Advanced Storage Systems (40638)

Prof. Asadi

System Call Vulnerabilities in Linux Storage Stack

(Under Supervision of **Dr. Eslamimehr**)

Sepand Haghighi and **Nima Mohammadi**

The quality for core of an OS kernel is usually very high. However, the same can not be stated for the kernel modules, being device drivers or filesystems, mainly due to their rapid pace of development or not being adequately maintained in face of evolving OS kernel. A lot of the defects which are present in a program are not visible to the compiler. Finding these defects involves a wide array of security testing, including both dynamic and static source code analysis among others. The aim of this project is to employ numerous static code analysis tools to build a body of potential defects that are to be used as cues for dynamic analysis. The currents study focuses on Linux kernel and attempts to make use of the proposed framework to inspect kernel-space filesystem modules of the widely-used open source system.

1 Phase One: Static Code Analysis

Static code analysis tools (SCATs) analyze the code without executing it, hence the name static [1]. It examines the source code of the program and reasons over all possible behavior that the program may exhibit [2]. The history of SCATs goes back to Lint written by Stephan Johnson at Bell Laboratories at 1970. Lint examines the C code and could find the bugs that escaped detection by the compiler. Overtime the necessity of code reviews became more apparent, but due to its labour-intensive nature, more and more software developers adopted more systematic and automatic software verification techniques, static analysis getting most of the attention. The first phase of our project encompasses applying multiple SCATs on filesystem implementations within Linux kernel. Contrary to common belief that opensource code is well reviewed and possible bugs are found due to wide spread use, reports show that the amount of bugs are growing uncontrollably[3]. The 'expansive' (i.e. continuation of adding new code) and 'unexplored' (i.e. lack of unbiased entity to statically analyze

Table 1.1: Results of each SCAT on Linux filesystems

FS	LOC	Smatch	CppCheck	Sparse	Blast	Frama-C	Flawfinder
ext2	6840	3	-	0	-	-	29
ext4	32754	30	-	0	-	-	103
btrfs	82450	84	-	69	-	-	191
hfs	4618	22	-	0	-	-	20
jffs2	13771	40	-	6	-	-	45
reiserfs	21742	40	-	6	-	-	159
ubifs	21988	6	-	0	-	-	63
udf	8980	26	-	0	-	-	63

codes) traits of opensource softwares resulted in a void which SCAT companies are hesitant to get involved in, which is to some extent imposed by the rate at which the software is released [4].
Following is a list of SCATs we have checked in this study. We have chosen 10 filesystem here, deliberately avoiding network filesystems. Running SCATs on Linux kernel is a very challenging taskk. Kernel is quite different from your typical software. Kernel is rather hardware-dependent and includes some inline assembly onto its code which renders some SCATs useless. Two such SCATs we stumbled upon that failed to analyze the intermediate code due existing inline assemble codes are CPAchecker and Frama-C. As modification of the kernel code would change the original program semantics, the analysis would become invalid and therefore we refrain from altering the code. Table 1.1 depicts the results of running SCATs on the filesystems. The actual potential errors found by these tools can be found on the appendices. The tests are performed on Linux 3.17-rc1.

1.1 Sparse

Sparse [5] is a static code analysis tool specifically written for Linux kernel by Linus Torvalds in 2003. Sparse is a semantic parser. It creates a sematic prase tree to validate C semantics. It performs lazy type evaluation. Kernel build system has support for sparse and provides a make option to compile the kernel with sparse checking enabled. While it might not be as powerful as other static analyzers, it still might be worthwhile to start testing with this tool. It works by providing a wrapper around CC (i.e. *cgcc* instead of *gcc*).

1.2 Smatch

Smatch is an open source static checker written in C and very similar to Sparse. It is based on the papers about the Stanford Checker. It was developed with Linux kernel in mind, but can be used for other projects as well. Smatch analyzes source to detect programming logic errors. It can detect logic errors such as, attempts to unlock already unlocked spinlock. It is actively used to detect logic errors in the Linux kernel sources.

