03411 Abstracts Collection
Language-Based Security
— Dagstuhl Seminar —

A. Banerjee[1], H. Mantel[2], D. Naumann[3] and A. Sabelfeld[4]

[1] Kansas State Univ., USA
ab@cis.ksu.edu
[2] DFKI Saarbrücken, D
Heiko.Mantel@inf.ethz.ch
[3] Stevens Inst. of Techn., Hoboken, USA
naumann@cs.stevens-tech.edu
[4] Cornell Univ., Ithaca, USA
andrei@cs.cornell.edu

Abstract. From October 5-10, 2003, the Dagstuhl Seminar 03411 "Language-Based Security" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper.

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

03411 Final Report – Language-Based Security

This paper summarizes the objectives and structure of a seminar with the same title, held from October 5th to 10th 2003 at Schloss Dagstuhl, Germany.

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

Joint work of: Banerjee, Anindya Mantel, Heiko Naumann, David A. Sabelfeld, Andrei

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2005/172

Analyzing Memory Accesses in x86 executables

Gogul Balaskrishnan (Univ. Wisconsin - Madison)

This talk concerns static-analysis algorithms for analyzing binary executables. The aim of the work is to recover intermediate representations (IRs) that are similar to those that can be created for a program written in a high-level language. Our goal is to perform this task for programs such as viruses, worms, and mobile code. For such programs, symbol-table and debugging information is either entirely absent, or cannot be relied upon if present; hence, the analysis described in the paper makes no use of symbol-table/debugging information.

The main analysis that will be discussed, called value-set analysis, tracks address-valued and integer-valued quantities simultaneously. It is related to pointer-analysis algorithms that have been developed for programs written in high-level languages, which determine an over-approximation of the set of variables whose addresses each pointer variable can hold. At the same time, value-set analysis is similar to range analysis and other numeric static-analysis algorithms that over-approximate the integer values that each variable can hold.

The techniques that will be described have been implemented as part of CodeSurfer/x86, a prototype tool for browsing ("surfing"), inspecting, and analyzing x86 executables.

Joint work of: Balaskrishnan, Gogul; Reps, Thomas;

Stack-based Access Control for Secure Information Flow

Anindya Banerjee (Kansas State University)

Access control mechanisms are often used with the intent of enforcing confidentiality and integrity policies, but few rigorous connections have been made between information flow and runtime access control.

The Java virtual machine and the .NET runtime system provide a dynamic access control mechanism in which permissions are granted to program units and a runtime mechanism checks permissions of code in the calling chain. We investigate a design pattern by which this mechanism can be used to achieve confidentiality and integrity goals: a single interface serves callers of more than one security level and dynamic access control prevents release of high information to low callers. Programs fitting this pattern would be rejected by previous flow analyses. We give a static analysis that admits them, using permission-dependent security types. The analysis is given for a class-based object-oriented language with features including inheritance, dynamic binding, recursive types and type casts. The analysis is shown to ensure a noninterference property formalizing confidentiality and integrity.

Joint work of: Banerjee, Anindya; Naumann, David A.;

Flexibility and Trustworthiness in Proof-Carrying Code

Andrew Bernard (CMU - Pittsburgh)

Proof-carrying code (PCC) is a framework for ensuring that untrusted programs are safe to install and execute. When using PCC, programs are required to contain a proof that allows the program text to be checked statically and efficiently for safe behavior. I will present an engineering improvement to PCC that simplifies the checking software and simultaneously enables the proof to be checked against an explicit temporal-logic security-policy specification.

To make this approach practical, our checking software reconstructs a safety proof from the trace of an untrusted logic program. Our experiments suggest that proof sizes can remain small if we spend additional time on proof checking.

Joint work of: Bernard, Andrew; Lee, Peter; Donohue, Michael; Magill, Stephen;

Automatic Proof of Strong Secrecy for Security Protocols

Bruno Blanchet (MPI für Informatik)

We present a new automatic technique for proving strong secrecy for security protocols. Strong secrecy means that an adversary cannot see any difference when the value of the secret changes. Strong secrecy detects implicit flows and flows of partial information. It is a particular case of observational equivalence, which gives nice compositionality properties and makes it easier to combine manual and automatic proofs. Our technique relies on an automatic translation of the protocol into Horn clauses, and a resolution algorithm on the clauses. It requires important extensions with respect to previous work for (syntactic) secrecy and authenticity. This technique can handle a wide range of cryptographic primitives, and yields proofs valid for an unbounded number of sessions and an unbounded message space; it is also flexible and efficient. We have proved its correctness, its termination on the large class of tagged protocols, and implemented it.

