



CASR: Your Life Vest in a Sea of Crashes

Speaker: Andrey Fedotov, Ph.D.

Rnd Team Lead, ISP RAS
infosec.exchange/@anfedotoff

Speaker: Alexey Vishnyakov, Ph.D.

Senior DevSecOps Engineer, Yandex Cloud
infosec.exchange/@VishnyaSweet



Fuzzing in Security Development Lifecycle



- Fuzzing is DAST that unveils input seeds causing the target application to crash or trigger sanitizer checks
- Google [OSS-Fuzz](#) provides continuous fuzzing for open source software where crash triaging and fixing are performed by OSS project maintainers
- When we deliver SDL practices to companies, crashes are mostly triaged in-house by product developers
- To reduce the burden on shoulders of software engineers, crash triaging should be thoughtfully automated in DevSecOps pipelines

Crash Triaging Problems



1.

Tons of crashes flooding the security board

2.

Removing duplicate crashes

3.

Crash debugging overhead

4.

Lots of different programming languages

5.

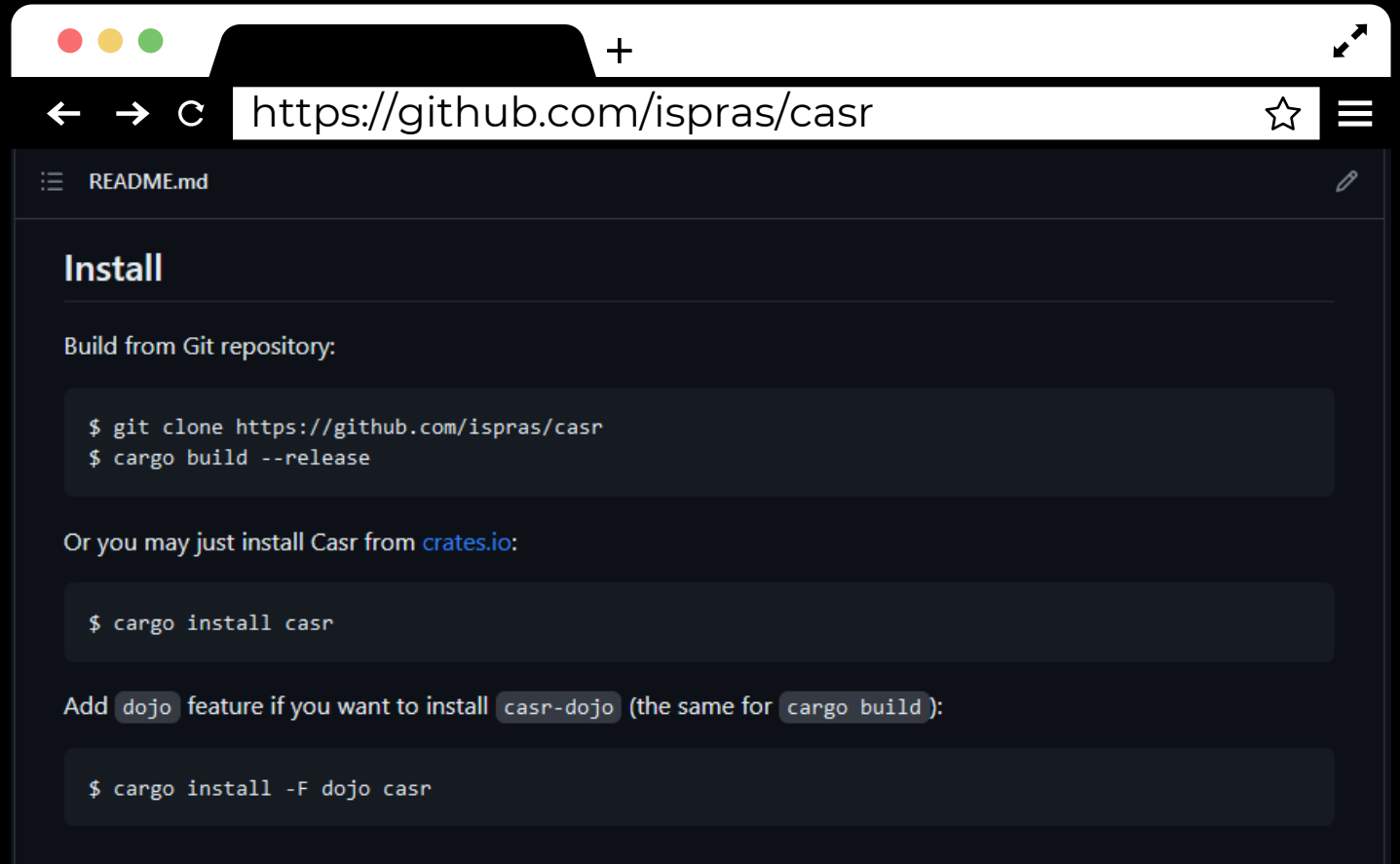
Huge variety of fuzzers

6.

Vulnerability management: estimating crash severity

Outline

- Existing Crash Triaging Tools Survey
- CASR: Crash Analysis and Severity Report
- Fuzzing Crash Triage Pipeline
- Undefined Behavior Sanitizer Errors Triage
- Vulnerability Management with DefectDojo
- Exporting Crash Reports to SARIF
- LibCASR: API for Crash Triage



```
README.md
```

Install

Build from Git repository:

```
$ git clone https://github.com/ispras/casr
$ cargo build --release
```

Or you may just install Casr from crates.io:

```
$ cargo install casr
```

Add `dojo` feature if you want to install `casr-doj` (the same for `cargo build`):

```
$ cargo install -F dojo casr
```



NO
FF
ONE
2023

Existing Crash Triaging Tools Survey

Crash Triaging Tools Survey



- GDB exploitable plugin: classifies crash severity
- AFLTriage: GDB stack trace hash based deduplication
- The tools above are abandoned
- Apport: Ubuntu crash reporting
- ClusterFuzz: OSS-Fuzz backend, crash deduplication based on Levenshtein distance between stack traces

The screenshot shows a GitHub repository page for the file `AFLplusplus/docs/fuzzing_in_depth.md`. The page is in the 'Preview' mode, showing the first line of the document: `casr-af1` from [CASR](#) tools provides comfortable triaging for crashes found by AFL++. Reports are clustered and contain severity and other information.

```
casr-af1 -i /path/to/af1/out/dir -o /path/to/casr/out/dir
```

NO
FF
ONE
2023

CASR: Crash Analysis and Severity Report



Motivation to Create CASR: Crash Analysis and Severity Report

Continuous fuzzing of open source projects
github.com/ispras/oss-sydr-fuzz:

- The more projects, the more crashes to handle
- CI fuzzing support for new program language often requires one more fuzzer/engine
- Different fuzzers have varying output

Vulnerability management:

- Which bug to fix first?



