



Practical Aspects of Massive-Scale Software Diversity

Master thesis for Dominik Lichtenauer
Matrikelnummer: 0455935, lichtenauer@gmail.com

Open networks and the computers on them are under constant attack from a variety of adversaries. Most of these attacks are enabled by software vulnerabilities, i.e., errors in operating systems, device drivers, shared libraries, and application programs that can be exploited to perform unauthorized operations on the computers running the software. Although considerable efforts have gone into finding and eliminating such errors, and although impressive advances have been made in doing so, the complexity of today's software systems is so great that a certain number of residual errors will probably always be present. The incidence of such errors tends to be proportional to the overall code size and decreases over time.

The existence of residual software errors becomes a significant threat when large numbers of computers are affected by identical vulnerabilities at the same time. Unfortunately, this is the situation today. We currently live in a software monoculture—for some widely used software, identical binary code is installed on millions, sometimes hundreds of millions, of computers. This makes widespread exploitation easy for an attacker, because the same attack vector is likely to succeed on a large number of targets.

This project is part of a larger effort at the University of California in Irvine (UCI) that aims to break the software monoculture assumption by giving each user a unique, but functionally identical binary so that a different attack needs to be constructed for each target. As a result, an attack intended for large scale use succeeds only on a small fraction of systems and is no longer able to rapidly spread throughout the internet. Attackers are then forced to modify their attack to successfully target each version of the binary. The attacker's job is made even harder because the attacker has no a priori knowledge of which version is used on a particular target. Therefore, the attacker must expend a significant amount of effort to get a large amount of targets.

A diversifying compiler based on the LLVM open source compilation system is under development at UCI. It will form the basis for this project. However, the diversifying compiler itself is not enough. There are several major practical issues hampering the general acceptance of software diversity as a viable defensive strategy:

- Generation/Distribution: How can software vendors efficiently generate and distribute hundreds of thousands of diverse variants of the same application?
- Testing: How is the software quality assurance process affected by diversification? How should the diversification system itself be tested? Is specialized testing of the software prior to diversification necessary? How should each generated variant of the software be tested?

- Debugging: Once variants have been distributed, how should bugs be reported and how should bug databases be organized, when every report refers to a unique piece of software? When a bug has been reported, how is actual debugging done on the unique software variant that the bug refers to? How are bugs that originate in the diversification process itself handled?
- Patching: Once bugs have been fixed, how are unique patches generated and distributed for all diverse variants of the software in a timely manner?

It is the goal of the work proposed here to design and implement a cloud-based diversification infrastructure that will provide efficient tools for generation, testing, debugging, and patching of diversified binaries.

This master's thesis is done at the Secure Systems and Software Laboratory, directed by Prof. Michael Franz, at the University of California in Irvine.

Advisor: Prof. Dr. Michael Franz (UC Irvine), O. Prof. Dr. Dr. h.c. Hanspeter Mössenböck

Start: June 2011