Pierangela Samarati
Peter Ryan
Dieter Gollmann
Refik Molva (Eds.)

LNCS 3193

Computer Security – ESORICS 2004

9th European Symposium on Research in Computer Security
Sophia Antipolis, France, September 2004
Proceedings

Springer

Pierangela Samarati Peter Ryan
Dieter Gollmann Refik Molva (Eds.)

Computer Security – ESORICS 2004

9th European Symposium
on Research in Computer Security
Sophia Antipolis, France, September 13 - 15, 2004
Proceedings

Springer

Volume Editors

Pierangela Samarati
Università degli Studi di Milano, Dipartimento di Tecnologie dell'Informazione
Via Bramante 65 - 26013 Crema, Italy
E-mail: samarati@dti.unimi.it

Peter Ryan
University of Newcastle upon Tyne, School of Computing Science
Newcastle upon Tyne, NE1 7RU, UK
E-mail: peter.ryan@ncl.ac.uk

Dieter Gollmann
Technische Universität Hamburg-Harburg
Harburger Schloßstraße 20, 21079 Hamburg, Germany
E-mail: diego@tu-harburg.de

Refik Molva
Institut Eurécom Corporate Communications Department
2229 Route des Crêtes, BP 193, 06904 Sophia Antipolis Cédex, France
E-mail: molva@eurecom.fr

Library of Congress Control Number: 2004111116

CR Subject Classification (1998): E.3, D.4.5, C.2.0, H.2.0, K.6.5, K.4.4

ISSN 0302-9743
ISBN 3-540-22987-6 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springeronline.com

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11315414 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3193

Preface

Foreword from the Program Chairs

These proceedings contain the papers selected for presentation at the 9th European Symposium on Research in Computer Security (ESORICS), held during September 13–15, 2004 in Sophia Antipolis, France.

In response to the call for papers 159 papers were submitted to the conference. These papers were evaluated on the basis of their significance, novelty, and technical quality. Each paper was reviewed by at least three members of the program committee. The program committee meeting was held electronically; there was an intensive discussion over a period of two weeks. Of the papers submitted, 27 were selected for presentation at the conference, giving an acceptance rate lower than 17%. The conference program also included an invited talk.

A workshop like this does not just happen; it depends on the volunteer efforts of a host of individuals. There is a long list of people who volunteered their time and energy to put together the workshop and who deserve special thanks. Thanks to all the members of the program committee, and the external reviewers, for all their hard work in the paper evaluation. Due to the large number of submissions the program committee members were really required to work hard in a short time frame, and we are very thankful to them for the commitment they showed with their active participation in the electronic discussion. We are also very grateful to all those people whose work ensured a smooth organization process: Refik Molva, who served as the General Chair, Marc Dacier, the Sponsoring Chair, Yves Roudier, who served as the Publicity Chair and maintained the Web pages, Sabrina de Capitani di Vimercati, who helped in the review process, Dieter Gollmann, who served as the Publication Chair and collated this volume, and Anne Duflos and Laurence Grammare for helping with the local arrangements.

Last, but certainly not least, our thanks go to all the authors who submitted papers and all the attendees. We hope you find the program stimulating.

Peter Ryan and Pierangela Samarati
(Program Co-chairs)

Foreword from the General Chair

Initially established as the European conference in research on computer security, ESORICS has reached the status of a main international event gathering researchers from all over the world. Taking place in a different European country every other year during its first seven occurrences, it has been a yearly conference since 2003.

ESORICS 2004 was organized by the Institut EURECOM and took place in Sophia Antipolis, France, September 13–15, 2004.

The organization of such an important event required a major effort and we wish to express our sincere appreciation to the organization committee members for their excellent work.

We would like to express our special appreciation to the Program Chairs Pierangela Samarati and Peter Ryan for coming up with a high-quality technical program that was the result of a complex evaluation process they handled very smoothly.

We are also indebted to the Institut EURECOM who not only allowed us and other organization committee members to dedicate considerable time and energy to the organization of this event, but also provided logistic and financial support to host it.

