Resilience of SIP against ML-based attacks

Master's Thesis

Supervisor: Prof. Dr. Alexander Pretschner
Advisor: Mohsen Ahmadvand, Aleieldin Salem
Email: {pretschn, ahmadvan, salem}@in.tum.de
Phone: +49 (89) 289 - 17836
Starting date: immediately

Fakultät für Informatik
Lehrstuhl 4
Software Engineering
Prof. Dr. Alexander Pretschner

Boltzmannstraße 3
85748 Garching bei München

Tel: +49 (89) 289 - 17836
https://www4.in.tum.de

Context

Software is (among others) susceptible to *code manipulation*/tampering attacks from attackers who have full control over the execution environment (Man-at-the-End) [1]. Against such attacks there exist a multitude of integrity protection (tamper-proofing) techniques. These techniques rely on code instrumentation to inject protection routines/guards into programs.

However, code instrumentation leaves recognizable patterns in the protected binaries that could potentially single out protection routines [2]. To mitigate pattern-based attacks, we inevitably have to utilize diversification obfuscation [3] on protection guards. How effectively such diversified guards can counter pattern-based attacks is questionable.

Goal

The goal of this thesis is to measure the effectiveness of obfuscation in countering pattern-based attacks on integrity protection guards. We base our study on guards that are generated by our SIP-toolchain[1] (namely, CFI [4], SC [5], OH [6] and SROH protection techniques).

The student will first try to identify patterns in protected binaries for each protection techniques. The effectiveness of pattern matching is then recorded as the baseline of the study. Thereafter, a dataset of obfuscated guards is generated. We use the obfuscation techniques available in Obfuscator-LLVM (`https://github.com/obfuscator-llvm/`) and Virt.SC (internal tool) to create the dataset. Each guard shall be obfuscated with all combinations of the available obfuscation techniques with enough number of iterations such that enough samples of each combination is observed.

We expect that the primitive pattern-based attacks will fail to recognize and identify any obfuscated guards. Consequently, we plan to utilize machine learning techniques, in a manner similar to [7], to identify the utilization of obfuscation to protect the guards and to localize the guard themselves within a given binary. To make a robust conclusion about the machine learning techniques that can be used to effectively tackle the problem at hand, we plan on comparing the performances of different techniques (e.g., extracted features, feature pre-/post-processing techniques, classification algorithms, etc.), against binaries of different sizes and complexities, protected with different guards and obfuscation techniques.

Working Plan

1. Implement pattern-based attacks to detect protection guards of SC, OH, SROH and CFI in protection binaries
2. Generate the dataset of obfuscated guards
3. Utilize machine learning to detect (and localize) patterns in obfuscated guards
4. Evaluate the performances of different machine learning techniques

Deliverables

- Docker container able to run a demo of the implementation, including instructions on how to run the demo
- The container should also include the source code of the implementation
- Technical report with comprehensive documentation of the implementation, (i.e., design decision, architecture description, API description and usage instructions)
- Final thesis report written in accordance with TUM guidelines

References

[1] C. S. Collberg and C. Thomborson, "Watermarking, tamper-proofing, and obfuscation-tools for software protection," *IEEE Transactions on software engineering*, vol. 28, no. 8, pp. 735–746, 2002.

[1] https://github.com/tum-i22/sip-toolchain

[2] M. Ahmadvand, A. Pretschner, and F. Kelbert, “A taxonomy of software integrity protection techniques,” ser. Advances in Computers. Elsevier, pp. –. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065245817300591

[3] B. Baudry and M. Monperrus, “The multiple facets of software diversity: Recent developments in year 2000 and beyond,” *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, p. 16, 2015.

[4] M. Abadi, M. Budiu, U. Erlingsson, and J. Ligatti, “Control-flow integrity,” in *Proceedings of the 12th ACM conference on Computer and communications security*. ACM, 2005, pp. 340–353.

[5] H. Chang and M. J. Atallah, “Protecting software code by guards,” in *ACM Workshop on Digital Rights Management*. Springer, 2001, pp. 160–175.

[6] Y. Chen, R. Venkatesan, M. Cary, R. Pang, S. Sinha, and M. H. Jakubowski, “Oblivious hashing: A stealthy software integrity verification primitive,” in *International Workshop on Information Hiding*. Springer, 2002, pp. 400–414.

[7] A. Salem and S. Banescu, “Metadata recovery from obfuscated programs using machine learning,” in *Proceedings of the 6th Workshop on Software Security, Protection, and Reverse Engineering*. ACM, 2016, p. 1.

Fakultät für Informatik
Lehrstuhl 4
Software Engineering
Prof. Dr. Alexander Pretschner

Boltzmannstraße 3
85748 Garching bei München

Tel: +49 (89) 289 - 17836
https://www4.in.tum.de