POSTER: Diversity for Software Resilience

Andrew S. Gearhart
The Johns Hopkins University
Applied Physics Laboratory
andrew.gearhart@jhuapl.edu

I. Diversity for Resilience

The current software "monoculture" presents a common avenue of attack against all computers running the vulnerable software [1]. Due to this concern, a taxonomy of software diversity techniques has been developed (see Larsen et al. [2]) and researchers have applied diversity as a defensive strategy to a range of attack types from SQL injection [3] to return-oriented programming (ROP) [4]–[6]. Two key areas of open research within the study of software diversity are (1) measuring the impact of diversity and (2) application of software diversity in n-variant systems to achieve cyber resilience.

Quantitative metrics to compare efficacy of software diversity strategies would optimally differentiate strategy types and also correlate with actual attacker effort. For example, doubling the distance between two diversity strategies with such a metric might be directly correlated with a proportionate increase in attacker effort to compromise a diversified software population. However, approximating attacker effort is a difficult problem due to the challenge of collecting reliable measures of effort from a sufficient number of human attackers. Many proxy approaches depend on a particular class of attacks, such as comparing the number of ROP gadgets that are shared between variants [7], [8]. These approaches have begun to address the problem of evaluating diversity strategies, but further investigation is required.

Second, software diversity can be used to protect a single physical platform by running several variants of an application simultaneously. This is known as n-variant execution [9], [10], and resilience is assumed to be incurred via the inability of common inputs to compromise the majority of variants. A monitoring application is typically used to analyze the system and behavior divergence of a portion of the variants is considered indicative of a potential fault. Managing overall performance, monitor design, and methods of detecting divergence are open challenges for n-variant systems.

II. Recent Results

Our work focuses on differentiating diversity strategies and investigating the practical implications of n-variant execution. In particular, we explore methods of comparing diversity strategies in a manner inspired by unstructured text analysis. Specifically, we generate several types of feature vectors (raw frequency counts, term frequency-inverse document frequency (TF-IDF), and doc2vec [11]) from disassembled binaries and evaluate these features (and a set of cluster-based distance metrics) for their ability to differentiate compiler-based diversity strategies. This is accomplished by generating a set of variants according to a particular diversity strategy, disassembling and normalizing these binaries, and then generating doc2vec representations. The resulting clusters can be compared to an undiversified baseline using Euclidean distance, and a separate distance value produced for each application in a benchmark suite. Figure 1 uses this approach to compare six diversity strategies (for details, see [7], [12]) over the GNU core utilities (103 applications).[1] We note that strategies that make small changes to the input source code (e.g., random scheduling) are most similar to the baseline, suggesting that this approach is viable to differentiate diversity techniques.

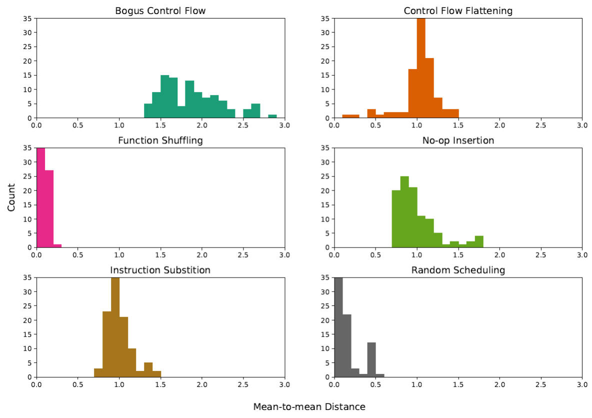

Fig. 1. Comparing Diversity Strategies to baseline

TABLE I
Overhead of n-variant execution for SPEC C/C++ benchmarks.

# Instances	Min (%)	Max (%)	Mean (%)	Median (%)
2	2.91	20.04	7.45	6.85
3	4.90	35.61	14.36	11.67
4	8.29	55.74	22.44	15.90

In addition to our clustering work, we have developed a prototype n-variant execution framework to aid further experimentation and design space exploration. Running the C/C++ SPEC benchmarks,[2] this framework currently results in a 16% runtime overhead when running four variants (Table I). In the future, we intend to use this framework to run server applications and potentially port it to embedded devices.

[1]http://www.gnu.org/s/coreutils
[2]https://spec.org/benchmarks.html

References

[1] D. Geer, R. Bace, P. Gutmann, P. Metzger, C. P. Pfleeger, J. S. Quarterman, and B. Schneier, "Cyber*In*security: The Cost of Monopoly," Computer & Communications Industry Association, Tech. Rep., September 2003.

[2] P. Larsen, A. Homescu, S. Brunthaler, and M. Franz, "SoK: Automated Software Diversity," in *2014 IEEE Symposium on Security and Privacy*, May 2014, pp. 276–291.

[3] S. Rauti, J. Teuhola, and V. Leppänen, "Diversifying SQL to Prevent Injection Attacks," in *Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA - Volume 01*, ser. TRUSTCOM '15, 2015, pp. 344–351.

[4] J. Hiser, A. Nguyen-Tuong, M. Co, M. Hall, and J. W. Davidson, "ILR: Where'd My Gadgets Go?" in *2012 IEEE Symposium on Security and Privacy*, May 2012, pp. 571–585.

[5] V. Pappas, M. Polychronakis, and A. D. Keromytis, "Smashing the Gadgets: Hindering Return-Oriented Programming Using In-place Code Randomization," in *2012 IEEE Symposium on Security and Privacy*, May 2012, pp. 601–615.

[6] C. Giuffrida, A. Kuijsten, and A. S. Tanenbaum, "Enhanced Operating System Security Through Efficient and Fine-grained Address Space Randomization," in *Proceedings of the 21st USENIX Security Symposium*, Oct. 2012, pp. 475–490.

[7] A. Homescu, S. Neisius, P. Larsen, S. Brunthaler, and M. Franz, "Profile-guided Automated Software Diversity," in *Proceedings of the 2013 IEEE/ACM International Symposium on Code Generation and Optimization*, ser. CGO '13, February 2013, pp. 1–11. [Online]. Available: http://dx.doi.org/10.1109/CGO.2013.6494997

[8] J. Coffman, D. M. Kelly, C. C. Wellons, and A. S. Gearhart, "ROP Gadget Prevalence and Survival Under Compiler-based Binary Diversification Schemes," in *Proceedings of the 2016 ACM Workshop on Software PROtection*, ser. SPRO '16, 2016, pp. 15–26.

[9] B. Cox, D. Evans, A. Filipi, J. Rowanhill, W. Hu, J. Davidson, J. Knight, A. Nguyen-Tuong, and J. Hiser, "N-Variant Systems: A Secretless Framework for Security through Diversity," in *Proceedings of the 15th USENIX Security Symposium*, August 2006, pp. 105–120.

[10] P. Hosek and C. Cadar, "VARAN the Unbelievable: An Efficient N-version Execution Framework," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '15. New York, NY, USA: ACM, 2015, pp. 339–353. [Online]. Available: http://doi.acm.org/10.1145/2694344.2694390

[11] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," in *Proceedings of the 31st International Conference on Machine Learning*, ser. ICML-14, 2014, pp. 1188–1196.

[12] P. Junod, J. Rinaldini, J. Wehrli, and J. Michielin, "Obfuscator-LLVM: Software Protection for the Masses," in *Proceedings of the 1st International Workshop on Software Protection*, ser. SPRO '15, May 2015, pp. 3–9.