Universidade Federal de Pernambuco – Graduação
em Ciências da Computação

Undergraduate Project Proposal

# Developing an Intermediate Representation for the Analysis of Binary Code

Student: Julio Auto de Medeiros
Supervisor: Prof. André Santos

Recife, April 27, 2007

# I. Context

For many years now, a considerable amount of effort has been put into making the automated analysis of software a powerful and practical tool in the workbench of professionals and researchers that deal with software. In this time, many formal approaches to software verification have been created and further research still goes on in an attempt to make these techniques capable to answer the most relevant questions in program analysis, some which are: Does this software have bugs? How can these bugs be triggered? How critical are they?

Furthermore, a branch of this research field (and a quite recent one) is of special importance: the analysis of binary code. Being capable of analyzing binary code enhances the possibility to answer all those intriguing questions without the need of the source code. All that is needed is the software in its most natural form: the bits and bytes that are ready to be fed into a program loader. It's in this brave and honored quest that this work is inserted.

The final implementation of this work's results shall be done upon the framework provided by the ERESI project. ERESI (ELF Reverse Engineering Software Interface) is an open-source project that has been developed for the past 6 years and that aims to help its users to gather relevant information from compiled programs using various means, such as disassembling or debugging/live analysis. One of the ultimate goals of ERESI is to be able to perform automated static analysis that can efficiently spot security breaches in binary software.

From its current state, an immediate step that the ERESI project has to take towards the development of its static analysis features is the conception and implementation of a low-level intermediate representation (LIR). This will result from a joint effort between this work's author and a third party, where the author's role (and, therefore, the main objective of this work) is the formalization of the operational semantics of a subset of the Intel IA-32 architecture instruction set and the writing of code that translates this machine code to the LIR with semantics information.

The ERESI framework runs on a variety of UNIX systems, working with the ELF executable format and having most of its features implemented mainly for Intel x86 and SPARC architectures. As such, any code resulting from this work shall target Linux over Intel x86 platform.

## II. Objectives

As previously stated, the main objective of this work is to make the LIR of the project capable of expressing the semantics of programs compiled for this specific architecture. Therefore, an implementation of code that can analyze the instructions and translate them to the LIR with semantics information naturally follows. It may also be desirable to describe the operational semantics of this kind of machine code using a formal notation such as SOS (Structured Operational Semantics). Therefore, some effort may also be put on this task.

The author will also be closely following the design of the IR itself, so the final report is expected to contain a study of the techniques and technologies involved with the conception of this IR, e.g. existing IRs from which this one derives and analysis structures that can be represented with our IR, as well as the reasons behind such design choices, i.e. how it can make the analysis and, ultimately, the verification process more powerful.

It's important to state that this work is, above all, very research-oriented. As such, reading and investigating play a part even bigger than writing and debugging proof-of-concept code. In this sense, it can be said that a big, although less 'tangible', objective of this work is to gain solid knowledge and compile information about state-of-art topics in program analysis.

As a brief overview of the technical aspects of the IR, the early conception points out that the design will make use of the SSA (Static Single Assignment) and SSI (Static Single Information) forms, introducing the concept of data-flow analysis into the project, which until now is only capable of performing very basic control-flow analyses. It's also targeted the construction of PSTs (Program Structure Trees) of SESE (Single Entry-Single Exit) regions and VSDGs (Value State Dependence Graphs), as means to enhance the potential of further analysis. A small bibliography at the end of this document can reference deeper information on this.

## III. Project Schedule

| | May | | | | Jun | | | | Jul | | | | Aug | | | | Sep |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | W1 | W2 | W3 | W4 | W1 | W2 | W3 | W4 | W1 | W2 | W3 | W4 | W1 | W2 | W3 | W4 | W1 |
| Gathering bibliographic references | | | | | | | | | | | | | | | | | |
| Researching references | | | | | | | | | | | | | | | | | |
| Studying Intel IA-32 Operational Semantics | | | | | | | | | | | | | | | | | |
| Writing Implementation | | | | | | | | | | | | | | | | | |
| Writing Final Report | | | | | | | | | | | | | | | | | |
| Work Presentation | | | | | | | | | | | | | | | | | |

## IV. Bibliography


- Vanegue J., Garnier T., Auto J., Roy S. & Lesniak R., “Next-Generation Debuggers for Reverse Engineering”, 2007
- Plotkin G. D., “A Structural Approach to Operational Semantics”, 1981
- Cytron R., Ferrante J., Rosen B., Wegman M. & Zadeck F., “Efficiently Computing Static Single Assignment Form and The Control Dependence Graph”, 1991
- Ananian C., “The Static Single Information Form”, 1999
- Johnson R., Pearson D. & Pingali K., “The Program Structure Tree: Computing Control Regions in Linear Time”, 1994
- Johnson N. & Mycroft A., “Combined Code Motion and Register Allocation using the Value State Dependence Graph”, 2003

## V. Signatures

_____
Julio Auto de Medeiros
Student

_____
Prof. André Santos
Supervisor