CASR History



- 1.** Casr (now **casr-core**): Core Dump Analysis and Severity Estimation
- 2.** Casr-Cluster: Crash Clustering for Linux Applications
- 3.** CASR tools: **casr-core**, **casr-gdb**, **casr-san**, **casr-python**, **casr-java**, **casr-ubsan**, **casr-afl**, **casr-libfuzzer**, **casr-cli**, **casr-dojo**
- 4.** LibCASR: Crash Triage API

casr-core: Core Dump Analysis and Severity Estimation

- Based on ideas from exploitable and apport
- CASR report with useful information: stack trace, register values, disassembly, severity estimation, opened files and network connections, etc.
- Online mode (apport like) and offline mode (GDB like)



CASR Crash Report



```
Crash Report for /decode_png_fuzz
Severity: NOT_EXPLOITABLE: heap-buffer-overflow(read)
Crash line: /libpng-1.6.37/png.c:90:18
```

```
▼ Date
  ◦ 2023-06-10T07:11:16.577326+03:00
▼ Uname
  ◦ Linux runner-qnfnlmdp-project-3-concurrent-4 5.15.0-72-generic #79~20.04.1-Ubuntu SMP Thu Apr 20 22:12:07 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
▼ OS
  ◦ Ubuntu
▼ OSRelease
  ◦ 20.04
▼ Architecture
  ◦ amd64
▶ ExecutablePath
▼ ProcCmdline
  ◦ /decode_png_fuzz -artifact_prefix=/builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/torchvision/decode_png-out/crashes/ -verbosity=2 -detect_leaks=0 -rss_limit_mb=15360 -timeout=300 -report_
▼ CrashSeverity
  ◦ NOT_EXPLOITABLE
  ◦ heap-buffer-overflow(read)
  ◦ Heap buffer overflow
  ◦ The target reads data past the end, or before the beginning, of the intended heap buffer.
▶ ProcEnviron
▼ Stacktrace
  ◦ #0 0x55c515 in MemcmpInterceptorCommon(void*, int (*)(void const*, void const*, unsigned long), void const*, void const*, unsigned long) /llvm-project-llvmorg-14.0.6/compiler-rt/lib/a
  ◦ #1 0x55ca0a in __interceptor_memcmp /llvm-project-llvmorg-14.0.6/compiler-rt/lib/asan/./sanitizer_common/sanitizer_common_interceptors.inc:892:10
  ◦ #2 0x13c8c171 in png_sig_cmp /libpng-1.6.37/png.c:90:18
  ◦ #3 0x6332b8 in vision::image::decode_png(at::Tensor const&, long, bool) /vision/torchvision/csrc/io/image/cpu/decode_png.cpp:52:18
  ◦ #4 0x6025c0 in LLVMFuzzerTestOneInput /vision/decode_png.cc:34:32
  ◦ #5 0x668bc1 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned long) /llvm-project-llvmorg-14.0.6/compiler-rt/lib/fuzzer/FuzzerLoop.cpp:611:15
  ◦ #6 0x65204c in fuzzer::RunOneTest(fuzzer::Fuzzer*, char const*, unsigned long) /llvm-project-llvmorg-14.0.6/compiler-rt/lib/fuzzer/FuzzerDriver.cpp:324:6
  ◦ #7 0x65819b in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char const*, unsigned long)) /llvm-project-llvmorg-14.0.6/compiler-rt/lib/fuzzer/FuzzerDriver.cpp:860:9
  ◦ #8 0x651da2 in main /llvm-project-llvmorg-14.0.6/compiler-rt/lib/fuzzer/FuzzerMain.cpp:20:10
  ◦ #9 0x7ffff7a66082 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x24082) (BuildId: 1878e6b475720c7c51969e69ab2d276fae6d1dee)
  ◦ #10 0x541cbd in _start (/decode_png_fuzz+0x541cbd)
▶ AsanReport
▼ Source
  ◦ 86
  ◦ 87     if (start + num_to_check > 8)
  ◦ 88         num_to_check = 8 - start;
  ◦ 89
  ◦ --->90     return ((int)(memcmp(&sig[start], &png_signature[start], num_to_check)));
  ◦ 91     }
  ◦ 92
  ◦ 93     #endif /* READ */
  ◦ 94
  ◦ 95     #if defined(PNG_READ_SUPPORTED) || defined(PNG_WRITE_SUPPORTED)
```

Press q to exit

Crash Severity Estimation

Highlight likely critical and likely not critical crashes:

- **EXPLOITABLE:** PC overwriting, possible CWE-123 (write-what-where)
- **PROBABLY_EXPLOITABLE:** SIGILL, EXPLOITABLE cases with NULL values
- **NOT_EXPLOITABLE:** SIGABRT, SIGFPE, panics, exceptions, etc.

