EXHIBIT B
SRI International  
Computer Science Laboratory

Technical Reports (Since 1987)


Note: Only CSL Technical Reports are listed here: for papers and books, follow the links in the "Programs and Activities" or "CSL Staff Members" sections on CSL's Home Page. Reports prior to 1987 are not listed here, but some are still available. We are striving to make all our recent technical reports available electronically, however most reports before 1994 are available only in hard copy.

To receive hard copies of SRI-CSL reports contact:

Judith Burgess  
Computer Science Laboratory  
SRI International  
333 Ravenswood Ave.  
Menlo Park, CA 94025, USA

Phone (415) 859-5924  
FAX (415) 859-2844  
email burgess@csli.sri.com

1996

- SRI-CSL-96-01, January 1996  
  *An Overview of Enclaves 1.0*, by Li Gong.

- SRI-CSL-96-03, June 1996  
  *Gateway Based Approach for Conducting Multiparty Multimedia Sessions over Heterogeneous Signaling Domaines*, by Madhu Sudan and Nachum Shacham. [Abstract and Postscript.]

- SRI-CSL-96-05, May 1996  
  *Architectures and Formal Representations for Secure Systems*, by Peter G. Neumann. [Abstract and Postscript.]


- SRI-CSL-96-09, November 1996  
  *Automated Theorem Proving by Test Set Induction*, by Adel Bouhoula. [Abstract]
1995

- SRI-CSL-95-01, March 1995
  Formal Methods and their Role in the Certification of Critical Systems by John Rushby. abstract, dvi or postscript

- SRI-CSL-95-02, January 1995

- SRI-CSL-95-03, January 1995
  Mechanized Formal Verification - Seven Papers by David Cyrluk, Patrick Lincoln, Steven P. Miller, Paliath Narendran, Sam Owre, Sreeranga Rajan, John Rushby, Natarajan Shankar, Jens Ulrik Skakkebaek, Mandayam Srivas, and Friedrich von Henke. abstract

1. Formal Verification for Fault-Tolerant Architectures: Prolegomena to the Design of PVS

2. Formal Verification of an Interactive Consistency Algorithm for the Draper FTP Architecture Under a Hybrid Fault Model, by Patrick Lincoln and John Rushby, from Compass '94 (Washington DC, June). abstract, dvi or postscript

3. A Formally Verified Algorithm for Clock Synchronization Under a Hybrid Fault Model, by John Rushby, presented at PODC '94 (Los Angeles, August 94). abstract, dvi or postscript


5. Ground Temporal Logic: A Logic for Hardware Verification, by David Cyrluk and Paliath Narendran, presented at CAV '94 (Stanford, CA, June). abstract or postscript only


7. Formal Verification of the AAMP5 Microprocessor: A Case Study in the Industrial Use of Formal Methods, by Steven P. Miller and Mandayam Srivas, presented at WIFT '95. abstract and postscript

- SRI-CSL-95-04, July 1995
  Formal Verification of a Commercial Microprocessor, Mandayam Srivas and Steven P. Miller. (Available in hard copy only.) abstract

- SRI-CSL-95-05, July 1995
  Execution Driven Distributed Simulation of Parallel Architectures, by Livio Ricciulli, Patrick
Lincoln, and Jose Meseguer. abstract and postscript.

- SRI-CSL-95-06, May 1995

- SRI-CSL-95-07, May 1995

- SRI-CSL-95-08, July 1995
  Correct Schema Transformations, Xiaolei Qian. abstract and postscript

- SRI-CSL-95-09, July 1995
  Query Folding, Xiaolei Qian. abstract and postscript

- SRI-CSL-95-10, June 1995

- SRI-CSL-95-11, July 1995
  Architecture and Techniques for Fast Computer Tomography, Iskender Agi. (Available in hard copy only.) abstract

- SRI-CSL-95-12, June 1995, Rev. April 1996
  Analyzing Tabular and State-Transition Specifications in PVS, Sam Owre, Natarajan Shankar and John Rushby. abstract

- SRI-CSL-95-17, December 1995 Finite-State Analysis of Space Shuttle Contingency Guidance Requirements, Judith Crow. abstract and postscript

- SRI-CSL-95-18, November 1995
  View-Based Access Control with High Assurance, Xiaolei Qian. abstract and postscript

1994

- SRI-CSL-94-01, January 1994
  General Logics and Logical Frameworks by Narciso Martí-Oliet and José Meseguer. abstract

- SRI-CSL-94-02, April 1994
  Semantic Interoperation: A Query Mediation Approach by Xiaolei Qian and Teresa Lunt. abstract

- SRI-CSL-94-03, March 1994
  Elements of Trusted Multicasting by Li Gong and Nachum Shacham. abstract or abstract or postscript

- SRI-CSL-94-04, March 1994
Implementing Adaptive Fault-Tolerant Services for Hybrid Faults by Jack Goldberg and Li Gong. abstract or abstract or postscript

- SRI-CSL-94-05, April 1994
  A Formal Approach to Correct Refinement of Software Architectures by Mark Moriconi, Xiaolei Qian, and Robert Riemenschneider.
  (superseded by SRI-CSL-94-13) abstract

- SRI-CSL-94-06, May 1994
  GLU Programmer's Guide by R. Jagannathan and Christopher Dodd abstract

- SRI-CSL-94-07, April 1994
  Action and Change in Rewriting Logic by Narciso Marti-Oliet and José Meseguer. abstract

- SRI-CSL-94-08, May 1994
  New Protocols for Third-Party-Based Authentication and Secure Broadcast by Li Gong. (Also includes two other papers). abstract or abstract or postscript

- SRI-CSL-94-09, July 1994
  Correctness and Composition of Software Architectures by Mark Moriconi and Xiaolei Qian. (Not yet published)

  SRI-CSL-94-10, October 1994
  Transformations in High-Level Synthesis: Formal Specification and Efficient Mechanical Verification by P. Sreeranga Rajan. abstract or postscript

- SRI-CSL-94-11, May 1994
  Specification, Transformation, and Programming of Concurrent Systems in Rewriting Logic by Patrick Lincoln, Narciso Martí-Oliet and José Meseguer abstract

- SRI-CSL-94-12, August 1994
  A Policy Framework for Multilevel Relational Databases by Xiaolei Qian and Teresa F. Lunt. abstract

- SRI-CSL-94-13, August 1994
  A Formal Approach to Correct Refinement of Software Architectures by M. Moriconi, X. Qian, and R. A. Riemenschneider. (Not yet published. Will supersede 94-05.)

