FlawFinder:
an static source code analysis tool

Prof. Breno Miranda, CIn-UFPE

Paulo César Pereira Gomes

Centro de Informática UFPE

Universidade Federal de Pernambuco

STELLANTIS

FACEPE
Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

Flawfinder: Tutorial, Advantages and Limitations

How to install?

- sudo apt install flawfinder
- pip install flawfinder
- ./flawfinder FILES-OR-DIRECTORY

✔Advantages ✔

- Easy to install and use;
- Fast execution;
- Informative outputs;
- Aimed at detecting code that could generate **security flaws**;
- Classifies flaws into **severity levels** according to **Common Weakness Enumeration (CWE)**

⚠ Limitations ⚠

- It has no GUI and does not create files directly;
- Susceptible to certain attacks, not recommended for use with unknown code;
- Susceptible to detecting **False Positives**;
- “does not use or have access to information about control flow, data flow, or data types”

Flawfinder: Input Flags

Selecting Input Data

--allowlink Allow the use of symbolic links; normally symbolic links are skipped. Don't use this option if you're analyzing code by others; attackers could do many things to cause problems for an analysis with this option enabled. For example, an attacker could insert symbolic links to files such as /etc/passwd (leaking information about the file) or create a circular loop, which would cause flawfinder to run "forever". Another problem with enabling this option is that if the same file is referenced multiple times using symbolic links, it will be analyzed multiple times (and thus reported multiple times). Note that flawfinder already includes some protection against symbolic links to special file types such as device file types (e.g., /dev/zero or C:\mystuff\com1). Note that for flawfinder version 1.01 and before, this was the default.

--followdotdir
Enter directories whose names begin with ".". Normally such directories are ignored, since they normally include version control private data (such as .git/ or .svn/), build metadata (such as .makepp), configuration information, and so on.

--nolink Ignored. Historically this disabled following symbolic links; this behavior is now the default.

--patch=*patchfile*

-P *patchfile* Examine the selected files or directories, but only report hits in lines that are added or modified as described in the given patch file. The patch file must be in a recognized unified diff format (e.g., the output of GNU "diff -u old new", "svn diff", or "git diff [commit]"). Flawfinder assumes that the patch has already been applied to the files. The patch file can also include changes to irrelevant files (they will simply be ignored). The line numbers given in the patch file are used to determine which lines were changed, so if you have modified the files since the patch file was created, regenerate the patch file first. Beware that the file names of the new files given in the patch file must match exactly, including upper/lower case, path prefix, and directory separator (\ vs. /). Only unified diff format is accepted (GNU diff, svn diff, and git diff output is okay); if you have a different format, again regenerate it first. Only hits that occur on resultant changed lines, or immediately above and below them, are reported. This option implies --neverignore. **Warning**: Do *not* pass a patch file without the **-P**, because flawfinder will then try to treat the file as a source file. This will often work, but the line numbers will be relative to the beginning of the patch file, not the positions in the source code. Note that you **must** also provide the actual files to analyze, and not just the patch file; when using **-P** files are only reported if they are both listed in the patch and also listed (directly or indirectly) in the list of files to analyze.

Flawfinder: Hit Flags

Selecting Hits to Display

--inputs

-I Show only functions that obtain data from outside the program; this also sets minlevel to 0.

--minlevel=*X*

-m *X* Set minimum risk level to X for inclusion in hitlist. This can be from 0 ("no risk") to 5 ("maximum risk"); the default is 1.

--falsepositive

-F Do not include hits that are likely to be false positives. Currently, this means that function names are ignored if they're not followed by "(", and that declarations of character arrays aren't noted. Thus, if you have use a variable named "access" everywhere, this will eliminate references to this ordinary variable. This isn't the default, because this also increases the likelihood of missing important hits; in particular, function names in #define clauses and calls through function pointers will be missed.

--neverignore

-n Never ignore security issues, even if they have an "ignore" directive in a comment.

--regexp=*PATTERN*

-e *PATTERN*

Only report hits with text that matches the regular expression pattern PATTERN. For example, to only report hits containing the text "CWE-120", use "--regex CWE-120". These option flag names are the same as grep.

Flawfinder: Output Flags

Selecting Output Format

--columns

-C Show the column number (as well as the file name and line number) of each hit; this is shown after the line number by adding a colon and the column number in the line (the first character in a line is column number 1). This is useful for editors that can jump to specific columns, or for integrating with other tools (such as those to further filter out false positives).

--context

-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. It will always generate a header row, followed by 0 or more data rows (one data row for each hit). Selecting this option automatically enables --quiet and --dataonly. The headers are mostly self-explanatory. "File" is the filename, "Line" is the line number, "Column" is the column (starting from 1), "Level" is the risk level (0-5, 5 is riskiest), "Category" is the general flawfinder category, "Name" is the name of the triggering rule, "Warning" is text explaining why it is a hit (finding), "Suggestion" is text suggesting how it might be fixed, "Note" is other explanatory notes, "CWEs" is the list of one or more CWEs, "Context" is the source code line triggering the hit, and "Fingerprint" is the SHA-256 hash of the context once its leading and trailing whitespace have been removed (the fingerprint may help detect and eliminate later duplications). If you use Python3, the hash is of the context when encoded as UTF-8.

--dataonly

-D Don't display the header and footer. Use this along with --quiet to see just the data itself.

--html

-H Format the output as HTML instead of as simple text.

--immediate

-i Immediately display hits (don't just wait until the end).

--singleline

-S Display as single line of text output for each hit. Useful for interacting with compilation tools.

--omittime 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.

--quiet

-Q Don't display status information (i.e., which files are being examined) while the analysis is going on.

THANK YOU!

Centro de Informática UFPE

Universidade Federal de Pernambuco

STELLANTIS

FACEPE
Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

References

- WHEELER, D. A. **Flawfinder**. Disponível em: <https://github.com/david-a-wheeler/flawfinder>.
- WHEELER, D. **Flawfinder**. Disponível em: <https://dwheeler.com/flawfinder/>.
- WHEELER, D. **Flawfinder (PDF)**. Disponível em: <https://dwheeler.com/flawfinder/flawfinder.pdf>