE-20)",Use fgets() instead,
,"CWE-120, CWE-20",             gets(command);,ad31c89942ebe61b76a5c099f58da9bace93cf21da7d2b4001c6c766904cc0
26
./commod.h,200,3,4,buffer,scanf,"The scanf() family's %s operation, without a limit specification, permits bu
ffer overflows (CWE-120, CWE-20)","Specify a limit to %s, or use a different input function",,"CWE-120, CWE-2
0","            scanf(""%s"",pattern);",a3f168e7af3dd5e554b714ddc05ba7a38d3e2ac7a0587dcdbd1dfae6fde79a93
./main.c,18,2,4,buffer,scanf,"The scanf() family's %s operation, without a limit specification, permits buffe
r overflows (CWE-120, CWE-20)","Specify a limit to %s, or use a different input function",,"CWE-120, CWE-20",
"       scanf(""%s"",&s);",63fc98372995b2eccb22fbd24a2bb7115bbafe3bd4be3ba98f927047722423b8
./main.h,103,2,4,shell,system,This causes a new program to execute and is difficult to use safely (CWE-78),tr
y using a library call that implements the same functionality if available,,CWE-78,"    system(""clear"");",1
a359598742f3e2ea54d60316691145e730f715d9a8efa699052cc2596e8775a
```



```
./main.c,176,9,1,buffer,fgetc,"Check buffer boundaries if used in a loop including recursive loops (CWE-120,
CWE-20)",,,"CWE-120, CWE-20",                              d = fgetc(vimrc); //read char ,b73f7dad60d8290c96581e
058c40ab76613981634dd0b124e33770907950aedf
./main.h,23,8,1,buffer,read,"Check buffer boundaries if used in a loop including recursive loops (CWE-120, CW
E-20)",,,"CWE-120, CWE-20","    if(read(0,&buf,1)<0)",4e3105cee08839c9ee84ddbc5952168d146eab92f694f1d659d2f0d
ec4973ccf
```



```
./main.h,96,45,1,buffer,strlen,Does not handle strings that are not \0-terminated; if given one it may perfor
m an over-read (it could cause a crash if unprotected) (CWE-126),,,CWE-126,"            if(strncmp(vimrc[i] ,
 ""set nu"",6) == 0 && strlen(vimrc[i]) == 6)",ecd6f4729070e407b4f28cbbc4fffd61125739e03def1427755b08b118df1f
33
root@attackdefense:~/github-repos/small-vim#
```

Issues Detected

- No checks for Buffer Overflows
- Checks are missing while opening files
- Buffer boundaries should be checked for ‘fgetc’ function

Learnings

Perform static code analysis using flawfinder tool.