Sophia Antipolis, September 2004 Refik Molva

Program Committee

Vijay Atluri	Rutgers University, USA
Giampaolo Bella	Università di Catania, Italy
Joachim Biskup	Universität Dortmund, Germany
Jan Camenisch	IBM Research, Switzerland
Germano Caronni	Sun Microsystems Laboratories, USA
David Chadwick	University of Salford, UK
Ernesto Damiani	University of Milan, Italy
Sabrina De Capitani di Vimercati	University of Milan, Italy
Yves Deswarte	LAAS-CNRS, France
Alberto Escudero-Pascual	Royal Institute of Technology, Sweden
Csilla Farkas	University of South Carolina, USA
Simon Foley	University College Cork, Ireland
Dieter Gollmann	TU Hamburg-Harburg, Germany
Joshua D. Guttman	MITRE, USA
Sushil Jajodia	George Mason University, USA
Sokratis K. Katsikas	University of the Aegean, Greece
Maciej Koutny	University of Newcastle upon Tyne, UK
Peng Liu	Pennsylvania State University, USA
Javier Lopez	University of Malaga, Spain
Roy Maxion	Carnegie Mellon University, USA
Patrick McDaniel	AT&T Labs-Research, USA
John McHugh	CERT/CC, USA
Catherine A. Meadows	Naval Research Lab, USA
Refik Molva	Institut Eurécom, France
Peng Ning	NC State University, USA
LouAnna Notargiacomo	The MITRE Corporation, USA
Eiji Okamoto	University of Tsukuba, Japan
Stefano Paraboschi	University of Bergamo, Italy
Andreas Pfitzmann	TU Dresden, Germany
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Jean-Jacques Quisquater	Microelectronic Laboratory, Belgium
Peter Ryan (co-chair)	University of Newcastle, UK
Pierangela Samarati (co-chair)	University of Milan, I
Steve Schneider	University of London, UK
Christoph Schuba	Sun Microsystems Inc., USA
Michael Steiner	IBM T.J. Watson Research Lab., USA
Paul Syverson	Naval Research Laboratory, USA
Kymie M. C. Tan	Carnegie Mellon University, USA
Dan Thomsen	Tresys Technology, USA
Moti Yung	Columbia University, USA

Additional Reviewers

Carlos Aguilar, Farid Ahmed, Ben Aziz, Walid Bagga, Endre Bangerter, Lejla Batina, Alex Biryukov, Rainer Böhme, R. Bouroulet, Laurent Bussard, David Byers, Alex Bystrov, Shiping Chen, Ioan Chisalita, Mathieu Ciet, Sebastian Clauß, Stefano Crosta, Roberto Delicata, Alex Dent, Thomas Dübendorfer, Claudiu Duma, Neil Evans, Ulrich Flegel, Elke Franz, Qijun Gu, James Heather, Almut Herzog, Manuel Hilty, John Iliadis, Ryszard Janicki, Mohamed Kaâniche, Ioanna Kantzavelou, Kevin Killourhy, Herbert Klimant, Stefan Köpsell, Spyros Kokolakis, Thomas Kriegelstein, Klaus Kursawe, Costas Lambrinoudakis, Thomas Leineweber, Benoît Libert, Donggang Liu, Pietro Michiardi, Jose A. Montenegro, Fabrice Mourlin, Vincent Nicomette, Melek Onen, Sassa Otenko, Giuseppe Pappalardo, Jörg Parthe, E. Pelz, Olivier Pereira, Thomas Quillinan, Josyula R. Rao, Douglas S. Reeves, Marc Rennhard, Pankaj Rohatgi, Rodrigo Roman, Yves Roudier, Dagmar Schönfeld, Diana Senn, Stefaan Seys, Barbara Sprick, Sandra Steinbrecher, Reto Strobl, Linying Su, Kun Sun, Eduard Turcan, Torben Weibert, Duminda Wijesekera, Sandra Wortmann, Dingbang Xu, Jun Xu, Meng Yu, Wanyu Zang, Christophe Zanon, Homgbin Zhou

Organisation Committee

Refik Molva (General Chair), Yves Roudier (Publicity Chair), Marc Dacier (Sponsoring Chair), Dieter Gollmann (Publication Chair), Anne Duflos (Conference Secretary), and Laurence Grammare (Communications)

ESORICS 2004 was supported by SAP, @sec, and Conseil Régional Provence Alpes Côte d'Azur.

Steering Committee

Elisa Bertino (University of Milan, I), Joachim Biskup (Universität Dortmund, D), Frédéric Cuppens (ENST-Bretagne, F), Marc Dacier (Eurecom, F), Yves Deswarte (LAAS-CNRS, F), Gérard Eizenberg (ONERA, F), Simon Foley (University College Cork, IE), Dieter Gollmann (TU Hamburg-Harburg, D), Franz-Peter Heider (debis IT Security Services, D), Jeremy Jacob (University of York, UK), Sokratis Katsikas (University of the Aegean, GR), Helmut Kurth (atsec, D), Peter Landrock (Cryptomathic, UK), Jean-Jacques Quisquater (UCL, B), Peter Ryan (University of Newcastle, UK: Steering Committee Chair), Pierangela Samarati (University of Milan, I: Steering Committee Vice-Chair), Einar Snekkenes (Gjøvik University College, N), Michael Waidner (IBM Research, CH).

Lecture Notes in Computer Science

For information about Vols. 1–3094

please contact your bookseller or Springer

Vol. 3232: R. Heery, L. Lyon (Eds.), Research and Advanced Technology for Digital Libraries. XV, 528 pages. 2004.

Vol. 3220: J.C. Lester, R.M. Vicari, F. Paragua\c cu (Eds.), Intelligent Tutoring Systems. XXI, 920 pages. 2004.

