Christopher Kruegel
Richard Lippmann
Andrew Clark (Eds.)

LNCS 4637

Recent Advances in Intrusion Detection

10th International Symposium, RAID 2007
Gold Coast, Australia, September 2007
Proceedings

Springer

TP309-53
R149
2007

Christopher Kruegel Richard Lippmann
Andrew Clark (Eds.)

Recent Advances in Intrusion Detection

10th International Symposium, RAID 2007
Gold Coast, Australia, September 5-7, 2007
Proceedings

华南理工大学
图书馆藏书

Springer

E2007003374

Volume Editors

Christopher Kruegel
Secure Systems Lab
Technical University of Vienna
1040 Vienna, Austria
E-mail: chris@seclab.tuwien.ac.at

Richard Lippmann
Lincoln Laboratory
Massachusetts Institute of Technology
Lexington, MA 02420-9108, USA
E-mail: lippmann@ll.mit.edu

Andrew Clark
Information Security Institute
Queensland University of Technology
Brisbane, QLD 4001, Australia
E-mail: a.clark@qut.edu.au

Library of Congress Control Number: 2007932747

CR Subject Classification (1998): K.6.5, K.4, E.3, C.2, D.4.6

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN	0302-9743
ISBN-10	3-540-74319-7 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-74319-4 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2007
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12110704 06/3180 5 4 3 2 1 0

Lecture Notes in Computer Science 4637

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK
Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA
Josef Kittler
University of Surrey, Guildford, UK
Jon M. Kleinberg
Cornell University, Ithaca, NY, USA
Friedemann Mattern
ETH Zurich, Switzerland
John C. Mitchell
Stanford University, CA, USA
Moni Naor
Weizmann Institute of Science, Rehovot, Israel
Oscar Nierstrasz
University of Bern, Switzerland
C. Pandu Rangan
Indian Institute of Technology, Madras, India
Bernhard Steffen
University of Dortmund, Germany
Madhu Sudan
Massachusetts Institute of Technology, MA, USA
Demetri Terzopoulos
University of California, Los Angeles, CA, USA
Doug Tygar
University of California, Berkeley, CA, USA
Moshe Y. Vardi
Rice University, Houston, TX, USA
Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

Lecture Notes in Computer Science

For information about Vols. 1–4556

please contact your bookseller or Springer

Vol. 4697: L. Choi, Y. Paek, S. Cho (Eds.), Advances in Computer Systems Architecture. XIII, 400 pages. 2007.

Vol. 4682: D.-S. Huang, L. Heutte, M. Loog (Eds.), Advanced Intelligent Computing Theories and Applications. XXVII, 1373 pages. 2007. (Sublibrary LNAI).

Vol. 4681: D.-S. Huang, L. Heutte, M. Loog (Eds.), Advanced Intelligent Computing Theories and Applications. XXVI, 1379 pages. 2007.

Vol. 4673: W.G. Kropatsch, M. Kampel, A. Hanbury (Eds.), Computer Analysis of Images and Patterns. XX, 1006 pages. 2007.

Vol. 4671: V. Malyshkin (Ed.), Parallel Computing Technologies. XIV, 635 pages. 2007.

Vol. 4660: S. Džeroski, J. Todorovski (Eds.), Computational Discovery of Scientific Knowledge. X, 327 pages. 2007. (Sublibrary LNAI).

Vol. 4651: F. Azevedo, P. Barahona, F. Fages, F. Rossi (Eds.), Recent Advances in Constraints. VIII, 185 pages. 2007. (Sublibrary LNAI).

Vol. 4647: R. Martin, M. Sabin, J. Winkler (Eds.), Mathematics of Surfaces XII. IX, 509 pages. 2007.

Vol. 4645: R. Giancarlo, S. Hannenhalli (Eds.), Algorithms in Bioinformatics. XIII, 432 pages. 2007. (Sublibrary LNBI).

Vol. 4643: M.-F. Sagot, M.E.M.T. Walter (Eds.), Advances in Bioinformatics and Computational Biology. XII, 177 pages. 2007. (Sublibrary LNBI).

Vol. 4639: E. Csuhaj-Varj'u, Z. Ésik (Eds.), Fundamentals of Computation Theory. XIV, 508 pages. 2007.

