WThreadAFL:Deterministic Greybox Fuzzing for Multi-thread Network Servers

Jianjun Lu, Bo Yu[†]
College of Computer, National University of Defense Technology
Changsha, China
{lujianjun16, yubo0615}@nudt.edu.cn

ABSTRACT

Network servers often spawn a worker thread to deal with the incoming connection while the main thread remains listening as background thread. Random scheduling of multiple threads causes variations of the coverage of edges, which constrains the performance of the fuzzer. In this work, we present WThreadAFL, a new grey-box fuzzer for fuzzing multi-thread network servers. WThreadAFL addresses the non-deterministic problem based on a lightweight thread identification approach. We distinguish between worker thread and background thread via thread-context instrumentation, and only update coverage feedback of the worker thread. The experimental results on four popular server benchmarks show that, WThreadAFL behaves more deterministic than network protocol grey-box fuzzer AFLNET, which increases the stability metric by 1.9% — 25.7% and cuts down coverage variable edges by 19.0%—39.8% within 24 hours.

CCS CONCEPTS

• Security and privacy → Network security.

KEYWORDS

Grey-box Fuzzing; Networks Protocols; Coverage Feedback

ACM Reference format:

Jianjun Lu, Bo Yu. 2022. WThreadAFL:Deterministic Greybox Fuzzing for Multi-thread Network Servers. In *Proceedings of 6th Asia-Pacific Workshop on Networking (APNet 2022). ACM, New York, Fuzhou, China,* 2 pages. https://doi.org/10.1145/3542637.3542649

1 Introduction

Network servers are an important part of the IT infrastructures. The security flaws of them are particularly harmful since network servers are exposed to the whole Internet, and attackers can exploit them remotely in anywhere. Thus, it is of great significance to find and repair the vulnerability of network servers.

Coverage-guided Greybox Fuzzing (CGF) is a relevant vulnerability discovery technique and has been widespread adopted in testing file-based system(such as AFL[1]).Recent research[2,3,4] proposes to apply CGF to detect vulnerability in network servers. For instance, AFLNET[2] expands AFL with a network communication plugin and automated protocol state model inferencing, and successfully fuzzing up to 10 sorts of protocols.

However, the non-deterministic behaviors in fuzzing network servers greatly affect efficiency. Network servers often spawn a sub-thread or sub-process to handle the incoming connection ("worker thread") while the main thread remains listening ("background thread"). The worker thread contains key data and logic which we really care about. But the AFLNET's coverage-guided design is unaware of multithreading and mixes up multiple executions. The uncertain coverage feedback makes fuzzer difficult discerning between meaningful and "phantom" effects of mutating the input files. Thus, the efficiency for network servers fuzzing is severely constrained by the current thread-unaware coverage feedback.

In this work, we present WThreadAFL, a new grey-box fuzzer for deterministically fuzzing network servers. We inject code into the target server at compile-time through source code instrumentation technique. The injected code infers pure coverage feedback of worker thread by: tracking thread creations and collect identity of new thread; on each basic block, determining whether to run on the worker thread; if so, updating code coverage, otherwise not. In this way, we avoid the disturbance of background thread and gain more deterministic feedback. In summary, the main contributions of this paper are as follows.

- A new feedback that identify and focus on worker thread without interference of background thread based on lightweight thread identification approach;
- A prototype tool implementation called WThreadAFL, which has higher stability and fewer various edges during fuzzing network servers;
- An evaluation of four popular protocol benchmarks, and the result shows that WThreadAFL performs more deterministic than original coverage-guided protocol fuzzer AFLNET.

2 DESIGN AND IMPLEMENTATION

2.1 Feedback to be Thread-aware

AFLNET instruments the start point of each basic block. Once a basic block is executed, the injected code acts as deputy to provide coverage feedback indicating a new path or a state. We refer to the deputy instructions as Feedback-Instrumentation. Since Feedback-Instrumentation does not record thread identities,

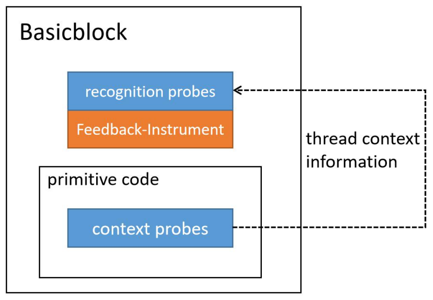

Figure 1 Thread-aware instrumentation in basic block

AFLNET cannot distinguish different threads. Therefore, we can add more deputy instructions to collect thread-context information for recognizing the worker thread by performing conditional judgment before Feedback-Instrumentation.