Vol. 3208: H.J. Ohlbach, S. Schaffert (Eds.), Principles and Practice of Semantic Web Reasoning. VII, 165 pages. 2004.

Vol. 3207: L.T. Jang, M. Guo, G.R. Gao, N.K. Jha, Embedded and Ubiquitous Computing. XX, 1116 pages. 2004.

Vol. 3206: P. Sojka, I. Kopecek, K. Pala (Eds.), Text, Speech and Dialogue. XIII, 667 pages. 2004. (Subseries LNAI).

Vol. 3205: N. Davies, E. Mynatt, I. Siio (Eds.), UbiComp 2004: Ubiquitous Computing. XVI, 452 pages. 2004.

Vol. 3203: J. Becker, M. Platzner, S. Vernalde (Eds.), Field Programmable Logic and Application. XXX, 1198 pages. 2004.

Vol. 3198: G.-J. de Vreede, L.A. Guerrero, G. Marín Raventós (Eds.), Groupware: Design, Implementation and Use. XI, 378 pages. 2004.

Vol. 3194: R. Camacho, R. King, A. Srinivasan (Eds.), Inductive Logic Programming. XI, 361 pages. 2004. (Subseries LNAI).

Vol. 3193: P. Samarati, P. Ryan, D. Gollmann, R. Molva (Eds.), Computer Security – ESORICS 2004. X, 457 pages. 2004.

Vol. 3192: C. Bussler, D. Fensel (Eds.), Artificial Intelligence: Methodology, Systems, and Applications. XIII, 522 pages. 2004. (Subseries LNAI).

Vol. 3189: P.-C. Yew, J. Xue (Eds.), Advances in Computer Systems Architecture. XVII, 598 pages. 2004.

Vol. 3186: Z. Bellahsène, T. Milo, M. Rys, D. Suciu, R. Unland (Eds.), Database and XML Technologies. X, 235 pages. 2004.

Vol. 3184: S. Katsikas, J. Lopez, G. Pernul (Eds.), Trust and Privacy in Digital Business. XI, 299 pages. 2004.

Vol. 3183: R. Traunmüller (Ed.), Electronic Government. XIX, 583 pages. 2004.

Vol. 3182: K. Bauknecht, M. Bichler, B. Pröll (Eds.), E-Commerce and Web Technologies. XI, 370 pages. 2004.

Vol. 3181: Y. Kambayashi, M. Mohania, W. W\"oß (Eds.), Data Warehousing and Knowledge Discovery. XIV, 412 pages. 2004.

Vol. 3180: F. Galindo, M. Takizawa, R. Traunmüller (Eds.), Database and Expert Systems Applications. XXI, 945 pages. 2004.

Vol. 3178: W. Jonker, M. Petkovic (Eds.), Secure Data Management. VIII, 219 pages. 2004.

Vol. 3177: Z.R. Yang, H. Yin, R. Everson (Eds.), Intelligent Data Engineering and Automated Learning – IDEAL 2004. XVIII, 852 pages. 2004.

Vol. 3175: C.E. Rasmussen, H.H. Bülthoff, B. Schölkopf, M.A. Giese (Eds.), Pattern Recognition. XVIII, 581 pages. 2004.

Vol. 3174: F. Yin, J. Wang, C. Guo (Eds.), Advances in Neural Networks - ISNN 2004. XXXV, 1021 pages. 2004.

Vol. 3172: M. Dorigo, M. Birattari, C. Blum, L. M.Gambardella, F. Mondada, T. Stützle (Eds.), Ant Colony, Optimization and Swarm Intelligence. XII, 434 pages. 2004.

Vol. 3170: P. Gardner, N. Yoshida (Eds.), CONCUR 2004 - Concurrency Theory. XIII, 529 pages. 2004.

Vol. 3166: M. Rauterberg (Ed.), Entertainment Computing – ICEC 2004. XXIII, 617 pages. 2004.

Vol. 3163: S. Marinai, A. Dengel (Eds.), Document Analysis Systems VI. XII, 564 pages. 2004.

Vol. 3159: U. Visser, Intelligent Information Integration for the Semantic Web. XIV, 150 pages. 2004. (Subseries LNAI).

Vol. 3158: I. Nikolaidis, M. Barbeau, E. Kranakis (Eds.), Ad-Hoc, Mobile, and Wireless Networks. IX, 344 pages. 2004.

Vol. 3157: C. Zhang, H. W. Guesgen, W.K. Yeap (Eds.), PRICAI 2004: Trends in Artificial Intelligence. XX, 1023 pages. 2004. (Subseries LNAI).

Vol. 3156: M. Joye, J.-J. Quisquater (Eds.), Cryptographic Hardware and Embedded Systems - CHES 2004. XIII, 455 pages. 2004.

Vol. 3155: P. Funk, P.A. González Calero (Eds.), Advances in Case-Based Reasoning. XIII, 822 pages. 2004. (Subseries LNAI).

