Manual and Automatic Program Analysis

Professor Julian Cohen
HockeyInJune@isis.poly.edu

Prerequisites

- Strong Code Comprehension Skills
- Strong Reverse Engineering Skills
- Strong Exploitation Skills
- Familiarity with Computer Science Concepts

Synopsis

Program Analysis describes the techniques of understanding properties of a computer program. In this class, we will cover various manual and automatic processes that allow us to gather all kinds of information about computer programs.

This class will build up to a final project where every student will have a custom working automatic input generator (similar to Microsoft SAGE or LLVM KLEE) that works without source code. This class will focus heavily on reading papers and code in order for students to understand the cutting-edge of program analysis.

Throughout the class each student will be responsible for comparing the techniques and methods we discuss and learn about to tools that exist in the community.

Optional **Textbook**: *Principles of Program Analysis* http://www.amazon.com/gp/product/3540654100/

Core Themes

- The Halting Problem and Rice's Theorem
- Code Analysis
- Automatic Testing
- Program Introspection
- Code Translation and Representation
- Code Coverage
- Data Flow Tracking
- Abstract Interpretation
- Symbolic Execution
- Constraint Solving
- Formal Verification

Resources

- https://github.com/isislab/Project-Ideas/wiki/Program-Analysis

Grading

- Project: 25%
- Midterm: 25%
- Final: 25%
- Homework: 25%

Office Hours

By appointment only.

Syllabus – Spring 2014

Week 1 – January 30: **Introduction** to Program Analysis.

- http://www.cs.umd.edu/class/fall2011/cmsc631/lectures/sym.pdf
- http://akira.ruc.dk/~madsr/webpub/absint.pdf
- http://www.cs.unm.edu/~moore/tr/02-12/zovi.pdf
- http://users.ece.cmu.edu/~ejschwar/papers/oakland10.pdf

Week 2 – February 6: Lecture and lab on **Microsoft SAGE** and **LLVM KLEE**.
Homework: Students will outline their final project.

- http://llvm.org/pubs/2008-12-OSDI-KLEE.pdf
- http://www.doc.ic.ac.uk/~cristic/talks/klee-stanford-2009.ppsx
- http://research.microsoft.com/en-us/um/people/pg/public_psfiles/sage-in-one-slide.pdf

Week 3 – February 13: Lecture and lab on **Reverse Engineering** and **Exploitation**.
Homework: Students will write an exploit for a DEFCON CTF binary.

Week 4 – February 20: Lecture on **Intermediate Representations** and **Abstract Interpretation**.
Homework: Students will design their own IR.

- Guest lecture by **Andrew Ruef**.

Week 5 & 6 – February 27 & March 6: Lecture and lab on **Crash Test Harnesses**.
Homework: Students will build their own crash analyzer.

- http://blogs.technet.com/b/srd/archive/2009/04/08/the-history-of-the-exploitable-crash-analyzer.aspx
- http://technet.microsoft.com/en-us/library/bb457063.aspx

Week 7 – March 13: Lecture on **Constraint Solving**.

- Gust lecture by **John Villamil**.

Week 8 – March 27: Lecture on **Symbolic Execution**.
Homework: Students will build their own constraint collector.

- Guest lecture by **Andrew Ruef**.

Week 9 – April 3: Lecture on **Code Coverage**.
Homework: Students will optimize constraints and solve for input values that improve code coverage.

- Guest lecture by **Chris Rohlf**.

Week 10 – April 10: Lecture on **Static Checking**.

- Guest Lecture by **Julien Vanegue**.

Week 11 & 12 – April 17 & April 24: Lecture and lab on **Automatic Exploit Generation**.
Homework: Students will automatically generate an exploit for the binary they exploited in Week 3.

- http://security.ece.cmu.edu/aeg/aeg-current.pdf
- http://users.ece.cmu.edu/~ejschwar/papers/usenix11.pdf

Week 13 – May 1: **Overflow**.

Week 14 – May 8: Final Project **Presentations**.

Project: Develop a custom working automatic input generator that works without source code.