Program Analysis for Security and Privacy
Report on the WS PASSWORD at ECOOP'06

Marco Pistoia[1] and Francesco Logozzo[2]

[1] IBM T. J. Watson Research Center, Hawthorne, New York, USA
pistoia@us.ibm.com
[2] École Normale Supérieure, Paris, France
Francesco.Logozzo@polytechnique.edu

Abstract. Software security has become more important than ever. Unfortunately, still now, the security of a software system is almost always retrofitted to an afterthought. When security problems arise, understanding and correcting them can be very challenging. On the one hand, the program analysis research community has created numerous static and dynamic analysis tools for performance optimization and bug detection in object-oriented programs. On the other hand, the security and privacy research community has been looking for solutions to automatically detect security problems, privacy violations, and access-control requirements of object-oriented programs. The purpose of the First Program Analysis for Security and Safety Workshop Discussion (PASSWORD 2006), co-located with the Twentieth European Conference on Object-Oriented Programming (ECOOP 2006), was to bring together members of the academic and industrial communities interested in applying analysis, testing, and verification to security and privacy problems, and to encourage program analysis researchers to see the applicability of their work to security and privacy—an area of research that still needs a lot of exploration. This paper summarizes the discussions and contributions of the PASSWORD workshop.

1 Introduction

Security has progressively become more interesting to the program analysis community. Both static and dynamic program analysis have been extensively applied to security issues, such as access control [6] and information flow [5,16]. However, there is still high potential for more research on how to apply analysis, testing, and verification to security and privacy problems, including:

1. Evaluation of security and privacy policies
2. Identification of vulnerabilities that could lead to denial of service attacks
3. Verification of access control
4. Computation of access control requirements
5. Identification of mutability, accessibility, and isolation policy violations
6. Verification of complete authorization
7. Intrusion detection
8. Secure programming

M. Südholt and C. Consel (Eds.): ECOOP 2006 Ws, LNCS 4379, pp. 59–68, 2007.

This paper summarizes the contributions of the First Program Analysis for Security and Safety Workshop Discussion (PASSWORD 2006), co-located with the Twentieth European Conference on Object-Oriented Programming (ECOOP 2006), which was held in Nantes, France in July 2006.

2 Objective and Call for Papers

With the advent of the Internet, software security has become more important than ever. Unfortunately, still now, the security of a software system is almost always an afterthought. When security problems arise, understanding and correcting them can be very challenging. On the one hand, the program analysis research community has created numerous static and dynamic analysis tools for performance optimization and bug detection in object-oriented programs. On the other hand, the security and privacy research community has been looking for solutions to automatically detect security problems, privacy violations, and access-control requirements of object-oriented programs. The purpose of PASSWORD 2006 was to bring together members of both these communities and to encourage program analysis researchers to see the applicability of their work to security and privacy—an area of research that still needs exploration.

Two types of papers were welcomed at PASSWORD 2006: *technical papers*, which present mature technical and research material, and *position, exploratory, or preliminary-work papers*, which may describe work in progress or new research ideas.

Topics of interest for PASSWORD included, but were not limited to:

- Analysis of cryptographic systems and implementations
- Analysis of network and security protocols
- Automatic detection of attacks against networks and machines
- Automated tools for source- and compiled-code analysis
- Authentication and authorization of users, systems, and applications
- Bug finding
- Detection of mutability, accessibility, and isolation policy violations
- Identification of denial-of-service attacks
- Input validation
- Intrusion and anomaly detection
- Language-based security
- Operating system security
- Privacy analysis
- Security in heterogeneous and large-scale environments
- Security in the presence of agents and mobile code
- Security policy analysis
- Static analysis for program verification
- Static analysis techniques for soundness, precision, and scalability

3 Program Organization and Participants

3.1 Organizers

PASSWORD 2006 was organized by Marco Pistoia and Francesco Logozzo.

