IS561: Project Proposal

(Fall 2017)

Timeline

1. Form your team.
2. Let us know your team members and project topic
   (Edit the Google Spreadsheet shared in KLMS) **due. 09/19/2017**.
3. Iterate with course instructors through emails regarding your project topic.
4. Finalize your project proposal by **10/03/2017**.

.

Submission Submit your written project proposals to KLMS. The proposal *must* be in a PDF format with the following name: "IS561-YourTeamName.pdf".

Guidelines The proposal should be a PDF document that includes: (1) motivation of your problem; (2) why the problem that you are trying to solve is important; (3) your proposed approach; and (4) expected evaluation.

In general you should use your expertise to choose an interesting project topic. Most students in this course are graduate-level students, and you should be able to apply your own expertise to find an interesting software security problems in your research domain. For example, if you are an expert in compiler, then you may want to build a compiler extension that dynamically monitors safety property of a program. If you are a system expert, you may try to build a fast and distributed malware detection system. You should also iterate *frequently* with course instructors before finalizing your topic.

List of Potential Projects

1. Identifying functions from binary code.
   - Given arbitrary binary code, can you return a list of functions in the binary? Notice, a simple pattern matching would involve both false positives and negatives.
2. Build your own fuzzer.
   - Build your own fuzzer to find a vulnerability. We encourage you to set a specific type of vulnerability that you want to find.
3. Improving CFG layout algorithm.
   - Many binary analysis tools such as IDA Pro employ their own CFG layout algorithm. Can you improve the readability of a CFG by employing a new layout algorithm? For example, you want to minimize the number of edge crossings. Finding an optimal solution to this problem is known to be NP-hard, so your goal is to devise some heuristics.
4. Checking the similarity between two binary code.

- Given two binaries (one is potentially an obfuscated binary of the original one), can you determine the similarity between them?