CERT/CC Blog

Vulnerability Insights

CERT Failure Observation Engine 2.0 Released

POSTED ON JULY 23, 2012 BY ALLEN HOUSEHOLDER [/AUTHOR/ALLEN-HOUSEHOLDER] IN TOOLS [HTTPS://INSIGHTS.SEI.CMU.EDU/CERT/TOOLS/]

Hi folks, Allen Householder from the CERT Vulnerability Analysis team here. Back in April, we released version 1.0 of the CERT Failure Observation Engine (FOE) [http://www.cert.org/vuls/discovery/foe.html], our fuzzing framework for Windows. Today we're announcing the release of FOE version 2.0. (Here's the download [http://www.cert.org/download/foe].) Although it has only been a few months since we announced FOE 1.0 [http://www.cert.org/blogs/certcc/2012/04/cert_failure_observation_engin.html], our development cycle is such that FOE 2.0 actually reflects nearly a year of additional improvements over the 1.0 release.

Our main focus in developing FOE 2.0 was to apply what we learned from creating the CERT Basic Fuzzing Framework version 2.5 [http://www.cert.org/vuls/discovery/bff.html] for Linux and OS X to improve our fuzzing capabilities on Windows. We are gradually converging our code bases for BFF and FOE to simplify maintenance and the incorporation of new features. We're not quite there yet, but FOE 2.0 reflects a significant step in that direction. Read on for more details.

Improved Support for Multiple Seed Files

FOE 1.0 performed a specified number of tests on a seed file before moving to the next one. This required the user to tune the configuration parameters to ensure sufficient coverage of the seed files. Eliminating the need for user tuning, FOE 2.0 uses BFF 2.5's SeedfileSet module to dynamically choose seed files based on observing the seed files' performance in producing unique crashes throughout a campaign. Using a machine learning technique, it automatically adjusts its efforts to focus on the seed files that produce the most unique crashes.

Crashes Found During Minimization Are Analyzed as Well

First introduced in BFF 2.5, FOE 2.0 now supports minimizer recycling. When FOE is minimizing a crash and it comes across a different crash than the one being minimized, it will queue that new crash up for further analysis after the current minimizer run completes. This helps to increase the crash yield since frequently a crashing test case is located near other crashing test cases in the input space.

Improved Machine Learning Implementation Applied to Both Seed File Selection and Rangefinder

Using the SeedfileSet and Rangefinder modules from BFF 2.5 allows us to take advantage of the improvements we made to the machine learning algorithms that control BFF's (and now FOE 2.0's) fuzzing strategy.

Minimizer Tuned for Performance

BFF 2.5's faster minimizer is now part of FOE 2.0.

Optional Minimization-to-String Feature

Introduced in BFF 2.5, this feature can help simplify the creation of proof-of-concept exploits from crashing test cases. Say you have a crashing test case and you want to get to a proof-of-concept exploit. When you load the crash into your debugger, the problem is that you can't easily tell which registers, stack values, or memory locations are under your control. But what if you could change the crashing test case so that it had only the bytes required to cause that crash, and the rest were all masked out with a fixed value, say "x" (0x78)? Then if you saw EIP=0x78787878, you'd know that you may already be a winner. The minimize-to-string option does just that.

Run this to get command-line usage of the minimizer:

```
tools\minimize.py --help
```

Run this to minimize a crashing test case to a string of 'x' characters:

```
tools\minimize.py --stringmode <crashing_testcase>
```

When minimizing to a string pattern, FOE will use the resulting byte map to create an additional minimized file that uses a unique pattern, similar to what Metasploit's `pattern_create()` provides. This helps answer the question of which bytes you are looking at in a debugger. Note that this file is not guaranteed to produce the same crash as the original string minimization.

Continues Handled Exceptions

Microsoft Windows' Structured Exception Handling (SEH) mechanism provides the ability for an application to handle some of its own exceptions. FOE 1.0 only ran the !exploitable tool on the first exception that was encountered. New in FOE 2.0, when a crash is encountered FOE will attempt to continue execution of the program and provide !exploitable output for subsequent exceptions. In some cases the first exception encountered is not as exploitable as one or more of the exceptions encountered upon continued execution.

