様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　２８　年　　６　月　　２　日現在

機関番号：　１２６０１
研究種目：　若手研究(B)
研究期間：　2013～2015
課題番号：　２５７３００３９

研究課題名（和文）Webアプリケーションのセキュリティ分析の自動化

研究課題名（英文）Automated security analysis of web applications

研究代表者
ＬＩ　Ｘｉｎ（LI, Xin）

東京大学・情報理工学（系）研究科・研究員

研究者番号：６０５１０６４１

交付決定額（研究期間全体）：（直接経費）　　2,300,000円

研究成果の概要（和文）：アクセスコントロールは安全性が重要なシステムにおいて、安全性を確保するための第一歩である。しかしながら、現在においても、webアプリケーションのためのセキュリティポリシーは人手で設計されており、このことは新たな脆弱性を導入する原因となっている。この研究では、webアプリケーションのための、自動的なセキュリティポリシー生成のアルゴリズムを提案した。また、このアルゴリズムを実現するために、新しい静的解析アルゴリズムと条件付きのプッシュダウンシステムモデル検査アルゴリズムを、提案した。今後、これに基づいてwebアプリケーションのためのセキュリティポリシーの自動生成器の開発を実装する予定である。

研究成果の概要（英文）：This research is to apply program analysis and formal verification techniques to automated web security analysis, with a focus on automated generation of access control policies for Java-centric web applications. During the project, we first designed a systematic approach to automated generation of access control policies for Java programs to pass the runtime authorization. Next, to put the technique into practice, we studied efficient algorithms of those analysis modules underpinning the algorithmic framework. Last but not least, we also developed algorithms for model checking recursively-typed higher-order grammars and their potential application to web security analysis will be studied as future work. In the future, we plan to further elaborate the proposed algorithms and techniques, and conduct empirical studies of them, whereby eventually build a practical system for web security analysis.

研究分野：　ソフトウェア

キーワード：　Web Security　Access Control　Model Checking　Program Analysis

１．研究開始当初の背景

(1) The security of web applications is more important than ever. Nowadays people reply on web applications for social networking, e-commerce, and a variety of daily activities. In the meantime, the number of web-based attacks has also increased sharply. These attacks motivated a great deal of research, among which static program analysis and formal verification techniques like model checking play an important role in detecting security vulnerabilities.

(2) Access control is the first step to securing safety-critical systems. In Java-centric web applications and Microsoft .NET Common Language Runtime, stack inspection based access control mechanism is employed for runtime authorization. Although the stack inspection mechanism is well studied, there is relatively little work on automated generation of access control policies, and in practice, system administrators have to manually configure access control policies based on domain-specific knowledge and trail-and-error testing. Since testing cannot cover all program runtime behaviors, a benign program may encounter runtime authorization failures if it is assigned with too few permission. On the other hand, a program may become vulnerable points for malicious attacks if it is granted more permission than necessary.

(3) It is not an easy task to automatically generate access control policies for web applications. It requires a variety of static analyses to fulfill the security analysis task, and it is not enough clarified how to precisely coordinate different analysis modules in the same analysis framework. Besides, to obey the Principle of Least Privilege, we need precise analysis algorithms for identifying optimal access control policies for each web components, whereas applying precise static program analysis to production-level web applications would face the long-standing dilemma of making a compromise between precision and scalability.

２．研究の目的

(1) The general objective of this research is to develop and apply static program analysis and formal verification techniques to automated web security analysis. We expect our techniques are useful for solving security analysis problems in fields other than web security analysis, such as security analysis of mobile computing software.

(2) Specially, as our first step, we focus on developing algorithms, techniques and tools that automatically generate optimal access control policies for server-side Java programs to pass runtime authorization. The techniques can help system administrators to configure access control policies when deploying web applications. In the meantime, the automatically generated access control policies can also be used to find malicious components from domains of low-level trust that attempt to access protected resources requiring high-level trust.

(3) There is relatively little work on automated generation of access control policies. We are aware of considerable research efforts from IBM researchers on automatically generating access control policies for web applications such as [①] and [②]. In their work, they invented many novel ideas for scaling-up their permission analysis framework, by making reasonable tradeoffs between precision and practical efficiency. Unfortunately, to our knowledge, their techniques are protected by the U.S. patents. As our final target, we aim to build an open source automatic generator for access control policies.