- **Marco Pistoia, Ph. D.** is a Research Staff Member in the Programming Languages and Software Engineering Department at the IBM T. J. Watson Research Center in Hawthorne, New York, USA. He has written ten books, filed thirteen patents, and published numerous papers and journal articles on all areas of Java and program analysis for security. Most recently, he has published his Ph.D. thesis, and has been the lead author of the books *Enterprise Java Security*, published by Addison-Wesley in 2004 (and now available in Chinese), and *Java 2 Network Security*, published by Prentice Hall PTR in 1999. He has published and presented at several conferences worldwide and has been invited to give lectures and teach at several universities and research centers. He received his Ph. D. in Mathematics from Polytechnic University, Brooklyn, New York in May 2005 with a thesis entitled *A Unified Mathematical Model for Stack- and Role-Based Authorization Systems*, and his Master of Science and Bachelor of Science degrees in Mathematics *summa cum laude* from the University of Rome, Italy in 1995, with a research thesis entitled *Theory of Reductive Algebraic Groups and Their Representations*.
- **Francesco Logozzo, Ph. D.** is a Postdoctoral Researcher at the École Normale Supérieure, Paris, France. He graduated from École Polytechnique, Paris, France in June 2004 with a thesis entitled *Modular Static Analysis of Object-oriented Languages*. His Ph. D. advisor was Dr. Radhia Cousot. He was a former student of the Scuola Normale Superiore of Pisa, Italy. He is the author of more than ten papers on static analysis of object-oriented languages. He co-chaired the First Workshop on Abstract Intepretation of Object-Oriented Languages (AIOOL 2005) and the Seventh Workshop of Formal Techniques for Java-like Programs (FTfJP 2005).

3.2 Program Committee Members

The committee members were program analysis and security experts from academia and industry:

- **Sabrina De Capitani Di Vimercati**, University of Milan, Milan, Italy
- **Stephen J. Fink**, IBM T. J. Watson Research Center, Hawthorne, New York, USA
- **Robert J. Flynn**, Polytechnic University, Brooklyn, New York, USA
- **Charles Hymans**, European Aeronautic Defence and Space Company, Paris, France
- **Trent Jaeger**, Pennsylvania State University, University Park, Pennsylvania, USA
- **Francesco Logozzo**, École Normale Supérieure, Paris, France
- **Nasir Memon**, Polytechnic University, Brooklyn, New York, USA

- **Greg Morrisett**, Harvard University, Cambridge, Massachusetts, USA
- **David A. Naumann**, Stevens Institute of Technology, Hoboken, New Jersey, USA
- **Marco Pistoia**, IBM T. J. Watson Research Center, Hawthorne, New York, USA
- **Jan Vitek**, Purdue University, West Lafayette, Indiana, USA
- **Eran Yahav**, IBM T. J. Watson Research Center, Hawthorne, New York, USA
- **Steve Zdancewic**, University of Pennsylvania, Philadelphia, Pennsylvania, USA
- **Xiaolan Zhang**, IBM T. J. Watson Research Center, Hawthorne, New York, USA
- **Roberto Zunino**, University of Pisa, Pisa, Italy

3.3 Participants

Participation to the PASSWORD 2006 workshop was allowed also through remote connection for people unable to attend in person. The list of participants included:

- **Paolina Centonze**, IBM T. J. Watson Research Center, Hawthorne, New York, USA
- **Tzi-cker Chiueh**, Computer Science Department, Stony Brook University, Stony Brook, New York, USA
- **Holger Grandy**, University of Augsburg, Augsburg, Germany
- **Francesco Logozzo**, École Normale Supérieure, Paris, France
- **Yi Lu**, School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia
- **Eric Madelaine**, INRIA, Centre Sophia Antipolis, Sophia Antipolis, France
- **Michael McIntosh**, IBM T. J. Watson Research Center, Hawthorne, New York, USA
- **Nicholas Nguyen**, Department of Informatics, University of Sussex, Brighton, UK
- **Marco Pistoia**, IBM T. J. Watson Research Center, Hawthorne, New York, USA
- **Jan Vitek**, Purdue University, West Lafayette, Indiana, USA
- **Jian Yin**, IBM T. J. Watson Research Center, Hawthorne, New York, USA
- **Xiaolan Zhang**, IBM T. J. Watson Research Center, Hawthorne, New York, USA

4 Summary of Contributions

An important area of application for program analysis is intrusion detection. Host-based intrusion detection systems attempt to identify attacks by discovering program behaviors that deviate from expected patterns. While the idea of performing behavior validation on-the-fly and terminating errant tasks as

soon as a violation is detected is appealing, this presents numerous practical and theoretical challenges. Vitek [20] focuses on automated intrusion detection techniques—techniques that do not require human intervention. Of particular interest are techniques that rely on, or leverage, programming language semantics to find novel ways of detecting attacks [3]. Vitek reviews the main attack models, describes the state of the art in host-based intrusion detection techniques, and concludes with a list of challenges for the research community.

