Quantitative Information Flow: From Theory to Practice?

Pasquale Malacaria

School of Electronic Engineering and Computer Science
Queen Mary University of London

Most computational systems share two basic properties: first they process information, second they allow for observations of this processing to be made. For example a program will typically process the inputs and allows its output to be observed on a screen. In a distributed system each unit processes information and will allow some observation to be made by the environment or other units, for example by message passing. In an election voters cast their vote, officials count the votes and the public can observe the election result.

The general aim of Quantitative Information Flow (QIF) [4,5] is to measure the amount of information about a source of information that can be gained by observations on some related system component. An application in the context of programming languages is for example to measure leakage of confidential information in programs.

The simplest example motivating QIF is a password checking program where secret data (stored in the high variable `h`) can be deduced by observing the final value of the low variable `l`:

```
if (h == l) l = 0 else l = 1
```

The above program is insecure and the aim of QIF is to measure its insecurity.

Further examples of this scenario is code handling medical records, code querying databases, arithmetic operations within a cryptographic package, RFID tag identification etc. In these examples we expect sound code to leak as little as possible. There are also positive applications of QIF, where the analysis should return high leakage; examples of this scenario are programs computing best fit representation for data, hash coding, image processing code, strategy game solving programs etc.

In both scenarios would be very useful to have reasoning techniques and tools, possibly automatic tools, able to measure how much information is leaked.

QIF attempts to answer a set of questions related to leakage, for example are we interested in the total possible amount leaked, or only if it leaks below or above a certain threshold? clearly the former question is in general harder than the latter because for the latter we can stop measuring leakage once reached the threshold. Also is the ration leaked or the amount itself the information we want? for example a 3 bits leak is one tenth for a 30 bits secret but is one hundred percent for a 3 bits secret.

The nature of what is leaked is also very important, typically does it leak always the same information? for example there is a very big difference if the code leaks always the same bit when it is run or if leaks a different bit every time is run. Other important questions are: at what rate, i.e. how fast, is information leaked? and what's the maximum leakage (the channel capacity) over all possible choice of probabilities for the inputs?

The fundamental question has to be what is exactly that we want to use as a measure [13].

The basic albeit crude unit of measure in the analysis is the number of states of the source of information that can be distinguished by the observations. Classical non-interference [8] amounts to not to be able to make any distinction by looking at the observations, whereas total leakage corresponds to be able to make all possible distinctions. The tricky bit is, of course, what lies in between these two extreme.

In the past years a number of works have refined this idea and provided a solid theoretical background for QIF [5,10,13]. The theory is based on Information Theoretical terms, the main being Shannon's entropy but also Renyi's entropies and guessability measures have been shown to be relevant. The Information Theoretical approach has been shown to be powerful and able to provide natural, intuitive and precise reasoning principles for program constructs including loops [9,10]. The theory has also been fruitfully applied to the analysis of anonymity protocols, where the same framework has been used to measure the loss of anonymity [2,3].

More recent works have investigated automation of such ideas, and verification techniques like abstract interpretation, bounded model checkers and SAT solvers have been applied in this context [1,7,11,12]. Implementation of the quantitative aspects of the analysis presents however a number of challenges [14], the main being scalability and handling of data structures.

Some of these challenges could probably be solved by appropriate abstraction techniques, some may be better addressed by statistical means [6].

The talk will give an overview of the field, the basic ideas, reasoning principles and problems.

References

1. Backes, M., Köpf, B., Rybalchenko, A.: Automatic Discovery and Quantification of Information Leaks. In: Proc. 30th IEEE Symposium on Security and Privacy S& P '09 (2009)
2. Chatzikokolakis, K., Palamidessi, C., Panangaden, P.: Anonymity protocols as noisy channels. Information and Computation 206(2-4), 378–401 (2008)
3. Chen, H., Malacaria, P.: Quantifying Maximal Loss of Anonymity in Protocols. In: Proceedings ACM Symposium on Information, Computer and Communication Security (2009)
4. Clark, D., Hunt, S., Malacaria, P.: A static analysis for quantifying information flow in a simple imperative language. Journal of Computer Security 15(3) (2007)
5. Clark, D., Hunt, S., Malacaria, P.: Quantitative information flow, relations and polymorphic types. Journal of Logic and Computation, Special Issue on Lambda-calculus, type theory and natural language 18(2), 181–199 (2005)

6. Kpf, B., Rybalchenko, A.: Approximation and Randomization for Quantitative Information-Flow Analysis. In: Proceedings of Computer Security Foundations Symposium 2010 (2010)
7. Heusser, J., Malacaria, P.: Applied Quantitative Information Flow and Statistical Databases. In: Proceedings of Workshop on Formal Aspects in Security and Trust, FAST 2009 (2009)
8. Goguen, J.A., Meseguer, J.: Security policies and security model. In: Proceedings of the 1982 IEEE Computer Society Symposium on Security and Privacy (1982)
9. Malacaria, P.: Assessing security threats of looping constructs. In: Proc. ACM Symposium on Principles of Programming Language (2007)
10. Malacaria, P.: Risk Assessment of Security Threats for Looping Constructs. Journal Of Computer Security 18(2) (2010)
11. McCamant, S., Ernst, M.D.: Quantitative information flow as network flow capacity. In: PLDI 2008, Proceedings of the ACM SIGPLAN 2008, Conference on Programming Language Design and Implementation, Tucson, AZ, USA (2008)
12. Mu, C., Clark, D.: An Abstraction Quantifying Information Flow over Probabilistic Semantics. In: Workshop on Quantitative Aspects of Programming Languages (QAPL), ETAPS (2009)
13. Smith, G.: On the Foundations of Quantitative Information Flow. In: de Alfaro, L. (ed.) FOSSACS 2009. LNCS, vol. 5504, pp. 288–302. Springer, Heidelberg (2009)
14. Yasuoka, H., Terauchi, T.: Quantitative Information Flow - Verification Hardness and Possibilities. In: Proceedings of Computer Security Foundations Symposium (2010)