３．研究の方法

(1) We are concerned with applying precise context-sensitive static program analysis and formal verification techniques to web security analysis. In contrast to testing, the static analysis and verification techniques conservatively model the program's runtime behaviors, given abstract interpretation is properly applied to the program beforehand. Besides, using precise context-sensitive analysis techniques ensures to generate optimal access control policies.

(2) Specially, we are concerned with investigating the reachability problem of conditional (weighted) pushdown systems that naturally model many security analysis problems such as stack inspection in which the program call stack is examined

at runtime. Reachability analysis of conditional pushdown systems is the cornerstone of access rights analysis, but the problem is proved to be intractable in general.

(3) For practical scalability, we study new algorithms for those analysis modules underpinning the algorithmic framework for permission analysis, driven by practical applications and demands, so as to scale the analysis to real-world instances.

4. 研究成果

(1) First, we designed a systematic approach to automated generation of access control policies for (sever-side) Java programs to pass the runtime authorization based on stack inspection. The proposed algorithmic framework for access rights analysis is designed using techniques of abstract interpretation and a variety of context-sensitive static program analyses. Our latest report for the algorithm design can be found at http://arxiv.org/pdf/1307.2964v2.pdf.

① One key to our technique is that, by sharing the same abstract interpretation of program calling contexts, different context-sensitive analysis modules required in permission analysis are glued in a unified analysis framework. The abstract interpretation of calling contexts also serves as a bridge to reason permissions and identify relevant permissions at stack inspection points.

② We combine context-sensitive call graph with dependency graph as the underlying model for program analysis, where the dependency graph essentially encodes data flow of permission objects. The reason why call graph does not suffice as usual is that permission objects can be created and referred to anywhere in the program, by either accessing the heap, i.e., field access, or by parameter passing of method calls that are finished before stack inspection. In either case, the data flow of permission objects is beyond the scope of the current call stack inspected by the runtime authorization.

③ Our program model is also based on context-sensitive call graph rather than ordinary call graph, to precisely handle dynamic features of Java languages like late binding. The program model is encoded as conditional weighted pushdown systems and the analysis algorithm is solved as model checking problems. We expect a good precision of our analysis due to its context-sensitive nature.

(2) To put the aforementioned algorithm into practice, we investigated efficient algorithms of those analysis modules underpinning the algorithmic framework, including:

① A sliding-window algorithm for on-the-fly program analysis in the framework of weighted pushdown systems, with a target application to points-to analysis of Java programs. Points-to analysis is the first step of a precise access rights analysis. We present a sliding-window algorithm for weighted pushdown systems that consists in a sequence of local analysis on subparts of the system, with no need for tackling the entire state space of the system at once. The computation cost of each local analysis is lightweight, and the precision of the whole system analysis is preserved by accumulating a series of local fixed points. Our algorithm specially takes into account analysis problems for which one could not assume a prior program control flow, such as Java points-to analysis, the disassembly of binary codes with indirect jumps, etc. Solving such analysis problems amounts to tackling an on-the-fly analysis problem where the underlying system is expanded when the analysis proceeds. We have implemented and evaluated the sliding-window algorithm with Java points- to analysis as an application. Our empirical study shows that the analysis by the sliding-window algorithm always outperforms the whole program analysis for runtime efficiency.

② A new saturation-based algorithm for reachability analysis of patterned conditional pushdown systems. We observe that the existing applications of conditional pushdown systems carry regular languages in terms of regular expressions that obey certain patterns. Therefore, we studied such patterned subclasses of conditional pushdown systems used in practice and presented new backward/forward saturation algorithms for solving reachability problems of them. Our new algorithms give rise to

alternative solutions to reachability analysis of the existing applications using conditional pushdown systems, such as reachability analysis of HTML5 parser specifications, stack inspection, etc. We leave it as future work for extensive empirical studies of the new algorithms driven by patterns.

