Trends in Information Security

Topic Description:
Formal Proof and Levels of Trust in Information Security

Dr Tim Storer and Prof. Chris Johnson,
School of Computing, University of Glasgow.
Johnson@dcs.gla.ac.uk
http://www.dcs.gla.ac.uk

Introduction

Systems are intended to perform useful work and/or help users to get their jobs done. Security mechanisms are designed to prevent the system from being used in inappropriate ways. Unfortunately discriminating between these two activities is quite hard as users often make novel, unexpected, but perfectly legitimate uses of a system. Consequently, we need to assess the overall impact on system operation and security of a proposed security mechanism. Adding a 5-step multi-factor authentication mechanism to a system does not mean that the overall system is harder to penetrate: it's quite possible that harder authentication causes legitimate users to implement 'work arounds' to get their work done, but also reduce the effective security of the system. We will never be certain that a system is going to be perfectly secure against all possible threats. We need to be clear about what parts of the system are intended to be secure against what kind of threats. Equally, we need to be explicit about which parts of the system or security methodology we are going to trust to be secure, i.e. not worry about, or hope that someone else is worrying about it. Formal analysis based on mathematical proof can be used to reason about the security of system components and also about the interfaces with these trusted components.

Different Perspectives

One paper that supports these arguments is Thompson's demonstration of the ease with which vulnerabilities could be inserted into machine code without them appearing in the corresponding source code through manipulation of the compiler. Interestingly, Wheeler [2005] showed a method for reducing, but not eliminating this problem. MacKenzie's [2001] social history of the efforts to achieve formal proofs of correctness in software engineering makes a similar observation about the limits of mathematical proofs generated by machines (how do you know the machine doesn't contain bugs that invalidates the proof?). The Tokeneer case study [Barnes et al., 2006] of developing a formal proof for a security system, is an interesting example of this problem. The authors attempt to show that formal proofs are feasible for achieving desirable security properties in real software systems, rather than toy examples. However, many of the proofs are actually incomplete or rely on informal argumentation.

A related strand of research looks at the use of deontic logic in system specification, Brunel [2004] for example. The idea here is that we should specify how systems should behave, by formulating functions as obligations imposed on the system, rather than features. In addition, we can specify consequences or penalties if the system fails to meet it's obligations. The argument is that this results in more robust system architectures that are better able to tolerate component or peer failure (whether that's really true or not is very much up for debate). The

notion of intentional access management is another example of this approach [Cao and Iverson, 2006]. A final paper that is definitely worth a read is Spafford's account of the design and implementation of the Morris Worm, arguably the first released on the Internet, [Spa ord, 1988]. The paper is a nice example of how malware is often quite poorly designed and implemented. It is important to understand how these threats work because they help to identify the theorems that we might need to prove during a formal analysis. In other words, we can use mathematical reasoning to show that a future system is robust against a particular class of threats.

Open Research Questions

The following list provides a subset of research questions in this area:

- **What formalisms provide the best support for information security?**
  Previous sections have mentioned the use of deontic logics that consider relationships between obligation and permission. They can be used to reason about situations in which people violate security obligations/policies. However, a large number of alternate notations have been developed including temporal and epistemic logics, as well as formal specification techniques such as Z and VDM. There are graphical approaches such as Petri Nets and State Charts. Many of these are supported by automated tools to support reasoning, including theorem provers and model checkers. Often these have only been applied to a limited number of security case studies hence it is difficult to make detailed comparisons of the strengths and weaknesses of these alternate techniques;

- **Can formal analysis be cost effective?**
  In most cases, formal analysis can only be applied by teams with considerable training in mathematical reasoning. They also require considerable amounts of time to analyse relatively simple sections of code. Hence, questions remain about whether these costs are justified in terms of the insights that formal analysis provides into the underlying security of complex systems;

- **What are the limits of formal and informal reasoning?**
  Previous sections have argued that in many cases formal analysis of secure systems is often supplemented and structured by a range of less formal arguments. In other words, we cannot afford to create complete formal specifications of complex applications. Hence we have to use natural language descriptions to explain why mathematical techniques were used in certain areas of an application and not in others. We may also have to use these natural language statements to explain why we believe those areas of a system that have not been subjected to formal analysis do not undermine the overall security if a system;

- **How best to integrate formal models into security cases?**
  Safety cases have been developed using languages like Kelly's [1998] Goal Structuring Notation. They provide a map of the arguments that explain why a system is acceptable safe. More recently, this approach has been used to develop security cases. – to explain why a system is acceptable secure. This notation can be usd to integrate formal and informal analysis following the approach described in previous paragraphs. Further

work is required to determine whether this approach is scalable and supports security management in real-world applications.

References

Janet Barnes, Randy Johnson, David Cooper, and Bill Everett. Engineering the tokeneer enclave protection software. In Proceedings of the 1st IEEE International Symposium on Secure Software Engineering, Washington DC, March 2006.

Julien Brunel. Deontic logic for the specification of system availability. In Proceedings of 6th school on MOdelling and VErifying parallel Processes (MOVEP'04), Brussels, Belgium, December 2004.

Xiang Cao and Lee Iverson. Intentional access management: Making access control usable for end-users. In Symposium On Usable Privacy and Security (SOUPS), pages 20{31, Pittsburgh, Pennsylvania, USA, July 2006. CMU Usable Privacy and Security Laboratory.

T. P. Kelly, Arguing Safety – A Systematic Approach to Safety Case Management, DPhil Thesis YCST99-05, Department of Computer Science, University of York, UK, 1998.

Donald MacKenzie. Mechanizing Proof: Computing, Risk and Trust. The MIT Press, Massachusetts Institute of Technology, Cambridge, Massachusetts 02142, 2001.

Eugene H. Spafford. The internet worm program: An analysis. Technical Report CSD-TR-823, Department of Computer Science, Purdue University, West Lafayette, IN 47907-2004, November 1988.

Ken Thompson. Reflections on trusting trust. Communications of the ACM, 27(8):761{763, August 1984.

David A. Wheeler. Countering trusting trust through diverse double-compiling. In 21st Annual Computer Security Applications Conference, pages 33{48, Tucson, Arizona, December 2005. IEEE Computer Society.