Poster: End-to-End Service for System Security Experimentation

Christophe Hauser
Information Sciences Institute
University of Southern California
hauser@isi.edu

Zhenkai Liang
National University of Singapore
liangzk@comp.nus.edu.sg

Stephen Schwab
Information Sciences Institute
University of Southern California
sschwab@isi.edu

Abstract—**Countering constantly evolving internet security threats such as worms, botnets or distributed denial of service attacks (DDOS), requires realistic and accurate modeling of software components as well as the environment in which they interact. To this end, security experimentation test beds such as DeterLab have been focusing on the ability to reproduce a large range of experimental scenarios in a safe and accessible manner. The span of current test bed environments varies from the emulation of network architectures and protocols to the modeling of smart power grid systems. With a focus on network and system infrastructures, test bed environments have been supporting research in cyber security across numerous fields of application over many years. However, while many aspects of cyber security can be approached from a networking angle, the task of accurately modeling vulnerable and malicious applications and their interactions with the environment requires an understanding of low-level operating system abstractions as well as inner-application constructs that current state of the art test bed environments fail to represent. To this end, we aim to extend Deter with state-of-the-art program analysis models and techniques and leverage the resulting extended environment to develop new capabilities and novel approaches in terms of vulnerability analysis, discovery and incidence response.**

1. Introduction

Experimentation is a critical process in system security research, where researchers allocate the required hardware and networking infrastructure, set up the system and software environments, analyze target security problems, and validate their solutions. Many solutions have been proposed to facilitate security experimentation, in terms of infrastructure allocation [1] and environment setup [2]. However, researchers still need to go through a time-consuming manual effort to setup the environment with a diverse range of tools to analyze security problems, which significantly hinders the efficiency and scalability of system security experimentation.

In recent years, program analysis techniques have made substantial progress, both to analyze the source code of applications, and to directly analyze their executable form in binary code. The result of these advances, which have been published in the form of academic papers and related tools [3], [4], [5], [6]) is now accessible for experts to use and experiment with. The potential applications of these tools and techniques range from vulnerability discovery and malware analysis to security assessment. However, despite this sharing of knowledge and transfer of technology, access to these advances remains difficult for most users, because of the particular skills that are required to successfully make a fruitful use of such complex specialized software, which are exposing numerous low-level abstractions and inner details about the underlying approaches. Researchers and students would nonetheless benefit from the result of program analysis as part of their projects, since these offer valuable implementation of concepts, automation and semantic reasoning that cannot directly be achieved by other means. As a result of this observation, we propose to bring program analysis capabilities within the Deter cyber security experimentation test bed, as a new infrastructure providing "program analysis as a service".

The current state of the art in terms of program analysis tools offers a range of advanced, but isolated analyses capabilities. For instance, the `angr` binary analysis platform currently has a powerful symbolic execution engine, but limited support for full system dynamic analysis. On the contrary, DECAF [7], the Dynamic Executable Code Analysis Framework (the successor of Bitblazes TEMU) brings powerful dynamic execution and tainting support. By bringing together tools with different, but complementary capabilities, the objective of the Deter program analysis environment is to offer analysis as a service in the Deter environment.

2. Goals

Usability – we propose a novel approach to conceptualize program analysis, with an organization of key principles in the form of semantic analysis building blocks, which, combined together, can be used by non-expert analysts to build analyses that fit a wide range of requirements. Each building block offers an abstraction of a program analysis concept, leaving freedom to the analyst to experiment with state of the art techniques without having to learn each concept in detail. The construction of such building blocks will be based on a review of current state of the art abstractions in program analysis, along with a specification of their scope of reliability, i.e., in which contexts each particular analysis is reliable. For example, a data dependence analysis based on pure symbolic execution is likely to explode in the presence of loops, but offers high accuracy, while the same analysis built in an abstract static domain such as value-set analysis guarantees termination, but offers a coarser level of details which may not be satisfactory in some situations.