Another interesting area of research is program analysis applied to access control and information flow. Pistoia [13] presents a static analysis framework for statically representing the execution of software programs and the flow of security information in those programs. The results of the analysis can be used to automatically identify security properties of software and evaluate security policies. The analysis can be applied to define or evaluate security policies in both Stack-Based Access Control (SBAC) systems [6], such as Java, Standard Edition (SE) [18] and .NET Common Language Runtime (CLR) [2], and Role-Based Access Control (RBAC) systems [14], such as Java, Enterprise Edition (EE) [17] and CLR. In an SBAC system, when access to a security-sensitive resource is attempted, all the callers on the current stack of execution must prove possession of the right to access that resource. In an RBAC system, access rights typically represent responsibilities inside an organization; rights are aggregated into sets called *roles*, which are then assigned to users and groups. For both SBAC and RBAC system, Pistoia's security-policy inference algorithm assumes that the program execution is represented as a call graph. Security requirements are determined at the authorization points and then propagated backwards in the call graph, performing set unions at the merge points. Another application of this technique is the automatic identification of portions of trusted code that should be made "privilege-asserting" to exempt client code in an SBAC system from unnecessary access-right requirements. A typical example is a library that accesses a configuration or log file. While the library is required to exhibit the necessary file-access right, its clients are not. To achieve this result, the portion of library code that wraps the code responsible for the file-system access can be made privilege-asserting. The problem is how to determine which code should be made privilege-asserting without allowing tainted variables to transfer untrusted data from unauthorzed code into the trusted privilege-asserting code—an integrity problem. Pistoia presents a static-analysis technique based on static program slicing to automatically detect optimum placement of privilege-asserting code without allowing unsanitized tainted data to contaminate trusted code [16].

Resource access control has its origins in the field of operating systems research, but more recently has gained attention for use in programming language research. With the growth of the global computing paradigms, this is particularly true of programming languages to support distributed applications. The essence of resource access control is to enable untrusted programs to execute in a system in a manner prescribed by the system policy. Systems policies represents allowable behaviors by third party code and can range in their sophistication

from simple access matrices to sophisticated RBAC schemes. Policies may not only protect access to privileged resources but also ensure that code respects dynamic behaviors in the form of specified protocols. Nguyen and Rathke [12] do not focus on the policy specification but rather on the policy enforcement mechanism. They make use of type and effect systems to extract effects as models, and to statically check that each thread separately satisfies a global policy before using a reduced state space exploration of the interleaved behavior models. The particular language used is a simply-typed λ-calculus augmented with constructs for thread creation and Java style monitor primitives with synchronized expressions. They make use of these by noticing that within a monitor the thread has mutually exclusive access to system resources and, by virtue of this, it is sufficient to verify the policy state only at strategic points within transactions, such as monitor entry points. This observation helps to reduce the size of the state space when it comes to checking interleavings of threads. Behavior models during model extraction are decorated with summaries which instruct the model checker on how to track the policy state. A summary takes the form of a mapping, which relates the policy state at the current point of execution, backwards, to the state at the beginning of either the current transaction, or, the most recent function call. Their type and effect system performs an interprocedural analysis, which tracks the state of the policy across the boundaries of transactions and function calls. The resource accesses are abstracted from the behavior models and replaced with summaries of policy state changes.

Grandy, Stenzel, and Reif [4] illustrate the mapping of abstract data types to a real programming language during a refinement of security protocol specifications. They show that new security and correctness problems arise on the concrete level and describe a possible solution. Their initial observation is that it is hard to get communication protocols secure on the design level. Research has brought a variety of methods to ensure security, for example model-checking-based approaches or specialized protocol analyzers. Their approach for the analysis of security protocols uses Abstract State Machines and interactive verification with an interactive theorem prover called KIV. However, ensuring security of a protocol on the design level is not sufficient. It is an equally important step to get a correct and secure implementation. The verification of sequential Java programs is supported in KIV with a calculus and semantics. Together with the established refinement approach of Downward Simulation, which has been adapted to Abstract State Machines, the correctness of a protocol implementation can be proven. An implementation is *correct* if it makes the same state changes and has the same input/output behavior as an abstract specification.

With the rapid adoption of the Service Oriented Architecture (SOA), sophisticated software systems are increasingly built by composing coarse-grained service components offered by different organizations through standard Web Services interfaces. The ability to quantify end-to-end security risks of composite software services is extremely valuable to businesses that increasingly rely on Web applications to interact with their customers and partners. Yin, Tang, Zhang, and

McIntosh [21] propose a framework that predicts the probability of end-to-end security breaches of a software service by using a combination of three models:

1. A software security model that describes the probability distribution of security bugs in individual components,
2. A service composition model that describes the interactions of components and the contribution of security bugs in individual components to the overall security of the service, and
3. A hacking exposure model that estimates hackers' knowledge of individual components and hence the probability that a security hole, if exists, may be exploited.