  SRI-CSL-TR-94-14, October 1994
  Fail-Stop Protocols: An Approach to Designing Secure Protocols by Li Gong. abstract or postscript

  Efficient Network Authentication Protocols: Lower Bounds and Optimal Implementations by Li Gong. abstract or postscript
• SRI-CSL-93-01, March 1993
  *Critical System Properties: Analysis and Taxonomy* by John Rushby. abstract, postscript

• SRI-CSL-93-02, March 1993
  *A Formally Verified Algorithm for Interactive Consistency under a Hybrid Fault Model* by Patrick Lincoln and John Rushby. abstract, dvi or postscript

• SRI-CSL-93-03, May 1993
  *Multidimensional Problem Solving in Lucid* by A.A. Faustini and R. Jagannathan. abstract,

• SRI-CSL-93-04, May 1993
  *Eight Papers on Formal Verification* by Patrick Lincoln, Sam Owre, Natarajan Shankar, John Rushby, and Friedrich von Henke. abstract,
  1. *Formal Verification for Fault-Tolerant Architectures: Some Lessons Learned* abstract, dvi or postscript
  2. *Formal Verification of Algorithms for Critical Systems* abstract, dvi or postscript
  3. *Mechanical Verification of a Schematic Byzantine Fault-Tolerant Clock Synchronization Algorithm* abstract, dvi or postscript
  5. *PVS: A Prototype Verification System* abstract, dvi or postscript
  6. *A Formally Verified Algorithm for Interactive Consistency Under a Hybrid Fault Model* abstract, dvi or postscript
  7. *The Formal Verification of an Algorithm for Interactive Consistency under a Hybrid Fault Model* abstract, dvi or postscript
  8. *Verification of Real-Time Systems Using PVS* abstract, dvi or postscript

• SRI-CSL-93-05, Aug. 1993
  *Rewriting Logic as a Logical and Semantic Framework* by Narciso Martí-Oliet and José Meseguer.

• SRI-CSL-93-06, November 1993
  *A Model-Theoretic Semantics of the Multilevel Secure Relational Model* by Xiaolei Qian.

• SRI-CSL-93-07, November 1993
  *Formal Methods and the Certification of Critical Systems* by John Rushby. abstract, dvi or postscript

• SRI-CSL-93-08, Dec. 1993
  *A Lazy Approach to Compositional Verification* by N. Shankar. abstract, dvi or postscript

• SRI-CSL-93-09, Dec. 1993
  *Abstract Datatypes in PVS* by N. Shankar. abstract, dvi or postscript

• SRI-CSL-93-10, Dec. 1993
  *A Duration Calculus Proof Checker: Using PVS as a Semantic Framework* by J.U. Skakkebaek and N. Shankar. abstract, dvi or postscript

• SRI-CSL-93-11, Dec. 1993
• SRI-CSL-93-12, Dec. 1993
  Microprocessor Verification in PVS: A Methodology and Simple Example by David Cyrluk. abstract, or postscript only

• SRI-CSL-93-13, December 1993
  The Complexity and Composability of Secure Interoperation by Li Gong and Xiaolei Qian. dvi

1992

• SRI-CSL-92-01, January 1992
  Formal Verification of an Oral Messages Algorithm for Interactive Consistency, by John Rushby. abstract, dvi or postscript

• SRI-CSL-92-02, January 1992
  Noninterference, Transitivity, and Channel-Control Security Policies by John Rushby. abstract, dvi or postscript

• SRI-CSL-92-03, March 1992 (Draft only)
  Introducing OBJ by Joseph A. Goguen, Timothy Winkler, José Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud.

• SRI-CSL-92-04, March 1992
  Survey of Object-Oriented Analysis/Design Methodologies and Future CASE Frameworks by Donovan Hsieh.

• SRI-CSL-92-05, April 1992

• SRI-CSL-92-06, May 1992
  Multiparadigm Logic Programming by José Meseguer.

• SRI-CSL-92-07, July 1992

• SRI-CSL-92-08 July 1992
  A Logical Theory of Concurrent Objects and its Realization in the Maude Language by José Meseguer.

• SRI-CSL-92-09, Sept. 1992
  On the Semantics of Place/Transition Petri Nets by José Meseguer, Ugo Montanari, and Vladimiro Sassone.

• SRI-CSL-92-11, Sept. 1992
  Using Rewriting Logic to Specify, Program, Integrate and Reuse Open Concurrent Systems of Cooperating Agents by José Meseguer, K. Futatsugi, and Timothy Winkler.
• SRI-CSL-92-12, November, 1992
  Mechanized Verification of Real-Time Systems Using PVS, by N. Shankar. abstract, dvi or postscript

• SRI-CSL-92-13, December, 1992
  GLU: A Hybrid Language for Parallel Applications Programming by R. Jagannathan and A.A. Faustini.

• SRI-CSL-92-14, December, 1992
  Solving the Inheritance Anomaly in Concurrent Object-Oriented Programming by José Meseguer.

• SRI-CSL-92-15, December 1992
  A Logical Semantics for Object-Oriented Databases by José Meseguer and Xiaolei Qian.

1991

• SRI-CSL-91-01, February 1991

• SRI-CSL-91-02, February 1991
  An Introduction to Formal Specification and Verification Using EHDM, by John Rushby, Friedrich von Henke, and Sam Owre. abstract, dvi or postscript

• SRI-CSL-91-03, June 1991

• SRI-CSL-91-04, June 1991
  Mechanical Verification of a Schematic Protocol for Byzantine Fault-Tolerant Clock Synchronization, by N. Shankar. abstract, dvi or postscript

• SRI-CSL-91-05, February 1991
  Conditional Rewriting Logic as a Unified Model of Concurrency, by José Meseguer.

• SRI-CSL-91-06, March 1991
  Final Algebras, Coencomputable Algebras, and Degrees of Unsolvability, by Lawrence S. Moss, José Meseguer, and Joseph A. Goguen.

• SRI-CSL-91-07, April 1991
  From Petri Nets to Linear Logic Through Categories: A Survey, by Narciso Martí-Oliet and José Meseguer.

• SRI-CSL-91-08, November 1991
  Parallel Programming in Maude, by José Meseguer and Timothy Winkler.

1990

• SRI-CSL-90-01, February 1990
  Duality in Closed and Linear Categories by Narciso Martí-Oliet and José Meseguer.

• SRI-CSL-90-02R, February 1990, Revised June 1990
  Rewriting as a Unified Model of Concurrency by José Meseguer.

• SRI-CSL-90-03, February 1990
  Compiling Concurrent Rewriting onto the Rewrite Rule Machine by Hitoshi Aida, Joseph Goguen, and José Meseguer.

• SRI-CSL-90-04, August, 1989
  Secure Knowledge-Based Systems by Teresa F. Lunt and Jonathan K. Millen.

• SRI-CSL-90-06, June, 1990
  Order-Sorted Algebra Solves the Constructor-Selector, Multiple Representation and Coercion Problems by José Meseguer and Joseph Goguen.

• SRI-CSL-90-07, July, 1990
  A Logical Theory of Concurrent Objects by José Meseguer.

