2012 IEEE Fifth International Conference on Software Testing, Verification and Validation

Model-Based Fuzz Testing

Ina Schieferdecker
Fraunhofer FOKUS/Freie Universität Berlin
Berlin, Germany
ina.schieferdecker@fokus.fraunhofer.de

***Abstract*—The European ITEA2 project DIAMONDS (Development and Industrial Application of Multi-Domain Security Testing Technologies) develops under the direction of Fraunhofer FOKUS, Berlin efficient and automated security test methods for security-critical, networked systems in various industrial domains such as industrial automation, banking and telecommunications. DIAMONDS develops methods to design objective, transparent, repeatable, and automated security tests that focus on system specifications and related risks. The project goals include the development of a security test pattern catalogue and the development of model-based security testing techniques such as risk-based testing and model-based fuzz testing. The project results are made available through publications and contributions to the standardization at ETSI and other standardization bodies. The presentation focusses on model-based fuzz testing, reviews the state of the art, compare it to similar approaches such as mutation testing, and presents first results on behaviour fuzzing for security testing.**

***Keywords*-model-based testing, security testing, fuzzing**

Extended Abstract

Model-based testing strives at automatically and systematically generating test cases from system models, usage models, test models, and alike. The basic idea is that instead of creating test cases manually selected algorithms are generating them automatically. Usually, there is an infinite number of possible tests that can be generated from a model, so that test designers choose test generation directives or selection criteria to limit the number of generated tests to a small finite number by e.g. selecting highest-priority tests or by ensuring specific coverage of model structures or of other artifacts such as requirement documents. For security testing, approaches based on such coverage metrics are not sufficient: Because of the difficulties in having adequate hypotheses where to check for vulnerabilities, rather randomized approaches outperform traditional MBT approaches.

Fuzz testing is a commonly used approach for security testing of software-based systems. It is a black-box testing technique in which the system under test is stressed with invalid, unexpected or random inputs at its interfaces. The purpose is to reveal system vulnerabilities by bringing the systems into failure modes. The origin of fuzzing is based on a complete randomized approach [1]. Hence, fuzz testing is likewise confronted with the infinite number of randomized tests.

Smarter approaches for determining the tests are being investigated: block-based and model-based fuzzers use their knowledge about the message structure to systematically generate messages containing invalid data among valid data. Model-based fuzz testing adds to this the fuzzing of system behavior: it uses fuzz operators to randomize sequential or concurrent system behavior. Empirical studies have shown that model-based fuzzing outperforms traditional, brute force fuzzing [2].

We are using behavioral fuzzing on scenario models which are specified by sequence diagrams [3]. Behaviour fuzzing does not only reflect the generation of atypical messages but also changes the typical appearance and order of messages. For example a valid and approved sequence of messages can be turned into an atypical and unknown sequence by rearranging messages, repeating and dropping them or just by changing the type of message. In addition, the fuzzing of behavior sequences can be combined with the fuzzing of the input data or the fuzzing of the load given in parallel to the system.

Fuzz testing, mutation testing, fault injection, and others all base on a comparable idea of applying rather randomized, potentially erroneous inputs to a system under test. It remains to be seen, if a combination can lead further improvements for security tests by generating more efficient malicious inputs.

The author would like to thank Martin Schneider and Jürgen Großmann, Fraunhofer FOKUS for the joint work on model-based fuzz testing as well as the whole DIAMONDS team.

References

[1] B. P. Miller, L. Fredriksen, and B. So, "An empirical study of the reliability of unix utilities," in *In Proceedings of the Workshop of Parallel and Distributed Debugging*. Academic Medicine, 1990, pp. pages ix–xxi,.

[2] A. Takanen, J. DeMott, and C. Miller, *Fuzzing for software security testing and quality assurance*, ser. Artech House information security and privacy series. Artech House, 2008. [Online]. Available: http://books.google.de/books?id=tMuAc_y9dFYC

[3] I. Schieferdecker, J. Großmann, and M. Schneider, "Model-based security testing," in *In Proceedings of the Model-Based Testing Workshop at ETAPS 2012*. EPTCS, 2012, pp. 1–12.

978-0-7695-4670-4/12 $26.00 © 2012 IEEE
DOI 10.1109/ICST.2012.180