Comparing the system call sequence of a network application against a sandboxing policy is a popular approach to detecting control-hijacking attack, in which the attacker exploits such software vulnerabilities as buffer overflow to take over the control of a victim application and possibly the underlying machine. The long-standing technical barrier to the acceptance of this system call monitoring approach is how to derive accurate sandboxing policies for Windows applications whose source code is unavailable. In fact, many commercial computer security companies take advantage of this fact and fashion a business model in which their users have to pay a subscription fee to receive periodic updates on the application sandboxing policies, much like anti-virus signatures. Li, Lam, and Chiueh [7] describe the design, implementation and evaluation of a sandboxing system called BASS that can automatically extract a highly accurate application-specific sandboxing policy from a Win32/X86 binary, and enforce the extracted policy at run time with low performance overhead. BASS is built on a binary interpretation and analysis infrastructure called BIRD, which can handle application binaries with dynamically linked libraries, exception handlers and multi-threading, and has been shown to work correctly for a large number of commercially distributed Windows-based network applications, including IIS and Apache.

Centonze [1,11] observes that, although RBAC allows restricting access to privileged operations, a deployer may actually intend to restrict access to privileged data. Centonze describes a theoretical foundation for correlating an operation-based RBAC policy with a data-based RBAC policy. Relying on a location-consistency property, Centonze shows how to infer whether an operation-based RBAC policy is equivalent to any data-based RBAC policy.

Centonze and Pistoia discuss the pros and cons of static and dynamic analysis for security. Dynamic analysis is typically unsound. Its precision depends on the completeness of the test case suite used during testing. In absence of a complete suite of test cases, the results of dynamic analysis are going to be incomplete. For an analysis that is aimed at detecting certain security requirements of an application, for example authorization requirements [6,16,14,15,1,10], missing certain requirements can cause run-time failures. On the other hand, static analysis is potentially conservative, and may detect unfeasible security requirements. For example, an authorization-requirement analysis may report unnecessary access-right requirements—a violation of the Principle of Least Privilege [19]. Therefore,

Centonze and Pistoia observe that combining static and dynamic analysis is often crucial when it comes to security analyses, and these two approaches should be combined.

Logozzo describes how automatic inference of class invariants may be relevant to security issues [8,9].

Static program analysis of incomplete programs is still an open issue. Yet, most of the programs that need to be analyzed for security are libraries. Another point of discussion that deserves further research is a better quantification of the tradeoffs between precision and scalability of static program analysis for security.

5 Workshop Material

All the papers and presentations of the PASSWORD 2006 workshop are available at the workshop Web site at `http://research.ihost.com/password/`.

6 IBM Research Best Paper Student Award

The *Security and Privacy* and *Programming Languages and Software Engineering* departments at the IBM T. J. Watson Research Center in Hawthorne, New York, USA jointly sponsored the *IBM Research Best Paper Student Award*. The purpose of this recognition was to encourage talented students to submit papers with high research contents. To qualify for this award, at least one of the lead authors of the paper had to be a full-time undergraduate or graduate university student at the time the paper was submitted. Based on the research quality and originality of the paper *Automatic Application-Specific Sandboxing for Win32/X86 Binaries* [7], the Program Committee unanimously decided to confer the IBM Research Best Paper Student Award on Wei Li, lead author of the paper and Ph. D. student at Stony Brook University, Stony Brook, New York, USA.

References

1. Paolina Centonze, Gleb Naumovich, Stephen J. Fink, and Marco Pistoia. Role-Based Access Control Consistency Validation. In *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA '06)*, Portland, Maine, USA, July 2006.
2. Adam Freeman and Allen Jones. *Programming .NET Security*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, June 2003.
3. Rajeev Gopalakrishna, Eugene H. Spafford, and Jan Vitek. Efficient Intrusion Detection Using Automaton Inlining. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 18–31, Oakland, CA, USA, May 2005. IEEE Computer Society.
4. Holger Grandy, Kurt Stenzel, and Wolfgang Reif. Refinement of Security Protocol Data Types to Java. In *First Program Analysis for Security and Safety Workshop Discussion (PASSWORD 2006), co-located with the Twentieth European Conference on Object-Oriented Programming (ECOOP 2006)*, Nantes, France, July 2006.