1.3 BLAST

BLAST (Berkeley Lazy Abstraction Software verification Tool) is an automatic verification tool for checking temporal safety properties of C programs [6]. Given a C program and a temporal safety property, BLAST either statically proves that the program satisfies the safety property, or provides an execution path that exhibits a violation of the property (or, since the problem is undecidable, does not terminate, or fails). BLAST constructs, explores, and refines abstractions of the program state space based on lazy predicate abstraction and interpolation-based predicate discovery.

1.4 CPPCHECK

CppCheck claims to do scope check, bound checking, check of deprecated functions, memory leaks, redundant if, bad usage of the function strtol, bad usage of the function sprintf (overlapping data), division by zero, unsigned division, unused struct member, passing parameter by value, check how signed char variables are used, condition that is always true/-false, unusual pointer arithmetic, dereferencing a null pointer, and incomplete statement [7]. CppCheck [8] is generally not wrong about the errors it report but the chances are there that it reports less number of errors that the actual number of errors present in the code. This means it aims at minimal false positives but can have many false negatives

1.5 FLAWFINDER

FlawFinder is a simple yet efficient and quick tool, developed by David Wheeler, that scans C/C++ source code for calls to typical vulnerable library functions. It categorizes the security "flaws" by risk level. Flawfinder is written in Python, but somehow proves to be very fast. Flawfinder works by using a built-in database of C/C++ functions with well-known problems [7], including but not limited to buffer overflow risks, format string problems and race conditions. It then produces a list of hits, sorted by the severity. Not every hit is actually a security vulnerability, and not every security vulnerability is necessarily found. It is worth noting that Flawfinder does not really understand the code it is analyzing, that is it has no grasp of the semantics of the code at all. Flawfinder simply does pattern matching with known security vulnerabilities.

1.6 CPACHECKER

CPAchecker is a framework and tool for formal software verification, and program analysis, of C programs. Some of its ideas and concepts, for example lazy abstraction, were inherited from the software model checker BLAST. CPAchecker is based on the idea of configurable program analysis which is a concept that allows expression of both model checking and program analysis with one formalism. When executed, CPAchecker performs a reachability analysis, i.e., it checks whether a certain state, which violates a given specification, can potentially be reached.

2 Common Bugs Found by SCATs

The static analysis tools tend to be very verbose and report many bugs many of which turning out to be false positive. Managing these reports becomes even harder in case there are different SCATs to be analysed. The format of the generated warnings are not consistent along various tools. We have developed a Python code which parses the outputs of different SCATs and try to find the common bugs reported by two or more code analysis tools. To our utter astonishment, the python script only found one common potential bug in only one filesystem, namely ReiserFS, inside file `fs/reiserfs/inode.c` line 1810. Sparse states that "variable length array is used" while Flawfinder warns that it "does not check for buffer overflows when copying to destination". We had speculated to find more than one common bug and to sort them by the number of reports, showing the confidence level for the bug being genuine.

3 Manually inspecting Bugs

At this stage we manually reviewed the portions of the codes reported by the SCATs to find if there is a plausible input that could lead to this bug and the consequence of the potential fault ever happening. Based on our inspection, to best of our knowledge, most of these bugs are false positive or very hard to come up with a possible trace to confirm their occurrence manually. Some reports are due to nested pointers whose dereferencing could cause an exception. These are very hard to assess due to many conditional statements involved.

4 Phase Two: Dynamic Code Analysis

The objective in dynamic analysis is to find security errors in a program while it is running. It derives properties that hold for a handful of executions by running the program (usually through instrumentation). This is in contrast with static analysis which examines source code to derive properties that hold for all execution [9].
A dynamic analysis tool may simulate a malicious user or some critical conditions by providing the program with inputs corresponding extreme cases that might not occur during usual runs. Dynamic analysis provides guidance on proactive steps which can be taken to produce longer-term robust software.
Our goal is to use dynamic analysis in hope of finding bugs which is quite challenging as filesystem modules are executed in kernel space.

References

[1] M. D. Ernst, “Static and dynamic analysis: Synergy and duality,” in *WODA 2003: ICSE Workshop on Dynamic Analysis*, pp. 24–27, 2003.

