Projects

CSG 399

February 24, 2006

You should be starting to think about a possible project for the course. (See below for the approximate timeline you should keep in mind.) Please get in touch with me by email by the break, to decide what you would like to work on. Once you have chosen a project, I will give you 2 or 3 papers on the topic, and your job will be to read, understand, summarize, critique, and finally give a one-hour conference-style presentation about the papers in class. I will also ask you for a 5-10 pages writeup to go with your presentation, so that I have something concrete to look at after your talk.

The presentation should explain the problem that the papers investigate, and emphasize the core message that has appeared throughout this course, that reasoning about security requires one to carefully develop a formal model, carefully develop a specification, and design methods for verifying that the model satisfies the specification. Therefore, when reading your papers, you should constantly keep in mind the following three questions:

1. What model of systems are the authors using or proposing (and is that a reasonable choice)?
2. What specifications are the authors proposing to capture the properties they care about (and are they reasonable)?
3. What are the proposed verification techniques, if any (and what are their advantages/disadvantages)?

Timeline

Here is a rough timeline of what I have in mind:

- **By March 3rd**: Choose your project

- **Before March 31st**: Arrange a one-on-one meeting with me, where you will describe to me what you have extracted from the papers, how you have understood what the authors are doing, and go over what you propose to put in your presentation. If you have never put a presentation together before, that conversation should help you structure it, and I will provide further help if you need it. In any case, this conversation will convince me that you have read the papers and understood them.

- **From April 1st on**: Student presentations in class

There may be more students presentation than actual formal class time between April 1st and the end of classes. Meaning that we will have to find some time somewhere. We will discuss this in class towards the end of march, to see what can be done about it.

Some Project Suggestions

If you have some difficulty coming up with a project, here are some suggestions. The papers I give are purely indicative, and there is enough literature in any of these topics for you to find something to your liking. I do not have handy references for all those topics, so feel free to get in touch with me if one of the description with no associated paper interests you.

Another alternative for ideas is to look at past proceedings of the *Computer Security Foundations Workshop* (CSFW), at `http://www.csl.sri.com/programs/security/csfw/`, or the *IEEE Symposium on Security and Privacy*, at `http://www.ieee-security.org/TC/SP-Index.html`. Many of the papers at those conferences are available online. (And if not, generally a polite email to the authors will get you a copy.)

Strand Spaces. Strand spaces is a framework developed at MITRE for reasoning about security protocols. The selling point of the approach is that it provides for simple and intuitive proofs of how messages flow in a system. Sample papers include [17, 8].

Model Checking. Model checking is a automated verification technique often used in hardware verification and some forms of software verification to check that an abstract model satisfies a property expressed as a temporal properties of traces of events; that is, properties of the form: "if X occurs in the trace, then afterwards, Y always occurs", or "X occurs as long as Y does

not". This technique has been applied to protocol analysis, sample papers include [13, 12].

Types for Protocol Analysis. As I mentioned in class, it is possible to reason about processes in the spi calculus not just by analyzing the processes by equivalence, but also by imposing a type system developed so that a process type-checks if it satisfies a particular security property. Sample papers include [1, 7].

Logic Programming Approaches. Bruno Blanchet developed ProVerif, a protocol verification tool for the verification of security protocols based on logic programming (think, Prolog). It works quite well, and can take a variety of input languages, including a variant of the spi calculus. Sample papers include [4, 5]

Multiset Rewriting. Another protocol verification framework, this time based on rewrite rules and linear logic. Sample papers include [6, 3]

Constraints-Based Approaches. Another approach to analyzing protocols is to start from the end, so to speak. Very roughly speaking, describe what a bad trace looks like, and show that such a bad trace cannot occur under the constraints imposed by the protocol being analyzed.

Computational Soundness. The big assumption of Dolev-Yao style protocol analysis is that the cryptography is perfect: attackers cannot extract any information from an encrypted piece of data if they do not have the key. How far off from reality is this? There is a need to explore whether or not this assumption is really a problem or not, by comparing the Dolev-Yao attacker assumption with a more careful account of cryptography. The paper that started it all is [2].