2.2 Instrumentation probes

To identify the worker thread in basic block level, we insert probes into the instructions of the target server. Probes consists of context probes and recognition probes, which are weaved at specific program sites that create new threads, and that in the front of entry instruction of each basic block. As the figure 1 shows, Context probes collect the context information of new threads, and record it as identifier. Recognition probes recognize if the current thread is worker thread by reading the identifier, and then determine whether to carry out the Feedback-Instrumentation or not. In this way, WThreadAFL minimizes the " noise " of background thread, and drive fuzzing with more deterministic coverage feedback.

2.3 Implementation

WThreadAFL is developed on top of AFLNET[2] and LLVM[5]. For coverage feedback, we expand the technical solution provided by AFLNET. Besides the original Feedback-Instrumentation, we add further instrumentation to inject probes in the program, as discussed earlier. The context probes are inserted on thread creation sites that invoke the standard APIs such as *fork* or *pthread_create*, in order to acquire thread information. And the recognition probes are introduced to the beginning of each basic block, so as to identify the worker thread. In this way, we successfully have the fuzzer focused on the worker thread and record its coverage feedback only.

3 EVALUATION

Configuration. We evaluate the performance of WThreadAFL using four multi-thread network servers: Forked-daapd, LightFTP, Pure-FTPd, Exim. All the programs have been widely used at present. Thus, we suppose these four programs are representative. We fuzzed each server for 24 hours and repeated 24 hours experiment 5 times for each test program to reduce the randomness of test results. All of our experiments was run on a 64-bit machine with 16 cores (2.1 GHz Intel Xeon Silver 4216), 125GB of RAM, and Ubuntu 16.04 as server OS.

TABLE 1 : Stability of two fuzzers

	AFLNET	WThreadAFL
Forked-daapd	4.44	6.36(↑1.92)
LightFTP	79.79	85.52(↑5.73)
Pure-FTPd	17.91	28.53(↑10.62)
Exim	33.87	59.54(↑25.70)

Comparison. We compared WThreadAFL with grey-box protocol fuzzer AFLNET on the four servers. Firstly, we measured the degree of the stability metrics. As Table 1 shows, when fuzzing multi-thread network servers, WThreadAFL behaved more deterministic than AFLNET, which increased the stability metric by 1.9% — 25.7%. Then we further investigated the amount of deterministic and variable edges during 24 hours testing. As Table 2 shows, for deterministic edges, WThreadAFL raised the number among three targets by 1.2% — 70.0%, and lessen the number of one target by 7.6%. Since the worker thread contains the main logic and data, and our technique ignores the coverage feedback of the background thread, we consider a slight decrease in quantity is acceptable. As for variable edges, WThreadAFL cuts down the number not less than 19% and up to 39.8%. That result illustrates that our work is useful for solving non-deterministic problem in protocol fuzzing.

TABLE 2 :Deterministic edges / Variable edges

	AFLNET	WThreadAFL	Ratio(%)
Forked-daapd	172/3973	159/2999	↓7.6 / ↓24.5
LightFTP	517/141	523/90	↑1.2 / ↓36.2
Pure-FTPd	271/1230	388/996	↑43.2 / ↓19.0
Exim	1610/3206	2737/1930	↑70.0 / ↓39.8

4 Conclusion

In this poster, we present WThreadAFL, a new grey-box fuzzer for fuzzing multi-thread network servers. Based on thread-aware instrumentation, we propose a novel technique to distinguish the worker thread, and gain deterministic coverage feedback during random scheduling of multiple threads. The experimental results on four popular server benchmarks demonstrate the effect of WThreadAFL.

REFERENCES

[1] Michal Zalewski. [n.d.]. Technical "whitepaper" for afl-fuzz.
[2] Van-Thuan Pham, Marcel Böhme, and Abhik Roychoudhury. 2020. AFLNET: A Greybox Fuzzer for Network Protocols. In Intl. Conf. on Software Testing, Verification and Validation (Testing Tools Track).
[3] Sarosys LLC. Framework - arachni - web application security scanner framework, 2019.
[4] Li J, Li S, Sun G, et al. SNPSFuzzer: A Fast Greybox Fuzzer for Stateful Network Protocols using Snapshots[J]. arXiv preprint arXiv:2202.03643, 2022.
[5] Chris Lattner and Vikram Adve. LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation. In Proc. of the 2004 Int. Symposium on Code Generation and Optimization (CGO'04), Mar 2004.