Homework 4: Project

In this homework you will create and finish a project.
This homework is intended to be done in a group.

Suggestions for Deciding on a Project

To decide on a project, we suggest that you first think about what kinds of security problems you want to help prevent, then think about what kind of tool could best help do that.

Alternative Kinds of Projects

We will consider two types of projects:

- designing and building a software tool, or
- evaluating a tool, and its competition, that is intended to solve a specific problem.

The first kind of project is most appropriate for those interested in technical aspects of secure software development. The second is most appropriate for those interested in applying or managing security software instead of developing it.

Security Problems

Security problems could be any of the problems we discussed in class this semester. The following list of possible problems draws on those problems and also problems from the book *24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them* [1]. Note that you are welcome to do a project on a different problem, the following list only has some possible suggestions.

- Enforcing safety policies, expressed as finite automata.
- Preventing code injection attacks.
- Preventing web-server vulnerabilities such as XSS, XSRF, and Response Splitting.
- Preventing buffer overflows (in C or C++).
- Preventing format string problems (in C or C++).
- Preventing release of confidential information (such as PII).
- Preventing insecure use of security libraries (such as cryptographic libraries).
- Preventing networking problems, such as protecting network traffic, properly using PKI, and properly using DNS.
- Preventing theft of funds from Smart Contracts in blockchains like Ethereum.

Extensible Tool Frameworks

If you are building a tool, then after you decide on the kind of security bugs you want to address, you might consider some of the following extensible tool frameworks as a basis for your project. You would need to select a tool or tool framework to build on that would help solve your particular problem. (I have tried to add links for each system, which should be clickable in the PDF.) Note that you are welcome to do a project using a different tool framework; the following list only has some possible suggestions.

If you are evaluating tools, these links may give you some leads on finding tools to evaluate.

- Java PathFinder is an extensible symbolic execution framework.
- The CREST tool, which is an "automatic test generation tool for C".
- The jCUTE tool, which is a "Java Concolic Testing Engine."
- The KLEE symbolic virtual machine, which is built on top of the LLVM compiler infrastructure.
- The Clang Static Analyzer, which "is a source code analysis tool that finds bugs in C, C++, and Objective-C programs."
- Clang Tidy "is an extensible framework for diagnosing and fixing typical programming errors, like style violations, interface misuse, or bugs that can be deduced via static analysis. clang-tidy is modular and provides a convenient interface for writing new checks."
- BitBlaze, which "aims to design and develop a powerful binary analysis platform and employ the platform in order to (1) analyze and develop novel COTS protection and diagnostic mechanisms and (2) analyze, understand, and develop defenses against malicious code."

  The BitBlaze website also describes several tool efforts, which may be inspirational for your project.
- FuzzBALL "is a symbolic execution tool for x86 (and a little ARM) binary code, based on the BitBlaze Vine library."
- The Antlr parser generator tool, which can help automate lexical analysis and parsing.

Ask the instructor for suggestions of other possible tools if you need other kinds of tools to help with your project.

The Assignment

1. (6 points) Send a list of the names of your group members for this homework by email to the instructor, with a cc to all group members. The subject of this email must be "CIS 6614 Group membership for homework 4" and it must be sent by email (not on webcourses or by some other means).
2. [Plan] [Architect] Write a proposal for your project and turn that in on Webcourses.

   The proposal should be no more than 2 pages, although one page is fine. It should contain the following parts.

   (a) (5 points) A title that says something about the problem and approach, and perhaps the significance of the project. An example would be something like "Preventing Integer Overflows that cause Excessive Allocations using Symbolic Execution".

   If your project is evaluating tools, then you will still need to identify the problem that these tools are solving, and put that in the title. For example it might be something like "Evaluation of Tools that Prevent Integer Overflow Attacks."

   (b) (20 points) A brief description of the software security problem that your project will attempt to solve (or evaluate tools for) and why it is important. This should be a paragraph (or two) that is clearly labeled as describing the problem.

   (c) (20 points) An informal summary of the attack model describing the assumed capabilities of an attacker, which your project (or the tools you are evaluating) will defend against. This should be a paragraph, and may contain a list of capabilities.

(d) (5 points) A single sentence describing what kind of project you will be creating, which should be either a software tool of an evaluation of tools that solve that the problem identified earlier.

