Software Reuse

What is software reuse

· Reuse is a general common-sense principle, pervasively applied in all human activities
· Software reuse is an act of reuse of previously built software artifact/asset to fulfill need in new development task or sub-task

Why Reuse

· Avoiding wasting time and money redoing same task over and over
· Quality standard of previously completed task carries over to new occurrences of same need
What to reuse

Software assets
- Domain/business analysis/model
- Application requirements
- Application architectures and generic architecture patterns
- Application design/model and generic design patterns
- Source code
  - Libraries, modules, packages
  - Classes, templates, idioms
  - Objects, functions, subroutines, data
- Executable code: executable components, bytecode, binary code
- Documentation and meta-data
- Test suites and test generators

The Problem: Binary Executable Transforms

Credit: This slide is from Dan Roelker from a DARPA colloquium in 2011

Our Goal

Mission
Reusing binary code

- Given: x86 binary, functionality specification
- Identify: modular subroutines performing the functionality
- Extract: reusable C functions for the identified subroutine

About Binary Code
- Rich in every platform, with huge quantity ✓
- End users have everything ✓
- Difficulty to analyze the machine code ✗
Figure 1. Our assembly function reuse approach. The core of our approach is the function extraction step implemented by BCR. The three dark gray modules in function extraction have been specifically designed for this work. The execution monitor, disassembler, and semantics inference module (light gray) are reused from previous systems.

Challenges

Component Identification
- How to specify functionalities to reuse?
- How to identify relevant binary artifacts with accuracy and completeness?
- How to identify re-usable subroutines from relevant artifacts?

Component Extraction
- Handling indirect jumps and function pointers
- Inferring variables and their types
- Inferring calling interface of extracted subroutines

Existing Techniques

Decompilation
- HexRay

Function extraction
- BCR
- Inspector Gadget
- Virtuoso

Binary reverse engineering
- Data structures: REWARDS, TIE, Howard
- Control flow graphs: CFG generation

Outline

Background
Binary Code Reuse
Existing Techniques
Summary
Appendix: Graduate Study

BCR

Credit: This diagram is from the BCR paper in [NDSS 2010]
We can reuse the legacy binary code to build new software.
Imagine a circle that contains all of human knowledge

By the time you finish elementary school, you know a little

credit: http://matt.might.net/articles/phd-school-in-pictures/
By the time you finish high school, you know a bit more

With a bachelor’s degree, you gain a specialty

A master’s degree deepens that specialty

Reading research papers takes you to the edge of human knowledge
Once you’re at the boundary, you focus

You push at the boundary for a few years

Until one day, the boundary gives way

And, that dent you’ve made is called a Ph.D.
Of course, the world looks different to you now

So, don’t forget the bigger picture