CERT/CC Blog

Vulnerability Insights

A Look Inside CERT Fuzzing Tools

POSTED ON NOVEMBER 5, 2012 BY ALLEN HOUSEHOLDER [/AUTHOR/ALLEN-HOUSEHOLDER] IN TOOLS [HTTPS://INSIGHTS.SEI.CMU.EDU/CERT/TOOLS/]

Hi, this is Allen Householder of the CERT Vulnerability Analysis team [http://www.cert.org/vuls/]. If you've been following this blog for a while, you are probably familiar with our fuzzing tools: Dranzer [http://www.cert.org/vuls/discovery/dranzer.html], the CERT Basic Fuzzing Framework (BFF) [http://www.cert.org/vuls/discovery/bff.html], and the CERT Failure Observation Engine (FOE) [http://www.cert.org/vuls/discovery/foe.html]. While creating tools that can find and analyze vulnerabilities makes up a significant portion of our work in the CERT Vulnerability Analysis team [http://www.cert.org/vuls/], our focus [http://www.cert.org/vuls/discovery/] is on developing and communicating the knowledge we've built into those systems.

To that end, we recently published a pair of reports that describe a few of the heuristics and algorithms implemented in the BFF and FOE fuzzing tools. We briefly mentioned these techniques in the release announcements [http://www.cert.org/blogs/certcc/2012/10/updates_to_cert_fuzzing_tools.html] for the tools, but did not describe how they work in detail. Abstracts and links to the reports can be found below.

Probability-Based Parameter Selection for Black-Box Fuzz Testing
[http://www.sei.cmu.edu/library/abstracts/reports/12tn019.cfm]
Dynamic, randomized-input functional testing, or black-box fuzz testing, is an effective technique for finding security vulnerabilities in software applications. Parameters for an invocation of black-box fuzz testing generally include known-good input to use as a basis for randomization (i.e., a seed file) and a specification of how much of the seed file to randomize (i.e., the range). This report describes an algorithm that applies basic statistical theory to the parameter selection problem and automates selection of seed files and ranges. This algorithm was implemented in an open-source, file-interface testing tool and was used to find

and mitigate vulnerabilities in several software applications. This report generalizes the parameter selection problem, explains the algorithm, and analyzes empirical data collected from the implementation. Results of using the algorithm show a marked improvement in the efficiency of discovering unique application errors over basic parameter selection techniques.

Well There's Your Problem: Isolating the Crash-Inducing Bits in a Fuzzed File
[http://www.sei.cmu.edu/library/abstracts/reports/12tn018.cfm]
Mutational input testing (fuzzing, and in particular dumb fuzzing) is an effective technique for discovering vulnerabilities in software. However, many of the bitwise changes in fuzzed input files are not relevant to the actual software crashes found. This report describes an algorithm that efficiently reverts bits from the fuzzed file to those found in the original seed file, keeping only the minimal bits required to recreate the crash under investigation. This technique reduces the complexity of analyzing a crashing test case by eliminating the changes to the seed file that are not essential to the crash being evaluated.

By the way, we're hiring. We've got a few different vulnerability analysis openings along with a variety of other technical security positions posted at www.cert.org/jobs/ [http://www.cert.org/jobs/] . If improving how vulnerabilities are discovered, analyzed, and remediated is your thing, we'd like to talk to you.

About the Author

Allen Householder

Contact Allen Householder [https://www.sei.cmu.edu/contact.cfm]
Visit the SEI Digital Library for other publications by Allen
[https://resources.sei.cmu.edu/library/author.cfm?authorID=4483]
View other blog posts by Allen Householder [/author/allen-householder]