Final Report on Seminar 03411: Language-Based Security

Anindya Banerjee, Heiko Mantel, David Naumann, Andrei Sabelfeld

Summary

Modern computing systems are particularly vulnerable to security attacks at the *application level*. Traditionally, security mechanisms have been based on low-level protection such as OS-based monitoring and access control. However, application-level attacks (e.g., the widely-publicized Lovebug and Melissa viruses executed on behalf of a mailer application) operate at a higher-level and circumvent the security mechanisms. Not only is malicious code a threat to security, but also unintended bugs in the specification and implementation of systems can lead to equally disastrous effects.

Application-level security is becoming an increasingly popular area of research because there is an increasing demand for applications to provide high assurance that particular security policies are followed. An effective way to achieve high assurance is to counter security threats at the same level as attacks—the application level. Because applications are typically specified and implemented in programming languages, this area is known as *language-based security*. A direct benefit of language-based security is the ability to naturally express security policies and enforcement mechanisms using the techniques of the well-developed area of *programming languages*. These techniques facilitate rigorous specifications of security policies as well as their mechanical verification.

Language-based techniques are gradually entering standard security practices. For example, the Java byte-code verifier is a language-based enforcement mechanism of particular integrity properties. As another example, the Java Virtual Machine and the .NET runtime system provide a dynamic access control mechanism that inspects the runtime stack to check whether permissions have been granted to code in the calling chain.

Despite such forays into mainstream security practices, there are a number of open issues in language-based security. One problem is to preserve the *confidentiality* of data by programs. This involves specification and enforcement of a property that guarantees that a program's public outputs do not (explicitly or implicitly) reveal information about the program's secret inputs. Recent technical advances allow enforcing confidentiality using a variety of language-based techniques e.g., *type systems, data-flow and control-flow analysis, abstract interpretation, model checking*, etc.

While more and more realistic security properties for more and more expressive languages are being considered, there are critical challenges remaining in the area of language-based security in general and in the area of program confidentiality in particular. To name just a few:

- security in concurrent and distributed systems
- certifying compilation
- minimization of the trusted computing base
- compositionality of security properties
- system-wide security
- high assurance in the presence of downgrading
- security analysis for machine languages
- security protocols

To gain insight on these challenges, and with the ultimate goal to enhance standard security practices

with language-based protection mechanisms, a Dagstuhl seminar on "Language-based Security" was held October 5 - 10, 2003.

The seminar was attended by 59 researchers from 10 countries. There were 39 technical presentations and 3 tutorials by the seminar participants. When the organizers sent out the initial invitations, they had expected between 35 to 45 participants. In the end, there were close to 60, and the seminar was oversubscribed. Except for the hour-long invited talks, most presentations were 30 minutes long. Each full day ran from 9 AM to 6 PM. The large number of talks posed a scheduling challenge for the organizers. But thanks to some participants who chose not to talk and to some who agreed to make short (15-20 mins.) presentations, the scheduling became more manageable. The talks followed a standard conference format with questions/discussions during and at the end. Breaks between talks also facilitated discussions. The busy schedule made it at times difficult to have too many discussions during a talk. Therefore, two provocative discussion sessions were arranged. An evening discussion led by Peter Ryan, focused on providing a definition to language-based security and open issues in the area. Another discussion led by Greg Morrisett, focused on security issues and open problems for next generation virtual machines.

In addition to discussions over drinks, cards and billiards, it was quite common to find groups of participants working past midnight. Many chose to keep working despite diversions like the Wednesday afternoon excursion. The organizers are grateful that it was possible to arrange an organ concert by participant Michael Clarkson at the Dagstuhl chapel. This was an unusual bonus to the busy academic schedule.

Scientific Highlights

The main theme of the seminar was further elucidated by way of three one-hour tutorials on three main application areas of language-based security.

- **Language-based Information-Flow Security** by Andrei Sabelfeld, Cornell University.
- **Typed Assembly Language (Background)** by Greg Morrisett, Cornell University.
- **Protocol Analysis** by Dieter Gollmann, TU Hamburg-Harburg.