Type-based Security in Ambient-based Calculi: A Calculus of Bounded Capacities

Michele Bugliesi (Univ. Ca' Foscari - Venezia)

Resource control has attracted increasing interest in foundational research on distributed systems. In this talk, I focus focuses on space control and describe an analysis of space usage in the context of an ambient-like calculus with bounded capacities and weighed processes, where migration and activation require space. A type system complements the dynamics of the calculus by providing static guarantees that the intended capacity bounds are preserved throughout the computation.

Joint work of: Bugliesi, Michele; Barbanera, Franco; Dezani, Mariangiola; Sassone, Vladimiro;

Correspondence Assertions for Process Synchronization in Concurrent Communications

Adriana Compagnoni (Stevens Institute of Technology)

High-level specification of patterns of communications such as protocols can be modeled elegantly by means of session types. However, a number of examples suggest that session types fall short when finer precision on protocol specification is required. In order to increase the expressiveness of session types we appeal to the theory of correspondence assertions. The resulting type discipline augments the types of long term channels with effects and thus yields types which may depend on messages read/written prior within the same session. We prove that evaluation preserves typability and that well-typed processes are safe. Also, we illustrate how the resulting theory allows us to address the shortcomings present in the pure theory of session types.

In Proceedings of FOCLASA 2003.

Joint work of: Bonelli, Eduardo; Compagnoni, Adriana; Gunter, Elsa

Information Leakage Analysis in Mobile Ambients

Agostino Cortesi (Univ. Ca' Foscari - Venezia)

A multilevel security policy is considered in the scenario of mobile systems, and modeled within "pure" Mobile Ambients calculus, in which no communication channels are present and the only possible actions are represented by the moves performed by mobile processes. Information flow is defined in terms of the possibility for a confidential ambient/data to move outside a security boundary. A control flow analysis is presented, as a refinement of the Hansen-Jensen-Nielsons's CFA, that allows to capture boundary crossings with better accuracy. In this way, both direct and information leakage may be statically detected.

Joint work of: Cortesi, Agostino; Braghin, Chiara; Focardi, Riccardo;

Information Flow Control for Cryptographic Applets

Mads Dam (Swedish Institute of CS)

We consider the problem of protecting confidentiality of critical data such as private keys, pin's, or application data in the context of cryptographic protocols.

In this case multi-level security is not in general applicable. Based on earlier work on admissible interference we propose an automaton-based framework for describing flows of sensitive data, involving dynamically changing security levels. We give two successive generalisations of Volpano-Smith type non-interference in terms of bisimulation-like relations, corresponding to a form of intransitive non-interference in one case, and controlled downgrading in another. A preliminary study based on a JavaCard 2.0 implementation of PKCS#11 seems to indicate that the flow automata we propose are expressive enough to capture the intended flows of the private RSA exponent in a compact and natural way.

Joint work of: Dam, Mads; Giambiagi, Pablo;

Definition of Memory Safety

Drew Dean (SRI - Menlo Park)

The term "memory safety" is often used, but there is no canonical definition of it. We first identify requirements for a useful defintion from the viewpoint of computer security. In work in progress, we propose a definition based on a coloring invariant that meets 2 of the 3 requirements, and may be extensible to meet the third requirement.

Approximate Non-Interference

Alessandra Di Pierro (Università di Pisa)

In this work we lay the semantic basis for a quantitative security analysis of probabilistic concurrent systems by introducing notions of approximate confinement based on various process equivalences.

We re-cast the operational semantics classically expressed via probabilistic transition systems (PTS) in terms of linear operators and we show a technique for defining approximate semantics as probabilistic abstract interpretations of the PTS semantics.

An operator norm is then used to quantify this approximation.

This provides a quantitative measure of the indistinguishability of two processes and therefore of their confinement.

To this quantity we then give a statistical interpretation which relates it to the number of tests needed to breach the security of the system.

