ELSEVIER

Computer Networks 31 (1999) 2465–2475

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

A large scale distributed intrusion detection framework based on attack strategy analysis

Ming-Yuh Huang *, Robert J. Jasper, Thomas M. Wicks

Applied Research and Technology, The Boeing Company, Rm 27C1, P.O. Box 3707, M/S 7L-20 Seattle, WA 98124-2207, USA

Abstract

This paper describes a large-scale distributed intrusion detection (ID) architecture based on intrusion detection system (IDS) agents and collaborative attack strategy analysis. This architecture couples distributed IDS agents performing local event analysis with cooperative global ID. Other agent-based approaches have highlighted several advantages over monolithic architectures. This approach specifically focuses on cooperative IDS agents working together by analyzing the intruder's attack strategy and separating local event processing from global analysis. We believe that focusing on the intruder's intent (attack strategy) provides a theme that will help distributed IDS to work together. Furthermore, strategy analysis creates an opportunity for IDS agents to pro-actively look ahead for data most pertinent to current case development. This look ahead adaptive auditing behavior focuses limited system resources on collecting and auditing those events which are most likely to reveal intrusions.

Keywords: Security; Intrusion; Detection; Strategy; Agents

1. The difficulty of large scale distributed intrusion detection

Developers of large-scale distributed intrusion detection (ID) software face many difficult challenges:

- widely distributed and heterogeneous environment,
- voluminous, noisy and volatile data,
- incomplete information for decision making,
- diverse sensor technology,
- difficulty in communication, coordination, command-and-control,
- trust between intrusion detection system (IDS) agents,
- changing attack patterns.

IDS employ a variety of techniques. *Misuse detection* techniques recognize common attacks by comparing on-going activities against patterns of past intrusions. *Anomaly detection* techniques work by constructing normal operation profiles and discovering significant deviations to detect suspicious activities [12]. The techniques typically focus on analyzing local events in isolation and therefore tend not to scale to large, dynamic environments under coordinated attacks. In this paper, we propose a new approach focused on addressing these issues.

* Corresponding author: Tel.: +1-425-865-2490; fax: +1-425-865-2966.

E-mail addresses: ming-yuh.huang@boeing.com (M.-Y. Huang), robert.jasper@boeing.com (R.J. Jasper).

1389-1286/99/$ - see front matter
PII: S1389-1286(99)00114-0

2. Whats missing in todays centralized, one way event processing architecture

Large-scale heterogeneous networks generate large amounts of temporal event data in diverse formats. Much of this data is unrelated to security but represents system and network faults. An IDS must carefully distinguish between security and non-security data to be effective.

For example, determining the severity of a "file-access-violation" event requires contextual information. The violated file could be a regular user file on a desktop workstation, a critical file on a sensitive data server, or an important configuration file for a gateway machine. In addition, machine configuration at the time of event generation is also relevant. The locations of the files, auditing parameters, setup for applications/users/groups, and many other system attributes vary from system to system over time. These factors may affect the analysis outcome. In fact, system/network misconfiguration accounts for large percentage of the false-positives in many IDS. For example, NT and Unix have different ways of establishing user groups. NT allows users to set up an access control list (ACL) for individual files based on user defined *customized groups*. Consequently, certain NT group-oriented "file-access-violation" carries a very different meaning from that of UNIX's. To interpret this information correctly, IDS actually needs to know the source of the event. For example, is it an NT event based on a user defined customized group?

Today's centralized data/event gathering and processing ID architecture relies on a passive and one-way information-processing approach. This approach uses sensors for data gathering placed at multiple locations in the network. More advanced sensors might have the capability to receive commands or down load attack patterns from the central control system. Nevertheless, much of the analysis and correlation are done at a centralized location. This architecture faces considerable drawbacks when applied to sophisticated attacks on large and complex networks which places too much of a burden on the central machine.

By the time data arrives at the centralized location the contextual information needed to properly analyze the event has already been lost. That information existed only in the original environments where the data was generated. Worse even, the time latency might have made it impossible to go back and collect critical environmental information to confirm or exonerate the suspicions. Another almost insurmountable challenge is the difficulty in correlating voluminous distributed events. Lost context and the difficulty in translating/mapping event formats causes the potential loss of significance of many events. Without the right interpretation, it is unlikely that an IDS can perform an adequate job.