Polynomial-Time Attackers. A natural follow-up of the previous topic, although in reality it is quite orthogonal. If we wanted to analyze protocols not so much in the presence of a Dolev-Yao adversary, but an adversary that can perform arbitrary probabilistic polynomial time computations, including guessing values used in the protocol, what would such a framework would look like? One approach is to define a probabilistic form of the pi calculus with a notion of polynomial-time boundedness built-in. Sample papers include [11, 14].

And leaving the realm of protocol analysis:

Intrusion Detection. Intrusion detection is the problem of detecting an intrusion on the system, typically as the system is running, by for instance monitoring messages exchanged on the network, or the system calls performed. Many systems have been proposed for doing this, and some people are trying to analyze them formally.

Enforceable Security Policies. A security policy is, generally construed, a classification of good and bad traces in a system. (A good trace satisfies the policy, a bad trace does not.) Given such a general definition of security policy, can you always enforce a security policy by monitoring the execution of a system? It turns out that the class of security policies that can be enforced by execution monitoring is a proper subclass of all security policies. But there is some debate as to what exactly execution monitoring means; sample papers include [16, 10].

Digital Rights Management. There is a lot of work in standard bodies nowadays to describe languages in which to express digital rights licenses: ODRL, XrML, and so on. There is also a lot of work in trying to understand what exactly these languages are capturing, and what they are not. Sample papers include [9, 15].

Access Control Models. What are appropriate formal models for reasoning about access control policies (think unix: which user is allowed to read what file, a very simple access control policy). More generally, can we come up with general models of access control that recover most existing approaches?

References

[1] M. Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 46(5):749–786, 1999.

[2] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[3] S. Bistarelli, I. Cervesato, G. Lenzini, and F. Martinelli. Relating multiset rewriting and process algebras for security protocol analysis. *Journal of Computer Security*, 13(1):3–47, 2005.

[4] B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Computer Society Press, 2001.

[5] B. Blanchet. From secrecy to authenticity in security protocols. In *Proc. 9th International Static Analysis Symposium (SAS'02)*, volume 2477 of *Lecture Notes in Computer Science*, pages 342–359. Springer-Verlag, 2002.

[6] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *Proc. 12th IEEE Computer Security Foundations Workshop (CSFW'99)*, pages 55–69. IEEE Computer Society Press, 1999.

[7] A. D. Gordon and A. Jeffrey. Authenticity by typing for security protocols. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 145–159. IEEE Computer Society Press, 2001.

[8] J. Y. Halpern and R. Pucella. On the relationship between strand spaces and multi-agent systems. *ACM Transactions on Information and System Security*, 6(1):43–70, 2003.

[9] J. Y. Halpern and V. Weissman. A formal foundation for XrML. In *Proc. 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, pages 251–263, 2004.

[10] K. W. Hamlen, G. Morrisett, and F. B. Schneider. Computability classes for enforcement mechanisms. *ACM Transactions on Programming Languages and Systems*, 28(1):175–205, 2006.

[11] P. Lincoln, J. C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *Proc. 5th ACM Conference on Computer and Communications Security (CCS'98)*, pages 112–121, 1998.

[12] W. Marrero, E. Clarke, and S. Jha. A model checker for authentication protocols. In *Proc. DIMACS Workshop on Cryptographic Protocol Design and Verification*, 1997. Available from `http://dimacs.rutgers.edu/Workshops/Security/`.

[13] J. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murφ. In *Proc. 1997 IEEE Symposium on Security and Privacy*, pages 141–151. IEEE Computer Society Press, 1997.

[14] J. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A probabilistic polynomial-time calculus for analysis of cryptographic protocols. In *Proc. 17th Annual Conference on the Mathematical Foundations of Programming Semantics*, volume 45 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2001.

[15] R. Pucella and V. Weissman. A formal foundation for ODRL. In *Proc. Workshop on Issues in the Theory of Security (WITS'04)*, 2004.

[16] F. B. Schneider. Enforceable security policies. *ACM Transactions on Information and System Security*, 3(1):30–50, 2000.

[17] F. J. Thayer, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.