Provide short description about crash:

- Extract panic (Rust/Go) and exception (C++/Python/Java) messages

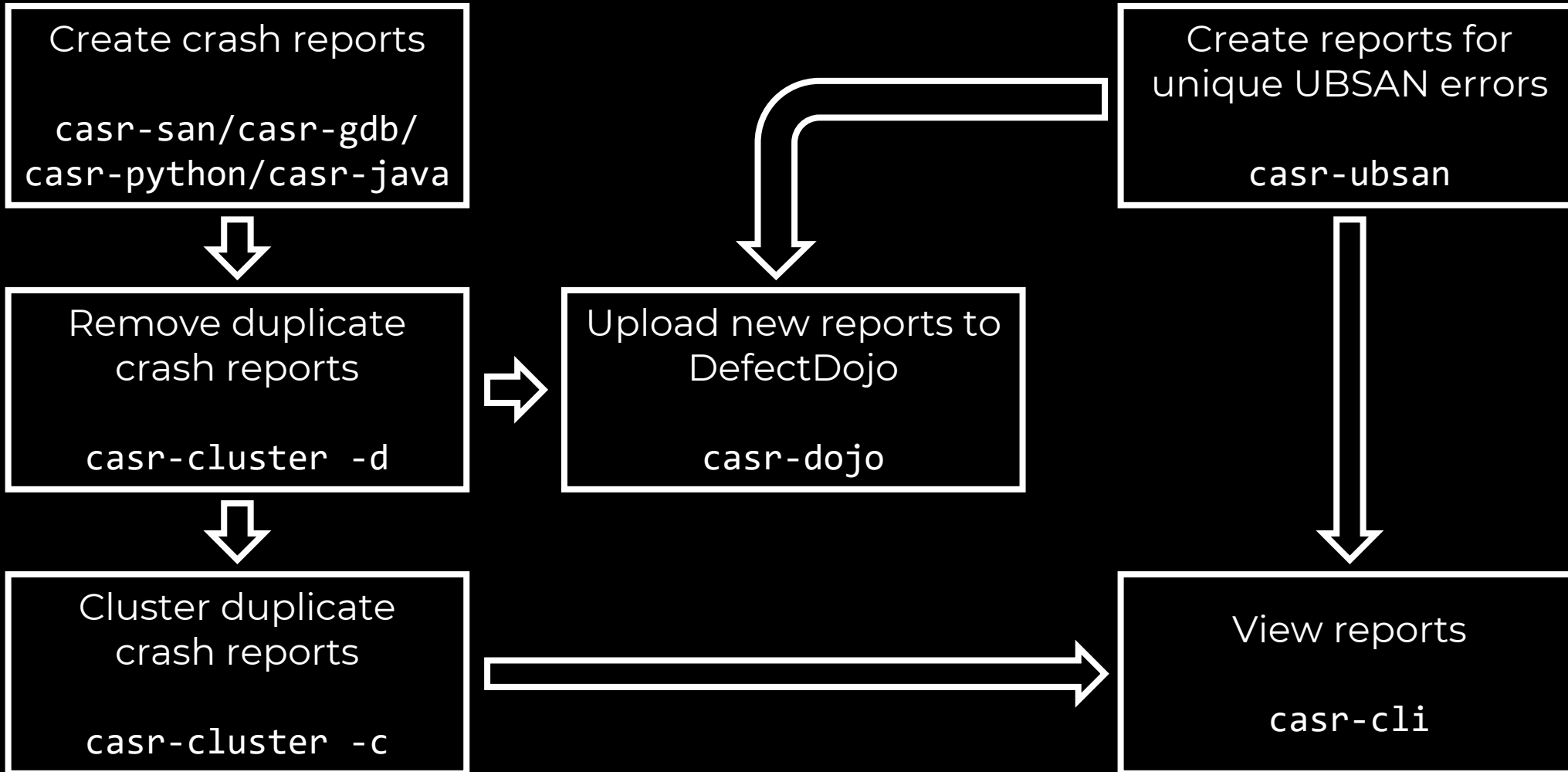




NO
FF
ONE
2023

Fuzzing Crash Triage Pipeline

Fuzzing Crash Triage Pipeline



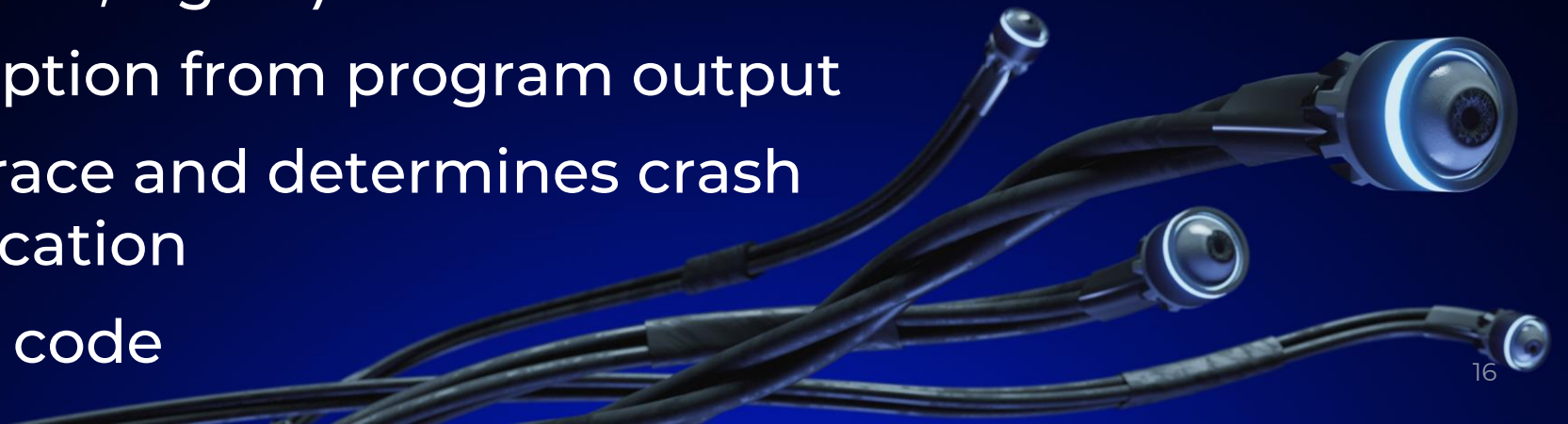
casr-san: Create CASR Reports for C/C++/Rust/Go



