Assignment 4, 2019
Texas A&M University Computer Science

Introduction

Every reverse engineer must be able to first identify the problem and its scope when reversing a binary, because time and resources are usually limited. A static analysis of the problem scope generates a hypothesis, which defines the tricks and tools required to solve the problem. During the trial and error processes of these tricks, the hypothesis can change, because of possible code obfuscation. Thus, a reverse engineer must be able to think critically and openly as they attempt to reverse engineer a binary.

This assignment focuses on creating a hypothesis of the binaries design, reverse engineering problems, and re-creating the source code of the binary.

Problem 1 (35 points)

As mentioned in the introduction, a hypothesis is used to make an educated guess about the binaries behavior and structure. This can include information about imports, architecture, classes, compiler, algorithms, functions, logic, and data structures. It is highly recommend to refer to the Reverse Engineering for Beginners book and the previous assignments, as this will provide information on functions, structures, c architecture, and obfuscations.

Starting from the main* of the program, identify the sequence of functions and their behavior. Create a flow chart linking all of these together and any sub processes that should be expanded.

Goal of the problem:

1. Create a hypothesis for each problem to be solved.
2. Highlight major information contained in each problem and/or the entire program.
3. Create a flow chart for the flow of the program starting with the main (it is recommended to also create one for each individual function) and describe each function called.

Example detailed information:

Call_000xxxx = int strcmp(char a,char* b)*
Strcmp takes two inputs and returns 0 if the strings are equal, or a non-zero value dependent on the differences of the ascii values of the first non-matching characters in the strings.

<u>*Do not describe the environment setup or any functions that come before the main*</u>

Problem 2 (40 points)

Solve each problem in the bomb lab by using the information in problem 1 as a guide. Record the answer and process for each problem solved. Problem 1 must be referenced to justify the answer (It is likely you may have to change some predictions in problem 1 as you complete each step).

Problem 3 (35 points)

Create source code that resembles the solutions to problem 1 and 2 that resembles the original binary file when compiled. All answers in part two must work on the new binary file that you created.

Goal of the problem:

1. Create source code that mirrors the bomb lab.
2. Test solutions on original binary with new.
3. Determine how similar these two binaries are by creating your own metric or by manipulating an existing metric from online (must sight). Explain the details behind this metric, why it works, and how it could be improved.
4. Discuss the similarities and dissimilarities between both binaries and why these may occur.

Report (30 points)

This must be a **minimum of 5 pages** in **11 point TIMES NEW ROMANS** font double column **or points will be lost**. Flow charts and diagrams are accepted, however, there must be at least be 5 pages of total text (Eye ball it and if it seems good, then it should be acceptable). Good writing can make up for a report less than 5 pages (double column). References MUST be included in the summary when using outside resources. Do so in **MLA format** and include a [ref #] in the report where the information is being used.

What is expected in the report:

1) Abstract - What is your purpose, method, and goal?
2) An introduction answering the following questions:
    a) What is the problem you are seeking to solve in this assignment?
    b) Why is it relevant?
    c) Introduction to how you plan to determine if you have created the same source code.
3) Problem 1 solution:
    a) Break down how you analyzed the code and diagrams explaining the overall structure of the binary.
4) Problem 2 solutions:
    a) Summary of work in problem 2. This includes screen shots, diagrams, or any other method to prove and show your work. For example, start with a summary of the problem and then provide an in-depth discussion on your hypothesis, answer, and challenges.
    b) Explain what type of bug in the program that allowed you to crack the code. Determine if current programs avoid these problems, if so, what do they use? (if not explain how they might avoid this)
5) Problem 3 solutions:
    a) Related work in binary analysis that is helpful
    b) New metrics or algorithms you created

c) Statistics of each binary
d) How you compared them
e) Results

Conclusion
References

NOTE: A summary of a problem is expected to explain in writing the problem, solution, and any challenges you occurred. Combine this with diagrams, screen shots, and anything else that would back up your claims in writing. The flow of the report should be smooth and understandable, similar to a research report.

TOTAL POSSIBLE POINTS

Problem 1	Problem 2	Problem 3	Report
35	40	35	40

WHAT TO HAND IN

The following is expected to be turned in on ecampus:

1) PDF Report to turn-it-in on ecampus (note that the individual solutions in each problem can be the same information that is included in the report.)
2) Separate PDF files for problems 1 and 2 (Different than report) in a zip folder that contains two sub folders called P1 and P2 **or points will be lost.**
3) Source code and the new binary for problem 3. Include information on the metrics used when determining the similarities between the old and new binaries.
4) References in **MLA** format (**or points will be lost**) for any tools, or knowledge used in the assignment. **DO NOT USE ANYTHING WITHOUT CITING THE WORK or you will receive a zero. Include this with the report.**