• SRI-CSL-90-08, August, 1990
  Decision Problems for Propositional Linear Logic by Patrick Lincoln, John Mitchell, Andre Sevedov, and Natarajan Shankar. dvi

• SRI-CSL-90-09, October, 1990
  On Location: Points about Regions by Judith S. Crow and Peter B. Ladkin. abstract, or postscript only

• SRI-CSL-90-10, October 1990

• SRI-CSL-90-11, November, 1990
  The GLU Programming Language by A.A. Faustini and R. Jagannathan.

• SRI-CSL-90-12, November, 1990
  Axiomatizing the Algebra of Net Computation and Processes by Pierpaolo Degano, José Meseguer, and Ugo Montanari.

• SRI-CSL-90-13, December 1990

• SRI-CSL-90-14, November 1990
  Conditional Rewriting Logic: Deduction, Models and Concurrency by José Meseguer.

• SRI-CSL-90-15, December 1990
  A Logic to Unify Semantic Network Knowledge Systems with Object-Oriented Database Models by Donovan Hsieh.

• SRI-CSL-90-16, December 1990
  Inclusions and Subtypes by Narciso Martí-Oliet and José Meseguer.
• SRI-CSL-90-17, December 1990

Architectural Design of the Rewrite Rule Machine Ensemble by Hitoshi Aida, Sany Leinwand, and José Meseguer.

1989

• SRI-CSL-89-1 January 1989

An Intensional Parallel Processing Language for Applications Programming by E.A. Ashcroft, A.A. Faustini, and R. Jagannathan.

• SRI-CSL-89-2 January 1989


• SRI-CSL-89-3R February 1989

Formal Verification of the Interactive Convergence Clock Synchronization Algorithm using EHDM by John Rushby and Friedrich von Henke. abstract, dvi or postscript

• SRI-CSL-89-4R March 1989, Rev. June 1989

From Petri Nets to Linear Logic by Narciso Martí-Oliet and José Meseguer.

• SRI-CSL-89-5 March 1989

General Logics by José Meseguer.

• SRI-CSL-89-6 March 1989

The Rewrite Rule Machine Project by Joseph Goguen, José Meseguer, Sany Leinwand, Timothy Winkler, and Hitoshi Aida.

• SRI-CSL-89-8 July 1989

A Categorical Manifesto by Joseph A. Goguen.

• SRI-CSL-89-9 Sept. 1989

A Practical Approach to Semantic Configuration Management by Mark Moriconi.

• SRI-CSL-89-10 July 1989

Order-Sorted Algebra I - Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations by Joseph A. Goguen and José Meseguer.

• SRI-CSL-89-11, December 1989

An Algebraic Axiomatization of Linear Logic Models by Narciso Martí-Oliet and José Meseguer.

1988

• SRI-CSL-88-1, January 1988

Higher Order Functions Considered Unnecessary for Higher Order Programming by Joseph Goguen.

• SRI-CSL-88-2R2, January 1988
  *What is Unification? A Categorical View of Substitution, Equation and Solution* by Joseph Goguen.

• SRI-CSL-88-3R, January 1988
  *Petri Nets Are Monoids* by José Meseguer and Ugo Montanari.

• SRI-CSL-88-4R2, August 1988
  *OBJ as a Theorem Prover with Applications to Hardware Verification* by Joseph Goguen.

• SRI-CSL-88-5, May 1988
  *A Descriptive and Prescriptive Model for Dataflow Semantics* by R. Jagannathan.

• SRI-CSL-88-7R, June 1988
  *Quality Measures and Assurance for AI Software* by John Rushby. abstract, dvi or postscript


• SRI-CSL-88-11 Sept. 1988
  *Software for the Rewrite Rule Machine* by Joseph A. Goguen and José Meseguer.

• SRI-CSL-88-13 Oct. 1988
  *Relating Models of Polymorphism* by José Meseguer.

• SRI-CSL-88-14 Nov. 1988
  *Reasoning about Design Changes* by Mark Moriconi and Gail A. Harrison.

---

1987

• SRI-CSL-87-1, May 1987
  *The Rewrite Rule Machine Project* by Joseph Goguen et al.

• SRI-CSL-87-2, May 1987
  *Concurrent Term Rewriting as a Model of Computation* by Joseph Goguen, Claude Kirchner, and José Meseguer.

• SRI-CSL-87-3, May 1987
  *An Abstract Machine for Fast Parallel Matching of Linear Patterns* by Ugo Montanari and Joseph Goguen.

• SRI-CSL-87-7, July 1987
  *Unifying Functional, Object-Oriented and Relational Programming with Logical Semantics* by Joseph Goguen and José Meseguer.
EXHIBIT C
Network-based Alert Management

Inventors: Phillip Andrew Portas, Cupertino, CA (US); Martin Wayne Fong, San Francisco, CA (US)

Assignee: SRI International, Inc., Menlo Park, CA (US)

Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 481 days.

Related U.S. Application Data

Continuation in part of application No. 09/188,739, filed on Nov 9, 1998, now Pat. No. 6,321,338

Field of Search: 173/201, 201; 709/224

References Cited

Primary Examiner—Thomas M. Heckler

Abstract

A method of managing alerts in a network including receiving alerts from network sensors, consolidating the alerts that are indicative of a common incident and generating output reflecting the consolidated alerts.

57 Claims, 6 Drawing Sheets
US 6,704,874 B1
Page 2

U.S. PATENT DOCUMENTS

6,005,467 A 12/1999 Ratcliff et al. ......... 709/224
6,052,709 A 4/2000 Paul ............. 709/220
6,070,244 A * 5/2000 Scheckl et al. ......... 713/201
6,096,845 B1 5/2002 Sugita .......... 370/449
6,194,141 A * 10/2002 Olden .......... 713/201
6,235,703 B1 2/2003 Joyce .......... 713/201

OTHER PUBLICATIONS


Debar et al., "A Neural Network Component for an Intrusion Detection System," © 1992 IEEE.


Dowell, "The Computerwatch Data Reduction Tool," AT&T Bell Laboratories, Whippany, New Jersey.


Vacarro et al., "Detection of Anomalous Computer Session Activity," © 1989 IEEE.


* cited by examiner
Fig. 2

1. Receive formatted alerts
2. Pass alerts through filters
3. Prioritize alerts
4. Topology vetting
5. Generate internal reports
6. Generate incident reports
7. Generate report output
8. Transport report output
<table>
<thead>
<tr>
<th>TYPE OF ATTACK</th>
<th>DENIAL OF SERVICE</th>
<th>INTEGRITY</th>
<th>SECURITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;PING OF DEATH&quot;</td>
<td>90</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>BUFFER OVERFLOW</td>
<td>80</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>WRITE - POLLING VIOLATION</td>
<td>10</td>
<td>90</td>
<td>70</td>
</tr>
<tr>
<td>USER1</td>
<td>0</td>
<td>20</td>
<td>80</td>
</tr>
<tr>
<td>USER2</td>
<td>80</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

**FIG. 3**
FIG. 4
NETWORK-BASED ALERT MANAGEMENT

This application claims priority under 35 USC §120 to U.S. patent application Ser. No. 09/188,739, filed on Nov. 9, 1998, now U.S. Pat. No. 6,251,336, the entire contents of which are hereby incorporated by reference.