[2] A. G. Bardas *et al.*, “Static code analysis,” *Journal of Information Systems and Operations Management*, vol. 4, no. 2, pp. 99–107, 2010.

[3] N. Palix, G. Thomas, S. Saha, C. Calvès, J. Lawall, and G. Muller, "Faults in linux: ten years later," in *ACM SIGPLAN Notices*, vol. 46, pp. 305–318, ACM, 2011.

[4] R. Kannavara, "Securing opensource code via static analysis," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, pp. 429–436, IEEE, 2012.

[5] D. Searls, "Sparse, linus & the lunatics," 2003.

[6] T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre, "Software verification with blast," in *International SPIN Workshop on Model Checking of Software*, pp. 235–239, Springer, 2003.

[7] L. Torri, G. Fachini, L. Steinfeld, V. Camara, L. Carro, and É. Cota, "An evaluation of free/open source static analysis tools applied to embedded software," in *2010 11th Latin American Test Workshop*, pp. 1–6, IEEE, 2010.

[8] D. Marjamäki, "Cppcheck: a tool for static c/c++ code analysis," 2013.

[9] T. Ball, *The Concept of Dynamic Analysis*, pp. 216–234. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999.

5 Appendix: Ext2

Table 5.1: Ext2

File	Line	Function	Description	Analyzer
ext2/dir.c	550	ext2_add_link()	info: ignoring unreachable code.	Smatch
ext2/inode.c	692	ext2_get_blocks()	error: we previously assumed 'partial' could be null (see line 644)	Smatch
ext2/super.c	736	ext2_max_size()	warn: impossible condition '(res >(9223372036854775807)) =>(s64min-s64max >s64max)'	Smatch
ext2/dir.c	262	-	potential overflows	FlawFinder
ext2/dir.c	274	-	potential overflows	FlawFinder
ext2/dir.c	569	-	Does not check for buffer overflows when copying	FlawFinder
ext2/dir.c	654	-	Does not check for buffer overflows when copying	FlawFinder
ext2/dir.c	662	-	Does not check for buffer overflows when copying	FlawFinder
ext2/ext2.h	456	-	potential overflows	FlawFinder
ext2/ext2.h	457	-	potential overflows	FlawFinder

ext2/ext2.mod.c	178	-	Does not check for buffer overflows when copying	FlawFinder
ext2/namei.c	199	-	Does not check for buffer overflows when copying	FlawFinder
ext2/super.c	1468	-	Does not check for buffer overflows when copying	FlawFinder
ext2/super.c	1510	-	Does not check for buffer overflows when copying	FlawFinder
ext2/xattr.c	80	-	potential overflows	FlawFinder
ext2/xattr.c	221	-	Does not check for buffer overflows when copying	FlawFinder
ext2/xattr.c	506	-	Does not check for buffer overflows when copying	FlawFinder
ext2/xattr.c	536	-	Does not check for buffer overflows when copying	FlawFinder
ext2/xattr.c	551	-	Does not check for buffer overflows when copying	FlawFinder
ext2/xattr.c	591	-	Does not check for buffer overflows when copying	FlawFinder
ext2/xattr.c	672	-	Does not check for buffer overflows when copying	FlawFinder
ext2/xattr.h	41	-	potential overflows	FlawFinder
ext2/xattr_security.c	18	-	Does not check for buffer overflows when copying	FlawFinder
ext2/xattr_security.c	19	-	Does not check for buffer overflows when copying	FlawFinder
ext2/xattr_trusted.c	22	-	Does not check for buffer overflows when copying	FlawFinder
ext2/xattr_trusted.c	23	-	Does not check for buffer overflows when copying	FlawFinder
ext2/xattr_user.c	24	-	Does not check for buffer overflows when copying	FlawFinder
ext2/xattr_user.c	25	-	Does not check for buffer overflows when copying	FlawFinder
ext2/ext2.mod.c	41	-	Does not check for buffer overflows when copying	FlawFinder
ext2/namei.c	173	-	Does not check for buffer overflows when copying	FlawFinder
ext2/xattr.c	163	-	Does not check for buffer overflows when copying	FlawFinder
ext2/xattr.c	388	-	Does not check for buffer overflows when copying	FlawFinder