Vol. 3154: R.L. Nord (Ed.), Software Product Lines. XIV, 334 pages. 2004.

Vol. 3153: J. Fiala, V. Koubek, J. Kratochvíl (Eds.), Mathematical Foundations of Computer Science 2004. XIV, 902 pages. 2004.

Vol. 3152: M. Franklin (Ed.), Advances in Cryptology – CRYPTO 2004. XI, 579 pages. 2004.

Vol. 3150: G.-Z. Yang, T. Jiang (Eds.), Medical Imaging and Augmented Reality. XII, 378 pages. 2004.

Vol. 3149: M. Danelutto, M. Vanneschi, D. Laforenza (Eds.), Euro-Par 2004 Parallel Processing. XXXIV, 1081 pages. 2004.

Vol. 3148: R. Giacobazzi (Ed.), Static Analysis. XI, 393 pages. 2004.

Vol. 3146: P. Érdi, A. Esposito, M. Marinaro, S. Scarpetta (Eds.), Computational Neuroscience: Cortical Dynamics. XI, 161 pages. 2004.

Vol. 3144: M. Papatriantafilou, P. Hunel (Eds.), Principles of Distributed Systems. XI, 246 pages. 2004.

Vol. 3143: W. Liu, Y. Shi, Q. Li (Eds.), Advances in Web-Based Learning – ICWL 2004. XIV, 459 pages. 2004.

Vol. 3142: J. Diaz, J. Karhumäki, A. Lepistö, D. Sannella (Eds.), Automata, Languages and Programming. XIX, 1253 pages. 2004.

Vol. 3140: N. Koch, P. Fraternali, M. Wirsing (Eds.), Web Engineering. XXI, 623 pages. 2004.

Vol. 3139: F. Iida, R. Pfeifer, L. Steels, Y. Kuniyoshi (Eds.), Embodied Artificial Intelligence. IX, 331 pages. 2004. (Subseries LNAI).

Vol. 3138: A. Fred, T. Caelli, R.P.W. Duin, A. Campilho, D.d. Ridder (Eds.), Structural, Syntactic, and Statistical Pattern Recognition. XXII, 1168 pages. 2004.

Vol. 3137: P. De Bra, W. Nejdl (Eds.), Adaptive Hypermedia and Adaptive Web-Based Systems. XIV, 442 pages. 2004.

Vol. 3136: F. Meziane, E. Métais (Eds.), Natural Language Processing and Information Systems. XII, 436 pages. 2004.

Vol. 3134: C. Zannier, H. Erdogmus, L. Lindstrom (Eds.), Extreme Programming and Agile Methods - XP/Agile Universe 2004. XIV, 233 pages. 2004.

Vol. 3133: A.D. Pimentel, S. Vassiliadis (Eds.), Computer Systems: Architectures, Modeling, and Simulation. XIII, 562 pages. 2004.

Vol. 3132: B. Demoen, V. Lifschitz (Eds.), Logic Programming. XII, 480 pages. 2004.

Vol. 3131: V. Torra, Y. Narukawa (Eds.), Modeling Decisions for Artificial Intelligence. XI, 327 pages. 2004. (Subseries LNAI).

Vol. 3130: A. Syropoulos, K. Berry, Y. Haralambous, B. Hughes, S. Peter, J. Plaice (Eds.), TeX, XML, and Digital Typography. VIII, 265 pages. 2004.

Vol. 3129: Q. Li, G. Wang, L. Feng (Eds.), Advances in Web-Age Information Management. XVII, 753 pages. 2004.

Vol. 3128: D. Asonov (Ed.), Querying Databases Privately. IX, 115 pages. 2004.

Vol. 3127: K.E. Wolff, H.D. Pfeiffer, H.S. Delugach (Eds.), Conceptual Structures at Work. XI, 403 pages. 2004. (Subseries LNAI).

Vol. 3126: P. Dini, P. Lorenz, J.N.d. Souza (Eds.), Service Assurance with Partial and Intermittent Resources. XI, 312 pages. 2004.

Vol. 3125: D. Kozen (Ed.), Mathematics of Program Construction. X, 401 pages. 2004.

Vol. 3124: J.N. de Souza, P. Dini, P. Lorenz (Eds.), Telecommunications and Networking - ICT 2004. XXVI, 1390 pages. 2004.

Vol. 3123: A. Belz, R. Evans, P. Piwek (Eds.), Natural Language Generation. X, 219 pages. 2004. (Subseries LNAI).

Vol. 3122: K. Jansen, S. Khanna, J.D.P. Rolim, D. Ron (Eds.), Approximation, Randomization, and Combinatorial Optimization. IX, 428 pages. 2004.

Vol. 3121: S. Nikoletseas, J.D.P. Rolim (Eds.), Algorithmic Aspects of Wireless Sensor Networks. X, 201 pages. 2004.

Vol. 3120: J. Shawe-Taylor, Y. Singer (Eds.), Learning Theory. X, 648 pages. 2004. (Subseries LNAI).

