Fuzzing Project report September 2015

Hanno Böck

September 2015

1 Linux / Gentoo system with Address Sanitizer

As already briefly mentioned in the last report I started to try an effort to create a full Linux system based on Gentoo with Address Sanitizer enabled. Address Sanitizer is a feature of the gcc and clang compilers that detects memory access bugs that are otherwise often hard to detect. It is designed as a debugging tool and has significant performance costs, but it could be used in production environments with high security requirements.

Just trying to build such a system already uncovered countless bugs. Many applications already show invalid memory access in their normal operation or in their test suite. In the course of this effort bugs have been fixed in core packages like bash [1], coreutils / shred [2] and man-db [3]. Further bugs have been reported in tcl [4] [5], iasl [6], xerces-c [7] [8], mhash [9], cdrtools (developer doesn't think it's a bug), star (same issue as cdrtools), nasm [10], yudit, dovecot [11], courier [12], proftpd [13] [14] [15], johntheripper [16] , tardy. Most of them appeared either purely by running their test suite or because they were used in the build process of other packages.

In many cases it turned out that issues found with Address Sanitizer were already fixed, but only the latest version contained the fix or the fix was not yet

[1] https://ftp.gnu.org/gnu/bash/bash-4.3-patches/bash43-041
[2] https://debbugs.gnu.org/cgi/bugreport.cgi?bug=20998
[3] https://savannah.nongnu.org/bugs/index.php?45854
[4] http://core.tcl.tk/tk/tktview/b1534b438bc711e848ad7ade3642ce0a6323fe8e
[5] http://core.tcl.tk/tk/tktview/9bad630c3163b4b2ef8781089ae27058c957a428
[6] https://bugs.acpica.org/show_bug.cgi?id=1184
[7] https://issues.apache.org/jira/browse/XERCESC-2050
[8] https://issues.apache.org/jira/browse/XERCESC-2051
[9] http://sourceforge.net/p/mhash/mailman/message/34273046/
[10] http://bugzilla.nasm.us/show_bug.cgi?id=3392315
[11] http://hg.dovecot.org/dovecot-2.2/rev/740935acc0f8
[12] https://blog.fuzzing-project.org/17-Courier-mail-server-Write-heap-overflow-in-mailbot-tool-and-out-of-bound
html
[13] http://bugs.proftpd.org/show_bug.cgi?id=4193
[14] http://bugs.proftpd.org/show_bug.cgi?id=4194
[15] http://bugs.proftpd.org/show_bug.cgi?id=4195
[16] http://www.openwall.com/lists/john-dev/2015/09/13/10

released (e. g. in screen, lcms, tmux, texinfo, binutils, python, mawk, gnulib). Therefore often the latest stable Version in Gentoo was missing the fix and just getting an update or backporting a patch from the upstream code was needed.

I still have a large batch of log files from compiles with test suite runs failing with different Address Sanitizer errors. It is unlikely that I will tackle them all by myself, at some point I will probably publish the log files and allow the community to have a look at them.

I think this effort and the large number of bugs that show up just by compiling applications and running their test suite shows that Address Sanitizer needs more publicity. Developers of C/C++ applications should be encourage to test their software with Address Sanitizer. It is a very easy thing to do that can uncover serious bugs. I have created a short tutorial which shows how to easily utilize some compiler flags to test software [17].

2 Undefined behavior sanitizer

Along with Address Sanitizer there are other sanitizing tools in gcc and clang. They include Thread Sanitizer (races), Memory Sanitizer (uninitialized memory, clang only), Leak Sanitizer (memory leaks, automatically enabled with Address Sanitizer in latest versions) and Undefined Behavior Sanitizer (C code with undefined outcome).

I had a closer look at the Undefined Behavior Sanitizer (ubsan). It will report code that by the definition of the C standard is undefined. The most common occurrences of undefined behavior are invalid shift operations and - to a lesser degree - signed integer overflows.

The problem with these issues is that they are extremely widespread and their impact is hard to classify. They often affect code that seems intuitively right unless the programmer knows subtle details of the C standard and that will most likely not cause any issues in practice, because the compiler will still produce the code that the programmer expected. As it is undefined behavior this is still correct. The compiler may very well do what the programmer expects even if its undefined. I am not aware that these issues ever caused any real world security threat.

What makes matters even more complicated is that fixing these issues often leads to more complicated and less readable code. This problem can for example be seen when trying to create a rotate operation without undefined behavior, Wikipedia has a description of that problem [18].

In some situations fixing undefined behavior also makes code measurably slower. In OpenSSL the symmetric block mode operations contain two code paths, one that is enabled upon setting a variable STRICT_ALIGNMENT. This avoids invalid memory alignments, but it will be slower, as operations that could happen in blocks of the size of an integer now happen byte-wise [19].

[17] `https://fuzzing-project.org/tutorial-cflags.html`
[18] **??**
[19] `https://github.com/openssl/openssl/blob/6218a1f57e7e25a6b9a798f00cf5f0e56a02ff31/`

Due to the large number of issues found even in core software and the lack of plausible attack scenarios I am unsure whether these things should be reported in masses to upstream developers. I started a discussion thread about these questions that on oss-security [20]. There were few replies, but I tend to agree with Solar Designer who wrote: "I think it's worth reporting these bugs primarily to more recent, cleaner, and better maintained projects, as well as to smaller projects, where it is realistic that all of these bugs would be fixed."

An invalid shift operation was fixed after my reports in zlib recently which is probably one of the most widely used free software libraries at all [21].

3 KASan

Since version 4.0 the Linux Kernel also includes the option to be built with Address Sanitizer.

