ForAllSecure

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.

From 2012 to 2017, the Department of Defense found vulnerabilities in nearly all weapons systems that were under development.

Today's vulnerability management solutions are reactive, leaving agencies and branches unequipped to handle the scale of vulnerabilities being found.

The GAO-19-128 report revealed only 1 of 20 cyber vulnerabilities identified in a previous assessment had been corrected.

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?

Boeing 787 Dreamliner

14 million lines of code
~1,866,666 test cases

F-22 Raptor

1.7 million lines of code
~266,666 test cases

F-35 Lightning II

24 million lines of code
~3,200,000 test cases

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.

As codebases grow in LoC with each new release, complexity increases exponentially, demanding new test cases and additional, continuous testing over time. Manual testing simply won't scale with today's rapid pace of development.

FORMULA

LoC / size_of_average_function * average_function_complexity * desired_path_coverage_percent
= 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?

"In today's world, you need to modify code if you're still fighting with it. The longer your code sits stationary, the more dormant it is. You could imagine in a future war where we are changing software on a daily basis as a necessary factor for winning. We have to get quality code for both the taxpayer and the warfighter."

Will Roper
Air Force Assistant Secretary for Acquisition, Technology, and Logistics

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.

DEVELOPMENTAL TEST AND EVALUATION

High false-positives waste scarce technical expertise on defect validation and triaging. Technical security experts need to be freed of boring, repetitive tasks to focus on creatively solving our toughest defense challenges.

OPERATIONAL TEST AND EVALUATION

Because scanners only uncover known vulnerabilities in third-party code, most assessments turn up a clean bill of health. This is because many defense systems only rely on proprietary code. Today, there's no way to force vendors to conduct testing themselves.

How Do We Cope?

John S. McCain National Defense Act calls for a report on the enhancement of software security for critical systems, recommending binary analysis and symbolic execution tools developed under the Cyber Grand Challenge of the Defense Advanced Research Projects Agency.

Why Fuzz?

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.

- Uncover deep defects
- Shift-left verification testing
- Actionable results with zero false-positives
- Verify and validate app supply chains
- Scale with autonomous test case generation
- Advanced fuzzing made easy

Want to learn more about Mayhem?

Get the Government and Defense Solution Brief.

Download

Know you're ready for Mayhem?

Request a personalized demo.

Request