# Understanding and Detecting Concurrency Attacks

Paper #82

The University of Hong Kong

@cs.hku.hk

## Abstract

## 1. Introduction

Concurrency program is hard to be correct. Concurrency bugs are common in modern multi-threaded programs[16, 17, 28, 29] including atomic violation, ..., and especially data race. Extant work well explore interleaving that cause concurrency bugs, and efficiently detect explicit concurrency bugs that direct to severe consequences such as execution order violation, wrong output and program crash.

Recent studies[21, 27] show rise of concern about *concurrency attacks*. By triggering concurrency bugs and employing subtle inputs, hackers may leverage the corrupted memory to construct attacks, including privilege escalations[], hijacking code execution[], bypassing security checks[], and breaking database integrity[21]. However, these vulnerabilities often hide in large amount of concurrency bug reports. For example, bug information leveraged by a xxx attack is hidden in 1000 race reports produced by TSAN[22], a famous and widely used data race detector. Also, to construct concurrency attacks, despite the inputs of leveraged concurrency bugs, attacker may need other crafted inputs to exploit the vulnerabilities. In CVE-2017-7533 exploited by our team, despite leveraging a data race bug to construct kernel heap overflow, we also require another thread to be the victim target, and finally achieve arbitrary code execution and get a root shell.

Unfortunately, although great progress has been made to detect and replay severe bugs (program crash[29]), extant work still lacks exploration of concurrency attacks from enormous concurrency bugs. Our study over several known concurrency bugs[1, 2] shows that even concurrency bugs have been successfully detected and reported, professionals still lack knowledge about how severe consequences these bugs may cause. For instance, *apache-25520*[1] has been reported over years and well studied by researchers[17]. We still firstly exploited a new heap overflow attack leveraging on this bug and break the HTML integrity.

## 2. Background

## 3. Overview

## 4. Reducing Benign Schedules

## 5. Discussions

## 6. Evaluation

## 7. Related Work

[]

## 8. Conclusion

## References

[1] Apache bug 25520. `https://bz.apache.org/bugzilla/show_bug.cgi?id=25520`.

[2] Apache bug 46215. `https://bz.apache.org/bugzilla/show_bug.cgi?id=46215`.

[3] S. Lu, S. Park, C. Hu, X. Ma, W. Jiang, Z. Li, R. A. Popa, and Y. Zhou. Muvi: automatically inferring multi-variable access correlations and detecting related semantic and concurrency bugs. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP '07)*, pages 103–116, 2007.

[4] S. Lu, S. Park, E. Seo, and Y. Zhou. Learning from mistakes: a comprehensive study on real world concurrency bug characteristics. In *Thirteenth International Conference on Architecture Support for Programming Languages and Operating Systems (ASPLOS '08)*, pages 329–339, Mar. 2008.

[5] P. B. Todd Warszawski. Acidrain: Concurrency-related attacks on database-backed web applications. In *Proceedings of the 2017 ACM SIGMOD International Conference on Management of Data*, pages 5–20. ACM, 2017.

[6] Threadsanitizer. `https://code.google.com/p/data-race-test/wiki/ThreadSanitizer`, 2015.

[7] J. Yang, A. Cui, S. Stolfo, and S. Sethumadhavan. Concurrency attacks. In *the Fourth USENIX Workshop on Hot Topics in Parallelism (HOTPAR '12)*, June 2012.

[8] W. Zhang, J. Lim, R. Olichandran, J. Scherpelz, G. Jin, S. Lu, and T. Reps. ConSeq: detecting concurrency bugs through se-

quential errors. In *Sixteenth International Conference on Architecture Support for Programming Languages and Operating Systems (ASPLOS '11)*, pages 251–264, Mar. 2011.

[9] W. Zhang, C. Sun, and S. Lu. ConMem: detecting severe concurrency bugs through an effect-oriented approach. In *Fifteenth International Conference on Architecture Support for Programming Languages and Operating Systems (ASPLOS '10)*, pages 179–192, Mar. 2010.