GOVERNMENT RIGHTS IN THIS INVENTION

This invention was made with U.S. government support under contract numbers F30601-96-C-0294 and F30602-99-C-0187 awarded by the U.S. Air Force Research Laboratory. The U.S. government has certain rights in this invention.

TECHNICAL FIELD

This invention relates to network-based alert management.

BACKGROUND

Computer networks may include one or more digital security monitors or sensors that automatically analyze traffic on the network to identify potentially suspicious activity. The sensors may be implemented in either software or hardware. Monitors may focus on security monitoring and/or on fault analysis.

Upon detecting suspicious activity, the sensors typically generate some kind of digital alert message or signal, and attempt to bring that message to the attention of network system managers whose responsibility it is to respond and react in an appropriate defensive manner against hostile digital attacks or to recover quickly from catastrophic failures.

SUMMARY

In an aspect, the invention features a method of managing alerts in a network including receiving alerts from network sensors, consolidating the alerts that are indicative of a common incident and generating output reflecting the consolidated alerts. Alerts are formatted into a standard alert format by the network sensors or an input receiving logic of an alert management system, or a combination of both. The alert format may be selected from a group of formats including IDIR, SNMP, HP OpenView, Attach Specification CIDF and GIDO. Alerts may be tagged with corresponding significance scores where the significance scores may include a priority measure for the corresponding alerts. The priority measure may be derived from a priority map that can be automatically generated or dynamically adjusted. The priority map may contain relative priority scores for resources availability, resource integrity and resource confidentiality.

In another aspect, the invention features a method of managing alerts including receiving alerts from a number of network sensors, filtering the alerts to produce one or more internal reports and consolidating the internal reports that are indicative of a common incident to incident report. Related incident reports may be correlated. The network sensors may format the received alerts. Filtering includes deleting alerts that do not match specified rules. The filtering rules may be dynamically adjusted. Filtering may also include tagging alerts with a significance score that can indicate a priority measure and relevance measure.

Among the advantages of the invention may be one or more of the following.

The alert manager can be tailored to a particular application by dynamically adding or removing data connections to sources of incoming alerts, and by dynamically varying the process modules, user filter classes, priority classes, topology classes, and output. Process modules may be added, modified, and deleted while the alert manager is active.

Output may be configured for a variety of graphical user interfaces (GUIs). In embodiments, useful, for example, for each category of attack the user can define different priorities as related to denial of service, security, and integrity.

Process modules are logical entities within the alert manager that can respond to an incoming alert in real time and virtual time, i.e., data within an application can cause the alert manager to respond.

The alert manager can act as a sender or receiver. In embodiments, useful, for example, the alert manager can listen to a specific port in a network or connect to an external process on a host computer and process its data.

The alert management process can be an interpretive process allowing the incorporation of new process classes and new rules.

The alert management process may provide a full solution for managing a diverse suite of multi-party security and fault monitoring services. Example targets of the alert management process are heterogeneous network computing environments that are subject to some perceived operational requirements for confidentiality, integrity, or availability. Inserted within the network are a suite of potential multi-party security and fault monitoring services such as intrusion detection systems, firewalls, security scanners, virus protection software, network management probes, load balancers, or network service appliances. The alert management process provides alert distributions within the monitored network through which security alerts, fault reports, and performance logs may be collected, processed and distributed to remote processing stations (e.g., Data Centers, Administrative Help Desks, MIS stations). Combined data produced by the security, fault, or performance monitoring services provide these remote processing stations detailed insight into the security posture, and more broadly the overall health, of the monitored network.

Value may be added to the content delivered by the alert management process to the remote processing station(s) that subscribe to alerts in the form of an advanced alert processing chain. For example, alerts received by the alert management process and prepared for forwarding to a remote processing station may be filtered using a dynamically downloadable message criteria specification.

In a further aspect, alerts may be tagged with a priority indication flag formulated against the remote processing station's alert processing policy and tagged with a relevance flag that indicates the likely severity of the attack with respect to the known internal topology of the monitored network.

In a further aspect of the invention, alerts may be aggregated (or consolidated) into single incident reports when found to be associated with a series of equivalent alerts produced by the same sensor or by other sensors, based upon equivalence criteria, and the incident reports forwarded to the remote processing station.

The alert management system is configurable with respect to the data needs and policies specified by the remote processing station. These processes are customizable on a per remote processing station basis. For example, two remote processing stations may in parallel subscribe to alerts from the alert management process, with each having individual filtering policies, prioritization schemes, and so forth, applied to the alert/incident reports it receives.

Other features and advantages will become apparent from the following description and from the claims.
DESCRIPTION OF DRAWINGS

FIG. 1 is a block diagram of a network based alert management system.

FIG. 2 is a flow diagram of an alert management process.

FIG. 3 is a block diagram of a priority database record.

FIG. 4 is a block diagram of the remote processing center.

FIG. 5 is a block diagram of a software architecture for the alert management system.

FIG. 6 is a block diagram of a computer platform.

Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

Referring to FIG. 1, a network based alert management system 10 includes a network 12, a network 14, and a network 16. Each of the networks 12–14 includes a number of computer systems collectively labeled 18, interconnected, for example, by an Ethernet cable 20. Each of the networks 12–14 includes security and fault monitoring systems generally labeled 22. Each security and fault monitoring system 22 is linked to an alert manager 24. The alert manager 24 is linked to one or more remote processing centers generally labeled 26. Each alert processing center 26 includes a remote management interface 36 (shown on only one center 26 by way of example). The remote management interface 36 provides a user (not shown) the capability of configuring reports produced by the alert manager 24.

The security and fault monitoring systems 22 may include, for example, intrusion detection systems, firewalls, security scanners, virus protection software, network management probes, load balancers, and network service appliances. Each of the security and fault monitoring systems 22 produces an alert stream in the form of, for example, security alerts, fault reports, and performance logs. The alert stream is sent to the alert manager 24 for collection, processing, and distribution to the remote processing center 26. Example remote processing centers 26 are security data centers, administrative help desks, and MIS stations.

In an embodiment, the remote processing center 26 subscribes to the alert manager 24 which in turns distributes specific collected and processed alert information to the remote processing center 26, more fully described below.

The networks 14, 16, and 16 being monitored by the security and fault monitoring systems 22 may include any computer network environment and topology such as local area networks (LAN), wide area networks (WAN), Ethernet, switched, and TCP/IP-based network environments. Network services occurring within the networks 12–16 include features common to many network operating systems such as mail, HTTP, ftp, remote log in, network file systems, finger, Kerberos, and SNMP. Each of the sensors 22 monitors various host and/or network activity within the networks 12–16, and each sensor 22, as discussed above, generates a stream of alerts, triggered by potentially malicious events, such as network packet data transfer commands, data transfer errors, network packet data transfer volume, and so forth. The alerts indicate a suspicion of possible malicious intrusion or other threat to operations within the networks 12–16.

