Bleeding-Edge Testing for Bleeding-Edge Defense

The Attack Surface is Expanding

Modern military technology are evermore reliant on software, exponentially expanding attack surfaces and leaving defense systems vulnerable to attack.

Grappling with the Scale of Vulnerabilities

According to Cybersecurity Ventures, newly reported zero-day exploits are predicted to rise from one-per-week in 2015 to one-per-day by 2021.

How Much Testing?

Carnegie Mellon University's Software Engineering Institute found the average code developed in the United States has 6,000 defects per million lines of code. Of those defects, 1 to 5 percent of defects are considered vulnerabilities. So, how much testing is required to mitigate vulnerabilities?

ForAllSecure projects that the number of test cases required is dependent on the number of lines of code in your codebase, average size of a function, average function complexity, and desired path coverage percentage.

FORMULA

\[ \text{LoC} / \text{size of average function} \times \text{average function complexity} \times \text{desired_path_coverage_percent} = \text{Number of test cases to cover all paths} \]

This formula projects a conservative estimation of the required number of test cases per line. This formula assumes a linear relationship between code base size and complexity. Typically, as codebases grow, complexity increase exponentially.

For the aircraft above, we assumed in these extrapolations:

- Average size of a function is 30 lines of code;
- Average function complexity is 8 (medium); and
- Desired path coverage is 50%.

Why Test?

Fuzzing is nearly three decades old, accepted by Silicon Valley's most successful tech behemoths. Teams at Google report that fuzzing finds 80% of their bugs, while the other 20% is uncovered by other forms of testing, or in production—meaning fuzzing finds bugs that are otherwise undetectable by other means. It's the next frontier of software security testing.

How Do We Cope?


What's Stopping Us?

Government agencies and branches rely on a portfolio of testing tools to address their Developmental Test and Evaluation (DT&E) and Operational Testing (OT&E) needs, but they weren't built for federal use cases. Several technical limitations stand in their way of productivity.

Uncover deep defects

Uncover deep defects with static analysis tools.

Automatically results with true false positives

Automatically results with true false positives from dynamic analysis tools.

I am unbreakable! I am not made of code

I am unbreakable! I am not made of code.

Shift left in validation testing

Shift left in validation testing.

Verify and validate app supply chains

Verify and validate app supply chains.

Advanced fuzzing made easy

Advanced fuzzing made easy.

Why Fuzz?

Fuzzing is nearly three decades old, accepted by Silicon Valley's most successful tech behemoths. Literally, teams at Google report that a fuzzer has uncovered bugs in other bugs. According to the EFF, however, there is no such thing as a fuzzer that will automatically uncover bugs in other bugs. In fact, Fuzzing is the only method that reliably finds bugs that are otherwise undetectable by other means. On the next front of security testing, we need a rewrite of human error.

Want to learn more about Mayhem?

Get the Government and Defense Solution Brief.

Know you're ready for Mayhem?

Request a personalized demo.