6 Appendix 2: Ext4

File	Line	Function	Description	Analyzer
ext4/inode.c	1640	__ext4_journalled_writepage()	info: ignoring unreachable code.	Smatch
ext4/inode.c	4571	ext4_setattr()	warn: inconsistent indenting	Smatch
ext4/inode.c	4630	ext4_getattr()	warn: should '(()->i_reserved_data_blocks) « (EXT4_SB(inode->i_sb))->s_cluster_bits' be a 64 bit type?	Smatch
ext4/inode.c	1640	__ext4_journalled_writepage()	info: ignoring unreachable code.	Smatch
ext4/inode.c	4571	ext4_setattr()	warn: inconsistent indenting	Smatch
ext4/inode.c	4630	ext4_getattr()	warn: should '(()->i_reserved_data_blocks) « (EXT4_SB(inode->i_sb))->s_cluster_bits' be a 64 bit type?	Smatch
ext4/super.c	2380	ext4_max_bitmap_size()	warn: impossible condition '(res > (9223372036854775807)) => (s64min-s64max > s64max)'	Smatch
ext4/super.c	3148	ext4_register_li_request()	error: we previously assumed 'ext4_li_info' could be null (see line 3130)	Smatch
ext4/super.c	4059	ext4_fill_super()	Function too hairy. Giving up.	Smatch
ext4/super.c	2380	ext4_max_bitmap_size()	warn: impossible condition '(res > (9223372036854775807)) => (s64min-s64max > s64max)'	Smatch
ext4/super.c	3148	ext4_register_li_request()	error: we previously assumed 'ext4_li_info' could be null (see line 3130)	Smatch
ext4/super.c	4059	ext4_fill_super()	Function too hairy. Giving up.	Smatch
ext4/migrate.c	549	ext4_ext_migrate()	warn: inconsistent indenting	Smatch
ext4/migrate.c	556	ext4_ext_migrate()	warn: inconsistent indenting	Smatch
ext4/migrate.c	563	ext4_ext_migrate()	warn: inconsistent indenting	Smatch
ext4/migrate.c	549	ext4_ext_migrate()	warn: inconsistent indenting	Smatch
ext4/migrate.c	556	ext4_ext_migrate()	warn: inconsistent indenting	Smatch
ext4/migrate.c	563	ext4_ext_migrate()	warn: inconsistent indenting	Smatch
ext4/mballoc.c	1442	mb_free_blocks()	warn: should '(block) « (EXT4_SB(sb))->s_cluster_bits' be a 64 bit type?	Smatch

ext4/mballoc.c	4334	ext4_mb_release_context()	warn: should '(ac->ac_b_ex.fe_len) « (sbi)->s_cluster_bits' be a 64 bit type?	Smatch
ext4/indirect.c	1462	ext4_ind_remove_space()	error: we previously assumed 'partial->bh' could be null (see line 1441)	Smatch
ext4/indirect.c	1474	ext4_ind_remove_space()	error: we previously assumed 'partial->bh' could be null (see line 1441)	Smatch
ext4/ext4.mod.c	415	-	If format strings can be influenced by an attacker, they can be exploited, and note that sprintf variations do not always \0-terminate (CWE-134)	FlawFinder
ext4/dir.c	346		leading to potential overflows	FlawFinder
ext4/dir.c	412		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/ext4.h	1099		leading to potential overflows	FlawFinder
ext4/ext4.h	1100		leading to potential overflows	FlawFinder
ext4/ext4.h	1240		leading to potential overflows	FlawFinder
ext4/ext4.h	1624		leading to potential overflows	FlawFinder
ext4/ext4.h	1638		leading to potential overflows	FlawFinder
ext4/ext4.h	1748		leading to potential overflows	FlawFinder
ext4/ext4.h	1882		leading to potential overflows	FlawFinder
ext4/ext4.h	1883		leading to potential overflows	FlawFinder
ext4/ext4.h	2197		leading to potential overflows	FlawFinder
ext4/ext4.h	2198		leading to potential overflows	FlawFinder
ext4/ext4.h	2649		leading to potential overflows	FlawFinder
ext4/ext4.mod.c	370		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/ext4_jbd2.c	130		leading to potential overflows	FlawFinder
ext4/extents.c	1805		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/extents.c	3188		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/file.c	214		leading to potential overflows	FlawFinder

