# Hacker Lexicon: What Is Fuzzing?

Author: Andy Greenberg.Andy Greenberg Security
WIRED | 2016-06-02

Hackers sometimes portray their work as a precise process of learning every detail of a system—even better than its designer—then reaching deep into it to exploit secret flaws. But just as often, it's practically the opposite, a fundamentally random process of poking at a machine and watching what happens. Refine that random poking to a careful craft of trial and error, and it becomes what hackers call "fuzzing"—a powerful tool for both computer exploitation and defense.

TL;DR: Fuzzing is the usually automated process of entering random data into a program and analyzing the results to find potentially exploitable bugs.

In the world of cybersecurity, fuzzing is the usually automated process of finding hackable software bugs by randomly feeding different permutations of data into a target program until one of those permutations reveals a vulnerability. It's an old but increasingly common process both for hackers seeking vulnerabilities to exploit and defenders trying to find them first to fix. And in an era when anyone can spin up powerful computing resources to bombard a victim application with junk data in search of a bug, it's become an essential front in the zero-day arms race.

Compared with traditional reverse engineering, "it's a kind of dumb science," says Pedram Amini, chief technology officer of the cybersecurity firm InQuest and a co-author of the book *Fuzzing: Brute Force Vulnerability Discovery*. "You're throwing a whole lot of data at a program, mutating it quickly and relying on your monitoring of the software to find when something bad has happened instead of meticulously mapping out the data flow to find a bug…It's a way of killing off a lot of bugs very quickly."

A hacker fuzzing Internet Explorer, for instance, might run Microsoft's browser in a debugger tool, so that they can track every command the program executes in the computer's memory. Then they'd point the browser to their own web server, one designed to run their fuzzing program. That fuzzer would create thousands or even millions of different web pages and load them in its browser target, trying variation after variation of HTML and javascript to see how the browser responds. After days or even weeks or months of those automated tests, the hacker would have logs of the thousands of times the browser crashed in response to one of the inputs.

Those crashes themselves don't represent useful attacks so much as annoyances; the real goal of fuzzing is not merely to crash a program, but to hijack it. So a hacker will scour their fuzz inputs that led to crashes to see what sorts of errors they caused. In some small set of cases, those crashes may have happened for an interesting reason—for example, because the input caused the program to run commands that are stored in the wrong place in memory. And in those cases the hacker might occasionally be able to write their own commands to that memory location, tricking the program into doing their bidding—the holy grail of hacking known as code execution. "You shake a tree really hard, and you use a bunch of filters," says Amini. "Eventually fruit will come out."

Fuzzing's method of using random data tweaks to dig up bugs was itself an accident. In 1987, University of Wisconsin at Madison professor Barton Miller was trying to use the desktop VAX computer in his office via a terminal in his home. But he was connecting to that UNIX machine over a phone line using an old-fashioned modem without error correction, and a thunderstorm kept introducing noise into the commands he was typing. Programs on the VAX kept crashing. "It seemed weird, and it triggered the idea we should study it," he says.

With a group of students, Miller created the first purpose-built fuzzing tool to try to exploit that method of haphazardly stumbling into security flaws, and they submitted a paper on it to conferences. “The software community slaughtered me. ‘Where’s your formal model?’ They’d say. I’d say, ‘I’m just trying to find bugs.’ I got raked over the coals,” he remembers. “Today, if you’re a hacker trying to crack a system, the first thing you do is fuzz test it.”

In fact, fuzzing has grown from a low-budget technique used by individual hackers to a kind of table-stakes security audit performed by major companies on their own code. Lone hackers can use services like Amazon to spin up armies of hundreds of computers that fuzz-test a program in parallel. And now companies like Google also devote their own significant server resources to throwing random code at programs to find their flaws, most recently using machine learning to refine the process. Companies like Peach Fuzzer and Codenomicon have even built businesses around the process.

All of that, Amini argues, has made fuzzing more relevant than ever. “Software shops are doing this work as a standard part of their development cycle,” he says. “It’s a great investment, and they’re helping to improve the world’s security by burning software cycles for everyone.

read:https://www.wired.com/2016/06/hacker-lexicon-fuzzing/