

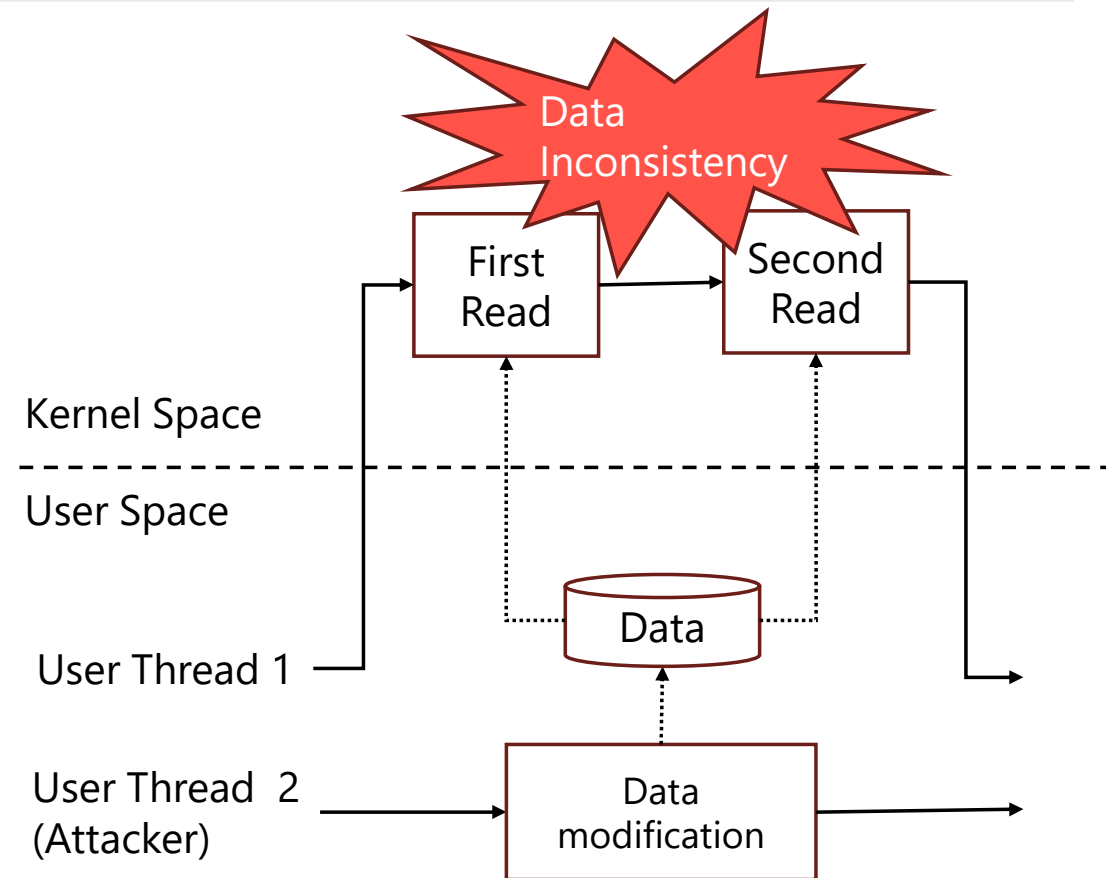


Toward a Practical Double-Fetch Checker for Clang Static Analyzer: Early Results and Future Directions for OS Security

Fumiya Shigemitsu
f-shigemitsu@esol.co.jp

Why Clang Static Analyzer, Why Double-Fetch?

- Great commercial tools \neq always care about what we need
 - Some checks we want are missing, unsupported, or unsuitable for our use-case
- Clang/LLVM = open, hackable, and already a part of our tool-chain
 - Clang Static Analyzer allows us to write custom rules to meet our needs
- We develop Real-Time Operating Systems (RTOS) – a kind of OS
 - Requires specialized validation
- Pick one serious OS bug → Double-Fetch
 - Serious, OS-specific, and ideal for learning custom checker design



Checker Overview

Memory Access Provenance

- Record where each load comes from.
- Tag each access with a unique ID.

Double Fetch Detection

- Detect when data with mismatched provenance tags is consumed.

Alias relationships

- Track tags through all symbol aliases.
- Ensure that any alias retains its original provenance tag.

Control-flow analysis

- Propagate tags into if/else paths.
- Make a branch “tagged” whenever its condition uses tagged data.

```
C df_simple_test.c 2, U
clang > test > Analysis > C df_simple_test.c > kernel_func
13 void kernel_func(MSG __user *uptr, int n) {
16
17     /*----First fetch (t1)----*/
18     get_user(x: &len1, p: &uptr->len); /*t1*/
19
20     /*
21     * malloc is a sink point in taint analysis: if len1 is tainted,
22     * it influences memory allocation size, which may lead to security
23     * issues.
24     */
25     buf = malloc(size: len1);
26     if (!buf)
27         return;
28
29     /*----Second fetch (t2)----*/
30     get_user(x: &len2, p: &uptr->len); /*t2*/
31
32     /*
33     * Use len2 to validate the size before copying text from user space.
34     * If len2 originates from a different structure than t1, this could
35     * indicate a double-fetch bug.
36     */
37     if (len2 < BUF_SIZE) {
38         copy_from_user(to: buf, from: uptr->text, n: len2);
39     }
40
41     free(ptr: buf);
42 }
```

```
PROBLEMS 2 TERMINAL OUTPUT ... bash - build + v [icons] ... < x
f-shigemitsu@sw-coreos-lin3:~/llvm-project/build$ bin/clang --analyze
-Xanalyzer -analyzer-checker=alpha.security.DoubleFetchChecker ../c
lang/test/Analysis/df_simple_test.c
../clang/test/Analysis/df_simple_test.c:37:9: warning: Double-fetch d
etected: use of untrusted data [alpha.security.DoubleFetchChecker]
    37 |         copy_from_user(buf, uptr->text, len2);
        |         ^~~~~~
1 warning generated.
f-shigemitsu@sw-coreos-lin3:~/llvm-project/build$
```

How The Checker Reduce False Positives

- Pointer changing
- Use of the double-fetched data

Reference:

- P. Wang, K. Lu, G. Li, and X. Zhou, "A survey of the double-fetch vulnerabilities," *Concurrency and Computation*, vol. 30, no. 6, p. e4345, Mar. 2018, doi: [10.1002/cpe.4345](https://doi.org/10.1002/cpe.4345).

Next Ideas

Future Ideas for RTOS/Hypervisor

- Product-specific API contract validation
- Interrupt Mask Duration policy checks
- Priority-Inversion detection
- Hypervisor Call validation

Additional Improvements for the Current Double-Fetch Checker:

- Application to large codebases
- Integration with the existing Clang Static Analyzer TOCTOU framework



Challenge with Passion