The alert manager 24 includes a receive-input logic module 28. In an embodiment, the receive-input logic 28 of the alert manager 24 subscribes, i.e., establishes a transport connection, to receive each of the alert streams produced by the sensors 22 through a secure electronic communication line (SSL) 30. The alert streams contain raw, i.e., unprocessed, alerts. The monitors 22 may format their respective alert streams in a variety of formats, such as IDIP, SNMP, HP Openview, an XML-based standard format (such as the Alert Specifications from ISOTF), Common Intrusion Detection Framework (CIDF), GIAC, or some other format. The receive-input logic 28 of the alert manager 24 is equipped with translation modules 32 to translate the original, raw alert streams from the monitors 22 into a common format for further processing, if the alerts do not arrive in the common format.

In another embodiment, the monitors 22 include conversion software (not shown), also referred to as “wrapper” software that translates a monitor’s raw alert stream into the common format used by the alert manager 24. The wrapper software can add data items of interest to the alert manager 24 by querying its network 12–16.

In another embodiment, a combination of monitors 22 having wrapper software and the receive-input logic 28 preprocessing raw alerts in the alert management network 10 are present to accommodate a heterogeneous base of monitors 22 that an end-user desires to manage.

The alert manager 24 includes an alert processing engine 34. Raw alerts received by the receive-input module 28 and formatted into the common format are sent to the alert processing engine 34.

Referring to FIG. 2, an alert management process 50 residing in the alert processing engine 34 includes receiving 52 formatted alerts from the receive-input logic 28. The formatted alerts are passed 54 through user-specified filters and alerts not matching criteria of the user-specified filters are discarded.

For example, a particular end-user subscriber may be responsible only for a portion of the overall operations network and may only wish to see alerts coming from a particular subset of monitors 22, e.g., from particular ports. Each end-user subscriber can interactively define his or her own customized user-specified filters using the remote management interface 36 of the remote processing center 26, fully described below.

The filtered alerts are prioritized 56, i.e., rated or scored according to priorities dynamically controlled by the user. In an embodiment, the priority of an alert is determined by analyzing the known, (relative) potential impact of the attack category identified with respect to each of various concerns such as confidentiality, data integrity, and system availability. Confidentiality involves allowing only authorized users to view network data. Data integrity involves allowing only authorized persons to change data. System availability involves providing users access to data whenever needed with minimum downtime.

Different categories of known computer intrusions and anomalies generally pose threats with differing levels of impact on each of the above three concerns. In addition, for different users and different applications, each of the concerns may be of different relative priority. For example, in a general Internet news/search portal like Yahoo! or Lycos, continuous availability may be a more important concern than confidentiality. Conversely, for a government intelligence database, confidentiality may be a greater priority than continuous availability. For an e-commerce business site, all three concerns may be of roughly equal seriousness and priority. An ultimate priority score assigned to a particular alert for a given end-user during prioritization 56 reflects a sum or combination of the identified attack’s potential adverse impact along each of the dimensions of interest (confidentiality, data integrity, and system availability),
weighted by the end-user’s individual profile of relative priority for each such dimension.

In an embodiment, a default priority profile is provided for each user or subscriber that assigns equal priority to confidentiality, data integrity, and system availability. In a preferred embodiment, the end-user may configure the priorities dynamically and modify the default values as desired, through the remote management interface 36 that gives the user the flexibility to customize priority assignments in a manner that reflects his/her unique concerns.

In another embodiment, users (or system developers) directly assign a relative priority score to each type of attack, instead of ranking more abstract properties such as integrity or availability, that allows more precise reflection of a user’s priorities regarding specific attacks, but requires more initial entry of detailed information.

In an embodiment, users may register a listing of critical services, identified by <host ID, protocol> pairs, as to whom potential attacks or operational failures are considered to be of especially high priority.

Management and alteration of filters and listings of critical services in accordance with each of the prioritization methodologies described above can are performed dynamically and interactively while alert manager 24 is in operation and as user priorities change using the remote management interface 36.

The alerts are topology vetted 58. Vetting 58 provides a relevance rating to alerts based on the topological vulnerability of the network being monitored to the type of attack signaled by the alert. Example topologies include the computing environment, what kind of operating system (O/S), network infrastructure, and so forth. In a preferred embodiment, vetting 58 utilizes a mapping between each network host and an enumeration of that host’s O/S and O/S version(s). Vetting step 58 further preferably utilizes a topology relevance table indicating the relevance of various types of attacks to each of the different possible O/S/version environments. Thus, to determine and assign a relevance score for a particular alert, the host ID (hostname/IP address) for the target of that alert can be used to retrieve its O/S/version information, and the O/S/version along with the attack type of the alert can be used to retrieve a relevance score from the topology table.

In an embodiment, the topology table of the network being monitored is dynamically configurable by end users through the remote management interface 36.

In another embodiment, automatic local area network (LAN) mapping is provided by a network topology scope application.

The relevance of various types of known attacks against different topologies is preferably specified in predefined maps, but dynamically configured using the remote management interface 36.

Internal reports are generated 60 from the output of filtering 54, prioritizing 56 and vetting 58. Internal reports generally include fewer alerts as compared with the original raw alert stream, as a result of the user-configured filtering 40. Internal reports also tag or associate each alert with priority and/or relevance scores as a result of priority mapping 56 and topology vetting 58, respectively.

The internal reports are used to generate 62 consolidated incident reports. A consolidated incident report adds perspective and reduces information clutter by merging/combining the internal reports for multiple alerts into a single incident report. In a preferred embodiment, generating 62 is carried out through report aggregation and equivalence recognition. Aggregation refers to combining alerts produced by a single sensor, whereas equivalence recognition refers to combining alerts from multiple sensors.

The underlying notion in both cases is that nominally different alerts may actually represent a single intrusion "incident" in the real world. By analogy, a single criminal intrusion into a physical property might trigger alarms on multiple sensors such as a door alarm and a motion detector that are instrumented on the same premises, but from an informational perspective both alarms are essentially signaling the same event.

In an embodiment, alert parameters examined for report aggregation include a variable combination of attack type, timestamp, monitor identification (ID), user ID, process ID, and <IP, port addresses> for the source and target of the suspicious activity.

When an internal report is generated 60 alerts are consolidated and the corresponding priority and relevance tags for the individual alerts are merged into single meta-priority/meta-relevance scores for the single incident. Different functions may be utilized for doing the priority blend, such as additive, min/max, average, and so forth. Duration of the overall incident is also preferably computed and associated with the incident, based on the time stamps of the various individual alerts involving the incident.

