Flawfinder – A Tool Analysis for Determining Security Weakness

Student: Siddique Reza Khan, remond_007@yahoo.com **Date:** 21.01.2019

Tool name (long/short)**:** *flawfinder*

Source (University / Institute / Company): https://dwheeler.com/flawfinder/

Target language(s): C/C++ source code

Platform (Windows, Unix/Linux, Mac/OS)**:** Flawfinder works on Unix-like systems (it's been tested on GNU/Linux), and on Windows by using Cygwin. It requires Python 2.7 or Python 3 to run [2].

Licence status: Flawfinder is released under the General Public License (GPL) version 2 or later, and thus is open source software and Free Software [2].

The functionality in a nutshell: [2] [6]
~~(What is the tool able to do and what not? Where are its limits?~~
~~Basic concepts/data structures: How does the tool work?)~~

Flawfinder is an easy and simple security audit tool. It is an intentionally simple tool, but user have found it useful.

Flawfinder has a built-in database of C/C++ functions with well-known problems and it is officially Common Weakness Enumeration (CWE) identifier(s) Compatible, such as buffer overflow risks (e.g., strcpy(), strcat(), gets(), sprintf(), and the scanf() family), format string problems ([v][f]printf(), [v]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()). The best thing is about the tool is, we don't need to create this database - it comes with the tool.

Flawfinder takes the source code text, and matches with database, while ignoring text inside comments and strings (except for flawfinder directives). Flawfinder also knows about gettext (a common library for internationalized programs), and this reduces the number of false hits in internationalized programs. Flawfinder works by doing simple lexical tokenization (skipping comments and correctly tokenizing strings), looking for token matches to the database (particularly to find function calls). Flawfinder is thus similar to RATS and ITS4, which also use simple lexical tokenization. Again, it doesn't do data flow or control flow analysis at all.

Flawfinder produces a list of "hits" (potential security flaws), sorted by risk; by default the riskiest hits are shown first. Flawfinder may be able to determine that the construct isn't risky at all, reducing false positives.

Nevertheless, flawfinder is fundamentally a naive program; it doesn't even know about the data types of function parameters, and it certainly doesn't do control flow or data flow analysis (see

the references below to other tools, like SPLINT, which do deeper analysis). Flawfinder can analyze software that we can't build; in some cases it can analyze files we can't even locally compile. In general, it is suggested *not* load or diff hitlists from untrusted sources with the tool.
Not every hit is actually a security vulnerability, and not every security vulnerability is necessarily found. Nevertheless, flawfinder can be a very useful aid in finding and removing security vulnerabilities. The documentation [6] points out various security issues when using the tool.

Experiences:

- Installation: (any problems?) Flawfinder is specifically designed to be easy to install and use. Problem to set-up Linux Platform in the virtual box machine.

- Stability:It was agreed to release the programs(on May 21, 2001) and till now it is running [2].

- Usability: It's very useful for quickly finding and removing at least some potential security problems [2].

- Beginner's learning curve: 40 to 60 hours depending on the competency on Linux environment and security background with programming in C/C++.

- User interface: (ASCII / GUI): Command line.

good

- Else: One have to have a ~~nice~~ understanding on the various security flaws of the C/C++ program functions such as bound checking, dynamic memory allocation, buffer overflow, format string, etc. Due to lack of understanding of mine, it did not possible to test all the options such as, Selecting Input Data options: --allowlink, --followdotdir, --nolink, --patch F | -P F; Selecting Output Format option: --error-level=LEVEL; and last but not the least Hitlist Management option:--diffhitlist=F, etc [6].

References: ~~(papers, books, slides, web pages)~~

[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