

ECE 1776 – Midterm Progress Report

Group Members:

Rita Chiu – 980290250

Jacky Mok – 990872301

Vicky Tsang – 981000580

Recap of Problem

We will try to verify the hypothesis that code vulnerabilities occur on less commonly executed paths by instrumenting programs with documented vulnerabilities and determine whether there is a correlation between the frequency of the paths executed and the location of the vulnerabilities.

Outline of Proposed Solution

- identify open source programs that have documented vulnerabilities
 - o There should be a mix of vulnerabilities (stack overflow, formatted string attack, resource leaks, etc...)
- Instrument program using EXE, inline C/C++ code or gprof
- Gather statistics on the frequency of path executions
- Compare the location of the vulnerabilities and hot paths
- Verify / argue against the thesis

Progress

We successfully identified several programs with a variety of documented vulnerabilities:

- zlib:
 - o DoS: compression library is vulnerable to a denial-of-service condition
 - o Buffer Overflow: zlib inflate() routine vulnerable to buffer overflow
- gzip:
 - o DoS: gzip contains an infinite loop vulnerability in its LZH handling
 - o Execution of arbitrary code: gzip contains a buffer overflow if the file size is over 1MB
 - o Execution of arbitrary code: gzip contains an array out-of-bounds vulnerability in make_table()
- open-ssh:
 - o Race condition: OpenSSH contains a race condition vulnerability
- perl:
 - o Formatted String Attack: Perl programs providing user-controlled I/O format strings may contain format string vulnerabilities

We tried to locate the source code for EXE to see if we can use it to profile paths as well as generate test cases where additional vulnerabilities may be found but we were unable to find the source code. Therefore, we decided to use gprof instead to profile which code

path is taken in the source code. We successfully compiled the source for the programs above with -g and -gp to enable gdb and gprof to be used. We are executing the tests on a Pentium III 1 Gz machine running Fedora Core release 4.

To gather statistics on the frequency of path executions, we will generate our own test cases with the goal of covering normal usage as well as corner cases for program execution. So far, we have successfully generated cases where the documented vulnerabilities are exploited. The challenge is to come up with a way to systematically generated test cases so that the statistics gathered from frequency of path execution is unbiased and useful.

Another difficulty that we're facing is crafting libraries to exploit vulnerabilities. Unlike string input, crafting a library to exploit vulnerability is not an easy task. We are still working towards creating a library that can successfully exploit the vulnerabilities documented.

Remaining Work Items

Although we have made steady progress so far, there is still a lot of work to be done. Test suites (and libraries) still need to be written in order to sufficiently test the programs as well as enable us to gather path execution frequencies. After the test suites are complete, we will run them through the programs and use gprof to identify the paths executed. We will then analyze and determine if there are any correlations between frequency of path execution and the location of vulnerabilities.

References

- [1] gzip - https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=204676
- [2] openssh - <http://www.kb.cert.org/vuls/id/851340>
- [3] zlib - <http://www.kb.cert.org/vuls/id/238678>
- [4] perl - <http://www.kb.cert.org/vuls/id/946969>