Software Engineering Institute
Carnegie Mellon University

SEI Insights

Home > CERT/CC Blog > Hacking the CERT FOE

CERT/CC Blog

Vulnerability Insights

Hacking the CERT FOE

POSTED ON NOVEMBER 26, 2013 BY WILL DORMANN [/AUTHOR/WILL-DORMANN] IN TOOLS [HTTPS://INSIGHTS.SEI.CMU.EDU/CERT/TOOLS/]

Hey folks, it's Will. Every now and then I encounter an app that doesn't play well with FOE [http://www.cert.org/vuls/discovery/foe.html]. You don't have to throw your hands up in defeat, though. Because FOE (and BFF [https://www.cert.org/vuls/discovery/bff.html]) are written in Python, it's pretty easy to modify them to do what you like.

The particular target application that I encountered does not take a filename as a command-line parameter. Instead, this application takes a parameter that is a directory that contains a configuration file that specifies the file to open.

FOE assumes that the fuzzed file is a command-line parameter, and every invocation of the target application uses a different path to the fuzzed file somewhere in the configured `fuzzdir`. With this particular target application, I'd like for FOE to put the fuzzed file in a fixed location for every iteration. As it turns out, this change takes just one additional line of code!

To patch FOE to behave as we like, we modify the `Fuzzer` object to write a copy of the fuzzed output to a location that we specify. In particular, find the `_postfuzz()` function in `certfuzz\fuzzers\fuzzer_base.py` and add this statement to the beginning:

```
write_file(self.fuzzed, '<target_path>')
```

For example, in my case I have

```
    def _postfuzz(self):

        logger.info('Copying fuzzed file to temp folder...')
        write_file(self.fuzzed, 'c:/temp/input')

        if self.options.get('fuzz_zip_container') or not self.sf.is_zip:
            return
```

And in my `foe.yaml` file, I have

```
cmdline_template: $PROGRAM c:/temp
```

The `_postfuzz()` function is what happens after the file contents are mutated. This architecture is how FOE is able to reconstruct a valid zip file with fuzzed contents, for example. In this case, we use `write_file`, which is part of the CERT `fuzztools` package. The byte array of the mutated file is `self.fuzzed`, and the second parameter is the output filename.

And that's it! One line of code added and FOE is now behaving as I want. Getting other FOE functionality, such as minimization, working with this scheme may require additional changes, but just the above changes are enough to start fuzzing. This is only one example of what can be done with FOE, but, really, your imagination is the limit!

FOE is open source, so if it doesn't do exactly what you want, feel free to change it. If you've got a good idea for how to make FOE better, please let us know [http://mailto:cert@cert.org?subject=CERT%2FCC%20Blog%20Comment%20INFO%23133478] .

About the Author

Will Dormann

Contact Will Dormann [https://www.sei.cmu.edu/contact.cfm]
Visit the SEI Digital Library for other publications by Will [https://resources.sei.cmu.edu/library/author.cfm?authorID=2547]
View other blog posts by Will Dormann [/author/will-dormann]