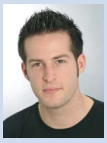
Michael Fisler

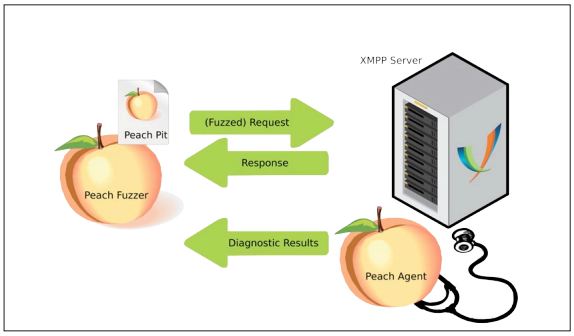


Kevin Lynn

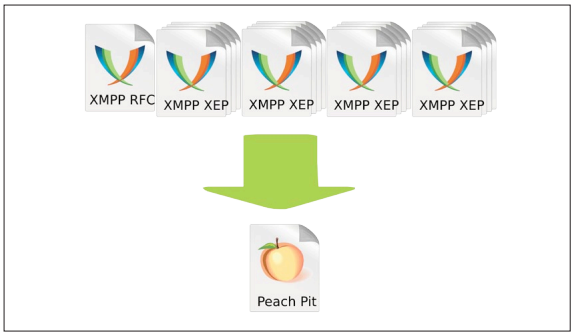
Graduate Candidates	Michael Fisler, Kevin Lynn
Examiner	Prof. Dr. Andreas Steffen
Co-Examiner	Dr. Ralf Hauser, PrivaSphere AG, Zurich
Subject Area	Sicherheit
Project Partner	Compass Security AG, Jona SG

IT Security: Fuzzing Windows Applications and Network Protocols

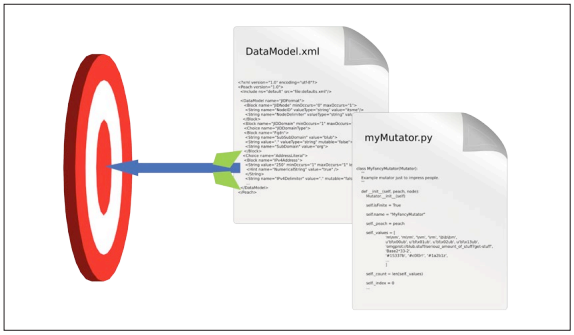
Evaluating fuzzing by means of an example protocol and a custom fuzzer



Our test set-up including the Peach Framework components Fuzzer and Agent (with its diagnostic and logging features) and the target server



The challenge of breaking down a protocol which spans hundreds of pages of documentation



Success-determining factors: Modelling details and mutation accuracy

Introduction: Fuzzing is a technique for detecting software flaws by intentionally sending invalid input to a target of evaluation, generally involving a high degree of automation. For software engineers it is crucial to identify and eliminate such flaws since they might be exploitable by a remote attacker. As a consequence an attacker could compromise the application as well as the operating system the software is running on, gain unauthorized access and steal or modify confidential data. While the basic idea behind fuzzing is simple, creating thorough, precise and effective fuzzers is a real challenge. With time and computing power being limiting factors, the success of fuzzing depends on the level of detail when modelling the protocol as well as the effectiveness of the data mutations performed. The goal of our thesis is to present methods to effectively fuzz a network (i.e. server) application using an example protocol and a custom fuzzer.

Approach/Technologies: By choosing the standards-based Extensible Messaging and Presence Protocol (XMPP), we selected a contemporary protocol that is gaining popularity in real-time communication applications and that also incorporates many advanced concepts found in other common networking protocols. The scope of the XMPP protocol including all its extensions is vast and therefore presents a major obstacle in terms of protocol modelling and target coverage. As our tool of choice, we selected the Peach Fuzzing Platform for fuzzer modelling. Peach is a free, powerful framework including many automation features. Partly due to the documentation, which in some areas is very scarce, Peach has quite a steep learning curve. By documenting our experience, successes and mistakes, we aim to reduce the initial effort required to become acquainted with Peach and its varied features. Using an arsenal of custom fuzzers and two different XMPP servers, we evaluated how to fuzz most effectively.

Result: In addition to general considerations such as the circumstances under which fuzzing makes sense and the limitations of fuzzing, we also present methods to tackle extensive XML-based protocols, and tuning Peach to achieve higher performance. We also point out strategies to achieve a high level of coverage and precision. In order to manage the protocol's complexity, we wrote scripts to facilitate fuzzer creation, allowing us to rip and process XML content from extension RFCs. Since Peach does not natively support XML-based protocols, we added custom components.