ext4/hash.c	159		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/inline.c	180		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/inline.c	194		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/inline.c	225		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/inline.c	240		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/inline.c	1126		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/inline.c	1218		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/inline.c	1371		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/inline.c	477		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/inline.c	1421		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/mballoc.c	359		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/mballoc.c	943		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/mballoc.c	2278		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/mballoc.c	2357		Does not check for buffer overflows when copying to destination	FlawFinder

ext4/mballoc.c	2433		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/migrate.c	359		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/mmp.c	154		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/namei.c	199		leading to potential overflows	FlawFinder
ext4/namei.c	201		leading to potential overflows	FlawFinder
ext4/namei.c	1481		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/namei.c	1687		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/namei.c	1811		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/namei.c	2037		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/namei.c	2360		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/namei.c	2929		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/page-io.c	58		leading to potential overflows	FlawFinder
ext4/namei.c	2929		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/resize.c	529		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/resize.c	852		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/resize.c	905		Does not check for buffer overflows when copying to destination	FlawFinder

ext4/resize.c	1099		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/super.c	476		leading to potential overflows	FlawFinder
ext4/super.c	505		leading to potential overflows	FlawFinder
ext4/super.c	506		leading to potential overflows	FlawFinder
ext4/super.c	545		leading to potential overflows	FlawFinder
ext4/super.c	722		leading to potential overflows	FlawFinder
ext4/super.c	1968		leading to potential overflows	FlawFinder
ext4/super.c	3919		leading to potential overflows	FlawFinder
ext4/super.c	4476		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/super.c	4480		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/super.c	4625		leading to potential overflows	FlawFinder
ext4/super.c	4772		leading to potential overflows	FlawFinder
ext4/super.c	5328		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/super.c	5377		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/xattr.c	72		leading to potential overflows	FlawFinder
ext4/xattr.c	303		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/xattr.c	346		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/xattr.c	643		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/xattr.c	663		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/xattr.c	709		Does not check for buffer overflows when copying to destination	FlawFinder

ext4/xattr.c	825		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/xattr.c	934		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/xattr.c	1401		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/xattr.c	1403		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/xattr.h	48		leading to potential overflowsleading to potential overflows	FlawFinder
ext4/xattr_security.c	23		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/xattr_security.c	24		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/xattr_user.c	25		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/xattr_user.c	26		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/ext4.mod.c	66		Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder
ext4/ext4.mod.c	209		Easily used incorrectly; doesn't always \0-terminate or check for invalid pointersEasily used incorrectly; doesn't always \0-terminate or check for invalid pointers	FlawFinder
ext4/inline.c	53		Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder
ext4/inline.c	79		Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder

ext4/inline.c	1361		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/namei.c	2347		Does not check for buffer overflows when copying to destination	FlawFinder
ext4/namei.c	2852		Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder
ext4/super.c	312		Easily used incorrectly; doesn't always \0-terminate or check for invalid pointersEasily used incorrectly; doesn't always \0-terminate or check for invalid pointers	FlawFinder
ext4/super.c	316		Easily used incorrectly; doesn't always \0-terminate or check for invalid pointersEasily used incorrectly; doesn't always \0-terminate or check for invalid pointers	FlawFinder
ext4/super.c	530		On some very old systems, snprintf is incorrectly implemented and permits buffer overflowsOn some very old systems, snprintf is incorrectly implemented and permits buffer overflows	FlawFinder
ext4/super.c	2480		On some very old systems, snprintf is incorrectly implemented and permits buffer overflowsOn some very old systems, snprintf is incorrectly implemented and permits buffer overflows	FlawFinder
ext4/super.c	2491		On some very old systems, snprintf is incorrectly implemented and permits buffer overflows	FlawFinder

