Preliminary Project Proposal

Firmware Module for Infer Static Analyzer

Yousef Elsendiony (YE2914) and Tony Butcher (ab4359)

Experiments and Direction of Project:

In this experiment we will develop a module for Infer targeted at finding firmware related failures in C language. Some of these common failures include stack overflows, null dereferencing, memory leaks, unwanted compiler optimizations, and cast exceptions. The midterm research paper included more details regarding why these are common failures in firmware. Our goal will be to find more bugs which traditional static analyzers are unable to discover.

Motivation for Project (Yousef):

Following up on the research paper, I think this project is doable and has value to the software and specifically firmware developer community. Debugging Production code has been proven to be a difficult problem to solve due to limited resources and data. It also is a hot topic in many conferences and a costly problem to solve. Lastly, I am currently working as a firmware developer, and I would love to see a static analysis tool which can catch more of the bugs that I encounter on my day to day on the job. If the tool proves effective, I would love to automated and/or integrated it into some of the firmware projects I work on.

Motivation for Project (Tony):

Following on with a technical project related to my privacy research paper is impractical given the constraints and the difficulty in identifying open source projects specifically targeting new developments in privacy. Working on a static analysis project aligns well with my computer security track and my interests in achieving more secure code earlier in the design process.

Relevance:

Catching firmware problems earlier in the development process makes fixing them easier and less costly and problematic. Static Analysis techniques are being revised and improved to help catch bugs earlier. Unfortunately, there is a lack of concentration on Firmware related errors which escape traditional static analyzers. Providing a tool which can cover some of these escape areas would be helpful and relevant to many firmware developers.

This is also an opportunity to contribute to a significantly recognized open source project if successful, which has greater potential to produce meaningful results than attempting to build a new tool over the course of a few weeks.

Evaluation:

The criterion would be to introduce sample bugs similar to infer and demonstrate how the modules we build can detect these sample firmware failures. Another Criterion for rating the effectiveness of our static analyzer would be to run against multiple firmware projects and examine the catch rate of errors. Lastly, we can compare the results from the above Infer runs against other static analyzers for the C language. A better or more effective analyzer should find a higher number of real bugs found and a lesser number of false bugs or noise.

Deliverables:

We plan on forking infer and trying to add the firmware modules to our forked repository. If we have success, initiating a pull request with the Infer folks would be a major win. Alternatively, if our bug findings do not generate notable results and our pull request is not accepted, we will provide a detailed analysis of what was done, what we found, any challenges we encountered, and why we did not succeed.

Documentation will be found in on the github page and the project report will describe more about the progression of the project. The forked repository will be provided and commit messages associated code pushes can be used to track progress on development.

Possible Artifacts could include the infer binaries with our models added for people to download and use. Initially I was thinking, it's probably best and most useful if we make 4-5 binaries for each module or feature supported. After some thought, I would like to see if we can leverage some continuous Integration (Travis CI is a great tool which works with github), to generate the binaries for each commit; however, it might be very time and resource consuming so we would need to test this.