Joint work of: Di Pierro, Alessandra; Hankin, Chris; Wiklicky, Herbert;

Type-Based Distributed Access Control

Dominic Duggan (Stevens Institute of Technology)

The Key-Based Decentralized Label Model (KDLM) is a type system that combines a weak form of information flow control, termed distributed access control, with typed cryptographic operations. The motivation is to have a type system that ensures access control and secure data sharing, while giving the application the responsibility to secure network communications, and to do this safely. KDLM introduces the notion of declassification certificates, a form of distributed declassification to support the declassification of encrypted data.

Joint work of: Duggan, Dominic; Chothia, Tom; Vitek, Jan;

Language-based Security in Authentication Protocols

Riccardo Focardi (Univ. Ca' Foscari - Venezia)

We propose a new method for the static analysis of entity authentication protocols. Our approach draws on an earlier work on a role-based characterization of the formal properties of this class of protocols. We develop our approach based on a dialect of the spi-calculus as the underlying formalism for expressing protocol narrations. Our analysis validates the honest protocol participants against static (hence decidable) and easily checked conditions that provide formal guarantees of entity authentication. The main result is that the validation of each component is provably sound and fully compositional: if all the protocol participants are successfully validated, then the protocol as a whole guarantees entity authentication in the presence of Dolev-Yao intruders.

Joint work of: Focardi, Riccardo; Bugliesi, Michele; Maffei, Matteo;

Types for Stack-Based Access Control

Jeffrey Foster (University of Maryland)

Stack-based access control is a mechanism used in systems such as Java and the Common Language Runtime to define and enforce fine-grain security policies. We develop a new type system for statically enforcing stack-based access control. In our system, key-pairs guard access to resources, and the association between key-pairs and resources can be changed at any program point (i.e., the binding is late). Our type system uses lexically scoped abstract names and ordering between them to allow local access control policies to be enforced in other parts of the code. In particular, this means that individual program components can locally refine access control policies, and the policies will be respected by the entire program.

The result is a static type system that can enforce, at compile time, a wide variety of useful, fine-grain access control patterns.

Joint work of: Foster, Jeffrey; Terauchi, Tachio; Aiken, Alex;

A Semantics for Web Services Authentication

Cedric Fournet (Microsoft Research UK)

We consider the problem of specifying and verifying cryptographic security protocols for XML web services. The security specification WS-Sec describes a range of XML security tokens, such as username tokens, public-key certificates, and digital signature blocks, amounting to a flexible vocabulary for expressing protocols. To describe the syntax of these tokens, we extend the usual XML data model with symbolic representations of cryptographic values. We use predicates on this data model to describe the semantics of security tokens and of sample protocols distributed with the WSE implementation of WS-Sec. By embedding our data model within Abadi and Fournet's applied pi calculus, we formulate and prove security properties with respect to the standard Dolev-Yao threat model. Moreover, we informally discuss issues not addressed by the formal model. To the best of our knowledge, this is the first approach to the specification and verification of security protocols based on a faithful account of the XML wire format.

Latest slides and papers are available from http://securing.ws and http://research.microsoft.com/ fournet

Joint work of: Fournet, Cedric; Bhargavan, Karthikeyan; Gordon, Andrew D.;

Inherently Safe Mobile Code Formats

Michael Franz (Univ. California - Irvine)

Java pioneered the concept of mobile code verification, checking a mobile program itself (instead of a digital signature) to test whether it is safe to run. Later came Proof Carrying Code, which enables to shift most of the verification effort from code consumer to code producer. We have invented a third alternative, by which unsafe programs cannot even be encoded in an intermediate representation to begin with, eliminating the need for verification altogether.

Abstract non-interference

Roberto Giacobazzi (Università di Verona)

In this paper we introduce the notion of parametric non-interference relatively to an observation. This means that no unauthorized flow of information is possible in programs from confidential to public data, relatively to what the attacker can observe on input/output.

The idea is to consider attackers as dataflow analyzers, whose task is to reveal properties of confidential resources by analyzing public ones. We prove that this notion can be fully specified in standard abstract interpretation framework, making the degree of security of a program a property of its semantics. We introduce systematic methods for extracting attackers from programs, providing a domain-theoretic characterization of the most precise attacker which cannot violate the security of a given program. These methods allow us both to compare attackers and program secrecy by comparing the corresponding abstractions in the lattice of abstract interpretations, and to design automatic program certification tools for security by abstract interpretation.