ext4/super.c	2492		On some very old systems, snprintf is incorrectly implemented and permits buffer overflows	FlawFinder
ext4/super.c	2503		On some very old systems, snprintf is incorrectly implemented and permits buffer overflows	FlawFinder
ext4/super.c	2504		On some very old systems, snprintf is incorrectly implemented and permits buffer overflows	FlawFinder
ext4/super.c	2533		On some very old systems, snprintf is incorrectly implemented and permits buffer overflows	FlawFinder
ext4/super.c	2554		On some very old systems, snprintf is incorrectly implemented and permits buffer overflows	FlawFinder
ext4/super.c	2592		On some very old systems, snprintf is incorrectly implemented and permits buffer overflows	FlawFinder
ext4/xattr.c	244		Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder
ext4/xattr.c	372		Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder
ext4/xattr.c	609		Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder
ext4/xattr.c	1018		Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder
ext4/xattr.c	1100		Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder

7 Appendix 3: JFFS2

File	Line	Function	Description	Analyzer
fs/jffs2/nodelist.c	1245	jffs2_add_frag_to_fragtree()	error: we previously assumed 'this' could be null	Smatch
fs/jffs2/read.c	154	jffs2_read_dnode()	warn: possible memory leak	Smatch
include/linux/mm.h	1245	stack_guard_page_end()	warn: bitwise AND condition is false here	Smatch
include/linux/mm.h	1239	vma_growsup()	warn: bitwise AND condition is false here	Smatch
fs/jffs2/nodemgmt.c	629	jffs2_mark_node_obsolete()	warn: inconsistent indenting	Smatch
fs/jffs2/nodemgmt.c	640	jffs2_mark_node_obsolete()	warn: inconsistent indenting	Smatch
fs/jffs2/gc.c	529	jffs2_garbage_collect_live()	error: we previously assumed 'frag' could be null	Smatch
fs/jffs2/wbuf.c	641	__jffs2_flush_wbuf()	warn: inconsistent indenting	Smatch
fs/jffs2/wbuf.c	649	__jffs2_flush_wbuf()	warn: was '== 0' instead of '='	Smatch
fs/jffs2/xattr.c	887	jffs2_build_xattr_subsystem	warning: the frame size of 1120 bytes is larger than 1024 bytes	Smatch
fs/jffs2/jffs2.mod.c	195		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/security.c	77		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/security.c	78		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/xattr.c	372		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/xattr_trusted.c	43		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/xattr_trusted.c	44		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/xattr_user.c	43		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/xattr_user.c	44		Does not check for buffer overflows when copying to destination	FlawFinder

fs/jffs2/compr.c	268		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/compr_lzo.c	57		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/compr_rtime.c	101		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/compr_rubin.c	314		leading to potential over-flows	FlawFinder
fs/jffs2/jffs2.mod.c	163		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/nodelist.h	258		leading to potential over-flows	FlawFinder
fs/jffs2/read.c	143		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/readinode.c	652		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/scan.c	519		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/scan.c	1070		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/summary.c	152		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/summary.c	300		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/summary.c	304		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/summary.c	468		Does not check for buffer overflows when copying to destination	FlawFinder

fs/jffs2/summary.c	748		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/wbuf.c	382		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/wbuf.c	451		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/wbuf.c	789		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/wbuf.c	1010		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/write.c	250		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/xattr.c	373		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/xattr.c	1070		Does not check for buffer overflows when copying to destination	FlawFinder
fs/jffs2/dir.c	98		\0-terminated; if given one it may perform an over-read	FlawFinder
fs/jffs2/dir.c	147		\0-terminated; if given one it may perform an over-read	FlawFinder
fs/jffs2/dir.c	289		\0-terminated; if given one it may perform an over-read	FlawFinder
fs/jffs2/gc.c	840		\0-terminated; if given one it may perform an over-read	FlawFinder
fs/jffs2/gc.c	891		\0-terminated; if given one it may perform an over-read	FlawFinder
fs/jffs2/jffs2.mod.c	42		\0-terminated; if given one it may perform an over-read	FlawFinder
fs/jffs2/readinode.c	587		Check buffer boundaries if used in a loop including recursive loops	FlawFinder
fs/jffs2/readinode.c	651		Check buffer boundaries if used in a loop including recursive loops	FlawFinder

