CS8803SS, Spring 2010, Homework #1

Nick Black
nickblack@linux.com

1 Buffer overflows.

overflow.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  static inline int
5  exploit_me_harder(FILE *fp,void *buf,size_t s){
6          return fgets(buf,BUFSIZ,fp) ? 0 : -1;
7  } // goodbye cruel world!
8
9  int main(void){
10         char buf[10];
11         int ret = exploit_me_harder(stdin,buf,sizeof(buf));
12         return printf("exploitable() returned %d\n",ret) < 0 ?
13                 EXIT_FAILURE : EXIT_SUCCESS;
14 }
```

Let's go ahead and gavage it...

```
[recombinator](0) $ ./overflow < /dev/zero
Segmentation fault
[recombinator](139) $
```

Could source code analysis have determined this issue existed? Certainly. Indeed, `gcc` alerts us to practice most dubious, even with only basic warning flags:

```
[recombinator](0) $ gcc -O2 -Wall -W -o overflow overflow.c
overflow.c: In function exploit_me_harder:
overflow.c:5: warning: unused parameter s
[recombinator](0) $
```

Sedulous attention to compiler warnings would have been sufficient.

Let's symptomatically treat the warning:

overflow-fixup.c

```
#include <stdio.h>
#include <stdlib.h>

static inline int
exploit_me_harder(FILE *fp,void *buf,size_t s){
        return fgets(buf,s,fp) ? 0 : -1;
} // goodbye cruel world!

int main(void){
        char buf[10];
        int ret = exploit_me_harder(stdin,buf,BUFSIZ);
        return printf("exploitable() returned %d\n",ret) < 0 ?
                EXIT_FAILURE : EXIT_SUCCESS;
}
```

`gcc`[1] no longer prints any warnings, even with extended bounds-checking and warning flags. Can the issue still be found? Splint doesn't quite manage:

```
[recombinator](0) $ splint +matchanyintegral overflow-fixup.c
Splint 3.1.2 --- 20 Feb 2009

overflow-fixup.c: (in function main)
overflow-fixup.c:11:36:
 Passed storage buf not completely defined
 (*buf is undefined): exploit_me_harder (..., buf, ...)
 Storage derivable from a parameter, return value or global
 is not defined. Use /*@out@*/ to denote passed or returned
 storage which need not be defined.

Finished checking --- 1 code warning
[recombinator](1) $
```

The warning isn't smart enough; while it does point to an issue, any number of correct fixups would continue to draw the flag. As noted in [2], it is useless in iterative development.

Sizing the array based on the results of, say, `rand(3)` is likely to cause these tools to shudder and begin belching smoke.

[1] gcc version 4.4.3 (Debian 4.4.3-2)

In order to correctly identify the problem, *without false positives*, analysis requires:

- Working interprocedural analysis.
- Either expert knowledge regarding `fgets(3)`, or access to its source.
- Expert knowledge regarding `FILE` objects. Imagine that this was machine-generated code, and thus prone to bizarre (but legal) activity. Presume that code immediately prior had definitively brought the `FILE` into the EOF state (as, if you'll allow, explicitly tested by a `feof(3)` guard immediately prior to this code). We'd not want to alert in such a case.
- Exponential time and space, since all possible execution paths reaching this function might have to be evaluated (we don't want to warn on dead code).
- *ad nauseam...*

Assuming a willingness to emit false positives, just iteratively strip away the refining factors listed above, eventually reaching a tool which warns on all possible programs[2]. Between these extremes lie a great many analyses, few so effective or general as hiring the right programmers.

Observing the program in a disassembler doesn't provide much insight, and we realize that a binary analysis tool would similarly require either knowledge of `fgets(3)` or interlibrary analysis. Assuming such traits, the issue is trivial to determine via value propagation [1].

2 Callgraph Chicanery.

I've no idea what Problem 2 is talking about. Is IDAPro horribly broken somehow? None of my tools show this behavior. I have a hard time believing they'd be allowed to for very long.

"Output from gdb"

```
[recombinator](0) $ gdb ./overflow GNU gdb (GDB)
7.0.1-debian Copyright (C) 2009 Free Software
Foundation, Inc.  License GPLv3+: GNU GPL version
3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change
and redistribute it.  There is NO WARRANTY,
to the extent permitted by law.  Type "show
copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from
/home/dank/local/gt/cs8803ss/hw1/overflow...done.
(gdb) step The program is not being run.
(gdb) start Temporary breakpoint 1 at 0x400580:
file overflow.c, line 9.  Starting program:
/home/dank/local/gt/cs8803ss/hw1/overflow
```

[2]It is rumored that Gimpel Software (makers of PC-LINT™) brought such a tool to market in the late '90s.

```
Temporary breakpoint 1, main () at overflow.c:9 9
int main(void){ (gdb) next 11                    int
ret = exploit_me_harder(stdin,buf,sizeof(buf));
(gdb) next ^C Program received signal SIGINT, Interrupt.
0x00000035130bf520 in
__read_nocancel () from /lib/libc.so.6 (gdb)
bt #0  0x00000035130bf520 in __read_nocancel ()
from /lib/libc.so.6 #1        0x000000351306d068 in
_IO_new_file_underflow (fp=<value optimized out>)
    at fileops.c:605
#2  0x000000351306e73e in _IO_default_uflow
(fp=<value optimized out>)
    at genops.c:440
#3  0x00000035130636de in _IO_getline_info
(fp=<value optimized out>,
    buf=<value optimized out>, n=<value
    optimized out>, delim=<value optimized
    out>, extract_delim=<value optimized out>,
    eof=<value optimized out>) at iogetline.c:74
#4  0x00000035130625b9 in _IO_fgets (buf=<value
optimized out>,
    n=<value optimized out>, fp=<value optimized
    out>) at iofgets.c:58
#5  0x0000000000400598 in exploit_me_harder ()
at overflow.c:6 #6  main () at overflow.c:11
(gdb) A debugging session is active.

        Inferior 1 [process 22340] will be
        killed.

Quit anyway? (y or n) y
[recombinator](0) $
```

Nope, no problems here, either…

"Output from valgrind"

```
[recombinator](0) $ valgrind --tool=callgrind ./overflow < zero
Callgrind, a call-graph generating cache profiler
Copyright (C) 2002-2009, and GNU GPL'd, by
Josef Weidendorfer et al.  ==22499== Using
Valgrind-3.5.0-Debian and LibVEX; rerun with -h
for copyright info
Command: ./overflow
```

```
 For interactive control, run 'callgrind_control -h'.
Process terminating with default action of signal 11 (SIGSEGV)
Access not within mapped region at address 0x7FF001000
at 0x351307C7DB: memcpy (memcpy.S:267)
by 0x35130636BC: _IO_getline_info (iogetline.c:114)
by 0x35130625B8: fgets (iofgets.c:58)
by 0x400597: main (overflow.c:6)
If you believe this happened as a result of a stack
overflow in your program's main thread (unlikely
but possible), you can try to increase the size of the
main thread stack using the --main-stacksize= flag.
The main thread stack size used in this run was 8388608.
Segmentation fault
[recombinator](139) $
```

References

[1] J. Yang, T. Kremenek, Y. Xie, and D. Engler. Meca: an extensible, expressive system and language for statically checking security properties. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 321–334, New York, NY, USA, 2003. ACM.

[2] M. Zitser, R. Lippmann, and T. Leek. Testing static analysis tools using exploitable buffer overflows from open source code. *SIGSOFT Softw. Eng. Notes*, 29(6):97–106, 2004.