CERT'S PODCASTS: SECURITY FOR BUSINESS LEADERS: SHOW NOTES

The Power of Fuzz Testing to Reduce Security Vulnerabilities

Key Message: To help identify and eliminate security vulnerabilities, subject all software that you build and buy to fuzz testing.

Executive Summary

Traditional software testing tends to focus on demonstrating that a software application or product functions as specified by its requirements and provides the features it was built to provide. The purpose of fuzz testing is to cause software to misbehave, break, or crash. It does so by providing invalid and unexpected input to an application. If not addressed, vulnerabilities exposed during fuzz testing can be used by attackers to take control of unsuspecting software and execute malicious code.

In this podcast, Will Dormann, a member of CERT's Cyber Threat and Vulnerability Analysis team, discusses fuzz testing. It is a relatively low cost/high value software testing technique that helps identify software security defects and vulnerabilities during software code and test. It can also be used to determine the security robustness of any software that you acquire or buy, including vendor software.

PART 1: WHY FUZZ TESTING?

Defining Fuzz Testing

Fuzz testing breaks or crashes a software application by providing invalid, malformed input. Such input can also cause the application to misbehave, performing in ways for which it was never intended.

Traditional software testing primarily makes sure that the software works as expected and provides the features for which it has been developed.

Fuzz Testing and Security

Fuzz testing can discover a number of software vulnerabilities, including those that attackers like to target (such as buffer overflows). If these are not removed before software goes into production, an attacker can cause a denial of service condition and can insert and execute malicious code.

Vendors that do not perform fuzz testing on their products prior to release are placing their customers and users at risk.

The Business Case for Fuzz Testing

All software developers want to produce high quality software. They want to eliminate any situation that could cause their application to crash.

Fuzz testing can be easily automated so it requires a fairly low level of staff resources. Results of fuzz testing tools can help developers identify the programming flaws that caused the software to crash or misbehave.

Fuzz testing can also be used to regression test software that has changed, to make sure no new vulnerabilities were introduced as a result of the change.

PART 2: FUZZ TESTING TECHNIQUES; CERT'S Dranzer TOOL FOR ActiveX CONTROLS

Categorizing Fuzz Testing Techniques

There are generally two types of fuzz testing. These refer to how a fuzz testing tool forms the malformed input that is used to test the software.

Testing Using a Smart Fuzzer

A smart fuzzer (also known as a generational fuzzer) generates input to an application from scratch. A tool that uses this approach generates data that is formatted in a way that makes sense to the application under test.

A smart fuzz testing tool used to test a PDF viewing application (such as Adobe Reader or Foxit) understands the specification for the PDF file format. It creates a test PDF file from scratch and mangles some of its contents. For example, the tool may provide a very large number for document height and width (greater than what the viewing application expects).

CERT’s Dranzer Tool

CERT has developed a smart fuzzing tool called Dranzer which is publicly available. Dranzer fully understands the ActiveX specification and generates test data to exercise ActiveX controls on a system running Microsoft Windows.

Testing Using a Dumb Fuzzer

A dumb fuzzer (also known as a mutational fuzzer) starts with a validly formed test file and changes bits of information to corrupt the file. Using the PDF file example, a dumb fuzzing test tool starts with a valid PDF file and modifies it by making random changes.

In a production environment, a software application may generate or receive files that are not compatible with other applications. In addition, files can be inadvertently corrupted. In order to combat this, the software application needs to be able to handle these situations without crashing.

PART 3: USE AUTOMATED FUZZ TESTING WHEN BUILDING AND BUYING SOFTWARE

Using Dranzer to Locate Security Vulnerabilities

CERT’s Dranzer tool has been used repeatedly to identify common software vulnerabilities in ActiveX controls such as buffer overflows and integer overflows.

A buffer overflow can allow an attacker to gain control over the application and cause the application to run the attacker’s code. Dranzer tests for a buffer overflow by generating a very large string and sending it to the application that is expecting a string parameter as valid data input.

Dranzer has been used successfully to identify vulnerabilities in software that uses ActiveX controls including Microsoft’s Internet Explorer. When users visit a website (using Internet Explorer) the website can exploit the ActiveX controls in Internet Explorer. The attacker in control of the website can then run malicious code on the user’s system.

Fuzz testing does identify software defects and vulnerabilities including many that affect software security.

Automation Is Key

The time and resources required to conduct fuzz testing depends on the size and type of the application being tested. That said, fuzz testing can be greatly automated which helps reduce cost and schedule.

A tool such as Dranzer can be used to retrieve ActiveX controls, test these controls, catalog the results into a database, and report these to the developer. Fuzz testing using file formats can also be automated.

Once a test tool environment is set up, a developer can let it run in the background, autonomously, while continuing to develop software or perform other tasks.

Use Fuzz Testing When Buying Software

Running fuzz testing tools on software that you buy under a contract or service-level agreement, as well as software provided by vendors, is highly recommended. Fuzz testing results provide a good indication of the quality of the software. Such results can be used to compare two or more competing products that you are evaluating.

Resources

CERT’s Vulnerability Analysis website

CERT’s Dranzer website

CERT's Basic Fuzzing Framework and blog

“Major malware campaign abuses unfixed PDF flaw.” Computerworld, April 28, 2010.

Open Web Application Security Project (OWASP) fuzzing website

Fuzzing.org

CERT’s Malware Apprenticeship Program