fs/jffs2/readinode.c	656		Check buffer boundaries if used in a loop including recursive loops	FlawFinder
fs/jffs2/readinode.c	661		Check buffer boundaries if used in a loop including recursive loops	FlawFinder
fs/jffs2/readinode.c	662		Check buffer boundaries if used in a loop including recursive loops	FlawFinder
fs/jffs2/readinode.c	663		Check buffer boundaries if used in a loop including recursive loops	FlawFinder
fs/jffs2/xattr.c:66	66		\0-terminated; if given one it may perform an over-read	FlawFinder
fs/jffs2/xattr.c:66	361		\0-terminated; if given one it may perform an over-read	FlawFinder
fs/jffs2/xattr.c:66	1101		\0-terminated; if given one it may perform an over-read	FlawFinder

8 Appendix 4: UDF

File	Line	Description	Analyzer
include/linux/mm.h	1245	warn: bitwise AND condition is false here	Smatch
include/linux/mm.h	1239	warn: bitwise AND condition is false here	Smatch
fs/udf/inode.c	971	error: buffer overflow 'laarr' 5 <= s32max	Smatch
fs/udf/inode.c	1028	warn: buffer overflow 'laarr' 5 <= 5	Smatch
fs/udf/inode.c	1050	warn: should 'numalloc « inode->i_sb->s_blocksize_bits' be a 64 bit type?	Smatch
fs/udf/super.c	709	warn: should 'sbi->s_session « sb->s_blocksize_bits' be a 64 bit type?	Smatch
fs/udf/super.c	826	info: ignoring unreachable code.	Smatch
fs/udf/inode.c	1692	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/inode.c	1716	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/inode.c	1754	Does not check for buffer overflows when copying to destination	FlawFinder

fs/udf/balloc.c	500	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/dir.c	147	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/dir.c	149	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/directory.c	47	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/directory.c	110	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/directory.c	138	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/directory.c	139	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/directory.c	150	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/file.c	49	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/file.c	75	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/inode.c	103	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/inode.c	124	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/inode.c	294	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/inode.c	317	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/inode.c	1329	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/inode.c	1386	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/inode.c	1398	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/inode.c	1412	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/inode.c	1630	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/inode.c	1707	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/inode.c	1726	Does not check for buffer overflows when copying to destination	FlawFinder

fs/udf/inode.c	1838	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/inode.c	1916	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	64	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	66	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	68	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	69	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	78	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	80	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	82	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	84	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	124	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	127	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	128	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	212	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	214	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	994	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/namei.c	1163	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/symlink.c	56	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/symlink.c	60	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/udf.mod.c	130	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/unicode.c	39	Does not check for buffer overflows when copying to destination	FlawFinder

fs/udf/unicode.c	61	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/unicode.c	78	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/unicode.c	420	Does not check for buffer overflows when copying to destination	FlawFinder
fs/udf/super.c	224	Ensure that umask is given most restrictive possible setting	FlawFinder
fs/udf/super.c	674	Ensure that umask is given most restrictive possible setting	FlawFinder
fs/udf/super.c	933	Easily used incorrectly; doesn't always \0-terminate or check for invalid pointers	FlawFinder
fs/udf/super.c	1385	Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder
fs/udf/super.c	1447	Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder
fs/udf/super.c	1464	Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder
fs/udf/super.c	1471	Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder
fs/udf/super.c	2097	Ensure that umask is given most restrictive possible setting	FlawFinder
fs/udf/udf.mod.c	82	Easily used incorrectly; doesn't always \0-terminate or check for invalid pointers	FlawFinder
fs/udf/unicode.c	33	Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder
fs/udf/unicode.c	35	Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder
fs/udf/unicode.c	39	Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder
fs/udf/unicode.c	41	Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder

fs/udf/unicode.c	43	Does not handle strings that are not \0-terminated; if given one it may perform an over-read	FlawFinder