Joint work of: Giacobazzi, Roberto; Mastroeni, Isabella;

Protocol Analysis (Tutorial)

Dieter Gollmann (TU Hamburg-Harburg)

Major issues in protocol analysis can be linked back to the influential paper by Dolev & Yao [IEEE-IT, 1983].

In terms of modeling environment and attacks, the Dolev-Yao model gives the adversary full control of the communications system. In terms of analytical methods, the adversary is restricted to algebraic manipulations on strings of symbols. Thus, protocol analysis in the Dolev-Yao model becomes a term-rewriting problem.

Modeling issues beyond Dolev-Yao touch on the distinction between outsider attacks (the traditional view in protocol analysis) and insider attacks (such as LoweŠs attack on the Needham-Schroeder public key protocol), or with scenarios where it is not assumed that the adversary has full control of the communications system (example: analysis of binding updates in Mobile IPv6).

As security in the Dolev-Yao model is translated into the word problem in type 0 languages, security is undecidable in general. Research has therefore either identified classes of protocols where efficient algorithms exist, or developed and extended techniques that work for restricted models or need not terminate for correct protocols.

In the evaluation of analysis methods, one should distinguish between tools for the analysis of protocols that should meet well established goals and use well established cryptographic mechanisms, and tools for the analysis of novel protocols. The latter case requires ŞagileŤ methodologies where it is easy to adjust the model of the environment and the formalization of security goals.

Safe and Flexible Memory Management in Cyclone

Michael Hicks (University of Maryland)

Cyclone is a type-safe programming language intended for applications requiring control over memory management. Our previous work on Cyclone included support for stack allocation, lexical region allocation, and a garbage-collected heap. We achieved safety (i.e., prevented dangling pointers) through a region-based type-and-effects system. This paper describes some new memory-management mechanisms that we have integrated into Cyclone: dynamic regions, unique pointers, and reference-counted objects. Our experience shows that these new mechanisms are well suited for the timely recovery of objects in situations where it is awkward to use lexical regions. Crucially, programmers can write reusable functions without unnecessarily restricting callers' choices among the variety of memory-management options. To achieve this goal, Cyclone employs a combination of polymorphism and scoped constructs that temporarily let us treat objects as if they were allocated in a lexical region. Experience writing two server applications using our new constructs was positive: compared to using conservative garbage collection, memory usage could be significantly reduced, and throughput improved by up to 42

Paper link at http://www.cs.umd.edu/ mwh/papers/cyclone-mm.pdf. Cyclone homepage at http://www.cs.cornell.edu/projects/cyclone/.

Joint work of: Hicks, Michael; Morrisett, Greg; Grossman, Dan; Jim, Trever;

Pi-Calculus, Flow Analysis and Non-Interference

Kohei Honda (Queen Mary College - London)

We consider how we can use the pi-calculus to have a precise analysis of the idea of information flow in programs. The idea is to make the best of a precise representability of diverse programming languages in typed processes, to allow an analysis which is both intuitive and lucid, combining both data and control flows. Extracting information flow in a given typed process in the π-calculus is easier than extracting from programs simply because operations — reading variables, calling procedures, communicating with remote objects, etc. — are all represented as sequences of tiny interactions of name passing processes. If time remains, we shall also discuss a theoretical underpinning of such an endeavour: the idea of polarity in interactions, as discussed in both pi-calculus studies and Linear Logic, plays a basic role in flow analyses of both sequential and concurrent processes.

Quantitative Analysis of Leakage of Confidential Information

Sebastian Hunt (City University - London)

Programs whose high-inputs have no effect on their low-outputs are said to satisfy the non-interference condition. We address the question of what more can be said about programs which do not satisfy non-interference. We show how Shannon's information theory can be used to define a measure of how much information a program transmits from its high-inputs to its outputs.

We describe a quantitative analysis for a While language; the analysis calculates bounds on the leakage into each variable under worst-case assumptions of the choice of low-inputs. This work improves upon the authors' previous work by dealing with programs which contain loops and by refining their previous treatments of arithmetic and boolean expressions.