Sabelfled's talk gave a survey of current research on information flow security, particularly focusing on static program analysis to enforce such policies. Four main subareas of research and open problems in information flow security were detailed: enriching language expressiveness, exploring the impact of concurrency, analyzing covert channels and refining security policies.

Morrisett's talk gave a survey of current research on typed assembly language. Typed assembly language is an idealized RISC-style assembly language with a formal operational semantics for a simple abstract machine. The goal is to provide type structure for machine code so that useful abstractions may be supplied to compilers to support specific security policies like memory safety.

Gollmann's talk gave a survey of approaches to protocol analysis and verification that are in some way linked to programming languages. Examples of topics considered were the Dolev-Yao intruder, protocols analyzed using CSP/FDR (e.g., Needham-Schroeder public key protocol), protocols specified using nominal calculi like the π-calculus, and the use of protocol analysis in the context of secure API's. The talk emphasized verification of protocols with respect to specified security goals in specified environments — the Dolev-Yao intruder model was discussed as one such example environment. Thus care must be taken in claiming discoveries of new flaws in a protocol: one needs to make sure that the protocol was indeed analyzed in an environment for which it was intended. The talk suggested that language-based protocol analysis tools are likely to be most useful for developers instantiating established protocol design techniques for use in standard environments —e.g., in web services.

Several significant areas of study were developed by the technical speakers. Noteworthy (but non-exhaustive) examples are:

- *Access control, cryptography, information flow and noninterference* as presented by Eijiro Sumii, University of Pennsylvania ("A bisimulation for dynamic sealing"), Tamara Rezk, INRIA, Sophia Antipolis ("Noninterference for the Java Virtual Machine"), Dominic Duggan, Stevens Institute of Technology ("Type-based distributed access control"), and Roberto Giacobazzi, Università di Verona ("Parameterized secrecy by abstract interpretation").

- *Specification and automatic validation of properties of security protocols*, as presented by Bruno Blanchet, MPI, Saarbrücken ("Automatic proof of strong secrecy for security protocols"), Andre Scedrov, University of Pennsylvania ("A probabilistic polynomial-time calculus for the analysis of cryptographic protocols), Flemming Nielson, Danish Technological University, ("Automatic validation of protocol narration"), Riccardo Focardi, Università Cá Foscari, Venezia ("Language-based security in authentication protocols"), and Luca Vigano, ETH Zürich ("An on-the-fly model checker for security protocol analysis").

- *Formulation of correctness properties for downgrading* as presented by David Sands, Chalmers University of Technology ("Controlled downgrading based on intransitive (non)interference"), and Reiner Hähnle, Chalmers University ("A theorem proving approach to secure information flow").

- *Memory safety* as presented by Drew Dean, Stanford Research International ("Definition of memory safety"), Michael Hicks, University of Maryland ("Safe and flexible memory management in Cyclone"), and Gogul Balakrishnan, University of Wisconsin ("Analyzing memory accesses in x86 executables").

- *Information flow policies in distributed systems and dynamic security policies* by Mads Dam, Swedish Institute of Computer Science ("Information flow control for cryptographic applets"), Andrew Myers, Cornell University ("Using information flow policies to construct secure distributed systems"), and Stephan Zdancewic, University of Pennsylvania ("First class principals in the decentralized label model").

A special mention must be made of the talk, "A semantics for web services authentication", presented by Cédric Fournet (Microsoft Research, Cambridge) which considered the problem of specifying and verifying cryptographic security protocols for XML web services. The protocols themselves are based on a faithful account of the XML wire format and are described as π-calculus processes. The work demonstrates a direct application of language-based security techniques in an area of practical importance, namely, web services.

Finally, the talks on downgrading presented initial results in an area that significantly impacts the entire direction of work on information flow security and noninterference and was a significant contribution made available by the Dagstuhl seminar.