- Run program with crashing seed and parse its output
- Extract stack trace from Address Sanitizer report
- Parse native stack trace for Go
- Disable address space randomization for deterministic addresses in stack trace
- Get panic or exception message when present
- Get termination signal (SIGSEGV/SIGBUS/SIGABRT/SIGILL/etc.)
- Get stack trace from GDB when it is missing in output (e.g., AFL++ fuzz target abort)
- Filter standard library function calls, get crash line, and collect source code
- Estimate severity according to error type (e.g., memory writes are considered EXPLOITABLE)

casr-gdb: Create CASR Reports from GDB Execution

- Helps determine whether program still crashes without sanitizers
- Uses github.com/anfedotoff/gdb-command for executing GDB commands in batch mode
- Gets stack trace, signal info, mappings, registers, and disassembly from GDB
- Estimates severity based on crash state (disassembly, registers, signal)
- Extracts panic/exception from program output
- Parses GDB stack trace and determines crash line in triaged application
- Reads crash source code



CASR Reports for Python and Java

- `casr-python` and `casr-java` create CASR reports for Python and Java exceptions
- Exception is thrown by Python/Java application or Atheris/Jazzer harness
- Exception messages and stack traces are parsed from program output
- Stack traces are filtered to detect a crash line and extract source code
- When crash occurs in C/C++ native extensions, `casr-san` is launched for report creation



Removing Duplicate Crashes

`casr-cluster -d`

- Load all CASR reports
- Filter out noise from stack traces: standard library calls, `__GI_raise`, sanitizers, fuzzer internals, panics, exceptions, etc.
- Remove recursive function calls
- Crashes are considered duplicate when their stack traces are identical after filtering
- Remove duplicate crash reports



casr-cluster: Crash Reports Clustering

Clustering method from Microsoft ReBucket [paper](#):

- Pairwise comparison of stack traces based on metrics
- Hierarchical clustering



Integrating CASR Pipeline with Fuzzers

Crash triaging pipeline is automated for AFL++ (`casr-afl`) and libFuzzer/go-fuzz/Atheris/Jazzer (`casr-libfuzzer`):

- Parallel CASR reports creation
- Deduplication and clustering
- Copying crashing input seeds next to CASR reports
- Printing clusters summary
- Additional GDB reports for target built without sanitizers

```
casr-libfuzzer -i crashes -o casr-out -- /fuzz_target
```

```
casr-afl -i afl-out -o casr-out -- /gdb_target @@
```



```

root@lobster:~/crashes# casr-libfuzzer -o /out -- /load_fuzzer
19:29:30 [INFO] Generating CASR reports...
19:29:30 [INFO] Using 6 threads
19:31:47 [INFO] Deduplicating CASR reports...
19:31:48 [INFO] Number of reports before deduplication: 362. Number of reports after deduplication: 44
19:31:48 [INFO] Clustering CASR reports...
19:31:48 [INFO] Number of clusters: 23
==> <cl1>
Crash: /out/cl1/crash-03363a6f6e4e82231553c6c2ae3dabecd113a96a
casrep: NOT_EXPLOITABLE: SourceAv: /xlnt/source/detail/cryptography/compound_document.cpp:723:19
Similar crashes: 1
Crash: /out/cl1/crash-13844280659f4d4852fd353d1a1ac1f8b5642c01
casrep: NOT_EXPLOITABLE: SourceAvNearNull: /xlnt/source/detail/cryptography/compound_document.cpp:126:31
Similar crashes: 1
Crash: /out/cl1/crash-0f56e6b956f8db16ada867974f48b9ca3893a949
casrep: NOT_EXPLOITABLE: heap-buffer-overflow(read): /xlnt/source/detail/cryptography/compound_document.cpp:131:44
Similar crashes: 1
Crash: /out/cl1/crash-196a469592792cab6d3db8bd042df3c7fe7eafae
casrep: NOT_EXPLOITABLE: out-of-memory: /xlnt/source/detail/cryptography/compound_document.cpp:722:15
Similar crashes: 1
Cluster summary -> SourceAvNearNull: 1 out-of-memory: 1 SourceAv: 1 heap-buffer-overflow(read): 1
==> <cl2>
Crash: /out/cl2/crash-2fecf7233cb18f42aae3e43f595a1d6cd4bdfb9a
casrep: NOT_EXPLOITABLE: heap-use-after-free(read): /xlnt/source/detail/cryptography/compound_document.cpp:723:19
Similar crashes: 1
Cluster summary -> heap-use-after-free(read): 1
==> <cl3>
Crash: /out/cl3/crash-229c54a6c9bb6ee3f45eb766ac4d2c0c2797e712
casrep: NOT_EXPLOITABLE: std::length_error: /xlnt/source/./source/detail/binary.hpp:319:26
Similar crashes: 1
Crash: /out/cl3/crash-306cb534266351278307f0588cf71bd9a3bd59f4
casrep: NOT_EXPLOITABLE: heap-buffer-overflow(read): /xlnt/source/detail/cryptography/compound_document.cpp:131:44
Similar crashes: 1
Cluster summary -> std::length_error: 1 heap-buffer-overflow(read): 1
==> <cl4>
Crash: /out/cl4/crash-18ebf7db76ffe9fc403faaebc0003d166e0ecc44
casrep: EXPLOITABLE: heap-buffer-overflow(write): /xlnt/source/detail/cryptography/base64.cpp:148:36
Similar crashes: 3
Crash: /out/cl4/crash-2d4c70be32af1f8f215bbed568ed0972cd41541d
casrep: EXPLOITABLE: heap-buffer-overflow(write): /xlnt/source/detail/cryptography/base64.cpp:176:32
Similar crashes: 5
Cluster summary -> heap-buffer-overflow(write): 8
==> <cl5>
Crash: /out/cl5/crash-9e8da1bb79829362bd4324e66626997f5067385e
casrep: NOT_EXPLOITABLE: heap-buffer-overflow(read): /xlnt/source/detail/cryptography/compound_document.cpp:162:40
Similar crashes: 1
Cluster summary -> heap-buffer-overflow(read): 1
==> <cl6>
Crash: /out/cl6/crash-3cdf0550a8765c86422c76eb5c123c2e3c67fe10
casrep: EXPLOITABLE: heap-buffer-overflow(write): /xlnt/source/./source/detail/binary.hpp:278:9
Similar crashes: 1
Crash: /out/cl6/crash-075ad405d58938179932c9c10ba73b03dc9fdee5
casrep: PROBABLY_EXPLOITABLE: DestAvNearNull: /xlnt/source/./source/detail/binary.hpp:278:9
Similar crashes: 1
Cluster summary -> DestAvNearNull: 1 heap-buffer-overflow(write): 1
==> <cl7>