Vol. 4637: C. Kruegel, R. Lippmann, A. Clark (Eds.), Recent Advances in Intrusion Detection. XII, 337 pages. 2007.

Vol. 4635: B. Kokinov, D.C. Richardson, T.R. Roth-Berghofer, L. Vieu (Eds.), Modeling and Using Context. XIV, 574 pages. 2007. (Sublibrary LNAI).

Vol. 4634: H.R. Nielson, G. Filé (Eds.), Static Analysis. XI, 469 pages. 2007.

Vol. 4633: M. Kamel, A. Campilho (Eds.), Image Analysis and Recognition. XII, 1312 pages. 2007.

Vol. 4632: R. Alhajj, H. Gao, X. Li, J. Li, O.R. Zaïane (Eds.), Advanced Data Mining and Applications. XV, 634 pages. 2007. (Sublibrary LNAI).

Vol. 4628: L.N. de Castro, F.J. Von Zuben, H. Knidel (Eds.), Artificial Immune Systems. XII, 438 pages. 2007.

Vol. 4627: M. Charikar, K. Jansen, O. Reingold, J.D.P. Rolim (Eds.), Approximation, Randomization, and Combinatorial Optimization. XII, 626 pages. 2007.

Vol. 4626: R.O. Weber, M.M. Richter (Eds.), Case-Based Reasoning Research and Development. XIII, 534 pages. 2007. (Sublibrary LNAI).

Vol. 4624: T. Mossakowski, U. Montanari, M. Haveraaen (Eds.), Algebra and Coalgebra in Computer Science. XI, 463 pages. 2007.

Vol. 4622: A. Menezes (Ed.), Advances in Cryptology - CRYPTO 2007. XIV, 631 pages. 2007.

Vol. 4619: F. Dehne, J.-R. Sack, N. Zeh (Eds.), Algorithms and Data Structures. XVI, 662 pages. 2007.

Vol. 4618: S.G. Akl, C.S. Calude, M.J. Dinneen, G. Rozenberg, H.T. Wareham (Eds.), Unconventional Computation. X, 243 pages. 2007.

Vol. 4617: V. Torra, Y. Narukawa, Y. Yoshida (Eds.), Modeling Decisions for Artificial Intelligence. XII, 502 pages. 2007. (Sublibrary LNAI).

Vol. 4616: A. Dress, Y. Xu, B. Zhu (Eds.), Combinatorial Optimization and Applications. XI, 390 pages. 2007.

Vol. 4615: R. de Lemos, C. Gacek, A. Romanovsky (Eds.), Architecting Dependable Systems IV. XIV, 435 pages. 2007.

Vol. 4613: F.P. Preparata, Q. Fang (Eds.), Frontiers in Algorithmics. XI, 348 pages. 2007.

Vol. 4612: I. Miguel, W. Ruml (Eds.), Abstraction, Reformulation, and Approximation. XI, 418 pages. 2007. (Sublibrary LNAI).

Vol. 4611: J. Indulska, J. Ma, L.T. Yang, T. Ungerer, J. Cao (Eds.), Ubiquitous Intelligence and Computing. XXIII, 1257 pages. 2007.

Vol. 4610: B. Xiao, L.T. Yang, J. Ma, C. Muller-Schloer, Y. Hua (Eds.), Autonomic and Trusted Computing. XVIII, 571 pages. 2007.

Vol. 4609: E. Ernst (Ed.), ECOOP 2007 – Object-Oriented Programming. XIII, 625 pages. 2007.

Vol. 4608: H.W. Schmidt, I. Crnkovic, G.T. Heineman, J.A. Stafford (Eds.), Component-Based Software Engineering. XII, 283 pages. 2007.

Vol. 4607: L. Baresi, P. Fraternali, G.-J. Houben (Eds.), Web Engineering. XVI, 576 pages. 2007.

Vol. 4606: A. Pras, M. van Sinderen (Eds.), Dependable and Adaptable Networks and Services. XIV, 149 pages. 2007.

Vol. 4605: D. Papadias, D. Zhang, G. Kollios (Eds.), Advances in Spatial and Temporal Databases. X, 479 pages. 2007.

Vol. 4604: U. Priss, S. Polovina, R. Hill (Eds.), Conceptual Structures: Knowledge Architectures for Smart Applications. XII, 514 pages. 2007. (Sublibrary LNAI).

