The Center for Cyber Defenders

Expanding computer security knowledge

FrogEye: Joint analysis of source code and binaries using Machine Learning

Dalton Cole, Missouri University of Science & Technology
Scott Heidbrink, Brigham Young University

Project Mentor: Danny Dunlavy, Org. 1461

Problem Statement

Source and binary code differ in information that security analysts are able to extract. We seek to understand how features of these two representations relate in order to aid analysts in discovering potential security threats.

Objectives

- Provide greater contextual information to software security analysts
- Allow analysis techniques to be cross compatible between source and binary code
- Apply Machine Learning techniques to alleviate security analysts' burden

Motivating Example

- Finding backdoors, unauthorized access to system, using binaries alone is often a challenging problem given the varying, or unavailability, of architectures. We conjecture that jointly analyzing source code and binaries can provide new perspective to security analysts for this problem.

Source Code Tool Comparisons

Features	Joern	Clang	CppCheck	FlawFinder	Vera++	cqmetrics	Flint++	Oclint	Frama-C
Analysis Ability									
Symbol Scope	✓	✓	✓	✗	✗	✗	○	○	○
Operand Metrics	○	○	○	✗	✗	✗	✗	○	○
Program Dependence	✓	✓	✗	✗	✗	✗	✗	○	✓
External Call Metrics	○	✓	○	✗	✗	○	✗	○	○
Requires Compiling Source	N	Y	N	N	N	N	N	Y	Y
Standardized output	✓	○	✗	✗	✓	○	✓	✓	○
Program									
Documentation	✓	✓	✓	✓	○	○	○	✓	○
Examples	✓	✓	○	✓	✓	○	✗	○	○
Extendable	✓	○	✓	○	✓	✗	✗	✓	✓
Maintained	✓	✓	✓	✗	○	✓	✓	✓	✓
Framework	Y	Y	N	N	N	N	N	N	Y
Cost to Vectorize (Hours)	-	-	1	1	1	0	1	1	-

Binary Tool Comparisons

Features	angr	BAP	Oxide	Pin	Manticore
Analysis Ability					
Static/Dynamic	○	✓	✓	✗	✗
Symbolic Execution	✓	✓	✗	✗	✓
Recognize functions	✓	✓	○	✗	✓
Identify strings and primitive data types	✓	✓	○	✗	✓
External Call Metrics	○	○	✓	○	○
Opcode Frequency	○	○	○	○	○
Standardized output	✓	✓	○	○	✗
Program					
Documentation	✓	✓	○	✓	✓
Examples	✓	✓	○	✓	○
Extendibility	✓	✓	✓	✓	✓
Maintained	✓	✓	✗	✓	✓

Approach

- Literature search for binary/source analysis tools
- Cross comparison between tools
- Restrict to static analysis as hardware may be unavailable
- Vectorizing tool output for use in Machine Learning algorithms
- Collect/generate benchmark dataset of C/C++ code
- Conduct experiments with Machine Learning algorithms

Challenges

- How useful are current analysis tools for use in machine learning?
- Do static analysis features correlate to security vulnerabilities?
- How correlated are features between binary and source?

Impact and Benefits

- Allows research in binary or source code analysis to be applied to the other field
- Provide greater understanding of relationship between source code and binaries for use in security assessments
- Aid security analysts in discovering security vulnerabilities
- Further the field of automatic vulnerability assessment

Results

- Tools chosen for deeper analysis
  - **Joern** [mlsec.org/joern]
    - Easy-to-use, comprehensive program graphs, easily extendable, doesn't require compilation
  - **Cametrics** [github.com/dspinellis/cqmetrics]
    - Quick vectorized output of large feature space
  - **Oclint** [oclint.org]
    - Easily vectorized output, common lint analysis
  - **Flawfinder** [dwheeler.com/flawfinder]
    - Quick CWE analysis with security level context, density information of found flaws
  - **angr** [angr.io]
    - Popular analysis tool, 3rd place cyber grand challenge, extendable framework, aimed at researchers in binary analysis
  - **BAP** [github.com/BinaryAnalysisPlatform/bap]
    - Mature binary analysis tool, numerous existing analysis tools, variety of supported plugin languages

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2017-7351D.