```



NO
FF
ONE
2023

Undefined Behavior Sanitizer Errors Triage



casr-ubsan: Triage Undefined Behavior Sanitizer Errors

- Run program with all seeds from corpus and crashes
- Extract UBSAN runtime errors
- Create CASR report for each error
- Deduplicate CASR reports based on crash line



```
/casr-out/corpus_0a7e4fe51b043113c48baf3be9f2344e6b99c9f6_identify.cpp_1868_24.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/identify.cpp:1868:24
/casr-out/corpus_c32a7ea7088bda26c3d2e4ec953703dcc36a4a3f_identify.cpp_2536_28.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/identify.cpp:2536:28
/casr-out/corpus_7ba4a0ae2b539add93cc3734d5d3d7fbdda2ca32_identify.cpp_631_52.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/identify.cpp:631:52
/casr-out/corpus_000c90f693e460b349b8ffba09abd7476b156ae3_identify_tools.cpp_97_20.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/identify_tools.cpp:97:20
/casr-out/corpus_0063adeb45f8fbbeed58d6ab8009faa5434fab42_kodak.cpp_103_17.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/kodak.cpp:103:17
/casr-out/corpus_03761a7aa5fdb62d991f4286d2ed2813b5932562_makernotes.cpp_189_17.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/makernotes.cpp:189:17
/casr-out/corpus_0d7798be846c08a33fc37b2c1a26ac7b08ef6901_makernotes.cpp_34_17.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/makernotes.cpp:34:17
/casr-out/corpus_004c9aaeefb79d11fa5043c54080b37b1cdaaf1_makernotes.cpp_520_17.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/makernotes.cpp:520:17
/casr-out/corpus_5da37c7106c36101936a65dfc6d360784c56f924_mediumformat.cpp_177_26.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/mediumformat.cpp:177:26
/casr-out/corpus_0b926951a80381fbfadb4a848228e1e41276a29b_mediumformat.cpp_194_27.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/mediumformat.cpp:194:27
/casr-out/corpus_03cbe953f0d445e49837c5b20c328b5a2065909d_mediumformat.cpp_268_19.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/mediumformat.cpp:268:19
/casr-out/corpus_03cbe953f0d445e49837c5b20c328b5a2065909d_mediumformat.cpp_36_3.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/mediumformat.cpp:36:3
/casr-out/corpus_03cbe953f0d445e49837c5b20c328b5a2065909d_mediumformat.cpp_41_17.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/mediumformat.cpp:41:17
/casr-out/corpus_03cbe953f0d445e49837c5b20c328b5a2065909d_mediumformat.cpp_48_5.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/mediumformat.cpp:48:5
/casr-out/corpus_135afb2f96da36333bf475e5a8e6c81aaf471685_misc_parsers.cpp_144_3.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/misc_parsers.cpp:144:3
/casr-out/corpus_49cfb68c2d8c8e9da7649f47636f4736ed39b6cd_misc_parsers.cpp_155_3.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/misc_parsers.cpp:155:3
/casr-out/corpus_0e8f52478be11b679d87254c2c6f497861cdd450_misc_parsers.cpp_171_19.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/misc_parsers.cpp:171:19
/casr-out/corpus_00fec81cfb9cf077b7ce56fb2ba87017ff2a643_pentax.cpp_472_43.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/pentax.cpp:472:43
/casr-out/corpus_0da286a79e0306929ac062092f25edf4697c62b2_tiff.cpp_1475_55.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/tiff.cpp:1475:55
/casr-out/corpus_0fceb8d8f96900e2212c0fe86038c6c923692ef3_tiff.cpp_253_29.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/tiff.cpp:253:29
/casr-out/corpus_2ea8d4eac160f7f9d81d2a63b3abece616d60fa_tiff.cpp_282_58.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/tiff.cpp:282:58
/casr-out/corpus_000083715a8e3d58c97c9d7592668566710180f9_tiff.cpp_52_17.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/tiff.cpp:52:17
/casr-out/corpus_0058d2eb403f042ac58c7d079d94c77f0b9ef5c6_tiff.cpp_566_17.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/tiff.cpp:566:17
/casr-out/corpus_5f465422dda913e39a7a3cb7627adfe8c5fded83_tiff.cpp_907_9.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/tiff.cpp:907:9
/casr-out/corpus_004c9efc45a8306c724cddd4c6812d85b2925a61_tiff.cpp_944_15.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/tiff.cpp:944:15
/casr-out/corpus_2c3fd2f72e570b137d8cd4c574232db0e6378ca7_open.cpp_586_9.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/Utils/open.cpp:586:9
/casr-out/corpus_000083715a8e3d58c97c9d7592668566710180f9_utils_dcrow.cpp_328_12.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/Utils/Utils_dcrow.cpp:328:12
/casr-out/corpus_000083715a8e3d58c97c9d7592668566710180f9_utils_dcrow.cpp_329_5.casrep: unsigned-integer-overflow: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/Utils/Utils_dcrow.cpp:329:5
SUMMARY -> float-cast-overflow: 14 implicit-integer-sign-change: 170 implicit-signed-integer-truncation: 84 implicit-signed-integer-truncation-or-sign-change: 12 implicit-unsigned-integer-truncation: 23 invalid-null-argument: 3 invalid-shift-base: 13 invalid-shift-exponent: 2 misaligned-pointer-use: 2 out-of-bounds-index: 1 signed-integer-overflow: 16 unsigned-integer-overflow: 3
```