Vol. 3118: K. Miesenberger, J. Klaus, W. Zagler, D. Burger (Eds.), Computer Helping People with Special Needs. XXIII, 1191 pages. 2004.

Vol. 3116: C. Rattray, S. Maharaj, C. Shankland (Eds.), Algebraic Methodology and Software Technology. XI, 569 pages. 2004.

Vol. 3114: R. Alur, D.A. Peled (Eds.), Computer Aided Verification. XII, 536 pages. 2004.

Vol. 3113: J. Karhumäki, H. Maurer, G. Paun, G. Rozenberg (Eds.), Theory Is Forever. X, 283 pages. 2004.

Vol. 3112: H. Williams, L. MacKinnon (Eds.), Key Technologies for Data Management. XII, 265 pages. 2004.

Vol. 3111: T. Hagerup, J. Katajainen (Eds.), Algorithm Theory - SWAT 2004. XI, 506 pages. 2004.

Vol. 3110: A. Juels (Ed.), Financial Cryptography. XI, 281 pages. 2004.

Vol. 3109: S.C. Sahinalp, S. Muthukrishnan, U. Dogrusoz (Eds.), Combinatorial Pattern Matching. XII, 486 pages. 2004.

Vol. 3108: H. Wang, J. Pieprzyk, V. Varadharajan (Eds.), Information Security and Privacy. XII, 494 pages. 2004.

Vol. 3107: J. Bosch, C. Krueger (Eds.), Software Reuse: Methods, Techniques and Tools. XI, 339 pages. 2004.

Vol. 3106: K.-Y. Chwa, J.I. Munro (Eds.), Computing and Combinatorics. XIII, 474 pages. 2004.

Vol. 3105: S. Göbel, U. Spierling, A. Hoffmann, I. Iurgel, O. Schneider, J. Dechau, A. Feix (Eds.), Technologies for Interactive Digital Storytelling and Entertainment. XVI, 304 pages. 2004.

Vol. 3104: R. Kralovic, O. Sykora (Eds.), Structural Information and Communication Complexity. X, 303 pages. 2004.

Vol. 3103: K. Deb, e. al. (Eds.), Genetic and Evolutionary Computation – GECCO 2004. XLIX, 1439 pages. 2004.

Vol. 3102: K. Deb, e. al. (Eds.), Genetic and Evolutionary Computation – GECCO 2004. L, 1445 pages. 2004.

Vol. 3101: M. Masoodian, S. Jones, B. Rogers (Eds.), Computer Human Interaction. XIV, 694 pages. 2004.

Vol. 3100: J.F. Peters, A. Skowron, J.W. Grzymała-Busse, B. Kostek, R.W. Świniarski, M.S. Szczuka (Eds.), Transactions on Rough Sets I. X, 405 pages. 2004.

Vol. 3099: J. Cortadella, W. Reisig (Eds.), Applications and Theory of Petri Nets 2004. XI, 505 pages. 2004.

Vol. 3098: J. Desel, W. Reisig, G. Rozenberg (Eds.), Lectures on Concurrency and Petri Nets. VIII, 849 pages. 2004.

Vol. 3097: D. Basin, M. Rusinowitch (Eds.), Automated Reasoning. XII, 493 pages. 2004. (Subseries LNAI).

Vol. 3096: G. Melnik, H. Holz (Eds.), Advances in Learning Software Organizations. X, 173 pages. 2004.

Vol. 3095: C. Bussler, D. Fensel, M.E. Orlowska, J. Yang (Eds.), Web Services, E-Business, and the Semantic Web. X, 147 pages. 2004.

Table of Contents

Incorporating Dynamic Constraints in the Flexible Authorization Framework 1
Shiping Chen, Duminda Wijesekera, and Sushil Jajodia

Access-Condition-Table-Driven Access Control for XML Databases 17
Naizhen Qi (Naishin Seki) and Michiharu Kudo

An Algebra for Composing Enterprise Privacy Policies 33
Michael Backes, Markus Dürmuth, and Rainer Steinwandt

Deriving, Attacking and Defending the GDOI Protocol 53
Catherine Meadows and Dusko Pavlovic

Better Privacy for Trusted Computing Platforms 73
Jan Camenisch

A Cryptographically Sound Dolev-Yao Style Security Proof of the Otway-Rees Protocol 89
Michael Backes

A Formalization of Anonymity and Onion Routing 109
S. Mauw, J.H.S. Verschuren, and E.P. de Vink

Breaking Cauchy Model-Based JPEG Steganography with First Order Statistics 125
Rainer Böhme and Andreas Westfeld

Comparison Between Two Practical Mix Designs 141
Claudia Díaz, Len Sassaman, and Evelyne Dewitte

Signature Bouquets: Immutability for Aggregated/Condensed Signatures .. 160
Einar Mykletun, Maithili Narasimha, and Gene Tsudik