试读结束，需要全本PDF请购买 www.ertongbook.com

Vol. 4603: F. Pfenning (Ed.), Automated Deduction – CADE-21. XII, 522 pages. 2007. (Sublibrary LNAI).

Vol. 4602: S. Barker, G.-J. Ahn (Eds.), Data and Applications Security XXI. X, 291 pages. 2007.

Vol. 4600: H. Comon-Lundh, C. Kirchner, H. Kirchner (Eds.), Rewriting, Computation and Proof. XVI, 273 pages. 2007.

Vol. 4599: S. Vassiliadis, M. Berekovic, T.D. Hämäläinen (Eds.), Embedded Computer Systems: Architectures, Modeling, and Simulation. XVIII, 466 pages. 2007.

Vol. 4598: G. Lin (Ed.), Computing and Combinatorics. XII, 570 pages. 2007.

Vol. 4597: P. Perner (Ed.), Advances in Data Mining. XI, 353 pages. 2007. (Sublibrary LNAI).

Vol. 4596: L. Arge, C. Cachin, T. Jurdziński, A. Tarlecki (Eds.), Automata, Languages and Programming. XVII, 953 pages. 2007.

Vol. 4595: D. Bošnački, S. Edelkamp (Eds.), Model Checking Software. X, 285 pages. 2007.

Vol. 4594: R. Bellazzi, A. Abu-Hanna, J. Hunter (Eds.), Artificial Intelligence in Medicine. XVI, 509 pages. 2007. (Sublibrary LNAI).

Vol. 4592: Z. Kedad, N. Lammari, E. Métais, F. Meziane, Y. Rezgui (Eds.), Natural Language Processing and Information Systems. XIV, 442 pages. 2007.

Vol. 4591: J. Davies, J. Gibbons (Eds.), Integrated Formal Methods. IX, 660 pages. 2007.

Vol. 4590: W. Damm, H. Hermanns (Eds.), Computer Aided Verification. XV, 562 pages. 2007.

Vol. 4589: J. Münch, P. Abrahamsson (Eds.), Product-Focused Software Process Improvement. XII, 414 pages. 2007.

Vol. 4588: T. Harju, J. Karhumäki, A. Lepistö (Eds.), Developments in Language Theory. XI, 423 pages. 2007.

Vol. 4587: R. Cooper, J. Kennedy (Eds.), Data Management. XIII, 259 pages. 2007.

Vol. 4586: J. Pieprzyk, H. Ghodosi, E. Dawson (Eds.), Information Security and Privacy. XIV, 476 pages. 2007.

Vol. 4585: M. Kryszkiewicz, J.F. Peters, H. Rybinski, A. Skowron (Eds.), Rough Sets and Intelligent Systems Paradigms. XIX, 836 pages. 2007. (Sublibrary LNAI).

Vol. 4584: N. Karssemeijer, B. Lelieveldt (Eds.), Information Processing in Medical Imaging. XX, 777 pages. 2007.

Vol. 4583: S.R. Della Rocca (Ed.), Typed Lambda Calculi and Applications. X, 397 pages. 2007.

Vol. 4582: J. Lopez, P. Samarati, J.L. Ferrer (Eds.), Public Key Infrastructure. XI, 375 pages. 2007.

Vol. 4581: A. Petrenko, M. Veanes, J. Tretmans, W. Grieskamp (Eds.), Testing of Software and Communicating Systems. XII, 379 pages. 2007.

Vol. 4580: B. Ma, K. Zhang (Eds.), Combinatorial Pattern Matching. XII, 366 pages. 2007.

Vol. 4579: B. M. Hämmerli, R. Sommer (Eds.), Detection of Intrusions and Malware, and Vulnerability Assessment. X, 251 pages. 2007.

Vol. 4578: F. Masulli, S. Mitra, G. Pasi (Eds.), Applications of Fuzzy Sets Theory. XVIII, 693 pages. 2007. (Sublibrary LNAI).

Vol. 4577: N. Sebe, Y. Liu, Y.-t. Zhuang, T.S. Huang (Eds.), Multimedia Content Analysis and Mining. XIII, 513 pages. 2007.

Vol. 4576: D. Leivant, R. de Queiroz (Eds.), Logic, Language, Information and Computation. X, 363 pages. 2007.

