ForAllSecure

What is Fuzz Testing?

The application attack surface is growing by 111 billion new lines of software code every year, with newly reported zero-day exploits rising from one-per-week in 2015 to one-per-day by 2021. As development speeds and deployment frequencies intensify, security must scale in tandem.

Positive testing, or deterministic testing, is a software testing process where an application is sent a valid set of inputs to ensure the application behaves as expected. Negative testing, or non-deterministic testing, is a testing process where an application is sent an invalid set of inputs to ensure the application remains stable in unexpected use cases. Both are necessary for developers to produce secure software applications.

Fuzz testing, or fuzzing, is a dynamic application security testing (DAST) technique for negative testing. Fuzzers send malformed inputs to applications with the objective of triggering bad behaviors, such as crashes, infinite loops, and/or memory leaks. These anomalous behaviors are often a sign of a previously unknown underlying software vulnerability.

Fuzz testing / dynamic testing is one of the most effective and better ROI methods for finding previously unknown software vulnerabilities. For example the Heartbleed vulnerability, which gave bad actors the opportunity to extract private information from websites, was discovered through fuzz testing. Another high-profile vulnerability in OpenWRT, which could allow attackers to compromise the embedded and networking devices running it, was discovered using ForAllSecure's Mayhem solution.

There are four types of fuzzers:.

- **Random fuzzers** send random inputs to an application. There is no systematic method to the generation of these test cases, and they do not resemble a valid input.
- **Template fuzzers** utilize manually supplied custom inputs and modify them to include anomalies. They are more effective than random fuzzers, because they resemble valid inputs.
- **Generational fuzzers** understand the inner workings of its input type. These tests are written to resemble a valid input, while evading common error-detection techniques. Protocol-based fuzzers are a common example of generational fuzzers.
- **Guided fuzzers** are intelligent, containing the capability to monitor and leverage the target's behavior to autonomously generate new, custom test cases on-the-fly. These fuzzers have scoring capabilities that measure the effectiveness of the test cases it sends. Guided fuzzers do rely on sample inputs, or a corpus, for initial guidance to explore a program however, thereafter, it monitors and leverages its target's behavioral feedback to generate new, customized test cases on-the-fly. These newly generated test cases aim to incrementally test new sections of code, checking the security of each new region it successfully penetrates.

The speed of guided fuzzers is undeniable. However, they struggle to break through complex conditional clauses within a program, limiting their testing depth. Without guidance, guided fuzzer randomly bounce around a program, struggling to reach deep into the code.

Continuous Fuzz Testing

Continuous fuzz testing has been a proven and accepted software security practice for years. It is also an advanced software testing technique, requiring technical savvy and budget to implement and maintain. As a result, fuzzing has been exclusive to technology behemoths, such as Google, Microsoft, Cloudflare, and the like. Fortunately, advancements in automation have improved usability dramatically, making fuzz testing increasingly accessible to the general public. Commercial organizations are recognizing its impact. Trends show fuzzing adoption rates are on the rise, particularly with security-fluent software developers, due to its benefits.

Advanced fuzzing includes symbolic execution. It solves complex branch conditions and generates a corpus, allowing the fuzzer to methodically resume its testing and analyze deeper into the program. With advanced fuzzing technologies, like ForAllSecure Mayhem, all organizations are able to employ proven dynamic negative testing without the personnel and expertise its predecessors required. In summary, the results are greater coverage and findings in less time, resources, and cost.

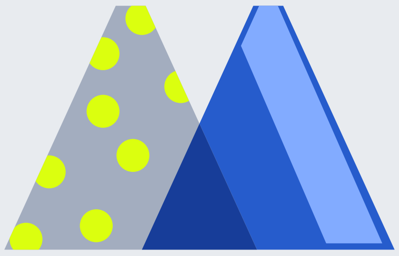
1,800
bugs and vulnerabilities in Office

11,687
bugs and vulnerabilities in Linux

27,000
bugs and vulnerabilities in Chrome and OSS

Guided Fuzzing
An instrumented intelligent fuzz testing technique that utilizes heuristics to generate inputs

Unique Inputs: ↓
Execution Speed: ↑



Symbolic Execution
A new technique that explores deeply by utilizing formal analysis to generate inputs logically

Unique Inputs: ↑
Execution Speed: ↓

Advanced Fuzzing
Unique Inputs: ↑
Execution Speed: ↑

Want to learn more?
Download "What is Advanced Fuzzing?"
for more details.

ForAllSecure