Joint work of: Hunt, Sebastian; Clark, David; Malacaria, Pasquale;

A Theorem Proving Approach to Analysis of Secure Information Flow

Reiner Hähnle (Chalmers UT - Göteborg)

Most attempts at analysing secure information flow in programs are based on domain-specific logics. Though computationally feasible, these approaches suffer from the need for abstraction and the high cost of building dedicated tools for real programming languages. We recast the information flow problem in a general program logic rather than a problem-specific one. We investigate the feasibility of this approach by showing how a general purpose tool for software verification can be used to perform information flow analyses. We are able to handle phenomena like method calls, loops, and object types for the target language Java Card. We are also able to prove insecurity of programs and to express declassification of information.

Joint work of: Hähnle, Reiner; Darvas, Ádám; Sands, David;

Relative Secrecy and Semantics of Declassification

Peeter Laud (University of Tartu)

The concept of relative secrecy covers the case where we want the program's public outputs to give away no information about the secret inputs, but at the same time we have determined that certain other outputs, giving away some information about the secret inputs, give away no sensitive information about them. In this case we may allow the public outputs to give away some information about the secret inputs, but nothing more than the aforementioned other outputs give.

In our talk we give the formal definition of relative secrecy and also show how to construct analyses for this property. As a useful application, we show how to use relative secrecy to give a possible formal meaning for declassify-statements that are used to mark that a certain variable at a certain program point does not contain sensitive information, even if a program analysis for secure information flow inferes that it does.

Joint work of: Laud, Peeter; Ülikool, Tartu;

Roles & Security

Gurvan Le Guernic (Kansas State University)

This talk present a work in progress concerning the declassification. In the majority of works on declassification, the main focus is given to the security label to which the information is declassify. Of course, this label is really important; but sometimes it can be quite difficult to know "what" is the value which is declassified.

The aim of this work is to add a usage information (role) to the data manipulated by the program. Among those usage informations is defined an interference relation. This relation gives some information about the source of a value with a given "role". This information is used when the data is declassified to ensure that the value declassified is really the one intended to be declassified.

Language-based Security and Formal Software Development

Heiko Mantel (ETH Zürich)

Language-based security techniques aim at analyzing the security of a given program after it has been developed. In contrast, the objective of a formal software development process is to ensure security already during the development (security-by-design). In this talk, I will discuss how these two approaches could be fruitfully integrated. More specifically, I will outline how language-based analysis techniques, such as security-type systems, can be used in combination with the more traditional verification techniques like unwinding in order to rigorously ensure that a system designed to be secure is indeed secure. This possibility arises from a formal connection between the security properties used in formal developments and the ones underlying language-based security techniques (joint work with Andrei Sabelfeld). I will also point out some other interesting directions that could be exploited on this basis, such as combining different analysis techniques to ensure system-wide security and analyzing programs written in multiple languages.

A language for real-time cryptographic protocol analysis

Fabio Martinelli (CNR - Pisa)

In this talk we present a language and a methodology for the analysis of real-time features of cryptographic protocols. We show how it is suitable to formally verify security properties of wireless/multicast protocols. We define also a compositional proof rule for establishing security properties of such protocols. The effectiveness of our approach is shown by defining and studying the timed integrity property for mu-TESLA, a well-known protocol for wireless sensor networks. We are able to deal with protocol specifications with an arbitrary number of agents (senders as well as receivers) running the protocol.

Joint work of: Martinelli, Fabio; Gorrieri, Roberto; Petrocchi, Marinella; Vaccarelli, Anna;

Composition of Cryptographic Protocols in a Probabilistic Process Calculus

Paulo Mateus (Instituto Superior Tecnico - Lisboa)

We describe a probabilistic polynomial-time process calculus for analyzing cryptographic protocols and use it to derive compositionality properties of protocols in the presence of computationally bounded adversaries.

We illustrate these concepts on oblivious transfer, an example from cryptography.

We also compare our approach with a framework based on interactive Turing machines.

Joint work of: Mateus, Paulo; Mitchell, J.; Scedrov, A.;

Typing Noninterference for a Reactive Language

Ana Matos (INRIA - Sophia Antipolis)

My talk is about the design of a type system for noninterference of programs in a reactive core language, as well of its soundness proof. The reactive primitives offer new expressive power with reflections on the forms of security leaks that are to be controlled. On one hand we are able to code interference examples analogous to those we find in concurrent imperative languages (Ex: PIN), and to use analogous reasonings when designing the typing rules. On the other hand, the deterministic nature of the reactive concurrency calls for more restrictions on the set of accepted programs. In addition to the impact on the type system itself, the deterministic nature of the language allows for an alternative definition of "secure programs", which is usually based on a bisimulation. Inspired by the "Determinacy => (Observation equivalence = Trace equivalence)" result by Engelfriet we establish an equivalence between the bisimulation definition and an alternative one using trace-like reasoning. This simplifies the task of proving the soundness of the type system.

