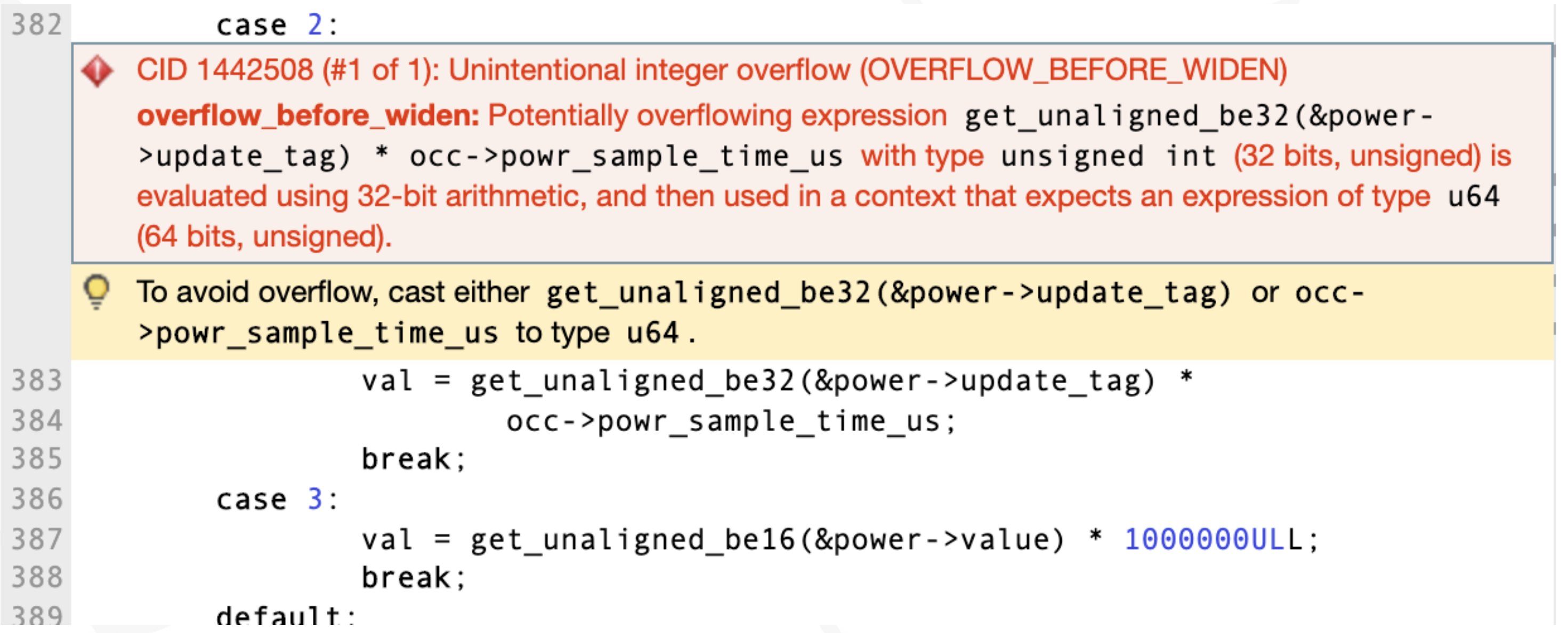
A Synopsis of Static Analysis Alerts On Open Source Software

Nasif Imtiaz, Laurie Williams
{simtiaz, lawilli3}@ncsu.edu
Department of Computer Science, North Carolina State University

How Do Developers Respond to Static Application Security Testing (SAST) Tool Alerts?

- Half of the state-of-the-art open source projects use SAST[1]
- Alerts are often *not* actionable[2] (important to developers to act upon)



```
        case 2:
```

CID 1442508 (#1 of 1): Unintentional integer overflow (OVERFLOW_BEFORE_WIDEN)
overflow_before_widen: Potentially overflowing expression `get_unaligned_be32(&power->update_tag) * occ->powr_sample_time_us` with type `unsigned int` (32 bits, unsigned) is evaluated using 32-bit arithmetic, and then used in a context that expects an expression of type `u64` (64 bits, unsigned).

To avoid overflow, cast either `get_unaligned_be32(&power->update_tag)` or `occ->powr_sample_time_us` to type `u64` .

```
                val = get_unaligned_be32(&power->update_tag) *
                        occ->powr_sample_time_us;
                break;
        case 3:
                val = get_unaligned_be16(&power->value) * 1000000ULL;
                break;
        default:
```

- The goal of this research is *to aid researchers in improving the usability of static application security testing tools by looking at what type of static analysis alerts are most likely to be acted on by OSS developers.*

Research Questions

1. What are the alert types that are most often introduced, triaged, and eliminated?
2. What are the alert types that are most likely to be unactionable?
3. What is the median lifespan of each alert type?
4. Are security alerts more likely to be triaged and eliminated than non-security alerts?

Dataset

- 5 projects – Linux, Firefox, Qt, Samba, Kodi
- Projects use a free SAST service – Coverity Scan
- Written in C/C++
- At least 100 analysis reports over last 5 year
- 24 alert categories
- Developers' triage history on Coverity Scan defect database

Findings

- Most Introduced → Control Flow Issues ← Mostly Intentional
- Most Triaged → Memory Illegal Access ← Mostly False Positive
- Most Eliminated → Performance Inefficiency ← Shortest Lifespan
- Most Likely Bug → Null Pointer Dereferences ← Short Lifespan
- Our cross-project comparison indicates that an alert being marked as a security issue by the tool does not affect its likelihood of getting fixed or its lifespan.

References

[1] Moritz Beller, Radjino Bholanath, Shane McIntosh, and Andy Zaidman. 2016. Analyzing the state of static analysis: A large-scale evaluation in open source software. In 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Vol. 1. IEEE, 470–481.
[2] C. Sadowski, E. Aftandilian, A. Eagle, L. Miller-Cushon, and C. Jaspan. Lessons from building static analysis tools at Google. CACM, 2018.

HOTSOS 2019
6TH ANNUAL HOT TOPICS in the SCIENCE OF SECURITY
APRIL 2-3, 2019 | NASHVILLE, TENNESSEE