Research Plan

Fu Song
January 11, 2016

1 Introduction

Global society increasingly relies on devices controlled by software, from TV sets and Mobile Phone to vehicle braking systems and aerospace systems. However, it is considered a "fact-of-life" that software routinely contains errors and vulnerabilities, sometimes malicious behaviors. These issues can come at great cost. For instance, the 1992 failure of the London Ambulance Dispatch Service is due to software errors and Stuxnet computer malware was used to attack Iran's nuclear program in 2007. Engineer classically use software testing as a technique to reduce the occurrence of flaws and analyze malicious behaviors. However, it is not feasible to run a program under all environments - especially given the nondeterminism present with concurrency and interaction - and hence, many flaws are suffered by users. Due to the above realities, my future research will mainly focus on foundations of automatic verification and its practical applications in cybersecurity. In the short term (3 years), I will concentrate on three themes: (1) automatic verification of correctness for infinite-state multi-agent systems, (2) security analysis of low-level binary executable programs and (3) secure programming methodology. The main objective is to propose effective and efficient methods, techniques and tools for cybersecurity.

2 Methodology

2.1 First Theme

I will investigate model-checking techniques for infinite-state multi-agent system, as model-checking is an technique that attempts to prove, formally and automatically, that a system behaves as expected under all conditions. To specify the behavior of multi-agent systems, a well-known logical formalism is Alternating-Time Temporal Logics [1] for which a model-checking algorithm for *finite-state* concurrent game structures is also given therein. Model-checking algorithms for various other temporal logics on *finite* multi-agent systems have been proposed in several works, e.g. [2, 4, 7]. More recently, on a different dimension, model-checking for ATL* on a class of *infinite-state* multi-agent systems with perfect recall and perfect information, i.e., *pushdown multi-agent systems*, was also studied in [11]. The authors introduced *pushdown game*

structures (PGSs) as the model, showed that the model-checking problem is 2EXPSPACE-hard, and proposed a model-checking algorithm in 3EXPTIME. In [6], we proved that this problem is indeed 3EXPTIME-complete. Moveover, we also studied the model-checking problem on PGSs for alternating-time μ-calculus (AMC) [1] which is more expressive than ATL*. We proved that this problem is EXPTIME-complete. However, it is shown in [9] that there are many interesting properties that cannot be expressed in ATL* and AMC. To overcome this limitation, many logics are proposed in the literature, among them, Strategy Logic (SL) is the most expressive one. I will investigate the model-checking problem of PGSs on SL and its subclasses. More precisely, we consider the SL logic with Nested-Goal, Boolean-Goal, Conjunctive-goal, Disjunctive-goal, One-Goal Strategy Logic, and Basic Strategy-Interaction Logic [10, 9, 5, 12] on PGSs with respect to both perfect and imperfect recall, perfect and imperfect information. These techniques will be implemented in a tool for automatic verification of infinite-state multi-agent systems.

2.2 Second Theme

It is extremely difficult to analyze low-level binary executable programs, especially for self-modified programs, in the absence of source code and debug information. To overcome this problem, based on the abstract interpretation framework [8] and the binary analysis framework BAP [3], I will propose an abstract interpretation framework for low-level binary executable programs with self-modification. The main challenge is how to deal with self-modified codes in the abstract interpretation. We have summarized types of self-modifications and solved one type of self-modifications which changes the target addresses of indirect jumps on the runtime. We plan to tackle other types of self-modifications and finally build a framework for analysis of self-modified low-level binary executable programs. This framework will be the cornerstone of advanced techniques investigation such as malware analysis, vulnerability detection, automated patch generations, etc.

2.3 Third Theme

Secure programming methodology is a desirable way to implement software programs avoiding vulnerability and attacks. However, software application scenario is usually too complicate (protection mechanisms, attacks) to precisely define a secure programming language. We plan to create a unified framework for creating: formal models of operating systems taking protection mechanisms into account (e.g., data execution prevention, address space layout randomization and control-flow integrity); formal models of C programming languages with editable features such as loop, recursion, memory allocation, pointer; formal models of attacks such as shell code injection, ROP exploits, etc. The framework and models will be implemented in a theorem prover isabelle or Coq. We will investigate operational semantics of these models so that we can reason about the relationship between attack techniques, protection mechanisms and the features

of C language. The verified principle of secure programming with respect to underlying attacks and protection mechanisms can be generated by proofs. The outcome of this framework could be applied to measure system security risk or guide the development of secure programs.

3 Team Building and Grands

To achieve the above goals, I have to build a research team which will consist of 2-3 post-docs, 2-3 Ph.D. and 4-6 master students. Each research topic will have 1 post-doc, 1 Ph.D. and 2 master students. In order to maintain such a team, besides the startup fund, I will try to find funds from the government and companies by research findings and industrial applications.

References

[1] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[2] Mustapha Bourahla and Mohamed Benmohamed. Model checking multi-agent systems. *Informatica (Slovenia)*, 29(2):189–198, 2005.

[3] David Brumley, Ivan Jager, Thanassis Avgerinos, and Edward J. Schwartz. BAP: A binary analysis platform. In *Computer Aided Verification*, July 2011.

[4] Nils Bulling and Wojciech Jamroga. Alternating epistemic mu-calculus. In *International Joint Conference on Artificial Intelligence*, pages 109–114, 2011.

[5] Petr Cermák, Alessio Lomuscio, and Aniello Murano. Verifying and synthesising multi-agent systems against one-goal strategy logic specifications. In *AAAI Conference on Artificial Intelligence*, pages 2038–2044, 2015.

[6] Taolue Chen, Fu Song, and Zhilin Wu. Global model checking on pushdown multi-agent systems. In *AAAI Conference on Artificial Intelligence*, 2016.

[7] Wojciech Jamroga and Aniello Murano. Module checking of strategic ability. In *International Conference on Autonomous Agents & Multiagent Systems*, pages 227–235, 2015.

[8] Johannes Kinder and Helmut Veith. Jakstab: A static analysis platform for binaries. In *Computer Aided Verification*, pages 423–427, 2008.

[9] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Transactions On Computational Logic*, 15(4):34:1–34:47, 2014.

[10] Fabio Mogavero, Aniello Murano, and Luigi Sauro. On the boundary of behavioral strategies. In *Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 263–272, 2013.

[11] Aniello Murano and Giuseppe Perelli. Pushdown multi-agent system verification. In *International Joint Conference on Artificial Intelligence*, pages 1090–1097, 2015.

[12] Farn Wang, Sven Schewe, and Chung-Hao Huang. An extension of ATL with strategy interaction. *ACM Trans. Program. Lang. Syst.*, 37(3):9, 2015.