Vol. 4575: T. Takagi, T. Okamoto, E. Okamoto, T. Okamoto (Eds.), Pairing-Based Cryptography – Pairing 2007. XI, 408 pages. 2007.

Vol. 4574: J. Derrick, J. Vain (Eds.), Formal Techniques for Networked and Distributed Systems – FORTE 2007. XI, 375 pages. 2007.

Vol. 4573: M. Kauers, M. Kerber, R. Miner, W. Windsteiger (Eds.), Towards Mechanized Mathematical Assistants. XIII, 407 pages. 2007. (Sublibrary LNAI).

Vol. 4572: F. Stajano, C. Meadows, S. Capkun, T. Moore (Eds.), Security and Privacy in Ad-hoc and Sensor Networks. X, 247 pages. 2007.

Vol. 4571: P. Perner (Ed.), Machine Learning and Data Mining in Pattern Recognition. XIV, 913 pages. 2007. (Sublibrary LNAI).

Vol. 4570: H.G. Okuno, M. Ali (Eds.), New Trends in Applied Artificial Intelligence. XXI, 1194 pages. 2007. (Sublibrary LNAI).

Vol. 4569: A. Butz, B. Fisher, A. Krüger, P. Olivier, S. Owada (Eds.), Smart Graphics. IX, 237 pages. 2007.

Vol. 4568: T. Ishida, S. R. Fussell, P. T. J. M. Vossen (Eds.), Intercultural Collaboration. XIII, 395 pages. 2007.

Vol. 4566: M.J. Dainoff (Ed.), Ergonomics and Health Aspects of Work with Computers. XVIII, 390 pages. 2007.

Vol. 4565: D.D. Schmorrow, L.M. Reeves (Eds.), Foundations of Augmented Cognition. XIX, 450 pages. 2007. (Sublibrary LNAI).

Vol. 4564: D. Schuler (Ed.), Online Communities and Social Computing. XVII, 520 pages. 2007.

Vol. 4563: R. Shumaker (Ed.), Virtual Reality. XXII, 762 pages. 2007.

Vol. 4562: D. Harris (Ed.), Engineering Psychology and Cognitive Ergonomics. XXIII, 879 pages. 2007. (Sublibrary LNAI).

Vol. 4561: V.G. Duffy (Ed.), Digital Human Modeling. XXIII, 1068 pages. 2007.

Vol. 4560: N. Aykin (Ed.), Usability and Internationalization, Part II. XVIII, 576 pages. 2007.

Vol. 4559: N. Aykin (Ed.), Usability and Internationalization, Part I. XVIII, 661 pages. 2007.

Vol. 4558: M.J. Smith, G. Salvendy (Eds.), Human Interface and the Management of Information, Part II. XXIII, 1162 pages. 2007.

Vol. 4557: M.J. Smith, G. Salvendy (Eds.), Human Interface and the Management of Information, Part I. XXII, 1030 pages. 2007.

¥516.00元

Preface

On behalf of the Program Committee, it is our pleasure to present the proceedings of the 10th Symposium on Recent Advances in Intrusion Detection (RAID 2007), which took place in Queensland, Australia, September 5–7, 2007. As in every year since 1998, the symposium brought together leading researchers and practitioners from academia, government, and industry to discuss intrusion detection research and practice.

This year, the RAID Program Committee received 101 paper submissions from all over the world. All submissions were carefully reviewed by at least three members of the Program Committee and judged on the basis of scientific novelty, importance to the field, and technical quality. The final selection took place at the Program Committee meeting held in Oakland, USA, May 22–23, 2007. Sixteen full papers and one short paper were selected for presentation and publication in the conference proceedings, placing RAID among the most competitive conferences in the area of computer security.

A successful symposium is the result of the joint effort of many people. In particular, we would like to thank all the authors who submitted papers, whether accepted or not. We also thank the Program Committee members and additional reviewers for their hard work in evaluating submissions. In addition, we want to thank the General Chair, George Mohay, for handling the conference arrangements, Rei Safavi-Naini for publicizing the conference, Andrew Clark for putting together the conference proceedings, and Ming-Yuh Huang for finding sponsor support.