The consolidated incident reports are used to generate 64 a report output. Formatting of the output report is based on subscriber-customized criteria that are defined using the remote management interface 36. The report output is transported 66 to the remote processing center 26.

Selection of a transport is under user control and managed using the remote management interface 36. The user may specify, for example, E-mail, XM, ITMI and/or writing out to a file. In an embodiment, the transport occurs over an SSL for display and assessment by the end-user.

The filtering 54, prioritization 56 and topology vetting 58 are event driven, i.e., each alert is processed and filtered/tagged as it arrives, one alert at a time. However, temporal clauses are utilized for aspects of report aggregation and equivalence recognition among multiple alerts. For example, as internal reports are generated 60 a sliding window is established during which additional records may be merged into the aggregate incident report. A single alert internal report may be sent to the remote processing center 26 indicating that it has witnessed the alert. A subsequent aggregate alert report, i.e. an incident report, covering that single alert as well as others, may also be forwarded to the remote processing center 26 to indicate a duration of the attack/incident, an aggregate count of individual alerts representing this incident, and an aggregate priority. In an embodiment, aggregate alert flushing may occur after some period of inactivity (e.g., "two minutes since last event"). The aggregate alert flushing is not event driven, but rather driven by an internal timeout recognized from a system clock (not shown) of the alert manager 24.

Referring to FIG. 3, an exemplary priority database record 80 used for prioritization 56 of filtered alerts includes example network attacks ping of death 82, buffer overflow 84 and write poll violation 86. For each of the attacks 82-86, a relative priority rating is assigned, namely, denial of service (system availability) 80, data integrity 90, and security (confidentiality) 92. By way of example, a first end-user 94 weighs denial of service at 0%, data integrity at 20%, and security at 80%. A second end-user 96 weights denial of service at 80%, data integrity at 10% and security...
at 10%. Thus, for the priority database record 80, the user 94 emphasizes a high concern (priority) with security, while the user 96 emphasizes a high concern (priority) with denial of service.

In this example, for first user 94 a "ping of death" alert 82 will have a priority score = (0.5*0)+ (0.2*10)+ (0.8*10)=27; whereas for second user 96 a "ping of death" alert 82 will receive a priority score = (0.8*0)+ (1*10)+ (0*1)+ (0.8*10)=74.

As is seen from the description above, (a) it is the relative value of these priority scores that has significance, not the absolute magnitudes, and (b) the priority values for alerts and for user preferences are subjective values that may vary from one application to another and from one user to another. As noted above, the alert priority map values and user priority profiles may be dynamically adjusted and customized by individual users via remote management interface 36.

Referring again to FIG. 1, the report output of the alert processing process 50 is stored at the remote processing center 26 in a database 38 contained in a storage device 40 for retrieval and reporting by the end user. In an embodiment, the report output is translated at the remote processing center 26 in accordance with a user-configurable database schema into an existing/legacy database management system (not shown) contained in the remote processing center 26 for convenience of the end-user, either manually by a database integration team or automatically using a database mediator/translator. The remote management interface 36 accesses the database management system and presents the report output to the end-user, such as by a graphical user interface (GUI) on a workstation 42.

In an embodiment, the alert management network 10 provides an open, dynamic infrastructure for alert processing and management. The alert manager 24 preferably includes functionality for dynamically generating, suspending, and configuring data connections and logical process modules, in response to interactive user commands issued via remote management interface 36. The remote management interface 36 preferably executes a Java application that generates command files, in response to end user requests, in the form of directives and any necessary data files, such as the priority database record 80, and so forth. The Java application communicates, e.g., via telnet, to the alert manager 24 and downloads the directives and data files. The alert processing engine 34, preferably a postscript interpreter in one embodiment, can process the directives dynamically. Many of the directives are preferably defined in terms of postscript code that resides locally in a library 44 in the alert manager 24. Applications running in alert manager 24 are written in modular fashion, allowing directives to accomplish meaningful change of logical behavior by instructing the alert manager 24 to terminate a particular process clause and activate a newly downloaded clause, for example.

By way of another example, through the operation of the alert processing engine 34 the alert manager 24 can dynamically establish and suspend connections to the various alert streams generated by the security and fault monitoring systems 22. Thus, the alert manager 24 can dynamically "plug into" (i.e., connect) new alert streams, such as alert streams from additional sensors newly deployed by an end-user, and likewise can dynamically suspend (permanently or temporarily) a connection to alert streams from sensors 22 that are removed, replaced, or taken offline, and so forth. Similarly, alert manager 24 can dynamically generate or suspend modules of the alert management process 50, and can dynamically adjust the configurable parameter settings of those modules.

In this manner, alert manager 24 is designed to be responsive to dynamic configuration requests initiated by end users using the remote management interface 36 of the remote processing center 26. As mentioned above, the remote management interface 36 provides an interactive interface at workstation 42 for end-users to specify desired modifications to the dynamically configurable aspects of alert manager 24.

Referring to FIG. 4, a block diagram of a software architecture 100 for a dynamic, open, alert management infrastructure in accordance with preferred embodiments of the present invention is shown. An infrastructure module 102 (labeled "eFlowgen") provides core infrastructure functionality, including implementation of the alert processing engine 34, and need not be specialized to alert management applications. An inline application code module 104 (in conjunction with an initialization module 106, described below) defines an alert management application, including the overall alert analysis and reporting process 50 described above with reference to FIG. 2. Initialization script module 106 complements application code module 104, by defining, for a particular application instance, the specifics of the input/output transport connections and specifics of the logical alert processing clauses corresponding to the process 50. A dynamic definitions module 108 represents dynamic changes submitted by users via the remote management interface 36, such as configuration changes and other extensions as previously discussed; the functionally dynamic definitions module 108 are comparable to initialization script module 106, except for being dynamically submitted and incorporated into the running application.

A detailed functional specification for a software infrastructure corresponding to eFlowgen module 102 is included in the Appendix, incorporated herein.

In another embodiment, referring to FIG. 5, the remote processing center 26 includes a correlation logic engine 110. The correlation logic engine 110 accesses and compares incident reports in database 38 and attempts to provide intelligent assistance to end-users in the analytical task of discovering patterns and making sense of alert data. The correlation logic engine 110 looks for key attribute relations in common for different incidents, such as incidents targeting a single host machine over a relatively short time frame, or incidents reflecting attacks or anomalies coming from a particular source machine. Automatically correlating separate incidents helps end-users recognize more quickly that a particular machine is under serious attack or that some other machine is a hostile "bad guy," for example, and the end-users can then take appropriate defensive action.

Another correlation technique residing in the correlation logic engine 110 looks for interrelated vulnerabilities, applying rule-based knowledge to look for groups of distinct incidents that can inferentially be interpreted as related parts of a single, coordinated attack. For example, rules matching patterns of incidents that look like a chain over time, where the target of an earlier incident becomes the source of a subsequent incident, may allow correlation logic engine 110 to conclude that these likely are unrelated incidents, and that a "worm" infection appears to be spreading.