Just booting a Kernel with KASan (Kernel Address Sanitizer) on my laptop already showed many warnings for out of bounds reads in the dmesg log. They were caused by two bugs in the Intel GPU driver. One was in a preprocessor macro, the code was already fixed in the upstream kernel code [22] (4.2 contains the fix). The other bug was caused by a wrong use of the counter variables in a nested loop. The loop checked the sorting of a table of GPU commands. However fixing the bug led to an unusable kernel because the tables were not properly sorted and thus the GPU driver would not start. In exchange with the Intel driver developers I was able to fix this issue with two patches that were quickly merged and will be part of Linux 4.3 [23] [24].

I did some further short tests with KASan and popular Kernel fuzzing tools (trinity, perf_fuzzer), but they didn't turn up any other issues. I also tested mounting corrupted file systems that were produced by fuzzing the corresponding file system tools with american fuzzy lop (see next chapter).

4 File system tools

I recently started fuzzing file system tools. For pretty much every file system supported in Linux there is usually a set of tools that at least include a tool to create and one to check the corresponding file system. These tools sometimes are called automatically on mounting a file system, therefore their parser should be

crypto/modes/cbc128.c

[20] http://seclists.org/oss-sec/2015/q3/42

[21] https://github.com/madler/zlib/commit/8a979f6c7986574e37316148cd8ca440c3bc08a3

[22] https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/include/drm/drm_atomic.h?id=60f207a5b6d8f23c2e8388b415e8d5c7311cc79d

[23] https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/drivers/gpu/drm/i915/i915_cmd_parser.c?id=9f58582c7ad64f025e7fc582461c5bfafb46818f

[24] https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/drivers/gpu/drm/i915/i915_cmd_parser.c?id=8453580cb8834dedffda86bcb64f13befc90eb03

considered security sensitive. By attaching a USB stick with a specially crafted file system these parser bugs may be exploitable.

This uncovered bugs in dosfstools, f2fs-tools, btrfs-progs, cramfs, xfsprogs, ntfs3g, squashfs-tools, reiserfsprogs. In reiserfsprogs just running the tools with Address Sanitizer enabled uncovered a bug. cramfs seems unmaintained (one of the bugs I found was already reported years ago), for squashfs-tools the situation is complicated. The original project is dead, there is a fork on github with unmerged pull requests that fix some of the bugs and a couple of bugs have been fixed within the package in fedora. This is an extreme example of how difficult things become when upstream projects are abandoned.

A problem was that some file systems have a very large minimum size. For example xfs file systems have to be at least 16 Megabytes, for f2fs the minimum size is 128 Megabytes. American Fuzzy Lop is mostly optimized for small inputs and rejects input files larger than one Megabyte.

In many cases it was still possible to fuzz these tools. Sometimes just truncating the input file was enough. In the case of f2fs-tools I had to remove a file size check from its code, then I was able to fuzz it with a truncated file system dump.

I also used the output samples of the fuzzing process to test the kernel file system code. On a kernel compiled with KASan I tried mounting and unmounting the respective file system images. To my surprise I didn't find any bugs with this strategy. This might indicate that the code for the file systems itself is much better maintained than the file system tools. But it might also just mean that the direct fuzzing with american fuzzy lop is much more successful in finding bugs than the indirect fuzzing with the samples. I am not aware of any methods to directly fuzz kernel code in a way like american fuzzy lop does it.

5 BIND Denial of Service vulnerability (CVE-2015-5722)

In July ISC, the company developing the BIND DNS server, released an update that fixed a denial of service vulnerability that could remotely crash a server through an assert [25] in the handling of TKEY queries.

This vulnerability was found via fuzzing with american fuzzy lop. I then started myself to have a look at BIND and fuzzed various of its command line. I discovered a similar bug in the parsing of DNSSEC keys [26]. While it is also only a denial of service issue through an assert it is possible to create a DNSSEC record that will cause a DNSSEC validating resolver to crash when trying to resolve that record. Therefore it could be used to crash a large number of DNS servers.

[25] https://www.isc.org/blogs/cve-2015-5477-an-error-in-handling-tkey-queries-can-cause-named-to-exit-with-a-req

[26] https://blog.fuzzing-project.org/22-BIND-Denial-of-Service-via-malformed-DNSSEC-key-CVE-2015-5722.html

The vulnerability was fixed with new releases of BIND on September 2nd [27]. In a blog post following that disclosure ISC wrote that they now incorporated testing with american fuzzy lop into their test suite [28].

6 Misc

Apart from the work mentioned I regularly run fuzzing jobs on a wide variety of software packages.

I recently found and reported fuzzing bugs in mutt [29], patchelf [30], qpdf [31], libcroco [32], poppler [33] [34], libxml2 [35] [36] [37] [38], libxslt [39].

[27] https://kb.isc.org/article/AA-01287
[28] https://www.isc.org/blogs/summer_security_vulnerabilities/
[29] http://dev.mutt.org/trac/ticket/3776
[30] https://github.com/NixOS/patchelf/issues/64
[31] https://github.com/qpdf/qpdf/issues/51
[32] https://bugzilla.gnome.org/show_bug.cgi?id=753034
[33] https://bugs.freedesktop.org/show_bug.cgi?id=91200
[34] https://bugs.freedesktop.org/show_bug.cgi?id=91186
[35] https://bugzilla.gnome.org/show_bug.cgi?id=751603
[36] https://bugzilla.gnome.org/show_bug.cgi?id=751631
[37] https://bugzilla.gnome.org/show_bug.cgi?id=751643
[38] https://bugzilla.gnome.org/show_bug.cgi?id=752191
[39] https://bugzilla.gnome.org/show_bug.cgi?id=751633