Finally, we extend our thanks to Northwest Security Institute, SAP, and CERT at the Software Engineering Institute, Carnegie Mellon University for their sponsorship and support.

September 2007

Christopher Kruegel
Richard Lippmann
Andrew Clark

Organization

RAID 2007 was organized by the Information Security Institute, Queensland University of Technology, Brisbane, Australia.

Conference Chairs

General Chair	George Mohay (Queensland University of Technology, Australia)
Program Chair	Christopher Kruegel (Technical University of Vienna, Austria)
Program Co-chair	Richard Lippmann (Massachusetts Institute of Technology, USA)
Publication Chair	Andrew Clark (Queensland University of Technology, Australia)
Publicity Chair	Rei Safavi-Naini (University of Calgary, Canada)
Sponsorship Chair	Ming-Yuh Huang (The Boeing Company, USA)

Program Committee

Michael Behringer	Cisco Systems Inc., France
Sungdeok Cha	Korea Advanced Institute of Science and Technology, Korea
Andrew Clark	Queensland University of Technology, Australia
Marc Dacier	Institut Eurécom, France
Hervé Debar	France Telecom R&D, France
Ulrich Flegel	University of Dortmund, Germany
Jonathon Giffin	Georgia Institute of Technology, USA
Thorsten Holz	University of Mannheim, Germany
Farnam Jahanian	University of Michigan and Arbor Networks, USA
Richard A. Kemmerer	University of California, Santa Barbara, USA
Kwok-Yan Lam	Tsinghua University, China
Wenke Lee	Georgia Institute of Technology, USA
Richard Lippmann	MIT Lincoln Laboratory, USA
Raffael Marty	ArcSight Inc., USA
Roy Maxion	Carnegie Mellon University, USA
Ludovic Mé	Supélec, France
George Mohay	Queensland University of Technology, Australia
Aloysius Mok	University of Texas, Austin, USA
Benjamin Morin	Supélec, France
Rei Safavi-Naini	University of Calgary, Canada
Vern Paxson	International Computer Science Institute and Lawrence Berkeley National Laboratory, USA

Robin Sommer	International Computer Science Institute and Lawrence Berkeley National Laboratory, USA
Dawn Song	Carnegie Mellon University, USA
Salvatore Stolfo	Columbia University, USA
Toshihiro Tabata	Okayama University, Japan
Vijay Varadharajan	Macquarie University, Australia
Giovanni Vigna	University of California, Santa Barbara, USA
Jianying Zhou	Institute for Infocomm Research, Singapore

Steering Committee

Marc Dacier (Chair)	Institut Eurécom, France
Hervé Debar	France Telecom R&D, France
Deborah Frincke	Pacific Northwest National Lab, USA
Ming-Yuh Huang	The Boeing Company, USA
Erland Jonsson	Chalmers University, Sweden
Wenke Lee	Georgia Institute of Technology, USA
Ludovic Mé	Supélec, France
Alfonso Valdes	SRI International, USA
Giovanni Vigna	University of California, Santa Barbara, USA
Andreas Wespi	IBM Research, Switzerland
Felix Wu	University of California, Davis, USA
Diego Zamboni	IBM Research, Switzerland

Local Organizing Committee

Matt Bradford	Queensland University of Technology, Australia
Andrew Clark	Queensland University of Technology, Australia
Elizabeth Hansford	Queensland University of Technology, Australia
James Mackie	Queensland University of Technology, Australia
George Mohay	Queensland University of Technology, Australia
Julien Vayssiére	SAP Research, Brisbane, Australia
Jacob Zimmermann	Queensland University of Technology, Australia

Additional Reviewers

Hirotake Abe
Michael Bailey
Venkat Balakrishnan
Gregory Banks
Michael Becher
John Bethencourt
David Brumley
Martim Carbone
Simon Chung
Siu-Leung Chung
Evan Cooke
Malcolm Corney
Scott Fluhrer
Felix Freiling
Debin Gao
Meng Ge
Markus Hagenbuchner
Jeffrey Horton
Yiyuan Huang
Grégoire Jacob
Frank Kargl
Kevin Killourhy
Kee-Eung Kim
Sang-Rok Kim