Joint work of: Matos, Ana; Boudol, Gerard; Castellani, Ilaria;

Typed Assembly Language (Background)

Greg Morrisett (Cornell University)

The promise of proof-carrying code (PCC) is that we can enforce a wide range of security policies on actual machine code using only a very small trusted computing base. However, PCC does not eliminate the hard problem of finding proofs; it rather shifts the burden from the code consumer to the code producer.

The combination of type-safe, high-level languages and certifying compilers provides a methodology for achieving the proofs needed in a PCC architecture. The desired policy is encoded as a type system at the source level. Programmers help to construct the initial source-level proof by rewriting their code until it type-checks (i.e., until a source-level type reconstruction engine can automatically construct the proof.) Often, this requires typing annotations that guide the type-checker. Certifying compilers conceptually transform the source-level proof in parallel with the code to produce a proof that the target level code has the desired properties.

The primary challenge in building a certifying compiler is designing type systems for low-level languages, including compiler intermediate representations and ultimately machine code. Today's type systems for virtual machines, such as the Java VM or Microsoft's Common Language Runtime, tend to be too high-level in order to keep the type system relatively simple. However, the invariants imposed by these high-level constructors prevent the expression of certain optimizations including tail-call elimination, instruction scheduling, and register allocation. They also tend to prevent efficient encodings of linguistic features such as continuations or covariant arrays.

The goal of the TAL project was to examine a "RISC" approach to the design of low-level type systems. By providing a set of orthogonal type constructors, the hope was that higher-level typing abstractions could be adequately encoded.

Keywords: Proof-carrying code, Typed Assembly Language

Using Information Flow Policies to Construct Secure Distributed Systems

Andrew Myers (Cornell University)

A challenging unsolved security problem is how to specify and enforce system-wide security policies; this problem is even more acute in distributed systems with mutual distrust. This talk describes a way to enforce policies for data confidentiality and integrity in such an environment. Programs annotated with security specifications are statically checked and then transformed by the compiler to run securely on a distributed system with untrusted hosts. The code and data of the computation are partitioned and replicated across the available hosts in accordance with the security specification and without placing undue trust in any host. The compiler automatically generates secure run-time protocols for communication among the replicated code partitions. A prototype implementation has been applied to various distributed programs.

Joint work of: Myers, Andrew; Zheng, Lantian; Chong, Stephen; Zdancewic, Steve;

Automatic Validation of Protocol Narration

Flemming Nielson (Technical University of Denmark)

We perform a systematic expansion of protocol narrations into terms of a process algebra in order to make precise some of the detailed checks that need to be made in a protocol. We then apply static analysis technology to develop an automatic validation procedure for protocols. Finally, we demonstrate that these techniques suffice for identifying a number of authentication flaws in symmetric key protocols such as Needham-Schroeder, Otway-Rees, Yahalom and Andrew Secure RPC.

Joint work of: Nielson, Flemming; Bodei, Chiara; Degano, Pierpaolo; Buchholtz, Mikael; Nielson, Hanne Riis;

Weighted Push Down Systems

Thomas Reps (Univ. Wisconsin - Madison)

This talk concerns the extension of pushdown systems to weighted pushdown systems, and the application of this formalism to two problems:
o Answering authorization questions for certificate sets expressed in SPKI/SDSI.
o Performing interprocedural dataflow analysis. (Actually, weighted pushdown systems support a generalized form of interprocedural dataflow analysis queries of the form “What is the meet-over-all-paths value to [or from] a set of stack configurations?”, where a set of stack configurations in a query is a regular set.)