③ An on-the-fly model checking algorithm for conditional weighted pushdown systems. Model checking conditional weighted pushdown systems was shown to be reduced to model checking weighted pushdown systems, and an offline algorithm was given that translates conditional weighted pushdown systems to weighted pushdown systems by synchronizing the underlying pushdown systems and finite state automata accepting regular conditions. The translation, however, can cause an exponential blowup of the system. Therefore, we present an on-the-fly model checking algorithm for conditional weighted pushdown systems that synchronizes the computing machineries on-demand while computing post-images of regular configurations. We developed an on-the-fly model checker for conditional weighted pushdown systems and applied it to models generated from the reachability analysis of the HTML5 parser specification. Our preliminary experiments show that, the on-the-fly algorithm drastically outperforms the offline algorithm regarding both space and time efficiency in practice.

(3) Apart from devoting efforts to efficient algorithms of pushdown systems, we also studied higher-model checking techniques that generalize finite-state and pushdown model checking. Specially, we designed and developed novel algorithms for model checking recursively-typed higher-order grammars that can be applied to verify safety properties of objected-oriented, concurrent, and higher-order functional programs. According to [③], the model checking problem of recursively-typed higher-order grammars is undecidable. We are concerned with a sound procedure for it, especially following the counterexample-guided abstraction refinement paradigm. Specially, our procedure is also relatively complete with respect to a regular set of term trees: the grammar is eventually proved to be safe if there exists a regular set of term trees that is a safety inductive invariants for the grammar. We have evaluated our tools on examples from verification problems of Featherweight Java programs and that of multi-threaded Boolean programs with recursion. For multi-threaded programs, we studied properties of mutual exclusion (e.g., the Peterson's algorithm), deadlock-freedom (e.g., for various solutions to the dining philosopher problem), and checking of assertion violation (e.g., for simplified variants of Bluetooth drivers). Its potential application to web security analysis will be studied as future work.

As future work, we plan to further elaborate the proposed algorithms and conduct extensive empirical studies of them. We will also investigate new verification techniques and analysis algorithms for access-rights analysis in the presence of subjects, as discussed in [②]. Eventually, we hope to build an open-source practical system for web security analysis.

<引用文献>

① Emmanuel Geay, Marco Pistoia, Takaaki Tateishi, Barbara G. Ryder, Julian Dolby. Modular string-sensitive permission analysis with demand-driven precision. Proceedings of 31st International Conference on Software Engineering (ICSE'09), pp. 177-187, 2009.

② Paolina Centonze, Marco Pistoia, Omer Tripp. Access-rights Analysis in the Presence of Subjects. Proceedings of the 29th European Conference on Object-Oriented Programming (ECOOP'15), pp. 222-246, 2015.

③ N. Kobayashi and A. Igarashi. Model checking higher-order programs with recursive types. In Proceedings of ESOP 2013, volume 7792 of Lecture Notes in Computer Science, Springer, 2013.

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 1 件）

① Hua Vy Le Thanh, Xin Li. An On-The-Fly Algorithm for Conditional Weighted Pushdown Systems. 情報処理学会論文誌:プログラミング、査読有、Vol. 7、No. 4、2014、pp. 1-7、
DOI:

http://doi.org/10.2197/ipsjtrans.7.132

〔学会発表〕（計　4　件）

① Xin Li. An On-The-Fly Algorithm for Conditional Weighted Pushdown Systems. 情報処理学会第 98 回プログラミング研究発表会、東京大学理学部 7 号館（東京)、2014 年 03 月 17 日~2014 年 03 月 18 日

② Xin Li. Generating Stack-based Access Control Policies. 情報処理学会第 103 回プログラミング研究発表会、産業技術総合研究所臨海都心センター（東京)、2015 年 03 月 09 日~2015 年 03 月 10 日

③ Xin Li. Automata-Based Abstraction Refinement for μHORS Model Checking. The 30th Annual IEEE Symposium on Logic in Computer Science. グランドプリンスホテル京都（京都府京都市左京区)、2015 年 07 月 06 日~2015 年 07 月 10 日

④ Xin Li. Automata-Based Abstraction Refinement for μHORS Model Checking. NII Shonan Meeting Seminar 063: Semantics and Verification of Object-Oriented Languages. 湘南国際村センター（神奈川県三浦郡葉山町)、2015 年 09 月 21 日~2015 年 09 月 25 日

〔その他〕
ホームページ等
http://www-kb.is.s.u-tokyo.ac.jp/~li-xin/

６．研究組織
(1)研究代表者
LI　Xin　（LI，　Xin）
東京大学・情報理工学（系）研究科・特任研究員
研究者番号：６０５１０６４１