Reproducibility – most state-of-the-art techniques, such as those presented in the academic literature, were evaluated by executing prototypes on size-limited datasets or in constrained environments. The ability to reproduce such experiments outside of their initial environments, and to gather statistics about the efficiency and efficacy of various approaches, or occurrences of certain types of vulnerabilities or malicious behavior in real-word software are interesting outcomes of research in that direction. However, the full automation of entire end-to-end analyses is out of the scope of this project, due to the extreme difficulty of scaling such analyses at the binary level without partly relying on program and environment specific knowledge provided by manual investigation and human judgement. This differentiates our approach from e.g., the SWAMP [8], which provides a collection of end-to-end analyses on the source code of applications. In order to best reproduce such results, we propose to build a database of existing known vulnerabilities, e.g., from Common Vulnerabilities and Exposures (CVE) entries provided by the National Vulnerability Database (NIST). It should be emphasized, however, that the sole knowledge of a CVE entry is not enough for accurate reproducibility, since such entries generally only provide a high-level description of the vulnerability, without any associated exploits or details about the vulnerable versions of the affected software. As a result, it is necessary for researchers to manually analyze the source code of the associated projects in order to gather a precise understanding of vulnerabilities, and this step is therefore a prerequisite in this project.

Scalability – relying on solid semantic analysis building blocks, tested and refined against real-world examples during the previous step of reproducibility, makes it possible to leverage the Deter infrastructure in order to perform large-scale experiments, and to offer automated analyses as an online service to remote users. By offering a range of sample automated analysis based on our system, we aim to run large scale experiments and discover new vulnerabilities in real-world binary software, collect new real-world vulnerable programs or malware samples, and generate usage statistics (from a usability standpoint, i.e., how researchers approach the task of analyzing software) and statistics about properties of existing vulnerabilities in real-world binary software.

Novelty – integrating these tools, techniques and datasets as part of a cyber security experimentation test bed, brings the opportunity to experiment with new approaches on large corpora of data, thus allowing academic research to advance the current state of the art in large-scale cyber experimentation.

Applications – by employing tools from multiple domains of program analysis, and by leveraging a database of vulnerable software (CVE), along with their respective exploits, we aim to provide an environment for experimentation, which allows users to reproduce, analyse and mitigate vulnerabilities as part of the same framework. For instance, while dynamic tracing allows a researcher to replay exploits and identify vulnerable program paths, static analysis and symbolic execution can be used to generalize the observed vulnerability, and to produce valuable input to vulnerability discovery algorithms, or to generate e.g., generic signatures of exploits, which are not bound to a specific environment. Currently, the vast majority of intrusion detection systems (IDS) designed for commercial use are based on signatures of exploits. Manually crafting these signatures requires a lot of human investigation. The ability to generate signatures automatically would be valuable to both increase coverage and reduce response time. Similarly, the framework that we propose would allow researchers to extend the scope of current analysis tools in the contexts of incident response and reverse engineering.

3. Related work

Similar efforts in the literature have focused on addressing the problem of scaling automated source code analysis. The SWAMP [8], or Software Assurance Marketplace allows code developers to assess the security of their code by running a number of end-to-end source-level analyses provided by third-party tools. Our approach, in contrast, is modular (i.e., users build their own analyses based on semantic analysis building blocks), semi-automated (requires knowledge from the user) and focuses on the binary form of applications (i.e., focuses on what is actually executed by the machine and does not require the source code to be available), thus extending the current scope to novel user-driven analyses of proprietary software and malware.

References

[1] J. Mirkovic and T. Benzel, "Deterlab Testbed for Cybersecurity Research and Education," *J. Comput. Sci. Coll.*, vol. 28, no. 4, pp. 163–163, Apr. 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2458539.2458567

[2] The Metasploit Framework. http://metasploit.com/.

[3] D. Song, D. Brumley, H. Yin, J. Caballero, I. Jager, M. G. Kang, Z. Liang, J. Newsome, P. Poosankam, and P. Saxena, "BitBlaze: a New Approach to Computer Security via Binary Analysis," in *International Conference on Information Systems Security*. Springer, 2008, pp. 1–25.

[4] Y. Shoshitaishvili, R. Wang, C. Salls, N. Stephens, M. Polino, A. Dutcher, J. Grosen, S. Feng, C. Hauser, C. Kruegel *et al.*, "SOK:(State of) The Art of War: Offensive Techniques in Binary Analysis," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 138–157.

[5] "angr, the Next Generation Binary Analysis Platform from UC Santa Barbara!". http://angr.io.

[6] A Robust Code Analysis Platform for C/C++. http://www.mlsec.org/joern/.

[7] A. Henderson, L. Yan, X. Hu, A. Prakash, H. Yin, and S. McCamant, "DECAF: a Platform-Neutral Whole-System Dynamic Binary Analysis Platform," *IEEE Transactions on Software Engineering*, 2016.

[8] Software Assurance Marketplace. http://continuousassurance.org/.