In an embodiment, the correlation logic engine 110 incorporates statistical inferential methods. The correlation logic engine 110 attempts to draw conclusions automatically based on received intrusion incident reports. The correlation logic engine 110 produces reports for the end-user indicating correlation found...
The alert manager 24 and other components of the alert management network 10 may be implemented and executed on a wide variety of digital computing platforms, including, but not limited to, workstation-class computer hardware and operating system software platforms such as Linux, Solaris, FreeBSD/Unix, and Windows-NT.

Referring to FIG. 6, a computer platform 120 suitable for hosting and executing the alert management process 50 includes a display device 122 connected to a computer 124. The computer 124 includes at least a memory 126 and a central processing unit (CPU) 128. The computer 124 includes a link to a storage device 130 and a network link 132.

The storage device 130 can store instructions that form an alert manager 24. The instructions may be transferred to the memory 126 and CPU 128 in the course of operation. The instructions for alert manager 24 can cause the display device 122 to display messages through an interface such as a graphical user interface (GUI). Further, instructions may be stored on a variety of mass storage devices (not shown).

Other embodiments are within the scope of the following claims:

What is claimed is:

1. A computer-implemented method of managing alerts in a network comprising:
   receiving alerts from network sensors;
   consolidating the alerts that are indicative of a common incident; and
   generating output reflecting the consolidated alerts.

2. The computer-implemented method of claim 1 further comprising formatting the alerts into a standard alert format.

3. The computer-implemented method of claim 2 wherein formatting the alerts into a standard alert format is performed by the network sensors.

4. The computer-implemented method of claim 2 wherein formatting the alerts into a standard alert format is performed by input-receiving logic of an alert management system.

5. The computer-implemented method of claim 2 wherein the alert format is selected from the following group of formats: [IP, SNIP, HP Openview, Attach Specification, CIDF, XML].

6. The computer-implemented method of claim 1 further comprising tagging the alerts with corresponding significance scores.

7. The computer-implemented method of claim 6 wherein the significance scores comprise a component indicating a priority measure for the corresponding alerts.

8. The computer-implemented method of claim 7 wherein the priority measure is derived using a priority map.

9. The computer-implemented method of claim 8 wherein the priority map is dynamically adjustable.

10. The computer-implemented method of claim 8 wherein the priority map comprises relative priority scores for resource availability, resource integrity, and resource confidentiality.

11. The computer-implemented method of claim 7 wherein the priority measure is derived based upon criticality of one or more resources targeted by the corresponding alerts.

12. The computer-implemented method of claim 6 wherein the significance scores comprise a component indicating a relevance measure for the corresponding alerts.

13. The computer-implemented method of claim 12 wherein the relevance measure is derived based upon a consideration of an operating environment topology for a target of an attack signaled by the corresponding alert.

14. The computer-implemented method of claim 13 wherein the relevance measure is derived using one or more topology tables.

15. The computer-implemented method of claim 14 wherein one or more elements of the topology tables are dynamically adjustable.

16. The computer-implemented method of claim 14 wherein one or more elements of the topology tables are automatically generated.

17. The computer-implemented method of claim 14 wherein the one or more topology tables comprise:
   a mapping between one or more network hosts and one or more corresponding environment features selected from the following group: {operating systems (o/s), o/s versions, hosted services/applications}; and
   a relevance rating for each of one or more types of attacks mapped against the corresponding environment features.

18. The computer-implemented method of claim 6 wherein the output reflecting the consolidated alerts includes a meta-significance score reflecting a blending of the significance scores for each of the consolidated alerts.

19. The computer-implemented method of claim 1 further comprising correlating common incidents.

20. The computer-implemented method of claim 1 further comprising filtering the alerts.

21. The computer-implemented method of claim 20 wherein filtering comprises comparing the alerts to user-specified filters.

22. The computer-implemented method of claim 21 wherein the user-specified filters are dynamically configurable.

23. The computer-implemented method of claim 1 wherein the consolidated alerts comprise alerts produced by a single network sensor.

24. The computer-implemented method of claim 1 wherein the consolidated alerts comprise alerts produced by different network sensors.

25. The computer-implemented method of claim 1 wherein consolidating the alerts further comprises identifying the alerts that are indicative of a common incident based upon one or more alert parameters selected from the following group: {attack type, timestamp, network security component identification (ID), user ID, process ID, IP address, port addresses for a source and a target of a suspicious activity}.

26. The computer-implemented method of claim 1 wherein generating output comprises generating one or more subscriber-specific reports.

27. The computer-implemented method of claim 26 wherein the subscriber-specific reports are based on one or more subscriber-customizable criteria.

28. The computer-implemented method of claim 27 wherein the subscriber-customizable criteria are dynamically configurable.

29. The computer-implemented method of claim 27 wherein the subscriber-customizable criteria comprise one or more transport options.

30. The computer-implemented method of claim 29 wherein the transport options comprise one or more options selected from the following group: {E-mail, XML, HTTPS, writing out to a file}.

31. The computer-implemented method of claim 1 wherein the output is automatically input to a data base management system.

32. The computer-implemented method of claim 1 further comprising sending the output over one or more secure communications links to one or more subscribers.
33 The computer-implemented method of claim 1 wherein receiving alerts further comprises dynamically modifying a set of network sensors from which the alerts are received.

34 The computer-implemented method of claim 1 wherein the network sensors comprise heterogeneous network sensors.

35 The computer-implemented method of claim 1 wherein the received alerts include one or more filtered alerts.

36 The computer-implemented method of claim 1 wherein the received alerts include one or more alerts tagged with corresponding significance scores.

37 The computer-implemented method of claim 1 wherein the received alerts include one or more consolidated alerts, as to which the method of claim 1 is applied in recursive fashion.

38 The computer-implemented method of claim 1 further comprising processing the alerts to produce one or more internal reports, and wherein consolidating comprises consolidating the internal reports that are indicative of a common incident to produce one or more incident reports.

39 A computer program, residing on a computer-readable medium, comprising instructions causing a computer to:

- receive alerts from a plurality of network sensors;
- consolidate the alerts that are indicative of a common incident;
- and generate output reflecting the consolidated alerts.

40 The computer program of claim 39 further comprising instructions causing a computer to:

- format the alerts;
- filter the alerts; and
- tag one or more of the alerts with corresponding significance scores.

41 The computer program of claim 39 wherein the network sensors comprise heterogeneous network sensors.

42 In a computer network that has a plurality of security or fault monitoring devices of various types, each of which generates an alert when an attack or anomalous incident is detected, a method for managing alerts comprising the steps of:

- ranking network resources and services based on their actual or perceived importance to effective operation of the network;
- receiving alerts from the security or fault monitoring devices;
- ranking each alert based on a potential or actual impact of each alert's underlying attack or incident on effective operation of the network;
- grouping alerts that may relate to a common attack or incident; and
- generating a report that shows at least a subset of the alert groups and that indicates a potential or actual impact of each alert group's underlying attack or incident on effective operation of the network.