casr-ubsan: Report Example

```
Crash Report for /load_from_memory_tiff_fuzzer
Severity: NOT_EXPLOITABLE: unsigned-integer-overflow
Crash line: /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/tiff.cpp:282:58
```

```
▶ Date
▶ Uname
▼ OS
  ◦ Ubuntu
▼ OSRelease
  ◦ 20.04
▼ Architecture
  ◦ amd64
▶ ExecutablePath
▼ ProcCmdline
  ◦ /load_from_memory_tiff_fuzzer corpus/2ea8d4eac160f7f9d81d2a63b3abeece616d60fa
▼ CrashSeverity
  ◦ NOT_EXPLOITABLE
  ◦ unsigned-integer-overflow
  ◦ unsigned integer overflow: 5046272 + 4294967295 cannot be represented in type 'unsigned int'
▶ ProcEnviron
▼ Stacktrace
  ◦ #0 0x2964690 in LibRaw::parse_tiff_ifd(int) /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/tiff.cpp:282:58
  ◦ #1 0x2a10a8c in LibRaw::parse_tiff(int) /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/tiff.cpp:1546:9
  ◦ #2 0x24eae79 in LibRaw::identify() /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/metadata/identify.cpp:512:14
  ◦ #3 0x10d6b03 in LibRaw::open_datastream(LibRaw_abstract_datastream*) /freeimage-svn/FreeImage/trunk/Source/LibRawLite/src/Utils/open.cpp:480:4
  ◦ #4 0x77fb68 in Validate(FreeImageIO*, void*) /freeimage-svn/FreeImage/trunk/Source/FreeImage/PluginRAW.cpp:645:21
  ◦ #5 0x59a6f0 in FreeImage_ValidateFIF /freeimage-svn/FreeImage/trunk/Source/FreeImage/Plugin.cpp:811:95
  ◦ #6 0x57a8dd in FreeImage_GetFileTypeFromHandle /freeimage-svn/FreeImage/trunk/Source/FreeImage/GetType.cpp:47:10
  ◦ #7 0x57af84 in FreeImage_GetFileTypeFromMemory /freeimage-svn/FreeImage/trunk/Source/FreeImage/GetType.cpp:109:10
  ◦ #8 0x524bcc in LLVMFuzzerTestOneInput /load_from_memory_tiff_fuzzer.cc:33:27
  ◦ #9 0x44b421 in fuzzer::Fuzzer::ExecuteCallback(unsigned char const*, unsigned long) /llvm-project-llvmorg-14.0.6/compiler-rt/lib/fuzzer/FuzzerLoop.cpp:611:15
  ◦ #10 0x43532c in fuzzer::RunOneTest(fuzzer::Fuzzer*, char const*, unsigned long) /llvm-project-llvmorg-14.0.6/compiler-rt/lib/fuzzer/FuzzerDriver.cpp:324:6
  ◦ #11 0x43b07b in fuzzer::FuzzerDriver(int*, char***, int (*)(unsigned char const*, unsigned long)) /llvm-project-llvmorg-14.0.6/compiler-rt/lib/fuzzer/FuzzerDriver.cpp:860:9
  ◦ #12 0x464632 in main /llvm-project-llvmorg-14.0.6/compiler-rt/lib/fuzzer/FuzzerMain.cpp:20:10
  ◦ #13 0x7f84bb5d2082 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x24082) (BuildId: 1878e6b475720c7c51969e69ab2d276fae6d1dee)
  ◦ #14 0x42fc4d in _start (/load_from_memory_tiff_fuzzer+0x42fc4d)
▶ UbsanReport
▼ Source
  ◦ 278         if ((utmp = get2()) ilm.LensID = utmp;
  ◦ 279         } else if ((imPana.LensManufacturer != 0xff) &&
  ◦ 280                 (imPana.LensManufacturer != 0xffffffff)) {
  ◦ 281             if ((utmp = (fgetc(ifp) << 8) | fgetc(ifp)))
  ◦ --->282                 ilm.LensID = (imPana.LensManufacturer << 16) + utmp;
  ◦ 283         }
  ◦ 284         break;
  ◦ 285         case 0x1203: /* 4611, FocalLengthIn35mmFormat, contained in 0x0120
  ◦ 286                     CameraIFD */
  ◦ 287             if (imgdata.lens.FocalLengthIn35mmFormat < 0.65f)
```

Press q to exit



NO
FF
ONE
2023

Vulnerability Management with DefectDojo

casr-doj: Upload New and Unique CASR Reports to DefectDojo



- Get all active, false positive, and out of scope findings from DefectDojo
- Compute filtered stack trace hashes (or get crash lines for UBSAN reports) for downloaded findings
- Upload new CASR reports to DefectDojo that have unique filtered stack trace hashes (or unique crash lines for UBSAN reports)
- Each finding will have a generated description with CASR report fields like crash line, severity, error description, source, stack trace, etc.
- Furthermore, casr-doj uploads CASR report, GDB CASR report, and crash seed files for corresponding finding

