発表概要

実行可能コードを対象とするスケーラブルかつ部分的パス依存なバッファオーバフロー静的検知

黒岩 将平[1,a)]　荒堀 喜貴[1]　権藤 克彦[1]

2019年1月17日発表

バッファオーバフロー検知は 20 年以上研究されているにもかかわらず脆弱性の中でも最も多いものの 1 つであり，セキュリティ上で深刻な問題を引き起こす原因になりうる．これまでに様々なバッファオーバフロー検知手法が提案されてきた．静的手法では記号実行を用いた手法があり，高い精度でセキュリティ上の脆弱性を検知できるが大規模なプログラムを網羅的に解析することが困難である．また，ソースコードやシンボル情報を必要とせずに大規模なプログラムにも対応した手法 (Kindermann, 2008) もあるが，スタック上で発生するバッファオーバフローのみを検知対象としている．そこで，本手法では実行ファイルのみを入力としたデバッグ情報やシンボル情報を必要とせずにスタックやヒープ，構造体内で発生するバッファオーバフローの検知手法を提案する．提案手法では x86 の実行ファイルを対象とし，関数内のメモリアクセスからメモリ上の値を過大近似する静的な数値とポインタ解析アルゴリズムの組み合わせた手法 Value-Set Analysis（VSA）(Balakrishnan and Reps, 2010) を元にした手法によりプログラム中の変数を復元し，プログラムのメモリアクセス範囲を計算することでバッファオーバフローの検知を目指す．さらに本手法では VSA に加えてバッファオーバフロー検知のために一部 Path-Sensitivity を加えた．実験では 100 万行を超えるコードの実行ファイルに対しても網羅的にバッファオーバフロー脆弱性の解析が可能なことを示す．

Presentation Abstract

Scalable and Partial Path Sensitive Static Detection of Buffer Overflows in Executables

Shohei Kuroiwa[1,a)]　Yoshitaka Arahori[1]　Katsuhiko Gondow[1]

Presented: January 17, 2019

Although Buffer overflow detection has been studied for more than 20 years, it is stil the most common source of security vulnerabilities. Various approaches has been proposed ever. Although symbolic analysis techniques can find such vulnerabilities with high precision, it does not scale to millions lines of code (MLOC). And it requires source code to analyze. (Kindermann, 2008) can analyze buffer overflows scalably without either source code or debug symbols, but it can not detect buffer overflows occured inside either heap allocated region or the field of structs. In this presentation, we employ Value-Set Analysis (VSA) (Balakrishnan and Reps, 2010) which is flow-sensitive and context-sensitive, a combined numeric-analysis and pointer-analysis algorithm to recover variables and overapproximate possible accessed locations to scalably detect possible buffer overflows which occurs on stack, heap, the field of structs. Moreover, we add partial path-sensitivity to VSA to improve the precision of our algorithm. Our experiments show that we can successfully analyze MLOC program.

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