Towards a Theory of Data Entanglement 177
James Aspnes, Joan Feigenbaum, Aleksandr Yampolskiy, and Sheng Zhong

Portable and Flexible Document Access Control Mechanisms 193
Mikhail Atallah and Marina Bykova

Possibilistic Information Flow Control in the Presence of Encrypted Communication 209
Dieter Hutter and Axel Schairer

Information Flow Control Revisited:
Noninfluence = Noninterference + Nonleakage 225
David von Oheimb

Security Property Based Administrative Controls 244
Jon A. Solworth and Robert H. Sloan

A Vector Model of Trust for Developing Trustworthy Systems 260
Indrajit Ray and Sudip Chakraborty

Parameterized Authentication 276
Michael J. Covington, Mustaque Ahamad, Irfan Essa, and H. Venkateswaran

Combinatorial Design of Key Distribution Mechanisms
for Wireless Sensor Networks 293
Seyit A. Çamtepe and Bülent Yener

Hindering Eavesdropping via IPv6 Opportunistic Encryption 309
Claude Castelluccia, Gabriel Montenegro, Julien Laganier, and Christoph Neumann

On the Role of Key Schedules in Attacks on Iterated Ciphers 322
Lars R. Knudsen and John E. Mathiassen

A Public-Key Encryption Scheme with Pseudo-random Ciphertexts 335
Bodo Möller

A Host Intrusion Prevention System for Windows Operating Systems 352
Roberto Battistoni, Emanuele Gabrielli, and Luigi V. Mancini

Re-establishing Trust in Compromised Systems:
Recovering from Rootkits That Trojan the System Call Table 369
Julian B. Grizzard, John G. Levine, and Henry L. Owen

ARCHERR: Runtime Environment Driven Program Safety 385
Ramkumar Chinchani, Anusha Iyer, Bharat Jayaraman, and Shambhu Upadhyaya

Sets, Bags, and Rock and Roll –
Analyzing Large Data Sets of Network Data 407
John M^c^Hugh

Redundancy and Diversity in Security 423
Bev Littlewood and Lorenzo Strigini

Discovering Novel Attack Strategies from INFOSEC Alerts 439
Xinzhou Qin and Wenke Lee

Author Index 457

Incorporating Dynamic Constraints in the Flexible Authorization Framework

Shiping Chen, Duminda Wijesekera, and Sushil Jajodia

Center for Secure Information Systems, George Mason University,
Fairfax, VA 22030-4444, USA
{schen3,dwijesek,jajodia}@gmu.edu

Abstract. Constraints are an integral part of access control policies. Depending upon their time of enforcement, they are categorized as static or dynamic; static constraints are enforced during the policy compilation time, and the dynamic constraints are enforced during run time. While there are several logic-based access control policy frameworks, they have a limited power in expressing and enforcing constraints (especially the dynamic constraints). We propose dynFAF, a constraint logic programming based approach for expressing and enforcing constraints. To make it more concrete, we present our approach as an extension to the *flexible authorization framework (FAF)* of Jajodia et al. [17]. We show that dynFAF satisfies standard safety and liveliness properties of a safety conscious software system.

1 Introduction

Constraints are a powerful mechanism for specifying high-level organizational policies [21]. Accordingly, most access control policies contain constraints, usually categorized as static or dynamic, referring to their time of enforcement by the access controller. As examples, consider the following two constraints: *an undergraduate student should not be permitted to grade qualifying examinations at the PhD level*, and *an author should not be allowed to review his/her own manuscript.* The first constraint can be enforced by prohibiting *grading* permissions on PhD examinations for every undergraduate student, thereby making it statically enforceable. The second constraint requires an access controller to evaluate if the requesting subject is also an author of the document to be reviewed when the request is made. This constraint cannot be evaluated prior to the request, making the constraint dynamically, but not statically, enforceable. Enforcing the latter kind of constraints over access permissions expressed as Horn clauses is the subject matter of this paper.

The past decade has seen several logic based *flexible* access control policy specification frameworks. Woo and Lam [25] propose the use of *default logic* for representing access control policies. To overcome the problems of undecidability and non-implementability that arise in Woo and Lam's approach, Jajodia et al. [17] propose an access control policy specification framework (FAF) based on a restricted class of logic programs, viz., those that are *locally stratifiable.*

P. Samarati et al. (Eds.): ESORICS 2004, LNCS 3193, pp. 1–16, 2004.

Bertino et al.'s framework [6] uses *C-Datalog* to express various access control policies [6]. Barker and Stuckey use *constraint logic programming* for multi-policy specification and implementation [4].

Although they are powerful in expressing access control policies, these frameworks have a limited power in specifying and enforcing constraints. For instance, Jajodia et al. [17] use an integrity rule (a logic rule with an `error()` head) to specify constraints. Barker and Stuckey [4] define some special consistency checking rules (with head of predicates *inconsistent_ssd, inconsistent_dsd*) to encode the separation of duty constraints. However, the enforcement of the constraints is left outside the framework; as a result, dynamic constraints cannot be enforced in the access control engine properly.