<input type="checkbox"/>	:	Critical	[XInt] [Load_fuzzer] Heap-Buffer-Overflow(write) in /xInt/s... </>	June 7, 2023	0	7	Admin User (admin)	API Test, Sydr-Fuzz DAST Report	Active
<input type="checkbox"/>	:	Critical	[XInt] [Load_fuzzer] Heap-Buffer-Overflow(write) in /xInt/s... </>	June 7, 2023	0	7	Admin User (admin)	API Test, Sydr-Fuzz DAST Report	Active
<input type="checkbox"/>	:	Critical	[XInt] [Load_fuzzer] Heap-Buffer-Overflow(write) in /xInt/s... </>	June 7, 2023	0	7	Admin User (admin)	API Test, Sydr-Fuzz DAST Report	Active
<input type="checkbox"/>	:	Critical	[XInt] [Load_fuzzer] Heap-Buffer-Overflow(write) in /xInt/s... </>	June 7, 2023	0	7	Admin User (admin)	API Test, Sydr-Fuzz DAST Report	Active
<input type="checkbox"/>	:	Critical	[XInt] [Load_fuzzer] Heap-Buffer-Overflow(write) in /xInt/s... </>	June 7, 2023	0	7	Admin User (admin)	API Test, Sydr-Fuzz DAST Report	Active
<input type="checkbox"/>	:	Critical	[XInt] [Load_fuzzer] Heap-Buffer-Overflow(write) in /xInt/s... </>	June 7, 2023	0	7	Admin User (admin)	API Test, Sydr-Fuzz DAST Report	Active
<input type="checkbox"/>	:	Critical	[XInt] [Load_fuzzer] Heap-Buffer-Overflow(write) in /xInt/s... </>	June 7, 2023	0	7	Admin User (admin)	API Test, Sydr-Fuzz DAST Report	Active
<input type="checkbox"/>	:	Critical	[XInt] [Load_fuzzer] Heap-Buffer-Overflow(write) in /xInt/s... </>	June 7, 2023	0	7	Admin User (admin)	API Test, Sydr-Fuzz DAST Report	Active
<input type="checkbox"/>	:	Critical	[XInt] [Load_fuzzer] Heap-Buffer-Overflow(write) in /xInt/s... </>	June 7, 2023	0	7	Admin User (admin)	API Test, Sydr-Fuzz DAST Report	Active
<input type="checkbox"/>	:	High	[XInt] [Load_fuzzer] DestAvNearNull in /xInt/source /../sour... </>	June 7, 2023	0	30	Admin User (admin)	API Test, Sydr-Fuzz DAST Report	Active
<input type="checkbox"/>	:	Medium	[XInt] [Load_fuzzer] Heap-Buffer-Overflow(read) in /xInt/so... </>	June 7, 2023	0	90	Admin User (admin)	API Test, Sydr-Fuzz DAST Report	Active
<input type="checkbox"/>	:	Medium	[XInt] [Load_fuzzer] Heap-Buffer-Overflow(read) in /xInt/so... </>	June 7, 2023	0	90	Admin User (admin)	API Test, Sydr-Fuzz DAST Report	Active
<input type="checkbox"/>	:	Medium	[XInt] [Load_fuzzer] SourceAv in /xInt/source/utlis /path.cp... </>	June 7, 2023	0	90	Admin User (admin)	API Test, Sydr-Fuzz DAST Report	Active

[XInt] [Load_fuzzer] Heap-Buffer-Overflow(write) in /xInt/source/./source/detail/binary.hpp:278:9
 Last Reviewed today by Admin User (admin), Last Status Update today, Created today


ID	Severity	SLA	Status	Type	Date discovered	Age	Reporter	CWE	Vulnerability Id	Found by
19	Critical	7	Active	Static	June 7, 2023	0 days	Admin User (admin)			API Test Sydr-Fuzz DAST Report

Location	Line Number
/xInt/source/./source/detail/binary.hpp	278

Similar Findings (2)



Description

Severity: EXPLOITABLE: heap-buffer-overflow(write): Heap buffer overflow

The target writes data past the end, or before the beginning, of the intended heap buffer.

GDB severity (without ASAN): EXPLOITABLE: DestAv: Access violation on destination operand

The target crashed on an access violation at an address matching the destination operand of the instruction. This likely indicates a write access violation, which means the attacker may control the write address and/or value.

Command: /load_fuzzer -artifact_prefix=/fuzz/sydr-fuzz-out/crashes/ -verbosity=2 -rss_limit_mb=8192 -timeout=10 -close_fd_mask=1 /fuzz/sydr-fuzz-out/crashes/crash-3cdf0550a8765c86422c76eb5c123c2e3c67fe10

OS: Ubuntu 20.04

Architecture: amd64

Source

```

274         {
275             throw xInt::exception("reading past end");
276         }
277
--->278         std::memcpy(data_ -> data() + offset_, reader.data() + reader.offset(), reader_element_count * sizeof(U));
279         offset_ += reader_element_count * sizeof(U) / sizeof(T);
280     }
```

NO
FF
ONE
2023

Exporting Crash Reports to SARIF



Export CASR Reports to SARIF



43 SARIF Results - xlint - Visual Studio Code

File Edit Selection View Go Run Terminal Help

source > detail > cryptography > compound_document.cpp > {} xlint > {} detail > read_sector<T>(sector_id, binary_writer<T>&)

```
564 stream_out_.rdub(stream_out_writer_.get());
565
566 return stream_out_;
567
568
569 template <typename T>
570 void compound_document::write_sector(binary_reader<T> &reader, sector_id id)
571 {
572     out_>seekp(static_cast<std::ptrdiff_t>(sector_data_start() + sector_size() * static
573     out_>write(reinterpret_cast<const char *>(reader.data() + reader.offset()),
574     static_cast<std::ptrdiff_t>(std::min(sector_size(), reader.bytes() - reader.offse
575 }
576
577 template <typename T>
578 void compound_document::write_short_sector(binary_reader<T> &reader, sector_id id)
579 {
580     auto chain = follow_chain(entries[0].start, sat_);
581     auto sector_id = chain[static_cast<std::size_t>(id) / (sector_size()) / short_sector_s
582     auto sector_offset = static_cast<std::size_t>(id) % (sector_size()) / short_sector_siz
583     out_>seekp(static_cast<std::ptrdiff_t>(sector_data_start() + sector_size() * static
584     out_>write(reinterpret_cast<const char *>(reader.data() + reader.offset()),
585     static_cast<std::ptrdiff_t>(std::min(short_sector_size(), reader.bytes() - reader
586 }
587
588 template <typename T>
589 void compound_document::read_sector(sector_id id, binary_writer<T> &writer)
590 {
591     in_>seekg(static_cast<std::ptrdiff_t>(sector_data_start() + sector_size() * static_c
592     std::vector<byte> sector(sector_size(), 0);
593     in_>read(reinterpret_cast<char *>(sector.data()), static_cast<std::ptrdiff_t>(sector
594     writer.append(sector);
595 }
596
597 template <typename T>
598 void compound_document::read_sector_chain(sector_id start, binary_writer<T> &writer)
599 {
600     for (auto link : follow_chain(start, sat_))
601     {
602         read_sector(link, writer);
603     }
604 }
605
606 template <typename T>
607 void compound_document::read_sector_chain(sector_id start, binary_writer<T> &writer, sect
608 {
609     auto chain = follow_chain(start, sat_);
610     for (auto i = std::size_t(0); i < count; ++i)
611     {
612         read_sector(chain[offset + i], writer);
613     }
614 }
615
616
617 template <typename T>
618 void compound_document::read_short_sector(sector_id id, binary_writer<T> &writer)
619 {
620     const auto container_chain = follow_chain(entries[0].start, sat_);
```