Christian Kreibich	James Newsome	Hongwei Sun
Junsup Lee	Yoshihiro Oyama	Rafael Timoteo de Sousa Jr
Minsoo Lee	Jon Oberheide	Eric Totel
Corrado Leita	Sorot Panichprecha	Uday Tupakula
Francois Lesueur	James Riordan	Jouni Viinikka
Zhuowei Li	Sebastian Schmerl	Heng Yin
Liang Lu	Jeong-Seok Seo	Stefano Zanero
Michael Meier	Wook Shin	Jacob Zimmermann
Z. Morley Mao	Takahiro Shinagawa	
Koichi Mouri	Sushant Sinha	

Table of Contents

试读结束，需要全本PDF请购买 www.ertongbook.com

Exploiting Execution Context for the Detection of Anomalous System Calls

Darren Mutz, William Robertson, Giovanni Vigna, and Richard Kemmerer

Computer Security Group
Department of Computer Science
University of California, Santa Barbara
{dhm,wkr,vigna,kemm}@cs.ucsb.edu

Abstract. Attacks against privileged applications can be detected by analyzing the stream of system calls issued during process execution. In the last few years, several approaches have been proposed to detect anomalous system calls. These approaches are mostly based on modeling acceptable system call sequences. Unfortunately, the techniques proposed so far are either vulnerable to certain evasion attacks or are too expensive to be practical. This paper presents a novel approach to the analysis of system calls that uses a composition of dynamic analysis and learning techniques to characterize anomalous system call invocations in terms of both the invocation context and the parameters passed to the system calls. Our technique provides a more precise detection model with respect to solutions proposed previously, and, in addition, it is able to detect data modification attacks, which cannot be detected using only system call sequence analysis.

Keywords: Intrusion Detection, System Call Argument Analysis, Execution Context.

1 Introduction

A recent thrust of intrusion detection research has considered model-based detection of attacks at the application level. Model-based systems operate by comparing the observed behavior of an application to *models* of normal behavior, which may be derived automatically via static analysis [8,23] or learned by analyzing the run-time behavior of applications [3,5,12,18,15]. In each case, attacks are detected when observed behavior diverges in some respect from the normal behavior captured by the model. In contrast to misuse-based approaches, where the analysis identifies attacks against applications using patterns of known malicious actions, model-based schemes have the advantage of being able to detect novel attacks, since attacks are not explicitly represented by the system. We note that this advantage typically comes at the cost of performance, precision, and explanatory capability, three properties that misuse-based approaches often achieve very well.

Most model-based intrusion detection systems monitor the sequence of system calls issued by an application, possibly taking into account some execution state. For example, the system described in [3] monitors pairs of system calls

© Springer-Verlag Berlin Heidelberg 2007

```
 1  void write_user_data(void)
 2  {
 3    FILE *fp;
 4    char user_filename[256];
 5    char user_data[256];
 6
 7    gets(user_filename);
 8
 9    if (privileged_file(user_filename)) {
10      fprintf(stderr, "Illegal␣filename.␣Exiting.\n");
11      exit(1);
12    }
13    else {
14      gets(user_data);      // overflow into user_filename
15
16      fp = fopen(user_filename, "w");
17
18      if (fp) {
19        fprintf(fp, "%s", user_data);
20        fclose(fp);
21      }
22    }
23  }
```

Fig. 1. Sample data modification attack

and records the application's stack configuration (that is, part of the history of function invocations). During the detection phase, the system checks if the observed pairs of system calls (and their associated stack configuration) match pairs recorded during the learning period. The systems described in [8] and [23] check call sequences against automata-based models derived from the application's source code or binary representation, and identify sequences that could not have been generated by the model.

Some of the shortcomings of sequence-based approaches were discussed in [2], where the problems of *incomplete sensitivity* and *incomplete sets of events* were introduced. Incomplete sensitivity affects models derived from static analysis. Due to the limitations of static analysis techniques, these models may accept impossible sequences of system calls (for example because branch predicates are not considered).

The problem of incomplete sets of events is more general, and it affects all approaches based on system call sequences. This problem stems from the fact that, in these systems, the manifestation of an attack must be characterized in terms of anomalies in the order in which system calls are executed. Changes in the ordering of system call invocations occur, for example, because foreign code is injected into the application (such as through a buffer overflow) or because the order in which instructions are executed is modified. Therefore, by modeling system call sequences, these approaches implicitly restrict themselves to only detecting attacks that modify the execution order as expressed by the application's code or by the execution histories observed during a training period. Unfortunately, an attacker can successfully compromise an application's goals by modifying the application's *data* without introducing anomalous paths in the application's execution.