To overcome these drawbacks, we propose a constraint logic programming based approach to express and enforce dynamic constraints. To make it more concrete, we present our approach as an extension to *Flexible Authorization Framework (FAF)* proposed by Jajodia et al. [17]. Our approach is applicable to other logic based access control frameworks because our constraint specification and enforcement modules are built on top of the existing framework modules. The proposed extension, called *dynFAF*, has two extra modules. First module, the *integrity constraint specification and derivation module (ISM)*, is responsible for specifying the atomic conflicts and deriving all possible complex conflicts in the system that represent the constraints. The second module, the *dynamic access grant module (DAM)*, is responsible for enforcing the constraints specified by ISM dynamically. In addition, DAM allows subjects to *relinquish* permissions that were granted to them. In our design, FAF composes the *static* component, and ISM and DAM compose the *dynamic* component of dynFAF.

We show that dynFAF satisfies safety and liveliness properties granting any access that does not violate derivable constraint, and denying those that do. Because FAF policies are stratified logic programs, they have a stable model semantics [14]. Our constraint specification policies taken together with FAF policies also have a local stratification, thereby admitting a stable model that extends the former. In addition, proposed dynamic access grant module enriches the syntax of the former by having yet another layer of constrained logic programs, that taken as a whole extends the former local stratification. Therefore, a dynFAF specification admits a well-founded model in which some predicate may result in an *undefined* truth in addition to the usual *true* or *false* values; however, our design ensures that any access requested of dynFAF returns only *true* or *false.*

The remainder of the paper is structured as follows. Section 2 contains a brief overview of FAF, followed by a description of its limitations. Section 3 presents the architecture of dynFAF, including the descriptions of ISM and DAM modules. Section 4 presents the semantics of dynFAF syntax. Section 5 shows that dynFAF satisfies the traditional safety and liveliness properties, and that the semantics of the granting and relinquishing access rights are enforced properly. Section 6 compares our work to those of others. Section 7 concludes the paper.

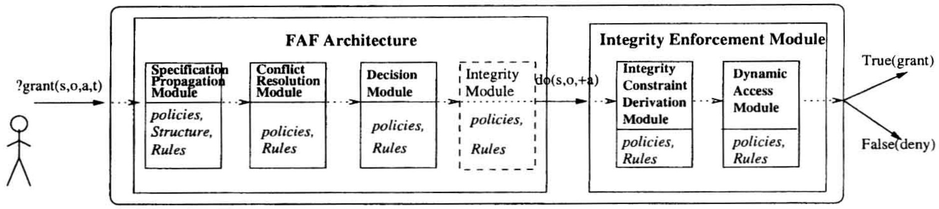

Fig. 1. dynFAF Architecture

2 Overview of FAF

FAF [17] is a logic-based framework to specify authorizations in the form of rules, based on four stages (each stage corresponds to a module) that are applied in a sequence, as shown in FAF Architecture part of Figure 1. In the first stage of the sequence, some basic facts such as authorization subject and object hierarchies (for example directory structures) and a set of authorizations along with rules to derive additional authorizations are given. The intent of this stage is to specify basic authorizations and use structural properties to derive new authorizations. Hence, they are called *specification and propagation policies.* Although propagation policies are flexible and expressive, they may result in *over-specification* resulting in conflicting authorizations. FAF uses *conflict resolution policies* to weed out these in the second stage. At the third stage, *decision policies* are applied in order to ensure the completeness of authorizations. The last stage consists of checking for integrity constraints, where all authorizations that violate integrity constraints will be denied. In addition, FAF ensures that every access request is either granted or rejected, thereby providing a built-in completeness property.

FAF syntax consists of terms that are built from constants and variables (no function symbols) and they belong to four sorts, viz., subjects, objects, actions, and roles. We use the capital letters with subscripts such as X_s, Y_o, X_a, and X_r to denote the respective variables belonging to them, and lower case letters such as `s`, `a`, `o`, and `r` for constants. FAF has the following predicates:

1. A ternary predicate `cando`(s, o, a), representing grantable or deniable requests (depending on the sign associated with the action) where s, o, and a are subject, object, and signed action terms, respectively.
2. A ternary predicate `dercando`(s, o, a), with the same arguments as `cando`. The predicate `dercando` represents authorizations derived by the system using inference rules modus ponens plus rule of stratified negation [2].
3. A ternary predicate `do`, with the same arguments as `cando`, representing the access control decisions made by FAF.
4. A 4-ary predicate `done`(s, o, a, t), meaning subject s has executed action a on object o at time t, t is a natural number.
5. Two binary predicate symbols `over`$_{AS}$ and `over`$_{AO}$, each taking two subject and object terms as arguments two object terms respectively.