9 Appendix 5: Python Code for Finding Common Bugs

```python
import os
import datetime
def print_line(number,char="-"):
    response=""
    for i in range(number):
        response=response+char
    response=response+"\n"
    return response
def over_line_handler(list_of_lines,name):
    com_1=[]
    com_2=[]
    com_3=[]
    for item in list_of_lines[0][0]:
        if item in list_of_lines[1][0]:
            com_1.append(item+" --> "+list_of_lines[0][1][list_of_lines
        if item in list_of_lines[2][0]:
            com_2.append(item+" --> "+list_of_lines[0][1][list_of_lines
    for item in list_of_lines[1][0]:
        if item in list_of_lines[2][0]:
            com_3.append(item+list_of_lines[1][1][list_of_lines[1][0].i
    file=open(os.path.join(os.getcwd(),"log/")+name[:-4]+".log","w")
    file.write(name[:-4]+"Log File\n")
    file.write("Generated :"+str(datetime.datetime.now())+"\n")
    file.write("Sparse Lines :("+str(len(list_of_lines[0][0]))+")\n")
    file.write(str(list_of_lines[0][0])+"\n")
    file.write("Smatch Lines :("+str(len(list_of_lines[1][0]))+")\n")
    file.write(str(list_of_lines[1][0]) + "\n")
    file.write("FlawFinder Lines :("+str(len(list_of_lines[2][0]))+")\
    file.write(str(list_of_lines[2][0]) + "\n")
    file.write(print_line(30, "*"))
    file.write("Sparse & Smatch : \n\n")
    if len(com_1)!=0:
        file.write("".join(com_1)+"\n")
    else:
        file.write("No Common Line\n")
    file.write(print_line(30,"*"))
    file.write("Sparse & FlawFinder : \n\n")
    if len(com_2)!=0:
        file.write("".join(com_2)+"\n")
    else:
        file.write("No Common Line\n")
    file.write(print_line(30,"*"))

    file.write("Smatch & FlawFinder : \n\n")
    if len(com_3)!=0:
        file.write("".join(com_3)+"\n")
    else:
        file.write("No Common Line\n")
    file.write(print_line(30,"*"))

    file.close()




def line_handler(file_name):
    file=open(file_name,"r")
    Smatch_lines=[]
    Flaw_lines=[]
    Sparse_lines=[]
    Smatch_file=[]
    Sparse_file=[]
    Flaw_file=[]
    temp=""
    for line in file:
        splited_line=line.split(",")
        if len(splited_line)>4:
            if splited_line[1]!="Line" and splited_line[4][:-1]=="Smatc
                Smatch_lines.append(splited_line[1])
                Smatch_file.append(splited_line[0])
            elif splited_line[1]!="Line" and splited_line[4][:-1]=="Fla
                Flaw_lines.append(splited_line[1])
                Flaw_file.append(splited_line[0])
            elif splited_line[1]!="Line" and splited_line[4][:-1]=="Spa
                Sparse_lines.append(splited_line[1])
                Sparse_file.append(splited_line[0])
        elif len(splited_line)>0 and len(splited_line)<=4:
            if splited_line[1]!="Line" and splited_line[len(splited_lir
                Smatch_lines.append(temp[1])
                Smatch_file.append(temp[0])
            elif splited_line[1]!="Line" and splited_line[len(splited_l
                Flaw_lines.append(temp[1])
                Flaw_file.append(temp[0])
            elif splited_line[1]!="Line" and splited_line[len(splited_l
                Sparse_lines.append(temp[1])
                Sparse_file.append(temp[0])

        temp=splited_line
    over_line_handler([[Sparse_lines,Sparse_file],[Smatch_lines,Smatch_

if __name__=="__main__":
    list_of_files=os.listdir()
    for item in list_of_files:
        if item.find(".csv")!=-1:
            line_handler(item)
    print("Done!")
```