(e) (30 points) If you are building a tool, then you should summarize the approach you plan to take to solve the problem. This should be a paragraph (or two) describing how the project will solve the problem. If you are building a software tool by extending or adapting some existing tools, then you should name those tools and briefly explain how they will be used in your project. If there are key technical insights needed, then explain what those insights are (or how you plan to discover them). You can say that you plan to adapt or use ideas from another source, such as a research paper, but in that case you should give an appropriate citation. Note that the approach does not have to be completely new, although novelty is a bonus.

If you are evaluating tools, this should be a paragraph (or two) describing the tools that your project will be evaluating, and why those tools are useful in practice.

(f) (15 points) A summary of any related work that you are using or extending (or improving upon), with appropriate citations. Related work should be work that is solving the same (or closely related) problem, or work that is evaluating tools for the same or a similar problem. For each such related work, state how you expect your solution will either use or improve on that work (as appropriate).

(g) (15 points) If you are building a tool, then given an overview of how you plan to validate or measure the extent to which your project solves the problem. There should be enough testing (or theoretical analysis) to show whether the project is solving that problem. Ideally there should also be demonstrations that the project is useful and reasonably efficient. If there are false negatives and false positives, then you should measure the rate of each.

If you are evaluating tools, then give details on the kind of experimental evaluation you plan to run, and what you expect that will help you (and the reader) learn about tools that solve this problem. You must indicate what you plan to measure about these tools and how you will perform these measurements in an objective way.

3. [Build] [Judge] Implement your proposal and write a report on it, turning both in on Webcourses. What you turn in should have the following parts:

(a) (100 points) If you are building a tool, then turn in your project sources. This should be the source code for a software tool that you built along with a way to build it (and install it), and any instructions (or documentation) needed to build it.

If you are evaluating tools, then this should be a detailed description of the experimental methods you followed. How you downloaded and built the tools involved and the source for the dataset you used for measurements.

In both cases, if it is more convenient, you can point at a github repository that is publically accessible for the sources.

(b) (100 points) Your project report. This report should ideally be around 10 pages in length, but it is fine to be of a different length. The report should contain: (1) A title, the date, and author information. (2) An introduction that summarizes the report, (3) A section describing the problem, with a title that uses the word "problem". (4) An optional section[1] that gives any background information needed for a computer scientist to understand the problem and approach. (5) A section on related work, defined as work that addresses the same problem, that explains for each such work, the extent to which that work's solution solves the problem and how your solution uses that technique or improves on it, (6) A section on the approach taken, that describes the tool or the evaluation in detail, with enough detail that a computer scientist could reproduce the solution approach or the experiment and understand it.[2] (7) A section on evaluation of how well the approach solves the problem or how well the evaluation covers the set of tools that solve the problem.[3] (8) An optional discussion section, that discusses any interesting issues that appeared during the project,

[1]Such an optional section can be omitted if it is not needed.

[2]This section should be self-contained, that is it should not rely on the reader having understood other papers to understand the approach. However, if you often refer to details in the source code, consider presenting the source code in a way that makes it easy for the reader to find the place the text is referring to.

[3]For tools, this could involve testing and/or experiments (and/or some theoretical analysis); for an evaluation, the could involve a discussion of threats to the validity of your evaluation. Ideally the evaluation presented should be enough to convince a computer scientist that the approach works as claimed in the conclusion.

which do not fit anywhere else. (9) A conclusion, that summarizes the problem. For a tool project, the conclusion should also describe the approach taken and the results of the evaluation (stating how well your project solves the problem). For an evaluation project, the conclusion should also describe the evaluation techniques used and any conclusions that can be definitively drawn from them. Optionally, there can be a paragraph on future work (work that could also address the same problem).

The report should be scientific in that it presents evidence for the reader to judge the project (for a tool: the approach and evaluation, for a tool evaluation: the experiments and conclusions). The report can be positive (if that is warranted) about the approach or evaluation, but should avoid seeming like a sales pitch that has no technical content. That is, include any negative results and use those negative results in drawing well-supported and rational conclusions.

References

[1] Michael Howard, David LeBlanc, and John Viega. *24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them.* McGraw-Hill, Inc., USA, 1 edition, 2009.