Consider, for example, the procedure `write_user_data` in Figure 1. Here, an overflow of the variable `user_data` at line 14 allows an attacker to overwrite the value contained in `user_filename`, which the application assumes was checked by the procedure invoked at line 9. Therefore, the attacker can leverage the overflow to append data of her choice to any file the application has access to. Note that the execution of this data modification attack does not affect the type or ordering of the system calls issued by the application.

To detect data modification attacks, models must include some representation of valid or normal program state. For example, prior work in [11] and [12] uses learning models to characterize "normal" system call argument values and to demonstrate that changes to program state as a result of an attack often manifest themselves as changes to the argument values of system calls. The assumption underlying this approach is that the goal of the attacker is to leverage the privileges of an application to change some security-relevant state in the underlying system (e.g., write chosen values to a file, execute a specific application, or change the permissions of a security-critical file). This type of activity may be readily observed as suspicious system call argument values.

One limitation of the argument modeling approach in [11], [12], and [15] is that models of normal argument values are built for each system call. That is, one set of models is created for `open`, another set for `execve`, and so on. As a result, a model captures the full range of argument values observed during all phases of the execution of an application. A better approach would be to train the models in a way that is specific to individual phases of a program's execution. For example, the arguments used during a program's initialization phase are likely to differ from those used during a production phase or termination phase. This can be achieved by differentiating program behavior using the *calling context* of a procedure – that is, the configuration of the application's call stack when a procedure is invoked. Similar techniques have been explored in the programming languages literature. Examples include improving profiling by considering a procedure's calling context [1], analyzing pointer variables more accurately [9], and improving lifetime predictions of dynamically allocated memory [16]. A common observation in these approaches is that the calling context of a procedure is often a powerful predictor of how the procedure and its data interact.

In this paper, we first propose and evaluate a metric for determining to what extent argument values are unique to a particular call stack for a given application. Our study, presented in Section 2, shows that this is predominantly the case, indicating that the argument modeling approach of [12] can be made more precise if models are built for each calling context in which a system call is issued by an application. Armed with this knowledge, we then introduce and evaluate a model-based detection system that builds separate argument models for each call stack in which an application issues a system call. Our experiments demonstrate that the trained models effectively generalize from the training data, performing well during a subsequent detection period.

This paper makes the following primary contributions:

- It analyzes the relationship between system call arguments and different calling contexts, and it introduces a novel metric to quantify the degree to which argument values exhibit uniqueness across contexts.
- It demonstrates that the application's call stack can be leveraged to add context to the argument values that appear at the system call interface. It

also demonstrates that the increased sensitivity of context-specific argument models results in better detection performance.
- It defines a technique to detect data modification attacks, which are not detected by previously proposed approaches based on system call sequences.
- It presents an extensive real-world evaluation encompassing over 44 million system call invocations collected over 64 days from 10 hosts.

The remainder of the paper is structured as follows. In Section 2 we introduce and apply a metric to characterize the degree to which system call argument values are unique to calling contexts in which system calls are issued. Then, in Section 3, we present our detection approach, which builds argument models that are specific to each calling context. Section 4 reports the results of evaluating the system empirically. Section 5 covers related work on system call-based anomaly detection. Finally, Section 6 draws conclusions and outlines future work.

2 System Call Argument and Calling Context Analysis

The effectiveness of system call analysis that includes call stack information is directly related to the number of contexts in which a given argument value associated with the invocations of a particular system call occurs. More specifically, if argument values appear in many contexts, essentially randomly, context-specific learning models are likely to offer no benefit. Furthermore, if each observed argument value appears (possibly multiple times) in only one context, we would expect system call argument analysis that includes call stack information to outperform context-insensitive models. In this section, we propose a metric to express the degree of context-uniqueness of argument values. We then use this metric to determine which applications are likely to be amenable to system call analysis that takes into account stack-specific behavior.