LOCATIONS 9 RULES 12 LOGS 1

Filter results

- compound_document.cpp source/detail/cryptography 22
 - 126 SourceAvNearNull: /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-b6ceb0128223a479...
 - 126 SourceAvNearNull: /load_sydr /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-b6ceb0128223a479...
 - 131 heap-buffer-overflow(read): /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-50fa1bd0...
 - 131 heap-buffer-overflow(read): /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-1dea9a96...
 - 162 heap-buffer-overflow(read): /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-28a198ee...
 - 162 heap-buffer-overflow(read): /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-975900d1...
 - 162 heap-buffer-overflow(read): /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-a914328c...
 - 162 heap-buffer-overflow(read): /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-af091513b...
 - 592 std::length_error: /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-061f4a6ee0752b9ba...
 - 592 std::length_error: /load_sydr /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-061f4a6ee0752b9ba...
 - 592 std::length_error: /load_sydr /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-388cadc865eea4e40...
 - 592 std::length_error: /load_sydr /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-388cadc865eea4e40...
 - 723 SourceAv: /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-1ed8211b0d8cf3ae4bec081...
 - 723 SourceAv: /load_sydr /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-1ed8211b0d8cf3ae4bec081...
 - 723 SourceAvNearNull: /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-b455e8d4b39dc2ab...
 - 723 SourceAvNearNull: /load_sydr /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-b455e8d4b39dc2ab...
 - 723 SourceAv: /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-8606aba06cc6efe7c28b6377...
 - 723 SourceAv: /load_sydr /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-8606aba06cc6efe7c28b6377...
 - 965 SourceAvNearNull: /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-4f0a45f6f86b1d16cca...
 - 965 AccessViolation: /load_sydr /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-4f0a45f6f86b1d16cca...
 - 975 SourceAv: /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-3433b5d76d6454bc3fa023...
 - 975 SourceAv: /load_sydr /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-3433b5d76d6454bc3fa023...
- checked.h third-party/utf.cpp/utf8 5
 - 216 utf8::invalid_utf16: /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-6b62b9e458f3190f...
 - 216 utf8::invalid_utf16: /load_sydr /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-7be94a1d94ba6a21...
 - 224 utf8::invalid_utf16: /load_sydr /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-6b62b9e458f3190f...
 - 224 utf8::invalid_utf16: /load_sydr /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-7be94a1d94ba6a21...
 - 224 utf8::invalid_utf16: /load_sydr /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-a345b01d88e51a2d...
- xlsx_consumer.cpp source/detail/serialization 4
 - 2044 SourceAvNearNull: /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-1a8513ab64eedcb8...
 - 2044 SourceAvNearNull: /load_sydr /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-1a8513ab64eedcb8...
 - 2969 std::invalid_argument: /load_af1 < /builds/dse/gitlab-jobs/oss-sydr-fuzz/projects/xlint/sydr-fuzz-af1++out/crashes/crash-50432025b5149...

Stacktrace

- #0 xlint::detail::compound_document::read_sector<int> (this=0x7fffffffaf00, id=<optimized out>, writer=...) compound_document.cpp 592:1
- #1 xlint::detail::compound_document::read_sat (this=0x2) compound_document.cpp 127:3:1
- #2 xlint::detail::compound_document::compound_document (this=0x7fffffffaf00, in=...) compound_document.cpp 515:1
- #3 (anonymous namespace)::decrypt_xlsx (bytes=..., password=...) xlsx_crypto_consumer.cpp 320:1
- #4 xlint::detail::decrypt_xlsx (data=..., password=...) xlsx_crypto_consumer.cpp 339:1
- #5 xlint::detail::xlsx_consumer::read (this=<optimized out>, source=..., password=...) xlsx_crypto_consumer.cpp 345:1
- #6 xlint::workbook::load (this=0x7fffffff570, stream=...) workbook.cpp 901:1
- #7 xlint::workbook::load (this=<optimized out>, data=...) workbook.cpp 919:1
- #8 LLVMFuzzerTestOneInput (data=0x15b3380 <__af1_fuzz_alt="320\317\021\340\241\261\032\341", size=<optimized out>) load_fuzzer.cc 9:1

NO
FF
ONE
2023

LibCASR: API for Crash Triage



LibCASR: Crash Triage API

LibCASR's Rust API:

- Stack trace parsing
- Crash report collection
- Crash triaging (deduplication and clustering)
- Crash severity estimation

Crash source: ASAN, UBSAN, GDB

Program languages: C/C++/Rust/Go/Python/Java

Architectures: x86/ARM/RISC-V

crates.io/crates/libcasr



Integrating LibCASR and LibAFL

```

/// parse ASAN error output emitted by the target command and compute the hash
pub fn parse_asan_output(&mut self, output: &str) {
    let mut hash = 0;
    if let Ok(st_vec) = AsanStacktrace::extract_stacktrace(output) {
        if let Ok(mut stacktrace) = AsanStacktrace::parse_stacktrace(&st_vec) {
            stacktrace.filter();
            let mut s = DefaultHasher::new();
            stacktrace.hash(&mut s);
            hash = s.finish();
        }
    }
    self.update_hash(hash);
}

```

Fuzzer example: github.com/anfedotoff/libafl_casr_forserkver_xlnt

Conclusion

CASR is a compound tool set and library that has plenty of benefits:

- Crash report creation with all needed information for manual analysis
- Significant reduction of crashes to be analyzed manually
- Integration with modern fuzzers (libFuzzer, AFL++, go-fuzz, Atheris, Jazzer) and fuzzing frameworks (LibAFL)
- Integration with DefectDojo vulnerability management system
- Support of multiple processor architectures (x86, amd64, arm32, aarch64, RISC-V)

Stargazing is very much appreciated!

github.com/ispras/casr



Questions?



github.com/ispras/casr