Joint work of: Reps, Thomas; Schwoon, S.; Jha, S.; Stubblebine, S.;

Non-Interference for the Java Virtual Machine

Tamara Rezk (INRIA - Sophia Antipolis)

The Java Platform combines static and dynamic mechanisms to enforce innocuity of applications, but these mechanisms fail to guarantee that confidential information will not leak to unauthorized principals. In order to provide stronger guarantees with respect to confidentiality, it is thus necessary to rely on alternative mechanisms, such as static analyses or type systems that enforce non-interference, a high-level security property that guarantees the absence of illicit information flows during a program execution. While Banerjee and Naumann have recently investigated the use of such type systems for the Java language, there lacks a similar study for the Java Virtual Machine, and more generally for “low-level languages [that] have not received much attention in studies of secure information flow” (Sabelfeld and Myers, 2003). We propose an information flow type system for a non-trivial fragment of the Java Virtual Machine that includes classes, arrays, methods, exceptions and subroutines, and show that it enforces non-interference. Our type system is compatible with the principles of bytecode verification, and could thus be used as the basis of an enhanced bytecode verifier for the Java Platform.

Joint work of: Rezk, Tamara; Barthe, Gilles;

Language-Based Information-Flow Security (Tutorial)

Andrei Sabelfeld (Cornell University)

Current standard security practices do not provide substantial assurance that the end-to-end behavior of a computing system satisfies important security policies such as confidentiality. An end-to-end confidentiality policy might assert that secret input data cannot be inferred by an attacker through the attacker's observations of system output; this policy regulates information flow.

Conventional security mechanisms such as access control and encryption do not directly address the enforcement of information-flow policies. Recently, a promising new approach has been developed: the use of programming-language techniques for specifying and enforcing information-flow policies. This talk is intended to give the big picture of recent and current research on information-flow security, particularly focusing on work that uses static program analysis to enforce information-flow policies.

The talk is based on a survey article (IEEE J-SAC, 21(1):5-19, Jan. 2003) written jointly with Andrew Myers, available via http://www.cs.cornell.edu/ andrei/Papers/jsac.ps

Joint work of: Sabelfeld, Andrei; Myers, Andrew C.;

Controlled Downgrading based on Intransitive (Non)Interference

David Sands (Chalmers UT - Göteborg)

Traditional noninterference cannot cope with common features of secure systems like channel control, information filtering, or explicit downgrading. Some recent research in language-based setting has addressed the derivation and use of weaker security conditions that could support some such features. However, a fully satisfactory solution to the problem has yet to be found. A key problem is to permit exceptions to a given security policy without permitting too much. In this article, we propose an approach that draws its underlying ideas from intransitive noninterference, a concept usually used on a more abstract specification level. Our results include a new bisimulation-based security condition that controls tightly where downgrading can occur and a sound security type system for checking this condition.

Joint work of: Sands, David; Mantel, Heiko;

A Probabilistic Polynomial-Time Calculus for the Analysis of Cryptographic Protocols

André Scedrov (University of Pennsylvania)

We describe properties of a process calculus that has been developed for the purpose of analyzing security protocols. The process calculus is a version of CCS, with bounded replication and probabilistic polynomial-time expressions allowed in messages and boolean tests. In order to avoid problems expressing cryptographic notions in the presence of nondeterminism, messages are scheduled probabilistically instead of nondeterministically. We prove that evaluation may be completed in probabilistic polynomial time and develop properties of a form of asymptotic protocol equivalence that allows security to be specified using observational equivalence, a standard relation from programming language theory that involves quantifying over possible environments that might interact with the protocol.

We also relate process equivalence to cryptographic concepts such as indistinguishability by polynomial-time statistical tests.

Joint work of: Scedrov, André; Mitchell, J.; Ramanathan, A.; Teague, V.; Mateus, P.; Lincoln, P.; Mitchell, M.;

Security Types for Multiagent Systems Development

Axel Schairer (DFKI Saarbrücken)

Starting from global confidentiality requirements for a whole multiagent system we first develop sufficient confidentiality requirements for single agents in a goal-directed way. This involves several design decisions. In a second step, the agents can then independently be developed to satisfy their respective requirements.

However, these two steps are interdependent and the intended functionality of the agents has to be taken into account to decompose the requirements in a sensible way.

We suggest using security types systems to aid in the decomposition.

The agents' algorithms are expressed as pseudo-programs and type-correctness is related to the confidentiality properties used to express the requirements. The global requirement corresponds to certain constraints on the type annotations of the pseudo-programs.

Propagation according to the type system adds constraints. The result is that the agents' individual requirements can be read off the resulting type annotations of the programs.

