Fuzzing in The Cloud (Position Statement)

Patrice Godefroid David Molnar
Microsoft Research
February 2010

In this short note, we argue that fuzzing in the cloud will revolutionize security testing.

It's more than elasticity. Fuzz testing is the process of repeatedly feeding modified inputs to a program in order to uncover security bugs (such as buffer overflows). Because fuzzing is typically easy to parallelize by running multiple instances of the program under test, it benefits from the elastic nature of cloud computing: if an organization wants to perform 100,000 test iterations, it only needs machines for the required time. This elasticity means that smaller teams or smaller companies can experiment with fuzz testing without a large capital cost.

Despite these benefits of elasticity, in the long run an organization that needs software security will find itself continuously testing. New releases of software need new security testing, whether by the authors of the software or by adversaries trying to find security holes. Over time, this repeated testing chips away at the "rent vs. own" argument for cloud infrastructure. An organization that does fuzz testing for each build of a piece of software will pass the break-even point for buying its own dedicated machines in a matter of months, even when the costs for cooling, administration, etc. are taken into account. Elasticity is *not* the "killer argument" for cloud fuzzing.

It's about shared infrastructure. Instead, the major win for cloud fuzzing comes from a *shared security infrastructure* for the entire picture of people, processes, and tools required to build secure software. Each application must be "enrolled" in a fuzz testing infrastructure, which includes defining how new inputs are passed to the program (test harnesses/drivers), how to detect security-related bugs (runtime checkers), and how to prioritize the resulting security flaws (bug triage). Beyond security testing, this infrastructure can include training for developers, as well as functions such as timely response to reports of security flaws. At Microsoft, the Security Development Lifecycle and the Trustworthy Computing organization are examples of shared security infrastructure that have grown over a period of many years.

A mature security infrastructure requires a range of human talents and organizational knowledge, and it cannot be assembled overnight. Once created, however, the marginal cost of supporting a new piece of software is relatively low. For example, once a new fuzz testing tool is available, it can be applied to applications already enrolled in the shared infrastructure without massive new investment. For another example, the cost to handle reports of security flaws can be amortized over all the applications supported by the infrastructure. Hosting security testing in the cloud simplifies the process of gathering information from each enrolled application, rolling out updates, and driving improvements in future development.

From nothing to something. Because the upfront cost of a security infrastructure is large, the leap from zero to *some* security process is the hardest. More and more organizations will face this leap due to compliance, risk management, and public relations impact of security failures on their business. For example, Adobe is currently under attack in the press due to its poor software security in Flash and Acrobat, which poses a threat to its platform dominance and thereby to its revenues. While Adobe's troubles stem from the large market share of Flash and PDF, the trend extends to other organizations, because security conscious customers want ways to measure the security risk introduced by outsourced code or even from packaged products. While compliance standards such as the Payment Card Industry standard or Sarbanes-Oxley today devote relatively little attention to software security, we expect this will change. If a major customer or compliance standard adopts software security requirements, organizations will find themselves making the leap into security testing with a short deadline under intense pressure. Cloud fuzzing makes it *possible* for a small organization to take this leap by leveraging shared infrastructure. As organizations join, the con-

tinuous improvement to the infrastructure will make it harder for others to replicate the infrastructure's performance, *even if they can buy the same amount of raw resources*.

It's happening already. Furthermore, centralization of infrastructure for security is an ongoing trend. Within Microsoft, the Windows organization adopted an "every team for itself" model in Vista. In Windows 7, the Windows Experience group (WEX) centralized security testing in a lab with hundreds of machines. For Windows 8, all Windows fuzz testing will take place in a central fuzzing lab. This model shields the rest of the Windows organization from the details of fuzzing and enables compliance with the Microsoft Security Development Lifecycle without needing dedicated security professionals in every Windows team.

Outside Microsoft, to take just two examples, IBM has introduced the Rational Developer Cloud, which makes testing resources with the Rational tools available on demand. Veracode offers on demand scanning of binaries to create a "Moody's score for code" that quantifies the risk of security bugs. Both offer more than simply migrating existing processes to the cloud: they change the way collaboration, response, and ship decisions are made by leveraging shared infrastructure.

Technology advances: Whitebox Fuzzing and Compositional Testing. Recent technology advances benefit from and further amplify these trends towards centralized fuzz testing. *Whitebox fuzzing* is a new approach to fuzzing pioneered at Microsoft in the SAGE tool and based on symbolic execution and constraint solving techniques. By observing what the program under test does with its inputs, whitebox fuzzing is able to drive its executions through code that was previously impossible to reach with traditional blackbox fuzzing. In the process, new security bugs can be found. In 2008-2009, the Windows 7 WEX organization deployed SAGE on a large scale as part of their centralized fuzzing lab: SAGE found 50% more bugs than all traditional fuzzers combined.

The next step is *compositional testing*, which creates *test summaries* from symbolic execution. These summaries are not only re-usable during a fuzzing session, but also apply across applications that share common components (such as DLLs) and over time (from one fuzzing session to the next). Compositional testing can result in a search algorithm that is *exponentially* faster than the current state of the art for whitebox fuzzing. Early experiments show that 90% or more of the current redundancies in whitebox fuzzing can be eliminated through compositional testing. Every test run in a centralized infrastructure can create new test summaries to improve all future test runs through this component.

Synergistic opportunities. We are developing a version of SAGE to leverage the gains from centralized infrastructure by collecting key statistics about whitebox fuzzing (code and taint coverage, x86 instructions not being handled, search bottlenecks, etc.). We are also currently building a general infrastructure to generate, store and re-use symbolic test summaries for large parts of the Windows operating system.

We envision that this data will be cross-checked and aggregated with other data sources such as MSRC data, Watson data, defect density models (adapted to security), code churn, the MSEC Locutus database, etc. The goal is to define a "fuzzing health index" that measures the quality of the security testing and evaluates the security risk from each piece of code.

Our data collection and analysis will improve the quality of tools and of the new processes they support. A bug found while testing Alice's code can improve the process and tools, so Bob benefits as well.

The whole is bigger than the sum of the parts. What is unique to the cloud? It is an infrastructure shared by multiple tenants. By sharing multiple data sources such as test results for different yet overlapping components of a product, the entire community benefits more than each of the individual tenants. These *formidable synergies* are the transformational forces that will drive the move towards *fuzzing in the cloud* and make it a reality.

Acknowledgements. This short note attempts to articulate why cloud computing might disrupt the security testing (i.e., fuzzing) world. We thank Jim Larus, Michael Levin and Matt Thomlinson for interesting comments.