Besides the tutorials and the talks, there were two open discussion sessions. Peter Ryan led a discussion on providing a definition for language-based security and challenging issues in language-based security. The consequences of a simple definition – application of semantics and programming languages to security – was explored in the following contexts: modeling user behavior, analyzing protocols, developing trustworthy code, formalizing software requirements and guiding software design by providing proper abstractions. Among the challenges discussed were compositionality of protocols and the connection between noninterference and access control.

Greg Morrisett led a discussion on next generation virtual machines. The main question considered was how one might design a secure operating system retaining the good features of abstract machines like the

Java Virtual Machine (JVM) or Microsoft's Common Language Runtime (CLR). Some issues focused on were: types for access control (capabilities), transfer of capabilities, resource control, JVM's thread model, exception handling and stack inspection.

Perspectives

The seminar was the first gathering of researchers working under the rubric of language-based security. The vibrant atmosphere at Dagstuhl provided an excellent opportunity for participants to be exposed to each other's research, to compare and contrast different approaches and to seek potential collaborations.

After five days of presentations, discussions, and debates, David Naumann led the discussion summarizing the main results of the seminar. Briefly stated, the conclusions are the following:

- Language-based security is a thriving research area with substantial intellectual content and significant applications and problems waiting to be solved. As the level of participation shows, there is a large community that has built around the topic. Participants deemed the Dagstuhl seminar a success and felt that a significant contribution has been made to further strengthen the community.
- Language-based security has a potential for substantially facilitating *security by design*. Applications are implemented in programming languages, systems are modeled at different levels of abstraction (using different languages), and security policies can be expressed and analyzed at each of these levels (e.g., by static analysis, model checking, formal verification). A closer integration of the various analysis techniques and their underlying security properties is the challenge we have to face for exploiting efficient language-based security techniques (e.g., security type systems) more systematically in a security-by-design approach.
- Critical open problems/questions emerging from seminar discussions include:
  - How to specify differing security goals for different parts of a system? For example, the same resource may have low integrity in one part of the system, hence performing a trusted action based on the resource is potentially dangerous. Yet, in another part of the system it may be safe to proceed with any action based on the resource.
  - How can specialized automated checking tools be integrated with constructive formal methods in high assurance software processes?
  - At what level (source or object) should the analysis and transformation of particular parts of a system be carried out?
  - How to handle dynamic information flow policies, e.g., information flow policies that change over time? How to connect to PKI?
  - How to specify and check security policies in extensible systems where components may be written in different languages? For example, how to specify confidentiality in a system where a webserver written in Java talks to a database implemented in SQL? Technical goals here may involve modular checking of security policies, composition of security policies, compiling to a common intermediate language and agreeing on a semantics for the common intermediate language.
  - Specification and verification of secrecy properties for security protocols (e.g., protocols for secure web services).
  - Integration of security protocols into compilers.

- Language-based treatment of mutual distrust.
- Language-based treatment of covert channels.
- How do language-based abstraction mechanisms interact with information flow?
- How can language-based techniques impact the design of next generation platforms (e.g., Microsoft's Common Language Runtime, or more speculatively, a secure operating system)? What security properties may one demand of a secure operating system?
- Development of a toolset to reduce the trusted computing base.
- What assurances can be provided in a system that employs downgrading?
- Empirical studies on the efficiency and usability of language-based methods in large systems. For example, for checking confidentiality using a type-based information flow analysis for a system where downloaded applets may call trusted library methods, it is possibly inefficient to annotate all library methods with information flow properties.

Keywords

Access control; information flow; noninterference; downgrading; protocol analysis; protocol verification; virtual machines; end-to-end security; memory safety; compositionality; semantic models.

Program Organizers

Anindya Banerjee (Kansas State University, USA)
Heiko Mantel (ETH Zürich, Switzerland)
David A. Naumann (Stevens Institute of Technology, USA)
Andrei Sabelfeld (Cornell University, USA)