Assignment 0: Attack!

1 Problem

You are to find an exploitable security vulnerability in an open source peer-to-peer network application, demonstrate that the vulnerability is real, provide a patch and finally issue a security advisory.

You are allowed to work on this project in groups of two. Furthermore, you should document (!) the steps of your security analysis. In case that you fail to find a vulnerability, your report can be used to demonstrate that you were just unlucky (but working hard and systematic in your approach). In this case, your report will be graded instead of the security advisory.

While this assignment is due towards the end of the quarter, you **must** get started **immediately**. Due dates are rather crowed at the end of the course.

2 Approach

Look for open source peer-to-peer projects on `freshmeat.net`. You can use those that focus on security (such as networks claiming to provide anonymity) or just general P2P networks. Do not select projects that nobody cares about (not in top 10,000 by popularity). Indirect exploits (bugs in a library used by the project) also count, but maybe harder too find. Note that whatever exploit you find has to match with the goals of the project that you are attacking. Any exploit that a reasonable maintainer of the respective software would want to fix counts.

One suggested approach for finding vulnerabilities is to use the `coverity` static analysis tool, especially on less-popular projects, and to investigate the reported bugs. In order to use `coverity`, you must be able to compile the software. If the Linux lab machines fail to satisfy the dependencies, feel free to request any Debian package to be installed on the system(s).

Another approach is to run the P2P software in conjunction with `zzuf` to introduce random permutations into the network traffic. If you are lucky, you may be able to crash it. If the protocol uses checksums, you may want to disable the checksum code to make sure that fuzzed traffic does not simply

get discarded. Depending on the computational requirements of the software combining `zzuf` with `valgrind` may increase your yield.

The first two methods maybe useful to find simple bugs that enable denial-of-service attacks or even arbitrary code execution. A third method would be to take a deep look at the P2P protocol specification and to base your exploit on the protocol instead of the concrete software implementation.

3 Implementation

You are to implement a simple program that demonstrates the vulnerability.

Furthermore, you should try to provide a patch that addresses the security problem (if possible).

4 Submission

You must submit the security advisory in ASCII to your subversion repository to the directory `courses/comp3704/s2009/$GROUP/`. Your advisory should include a reference to the original sources.

Also submit the patch (generated with `diff`) and the exploit code. Do not include generated files.