A Bisimulation for Dynamic Sealing

Eijiro Sumii (University of Pennsylvania)

We define lambda-seal, an untyped call-by-value lambda-calculus with primitives for sealing, and develop a bisimulation proof method that is sound and complete with respect to contextual equivalence. This provides a formal basis for reasoning about data abstraction in open, dynamic settings where static techniques such as type abstraction and logical relations are not applicable.

[Manuscript available at http://www.yl.is.s.u-tokyo.ac.jp/ sumii/pub/]

Joint work of: Sumii, Eijiro; Pierce, Benjamin C.;

An On-The-Fly Model-Checker for Security Protocol Analysis

Luca Vigano (ETH Zürich)

The on-the-fly model-checker OFMC is a state-of-the-art tool for analyzing security protocols, which owes its success to the combination of two ideas. The first is the modeling of protocols using lazy data-types in a higher-order functional programming language, which allows for the automated analysis of security properties using infinite-state model checking, where the model is explicitly built, on-the-fly, in a demand-driven fashion. The second is the integration of several search optimizations that are based on symbolic techniques and that significantly reduce the infinite search tree associated with a protocol without excluding any attacks. Extensive experimentation showed that our tool is state-of-the-art, both in terms of coverage and performance, and that it scales well to industrial-strength protocols.

Joint work of: Vigano, Luca; Basin, David; Mödersheim, Sebastian;

Analysing the Propagation of Computer Viruses

Herbert Wiklicky (Imperial College London)

We investigate the problem of how malicious code, i.e. computer viruses, can infect an entire network. Our approach is based on a probabilistic model describing the chances that a certain node infects some of its neighbours. We formalise this situation using a probabilistic version of a process calculus which is the core of KLAIM, a language for distributed and mobile computing based on interactions through distributed tuple spaces. The analysis we present exploits tecniques based on probabilistic abstract interpretation and provides an extimate of the probability of a virus propagation through a computer network.

Joint work of: Wiklicky, Herbert; Di Pierro, Alessandra; Hankin, Chris;

Secrecy and Fine-Grained Types for Higher-Order Mobile Code

Nobuko Yoshida (Imperial College London)

In wide area distributed systems it is now common for higher-order code to be transferred from one domain to another; the receiving host may initialise parameters and then execute the code in its local environment. In this talk we analise secrecy property of higher-order mobile code by a fine-grained typing system for a higher-order pi-calculus which can be used to control the effect of such migrating code on local environments.

Processes may be assigned different types depending on their intended use. This is in contrast to most of the previous work on typing processes where all processes are typed by a unique constant type, indicating essentially that they are well-typed relative to a particular environment. Our fine-grained typing facilitates the management of access rights and provides host protection from potentially malicious behaviour.

Our process type takes the form of an interface limiting the resources to which it has access, and the types at which they may be used. This framework is smoothly extendable to reason about fine-grained code security depending on untrust/trust distributed environments.

Security Issues in Coordination Models and Languages

Gianluigi Zavattaro (Università di Bologna)

In this presentation we describe how coordination models and languages are evolving in order to support the programming of new classes of networks with a high level of mobility and dynamicity (e.g. mobile ad hoc networks, internet applications, etc.).

In particular, we focus on the linguistic improvements that are currently proposed in order to support some form of secure coordination; this is a critical aspect because the applications in this class of networks are usually open to previously unknown agents, that could potentially present malicious behaviours.

First-Class Principals in the Decentralized Label Model

Stephan Zdancewic (University of Pennsylvania)

Type systems that enforce information-flow properties allow the programmer to express confidentiality and integrity policies on the data used in the program. Typically, these policies are static–the security labels on the data are determined at compile time. In practice, it is sometimes useful to allow the security policies to depend on run-time information, for instance to permit different information to be visible to users with root privileges. As this example shows, there must also be some way of connecting dynamic principals and authentication mechanisms.

This talk will present some work in progress on developing a security-typed language with dynamic principals. The type system, which incorporates a variant of Myers' and Liskov's decentralized label model, distinguishes between statically known principals and dynamic ones. An interesting question in this setting is how to regulate declassification when the principal whose authority is needed to perform the declassification may change at run time. I will sketch some ideas that suggest it is possible to smoothly integrate languages for information-flow security with public-key infrastructures for authentication.

Joint work of: Zdancewic, Stephan; Tse, Stephen;