発表概要

バイナリコードを対象とした高速かつ正確な動的 Use-After-Free 検出

石山 泰地[1,a)]　荒堀 喜貴[1,b)]　権藤 克彦[1,c)]

2019年1月17日発表

近年，Use-After-Free（UAF）脆弱性を悪用した攻撃が数多く存在している．UAF 脆弱性は解放済みのメモリ領域を参照しているダングリングポインタを使用することで発生し，これを悪用することで任意コードの実行や情報漏洩などが引き起こされる可能性がある．これまで C ソースコードに対して高精度かつ低オーバヘッドで UAF 脆弱性の悪用を防ぐ手法が多く提案されてきた．一方で，バイナリレベルでの UAF 検出はオーバヘッドが大きく，リアルタイムでの保護に向かないものが多い．また，バイナリレベルではスタックフレームの解放に起因する UAF の検出も難しい．本発表ではバイナリコードを対象とした高速かつ正確な UAF 検出手法を提案する．本手法は FreeSentry をバイナリコードを対象に再実装した上で，計装ツールへの最適化と，ポインタ追跡に使う命令とレジスタを限定することで高速化を達成した．また，call 命令や ret 命令などを監視することでスタックに起因する UAF の検出も可能にした．検出精度について，自作の 11 のテストケースと NIST SAMATE benchmark suite の 38 のテストケースで誤検出や検出漏れは発生しなかった．また，LAVA を UAF 用に改造したものを使用して精度を確認した．オーバヘッドについて，gzip に本手法を適用すると実行時間は 7 倍程度になった．

Presentation Abstract

Efficient and Precise Dynamic Use-After-Free Detection for Executables

Taichi Ishiyama[1,a)]　Yoshitaka Arahori[1,b)]　Katsuhiko Gondow[1,c)]

Presented: January 17, 2019

Use-After-Free (UAF) is a memory bug in which a (dangling) pointer to a freed memory objects is dereferenced. UAFs are a notorious source of many security vulnerabilities such as arbitrary code execution and information leakage. There have been proposed a number of efficient and precise detection techniques for UAFs. However, most of them are implemented via source-code instrumentation, limiting the range of code checked. Binary instrumentation-based UAF detectors are, on the other hand, neither efficient nor precise. Especially, they are too inefficient to be used as an online UAF detector, and not able to detect UAFs over stack frames. In this work, we propose an efficient and precise dynamic UAF detector based on binary instrumentation. We extend FreeSentry, a source-level UAF detector, to precisely check binary execution via monitoring stack frames, while taming runtime overheads by optimizing binary instrumentation and limiting registers over which to perform pointer-tracking. Our experiments with 18 small benchmarks and 38 NIST SAMATE benchmarks show that our approach incurs neither false positives nor false negatives. In addition, the runtime overhead imposed by our approach was about 7x for gzip.

This is the abstract of an unrefereed presentation, and it should not preclude subsequent publication.

[1] 東京工業大学
Tokyo Institute of Technology, Meguro, Tokyo 152–8550, Japan
[a)] ishiyama@sde.cs.titech.ac.jp
[b)] arahori@cs.titech.ac.jp
[c)] gondow@cs.titech.ac.jp