Homework 5

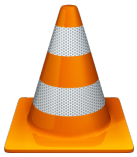
CIS 4930 / CIS 5930
Offensive Security
Spring 2014

Due **April 21** 2014, by MIDNIGHT
Worth: 100 points (~5% of final grade)

Electronic turn in (Turn in via email to instructor. Email address is redwood@cs.fsu.edu)
The email must be titled in the following format:
"[OCS2014] hw5 <your last name>". (where <your last name> is your last name)
Example: [OCS2014] hw5 redwood

This homework pertains to all the topics covered during our fuzzing lectures. You have the entire semester to do this homework, but in reality it should take 2-5 hours (though closer to 2 hours), as the target is sufficiently vulnerable.

Part A [worth 100 points]



Your goal for this homework is to find a 0-day bug, AND ethically disclose it. **If you do not ethically disclose it, no points will be given.** Your target is VLC Media player (http://www.videolan.org/vlc/index.html). The turn in for this part of the homework will be a media file that reliably produces a unhandled/unchecked crash in VLC. In other words, provide a file that reliably causes VLC to crash when opened, without VLC printing error messages / dialog warning boxes. You will then disclose the bug to the developers of VLC (this part is very important) as part of the homework. Furthermore VLC is entirely open source, which allows source code auditing, which will be needed for Part B (optional).

Attack surface of VLC

VLC has a vast attack surface, as it has many features (movie editing, custom skins, extensions, many accepted input formats for both video and audio). By far the biggest portion of the attack surface for VLC is its vast amount of libraries that handle audio/video encodings (see http://www.videolan.org/vlc/features.html). Furthermore each of these libraries have to work on Windows, Linux, Mac, and other platforms, which means there is a lot of room for things to be implemented differently (and that's where bugs commonly happen).

Related Efforts / Forewarning

Google has recently put a lot of effort (via mutational fuzzing) into fixing FFMpeg and Libav and has fixed over a thousand bugs in the ffmpeg libraries. You should read http://googleonlinesecurity.blogspot.com/2014/01/ffmpeg-and-thousand-fixes.html to familiarize yourself with their overall approach, **as well as to grab their sample files**. As a warning, FFmpeg and maybe Libav are likely not a good target sfor your efforts, as so many bugs have already been fixed.

Target Bugs

We are looking for crashes caused by the following categories of bugs

- NULL pointer dereferences,
- Invalid pointer arithmetic leading to SIGSEGV due to unmapped memory access,
- Out-of-bounds reads and writes to stack, heap and static-based arrays,
- Invalid free() calls,
- Double free() calls over the same pointer,
- Division errors,
- Assertion failures,
- Use of uninitialized memory.

Getting Started

First you absolutely should watch the 30 minute tutorial that Mitch Adair has put together on "Intro to Fuzzing" (See http://www.youtube.com/watch?v=P7xUEUJFSA0). The slides for the video are located at: https://csg.utdallas.edu/wp-content/uploads/2012/10/Fuzzing-Part-2.pdf

When done with that you will need to do the following:

1. Download VLC
2. Pick a fuzzer to use & set it up
    a. Suggested fuzzers:
        i. Mitch's fuzzer: http://rmadair.github.io/fuzzer/
        ii. http://peachfuzzer.com/
        iii. http://www.cert.org/vulnerability-analysis/tools/foe.cfm (Windows only)
        iv. http://www.cert.org/vulnerability-analysis/tools/bff.cfm (Linux / Mac)
    b. **Set up the fuzzer inside a VM (DO NOT DO THIS ON YOUR HOST OS)**, as your fuzzer will likely be mutating potentially malicious files (I make no guarantees of the safety of sample files you may find online; however, you should be safe doing this all inside a up to date virtual machine).
3. Download sample files (Do not pirate movies / copyrighted content)
    a. http://samples.mplayerhq.hu/
    b. http://www.filecrop.com/ (These two are mentioned at the end of the video)
4. Set it all up to fuzz VLC with the fuzzer of your choice given the input files

Coordinated Disclosure:

In order to prove to me that you've disclosed the bug properly, we're going to run all disclosures through the instructor (redwood@cs.fsu.edu) first, and he will be CC'd on your email to the developers for the bug(s) you've found.

When you've gotten to the point where you have a test case that reliably produces a crash from one of the above bugs, email the instructor so he can review and help you put together your disclosure report / email. emails here should be titled "[OCS2014] 0day redwood", where redwood would be your last name.

Turn in:

For Part A's turn in you will have to provide a brief writeup of what you did, the fuzzer you used, and anything interesting you'd like to share that you might have found. In addition to the writeup should be the turn-in file that triggers the bug you have found in VLC (You should tar.gz or 7z compress your file so that it doesn't get stripped).

(Extra Credit) Part B [worth 100 points of extra credit]

To earn the extra credit for PART B, you will have to root-source the bug, and identify it in the source code of VLC. A write-up demonstrating where in the source code this bug lies is expected. Your write-up should also provide some impact analysis, stating the risk of this bug, and how it might be exploited (if possible).

(Extra Credit) Part C [Automatic A+ in class]

To complete part C, you will have to develop "proof-of-concept" (POC) exploit for the bug (or a bug that you have found). The goal of your exploit is to spawn calc.exe (on windows), or a calculator for the target OS/architecture, against ALSR/DEP/NX. Simply popping a calculator is very common in POC exploits, and there is tons of different shellcode to do this. Your POC will be required to defeat ASLR and NX/DEP on the system. These topics will be covered in depth later in the class, so do not worry if you are not familiar with these terms.

To defeat ASLR you will likely need to find an infoleak / memleak bug in VLC, which adds a great deal of difficulty. These bugs are not found by fuzzers, as they are very subtle. We may cover finding infoleaks / memleaks in class if time permits; however, we may not. Defeating DEP/NX required ROP, which will be covered later in the semester as well.

By fuzzing VLC, finding bugs, developing a POC that defeats ASLR/DEP/NX, AND ethically disclosing your findings you will have demonstrated a mastery of the difficult core subjects in this course, and thus you will be awarded an A+. All other grades / assignments will be irrelevant and you will have won at Offensive Computer Security.