Before introducing our context-uniqueness metric, we need to define some notation. Let $S = \{s_1, s_2, \ldots\}$ be the set of monitored system calls, and let $A^{s_i} = \langle A_1^{s_i}, \ldots, A_n^{s_i} \rangle$ be the vector of formal arguments for system call s_i. Consistent with [6], we define the calling context of a system call invocation as the sequence of return addresses $C = \langle r_1, \ldots, r_l \rangle$ stored on the application's call stack at the time the system call invocation occurs. Each invocation s_{ij} of s_i has a concrete vector of values for A^{s_i} defined as $a^{s_{ij}} = \langle a_1^{s_{ij}}, \ldots, a_n^{s_{ij}} \rangle$, and two argument vectors $a^{s_{ij}}$ and $a^{s_{ij'}}$ are considered distinct if any of their subvalues $a_l^{s_{ij}}$ and $a_l^{s_{ij'}}$ differ.

We are interested in the set of argument vectors appearing in the invocation of a system call in a particular context. For this, we introduce the notion of an *argument set*. An argument set for a system call s_i in a context C is the set of all argument vectors $a^{s_{ij}}$ observed for the chosen system call when it is issued in the calling context C. This is denoted by $AS(C, s_i)$. The argument set for s_i across the entire application (i.e., ignoring the calling context) is denoted by $AS(*, s_i)$. We observe that if the set $AS(*, s_i)$ is partitioned by the subsets $\{AS(C_1, s_i), AS(C_2, s_i), \ldots\}$, then each recorded argument vector a^{s_i} occurs in only one calling context.

One potential route in the development of this metric would be to adapt cluster quality measures from the machine learning literature. Unfortunately, computing the distance between two argument vectors $a^{s_{ij}}$ and $a^{s_{ij}\prime}$ is problematic. For example, integer arguments that exhibit numeric similarity are often dissimilar in their semantic meaning. This occurs in cases where an integer argument is the logical OR of a collection of boolean flags. Computing string similarity also presents difficulties. For example, two filesystem paths may have large common substrings or a small Hamming distance, but correspond to files that have a very different meaning to the users of the system. For these reasons, we build our metric using argument vector equality only.

With this in mind, we would like to determine the number of contexts where each distinct argument vector is used. To measure this we define the *actual partitioning* value $AP(s_i)$, which is the sum over all recorded concrete argument vectors of the number of argument sets where each $a^{s_{ij}}$ appears during the period of monitoring. That is,

$$AP(s_i) = \sum_{j=1}^{K} \sum_{m=1}^{L} \mid \{a^{s_{ij}}\} \cap AS(C_m, s_i) \mid \tag{1}$$

where K is the number of distinct argument vector values recorded, and L is the number of distinct stack configurations observed during the monitoring period.

For our context-uniqueness metric, we would like to compare the actual partitioning value to both the optimal partitioning and the worst case partitioning values. For the optimal case, each argument vector should appear in as few contexts as possible. There are two cases to consider. In the case where the number of distinct argument vectors is greater or equal to the number of calling contexts ($K \geq L$), each argument value appears in only one context in the optimal partition of $AS(*, s_i)$. For the case when $K < L$, some argument vectors must appear in more than one context[1]. The optimal partitioning, in this case, is for each concrete argument vector to appear in L/K argument sets. Both cases can be expressed by specifying the number of argument sets where each argument vector is to appear as $max(L/K, 1)$.

We can now define the optimal partitioning value and the worst case partitioning value. Since there are K distinct argument vector values, the *optimal partitioning* value $OP(s_i)$ is defined as:

$$OP(s_i) = K * max(L/K, 1) = max(L, K) \tag{2}$$

To define the worst case, we need to know how many instances of each of the K distinct argument vectors $a^{s_{ij}} \in AS(*, s_i)$ were recorded during the monitoring period. We define the counter $cnt_{a^{s_{ij}}}$ as the number of times that a particular argument vector $a^{s_{ij}}$ occurs in the recorded invocations. The worst case partitioning is determined by distributing each of the K argument vectors in $AS(*, s_i)$ over as many contexts as possible. Although $a^{s_{ij}}$ can appear a maximum of $cnt_{a^{s_{ij}}}$ times, there are only L distinct contexts. Therefore, $a^{s_{ij}}$ appears

[1] If each distinct value appeared in only one context, then there would be contexts with no argument vectors.

试读结束，需要全本PDF请购买 www.ertongbook.com