43 In a computer network that has a plurality of security or fault monitoring devices of various types, each of which generates an alert when an attack or anomalous incident is detected, a method for managing alerts comprising the steps of:

- ranking network resources and services based on their actual or perceived importance to effective operation of the network;
- receiving alerts from the security or fault monitoring devices;
- ranking each alert based on a potential or actual impact of each alert's underlying attack or incident on effective operation of the network;
- grouping alerts that may relate to a common attack or incident; and
- generating a report that shows at least a subset of the alert groups and that indicates a potential or actual impact of each alert group's underlying attack or incident on effective operation of the network.

44 In a computer network that has a plurality of security or fault monitoring devices of various types, each of which generates an alert when an attack or anomalous incident is detected, a method for managing alerts comprising the steps of:

- receiving alerts from the security or fault monitoring devices;
- grouping alerts that may relate to a common attack or incident;
- ranking each alert group based on a potential or actual impact of each alert group's underlying attack or incident on effective operation of the network; and
- generating a report that shows at least a subset of the alert groups and that indicates a potential or actual impact of each alert group's underlying attack or incident on effective operation of the network.

45 The method of claim 44 wherein the security or fault monitoring devices are selected from the following group of devices:

- firewalls;
- intrusion detection systems;
- antivirus software;
- security scanners;
- network management probes;
- network service appliances;
- authentication services; and
- host and application security services.

46 The method of claim 44 further comprising the step of identifying critical network resources and services.

47 The method of claim 44 further comprising the step of ranking network resources and services based on their actual or perceived importance to effective operation of the network.

48 The method of claim 44 further comprising the step of identifying a set of alert classes or types.

49 The method of claim 48 wherein the set of alert classes or types is selected from the following group:

- privilege subversion;
- use subversion;
- denial of service;
- intelligence gathering;
- access violations;
- integrity violations;
- system environment corruption;
- user environment corruption;
- asset distress; and
- suspicious usage.

50 The method of claim 49 further comprising the step of ranking the alert classes or types based on actual or perceived impact of the underlying attacks or incidents on effective operation of the network.

51 The method of claim 44 wherein the alerts are grouped based on alert attributes selected from the following group:

- common source;
- common connection;
- common host-based session;
In a computer network, a method for ranking alerts that are indicative of an attack or an anomalous incident, the method comprising the steps of:

5. Identifying and ranking different types of attacks or incidents according to their actual or perceived impact on effective operation of the network;

identifying and ranking network resources or services according to their actual or perceived importance in effective operation of the network;

determining vulnerability of network resources to different types of attacks or incidents; and

assigning a relevance score to an alert based on the type of the underlying attack or incident, the target of the attack or incident, and the vulnerability of the target.

6. The method of claim 5 wherein an attack outcome measurement is also used in the assignment of the relevance score.

7. The method of claim 5 wherein the relevance score is assigned in part by associating dynamically adjustable weights with the type of the underlying attack or incident, the target of the attack or incident, and the vulnerability of the target.
EXHIBIT D
March 31, 2004

VIA FEDERAL EXPRESS

Mr Richard Macchia
Sr Vice-President and
Chief Financial Officer
Internet Security Systems, Inc
6303 Barfield Road
Atlanta, Georgia 30328

Re  SRI Network Security Patents

Dear Mr Macchia

I am the VP of Legal and Business Affairs and General Counsel of SRI International. For over 50 years, SRI has performed research and development work for thousands of government and commercial clients across a wide variety of fields, including information security. As discussed more fully below, SRI has a number of fundamental patents in the network-based intrusion detection and prevention area, and we would like to explore with you a productive and mutually-beneficial approach to realizing their value.

A little background is probably in order. Over the last decade, SRI has done extensive work for the government in the area of cybersecurity and network-based intrusion detection and prevention. Based upon that work, SRI now has five issued patents and several other pending applications, which cover, among other things, network-based surveillance, hierarchical event monitoring and analysis, and alert detection and management.

SRI's patents in this area have record priority dates going back to 1998. The scope and breadth of their claims vary; some are quite broad, while others are more narrowly focused on particular solutions. Based on our analysis of the prior art, as well as the PTO's approval after review of extensive prior art, we believe our claims to be valid.

SRI believes that a significant number of network-based security products, both software and appliances, infringe multiple claims of SRI's patents. We intend to embark on a
major licensing program, focusing on firewall, IDS and integrated network security software and appliances from major vendors in the field.

ISS is one of the companies to whose products -- including without limitation its Proventia™ integrated gateway and network security products and its RealSecure™ network security products — our patents are relevant. Other ISS products may also be implicated, though we haven't yet done the analysis.

We would like to discuss approaches to this situation that have the potential to be mutually beneficial both to ISS and to SRI. In that spirit, and after you and your counsel have had an opportunity to review the patents and their file histories, we would suggest that a meeting be arranged to discuss this matter face-to-face. We would agree, of course, that anything said by either SRI or ISS at such a meeting would be confidential and inadmissible.

I am enclosing copies of SRI's issued patents in this area. The file histories are easily available from the U.S. Patent Office. If you are unable to find any of the cited prior art, let us know and we would be happy to provide you with a copy.

I look forward to hearing from you, and to discussing this matter with you and your team in greater detail. Please be advised that we have initiated discussions regarding this matter with another company, and that there may be only a narrow window during which arrangements involving any kind of exclusivity may be available. If ISS is interested in engaging in discussions of such a nature, therefore, it would be useful to commence those discussions as promptly as possible.

Very truly yours,

Richard H. Abramson
VP Legal & Business Affairs

RHA/rhr

Enclosure
EXHIBIT E
April 28, 2004

Mr. Richard Macchia
Sr. Vice-President and
Chief Financial Officer
Internet Security Systems, Inc.
6303 Barfield Road
Atlanta, Georgia 30328

Re: SRI Network Security Patents

Dear Mr. Macchia

I have received no reply to my letter to you of April 2, 2004. In that letter, SRI notified ISS that ISS is infringing a number of patents owned by SRI, and proposed that representatives of our companies meet in order to determine whether a mutually beneficial solution might be reached.

Please let me know at your earliest opportunity whether ISS wishes to engage in such a discussion. While we understand that a review of the subject patents takes some time, SRI takes intellectual property matters such as this one very seriously, and would like to resolve this situation as expeditiously as possible. I therefore hope to hear from you in the near future. In the meantime, SRI reserves all of its rights in connection with this matter.

Sincerely,

[Signature]

Richard H. Abramson
VP Legal & Business Affairs

RHA/rhr

SRI International
333 Ravenswood Avenue ● Menlo Park, CA 94025
Phone 650 859 6068 ● Facsimile 650 859 3834 ● E-mail richard.abramson@sri.com