Calculating Quantitative Integrity and Secrecy for Imperative Programs

Tom Chothia, School of Computer Science, University of Birmingham, Birmingham, UK
Chris Novakovic, School of Computer Science, University of Birmingham, Birmingham, UK
Rajiv Ranjan Singh, Department of Computer Science, Shyam Lal College (Eve) University of Delhi, Delhi, India

ABSTRACT

This paper presents a framework for calculating measures of data integrity for programs in a small imperative language. The authors develop a Markov chain semantics for their language which calculates Clarkson and Schneider’s definitions of data contamination, data suppression, program suppression and program transmission. The authors then propose their own definition of program integrity for probabilistic specifications. These definitions are based on conditional mutual information and entropy; they present a result relating them to mutual information, which can be calculated by a number of existing tools. The authors extend a quantitative information flow tool (CH-IMP) to calculate these measures of integrity and demonstrate this tool with examples including error correcting codes, the Dining Cryptographers protocol and the attempts by a number of banks to influence the Libor rate.

Keywords: CH-IMP, Data Integrity, Dining Cryptographers Protocol, Libor Rate, Markov Chain Semantics, Small Imperative Language

1. INTRODUCTION

Data integrity is an important issue for programs, but this does not mean that programs must guarantee that all good data is perfectly preserved, or that no bad data can affect a program’s output in any way. It is often the case that a good service can be provided with a small but acceptable loss in the the integrity of data. Therefore, there is a need for a framework in which the level of integrity a system provides can be accurately measured.

Qualitative integrity has been well-studied (see e.g. the work of Birgisson et al. (Birgisson, Russo, & Sabelfeld, 2010), who provide a unified framework for qualitative integrity policies), but quantitative integrity measures have received far less attention. An exception to this is the

DOI: 10.4018/IJSSE.2015040102
work of Clarkson and Schneider, which defines the data integrity measures of “contamination” and “suppression” (Clarkson & Schneider, 2010, 2014). In this paper, we extend these measures to a small imperative language, give this language a Markov chain semantics, and show how these measures of integrity can be calculated for programs with multiple trusted and untrusted inputs and outputs. We extend an existing quantitative information flow (QIF) tool, CH-IMP (Chothia, Kawamoto, Novakovic, & Parker, 2013), to use this semantics to automatically calculate these measures.

Clarkson and Schneider’s definitions are based on conditional mutual information and entropy. Existing QIF tools can calculate the mutual information between a system’s secret values and observable outputs; we investigate whether mutual information can be used to calculate conditional mutual information. We find that for secrecy, in which the public inputs of a system must be unrelated to the secret inputs, conditional mutual information can be calculated using mutual information (and therefore that existing QIF tools can calculate conditional mutual information-based definitions of secrecy by simply appending a program’s public inputs to its public outputs). However, for integrity, the trusted data is considered to be public, and so the untrusted data sent to a system may depend on it, so conditional mutual information must be calculated explicitly.

The semantics we present makes it possible to calculate the integrity between multiple trusted and untrusted inputs and outputs. In doing so, we pay particular attention to what data is recorded by the semantics, and thus the probability distributions used to calculate the integrity measures. These decisions define the attacker model: CH-IMP models an attacker that wishes to learn anything they can about the secret values (including, e.g., the order in which they occur) and can only observe certain outputs from the system. For integrity, we consider a model in which we are only concerned that data is correctly preserved as it flows between variables. Thus, because of the different attacker models, the calculation of quantitative integrity is not the dual of the calculation of quantitative secrecy.

We combine the integrity model with our previous work on secrecy (Chothia et al., 2013) to make a single framework in which both integrity and secrecy can be measured, and therefore their trade off analysed. We have implemented a tool in OCaml to carry out this analysis and illustrate it with a number of examples, including examples based on error correcting codes, the Dining Cryptographers protocol and the attempts by a number of banks to influence the Libor rate (BBC, 2012). Our tool takes in a program written in our language CH-IMP-IQ, builds the state space, and from this calculates the probability distribution over trusted, untrusted, secrecy and public variables, from which it calculates the measures of integrity and secrecy described below. We have used our tool to analyse programs with a state space as large as $2^{22}$, which takes a few hours on a standard desktop machine.

The contamination and suppression measures of integrity tell us how much information about the trusted input of a system appears in the trusted outputs, however we may actually want the trusted outputs to contain less information than the inputs, for instance, we may have a program that calculates the average of a number of inputs. In this case, we would want high integrity programs to return a result as close to the average as possible, whereas the output of low integrity programs would deviate from it. To this end, Clarkson and Schneider define program suppression and program transmission. These measures take a deterministic program as a specification and measure how well an implementation program matches it. Clarkson and Schneider’s definitions do not work for probabilistic specification, so we propose a definition of program integrity which measures how well the outputs of a probabilistic program match the outputs of a probabilistic specification.

This paper is an extended version of a previous paper published at the Workshop on Quantitative Aspects in Security Assurance (QASA) (Chothia, Novakovic & Singh, 2014); the secrecy
Design and Transformation of a Domain-Specific Language for Reconfigurable Conveyor Systems
www.igi-global.com/chapter/design-transformation-domain-specific-language/77722?camid=4v1a

An Overview of RIAs Development Tools
www.igi-global.com/chapter/an-overview-of-rias-development-tools/188204?camid=4v1a