Dranzer's Button Clicker Now Included

The CERT Dranzer [http://www.cert.org/vuls/discovery/dranzer.html] tool for ActiveX fuzzing included a feature that automated button clicking for use when testing GUI applications. We've rolled that feature into FOE 2.0. Paraphrasing the Dranzer paper [http://www.cert.org/archive/pdf/dranzer.pdf]:

Many applications present new windows or dialogs when the program is initialized or certain methods are called. If these windows are not dismissed, the testing process can hang. FOE installs a global hook during the testing process to address this problem. FOE monitors all open windows and when a new window is activated, FOE determines if a window contains any buttons by looking for the class name "button" within the child windows. If the child window contains a button, the display name is obtained by using the `GetWindowText()` API. If a button contains a display name such as "OK," "No," or "Cancel," FOE clicks the button based on its priority as determined by its order in the "act upon" list. FOE will also close new windows, regardless of whether they have buttons or not. This design feature allows FOE to test many applications without hanging due to a window that is opened.

New drillresults.py Script for Picking Out Interesting Crashes

With some target applications, FOE may produce too many uniquely crashing test cases to investigate manually in a reasonable amount of time. We have provided a script called `drillresults.py` to pick out crashes that are most likely to be exploitable and list those cases in a ranked order. (The most exploitable cases are listed first.)

To use this script, run
```
tools\drillresults.py
```

For command-line usage, run
```
tools\drillresults.py --help
```

New Fuzzers Added

FOE 2.0 adds four new fuzzing strategies: ***drop***, ***insert***, ***truncate***, and ***verify***. The set of available fuzzing strategies now includes

- ***bytemut***: replace bytes with random values
- ***swap***: swap adjacent bytes
- ***copy***: copy seed file (do not mutate)
- ***bitmut***: flip random bits
- ***wave***: cycle through every possible single-byte value, sequentially
- ***drop***: removes one byte from the file for each position in the file
- ***insert***: inserts a random byte for each position in the file
- ***truncate***: truncates bytes from the end of the file
- ***verify***: do not mutate file; used for verifying crashing test cases

Odds and Ends

- FOE 2.0 has been refactored into object-oriented code.
- FOE 2.0 has been upgraded to python 2.7.
- Many other features and fixes from the CERT Basic Fuzzing Framework (BFF) v2.5 for Linux / OS X have been rolled in.
- See our blog post announcing BFF v2.5 for additional details on some of the features described above.

Vulnerabilities Found with FOE 2.0

We have found a number of vulnerabilities using FOE 2.0. For example, VU#118913 [http://www.kb.cert.org/vuls/id/118913] describes a set of vulnerabilities in Oracle Outside In [http://www.oracle.com/us/technologies/embedded/025613.htm] that affect a wide range of products including certain versions of AccessData Forensic Toolkit (FTK), ACD Systems Canvas, Avantstar Quick View Plus, Cisco Security Agent DLP, Guidance Encase Forensics, HP TRIM, IBM OmniFind Enterprise Edition, Kamel Fastlook 2009, kCura Relativity, Kroll Ontrack EasyRecovery and PowerControls, Lucion FileCenter, McAfee GroupShield and Host Data Loss Prevention, MarkLogic Server, Microsoft Exchange 2007 and 2010, and NewSoft America Presto! PageManager 9. See also CVE-2012-1766 [http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-1766], CVE-2012-1767 [http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-1767], CVE-2012-1768 [http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-1768] CVE-2012-1769, [http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-1769] CVE-2012-1770 [http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-1770], CVE-2012-1771 [http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-1771], CVE-2012-1772 [http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-1772], CVE-2012-1773 [http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-1773], CVE-2012-3106 [http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-3106], CVE-2012-3107 [http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-3107], CVE-2012-3108 [http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-3108], CVE-2012-3109 [http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-3109], and CVE-2012-3110 [http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-3110].

Quick Start

To install FOE 2.0, run FOE-2.0-setup.exe in a virtual machine. The installer should detect and attempt to download prerequisites and configure your environment appropriately.

Many options can be specified in the configuration file. See the included README.txt and `configs\examples` for more information. FOE can be downloaded on the CERT website [http://www.cert.org/vuls/discovery/foe.html].

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]