6. A predicate symbol without argument, error, symbolizing violation of an integrity constraint, where a rule with an error head must not have a satisfiable body.
7. Other terms and predicates necessary to model specific applications. For example, constants AOH, ASH denote object and subject hierarchies with in, where in(x,y,H) denotes that x $\leq$ y in hierarchy H. For example, we denote the fact that usr\local is below usr in the object hierarchy AOH by in(usr\local, usr, AOH).

Because any FAF specification is a locally stratified logic program, it has a unique stable model [14], and a well-founded model (as in Gelfond and Lifshitz). In addition, the well-founded model coincides with the unique stable model [3, 17]. Furthermore, the unique stable model can be computed in quadratic time data complexity [24]. See [17] for details.

2.1 Limitations of FAF

In its current design, FAF has these limitations. First, FAF expresses constraints using integrity rules of the kind error() $\leftarrow L_i, \ldots, L_n$ where error is an argument-less predicate that should not be valid in any model and $L_i, \ldots, L_n$ are other literals. Ensuring that granted permissions do not imply error is enforced outside of the logical machinery of FAF. Accordingly, it is not within the logical inference engine of FAF to avoid constraint violations. To elaborate further, FAF evaluates an access request as a query ?$\mathtt{do}(s, o, a)$, and ensures the completeness and consistency of the specification, consisting of the rules from the first three modules, by ensuring that one and only one of $\mathtt{do}(s, o, +a)$ or $\mathtt{do}(s, o, -a)$ evaluates to *true*. However, it is possible that both $(s, o, +a)$ and $(s, o, -a)$ could be rejected by the integrity enforcement module making the eventual outcomes incomplete, as the inference rules are unaware of rejections by the integrity enforcement module. Thus, the integrity enforcement needs to be brought inside the reasoning engine, as done in dynFAF.

Second, FAF does not allow constraint derivation, although this occurs in practice. For example, role based access control (RBAC) models have conflicting roles (say, r_1 and r_2) where a single subject assuming them simultaneously violate the policy. In addition, an application may want to declare junior roles of conflicting roles to be conflicting. That is, if roles r_3, r_4 are junior to roles r_1 and r_2, respectively, by satisfying the constrains $r_3 \leq r_1$ and $r_4 \leq r_2$, then no subject should be allowed to assume r_3 and r_4 simultaneously. Our extension facilitates constraint derivation.

Third, in FAF each access request is either granted or denied on its own merit. But some applications may want *controlled (don't care) nondeterminism*. For example, a subject is allowed to assume role r_1 or role r_2, but not both, with no preference for either. If we are to accept a unique stable model, then either r_1 or r_2, but not both can be assumed by the subject. dynFAF facilitates controlled nondeterminism in granting permissions.

Finally, and importantly, FAF does not consider an evolving access control system. That is, if $\texttt{do}(o, s, +a)$ is in the stable model of an authorization specification, the access request (o, s, a) is always permitted. In practice, an authorization may be relinquished by the user or the administrator some time after it is authorized (e.g., in workflow management systems). Consequently, some authorization that is not allowed at a point of time because of constraints restriction may be allowed later if the conflicting authorizations are relinquished. Notice that inserting a negative authorization $\texttt{cando}(o, s, -a)$ does not model this situation. The soon to be described *dynamic access grant module* of dynFAF provides this functionality.

3 dynFAF: A Constraint Logic Programming Based Extension to FAF

To address the problems described in the previous section, FAF is enhanced by adding an *integrity constraint specification and derivation module* (ISM) and a *dynamic access grant module* (DAM), grants accesses that avoid those conflicts. FAF enhanced with these two modules are referred to as *dynFAF*, shown in Figure 1. An access request for (s, o, a) is modeled in dynFAF as a predicate instance $\texttt{request}(s, o, \pm a, t)$, where $(s, o, +a, t)$ is a request to obtain permission for (s, o, a) at time t (equals to a query ?**grant**(s,o,a,t)) and $(s, o, -a, t)$ is a request to relinquish already existing permission for (s, o, a) at time t (equals to a query ?**relinquish**(s,o,a,t)). dynFAF ensures that any request for permission is statically granted by FAF, and granting it does not violate any constraints specified by ISM.

3.1 Integrity Constraint Specification and Derivation Module (ISM)

ISM uses two predicates:

- A binary predicate symbol, `conflict`, where `conflict(x,y)` is an atomic conflict where `x` and `y` can be either (subject,object,action) triples or object, action, or subject terms.
- A binary predicate symbol `derConflict`. `derConflict` has the same arguments as `conflict`. `derConflict(x,y)` is true iff `x,y` constitute a derivable conflict.

We use Horn clause rules to specify atomic conflicts and derivable conflicts based on already defined atomic conflicts. The corresponding rules are called *conflict rules* and *conflict derivation rules*, respectively. Each conflict rule has a `conflict` predicate as its head and some `cando`, `dercando`, `done` or `rel`-literals as its body.