5. Christian Hammer, Jens Krinke, and Gregor Snelting. Information Flow Control for Java Based on Path Conditions in Dependence Graphs. In *Proceedings of IEEE International Symposium on Secure Software Engineering*, Arlington, Virginia, USA, 2006.
6. Larry Koved, Marco Pistoia, and Aaron Kershenbaum. Access Rights Analysis for Java. In *Proceedings of the 17th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 359–372, Seattle, WA, USA, November 2002. ACM Press.
7. Wei Li, Lap chung Lam, and Tzi cker Chiueh. Application Specific Sandboxing for Win32/Intel Binaries. In *First Program Analysis for Security and Safety Workshop Discussion (PASSWORD 2006), co-located with the Twentieth European Conference on Object-Oriented Programming (ECOOP 2006)*, Nantes, France, July 2006.
8. Francesco Logozzo. Class-level modular analysis for object oriented languages. In *Proceedings of the 10th Static Analysis Symposium (SAS '03)*, volume 2694 of *Lectures Notes in Computer Science*. Springer-Verlag, June 2003.
9. Francesco Logozzo. Automatic inference of class invariants. In *Proceedings of the 5th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI '04)*, volume 2937 of *Lectures Notes in Computer Science*. Springer-Verlag, January 2004.
10. Gleb Naumovich. A Conservative Algorithm for Computing the Flow of Permissions in Java Programs. In *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA '02)*, pages 33–43, Rome, Italy, July 2002.
11. Gleb Naumovich and Paolina Centonze. Static Analysis of Role-Based Access Control in J2EE Applications. *SIGSOFT Software Engineering Notes*, 29(5):1–10, September 2004.
12. Nicholas Nguyen and Julian Rathke. Typed Static Analysis for Concurrent, Policy-Based, Resource Access Control. In *First Program Analysis for Security and Safety Workshop Discussion (PASSWORD 2006), co-located with the Twentieth European Conference on Object-Oriented Programming (ECOOP 2006)*, Nantes, France, July 2006.
13. Marco Pistoia. Keynote: Static Analysis for Stack-Inspection and Role-Based Access Control Systems. In *First Program Analysis for Security and Safety Workshop Discussion (PASSWORD 2006), co-located with the Twentieth European Conference on Object-Oriented Programming (ECOOP 2006)*, Nantes, France, July 2006.
14. Marco Pistoia, Stephen J. Fink, Robert J. Flynn, and Eran Yahav. When Role Models Have Flaws: Static Validation of Enterprise Security Policies. Technical Report RC24056 (W0609-065), IBM Corporation, Thomas J. Watson Research Center, Yorktown Heights, NY, USA, September 2006.
15. Marco Pistoia and Robert J. Flynn. Interprocedural Analysis for Automatic Evaluation of Role-Based Access Control Policies. Technical Report RC23846 (W0511-020), IBM Corporation, Thomas J. Watson Research Center, Yorktown Heights, NY, USA, November 2005.
16. Marco Pistoia, Robert J. Flynn, Larry Koved, and Vugranam C. Sreedhar. Interprocedural Analysis for Privileged Code Placement and Tainted Variable Detection. In *Proceedings of the 9th European Conference on Object-Oriented Programming*, Glasgow, Scotland, UK, July 2005. Springer-Verlag.
17. Marco Pistoia, Nataraj Nagaratnam, Larry Koved, and Anthony Nadalin. *Enterprise Java Security*. Addison-Wesley, Reading, MA, USA, February 2004.
18. Marco Pistoia, Duane Reller, Deepak Gupta, Milind Nagnur, and Ashok K. Ramani. *Java 2 Network Security*. Prentice Hall PTR, Upper Saddle River, NJ, USA, second edition, August 1999.

19. Jerome H. Saltzer and Michael D. Schroeder. The Protection of Information in Computer Systems. In *Proceedings of the IEEE*, volume 63, pages 1278–1308, September 1975.
20. Jan Vitek. Keynote: Advance in Intrusion Detection. In *First Program Analysis for Security and Safety Workshop Discussion (PASSWORD 2006), co-located with the Twentieth European Conference on Object-Oriented Programming (ECOOP 2006)*, Nantes, France, July 2006.
21. Jian Yin, Chunqiang Tang, Xiaolan Zhang, and Michael McIntosh. On Estimating the Security Risks of Composite Software Services. In *First Program Analysis for Security and Safety Workshop Discussion (PASSWORD 2006), co-located with the Twentieth European Conference on Object-Oriented Programming (ECOOP 2006)*, Nantes, France, July 2006.