Pattern recognition also becomes more difficult due to the large number of attack methods. Each method could manifest itself in a different set of events/formats when applied in a different environment (e.g., operating system and network). The number of potential permutations is large. The addition of system/network fault events further complicates the situation. Detecting intrusion, from both misuse detection and anomaly detection perspectives becomes extremely difficult. Therefore, *false positives* and *false negatives* become a real problem.

3. The basic anatomy of an intrusion

Today's intruders respect no boundaries. Intruders typically do not target just one machine or one network segment. An intruder also does not just damage systems and go away. Attacks cut across multiple machines and network segments by using several hacking methods and tools. Unfortunately, real life observations typically showcase only a small window of the entire attack process.

Intrusion attack steps do not occur in random order. An intruder usually starts with a set of tools to probe around and gather information. The selection of the hacking tools and the order of application depend heavily on the situation, data gathered, and the responses from the targeted system.

Fig. 1 shows how a typical *IP-spoofing* could take place. In this scenario, intruder host ***I*** wants to masquerade as ***A*** while talking to Victim ***V***.

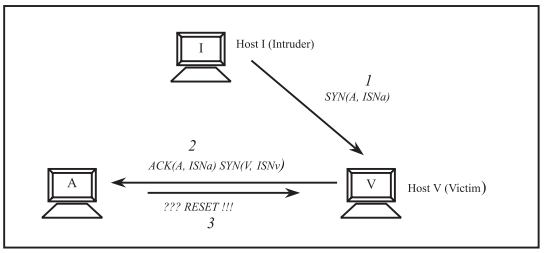

Fig. 1. A typical IP-spoofing.

1. ***I*** first sends an initialization request to ***V*** announcing its intent to talk while pretending to be ***A***.
2. In response to the request presumably coming from ***A***, ***V*** replies with an acknowledgment for a connection with ***A***.
3. The problem that ***I*** faces now is that when ***A*** sees an acknowledgment to a request not originated by itself, it will send out a reset.
4. This reset causes ***V*** to drop the connection with ***I*** thus preventing the spoofing from taking place.

To prevent ***A*** from reset, ***I*** starts with a *SYN-flood* attack against ***A*** (Fig. 2, step 1). This will keep ***A*** so busy that ***V***'s acknowledgment (step 3) will simply be ignored. Therefore, ***A*** sends no reset and ***I*** achieves the goal of spoofing.

This simple example illustrates the logical sequence of an attack with two single steps. This scenario can be extended further with the following steps:

1. Use SYN-flood to accomplish IP-spoofing, as in Fig. 2.
2. Use IP-spoofing to accomplish a Session Highjack, Telnet Highjack, Web-Spoofing. (Better yet, use tools such as *Hijack* or *TTYWatcher* that will do both, with a GUI.)
3. Use Session Highjacking to obtain system data.
4. Obtain encrypted password file for off-line password cracking (*Crack*, *L0phCrack*).
5. After entering system, hide one's presence (use *RootKit*).
6. After accessing system, spread out to other systems. (sniffing for passwords)
7. Install Trojan Horse (*ifconfig*) to hide the PROMISC flag. (since sniffing will cause the PROMISC flag to become on)
8. Modify file date/checksum (*fix*) to hide the Trojan Horse.
9. After accessing system, install a backdoor (use *RootKit*).

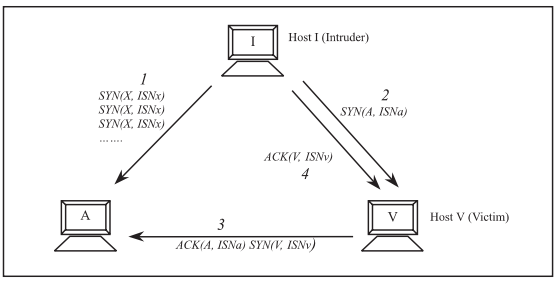

Fig. 2. SYN-flood ***A*** before IP-spoofing.

There are always multiple ways to invade a system. However, attackers follow steps and apply tools in a logical sequence. An attacker often starts with information gathering by probing around. Attackers first apply cautious but logical steps to explore the environment. Depending on the findings, the attacker follows another set of logical steps and applies tools to exploit the weakness. Our approach exploits this partial ordering of events to reveal the invasion.

4. Strategizing large scale distributed intrusion detection

Addressing the large scale distributed intrusion assessment/detection requires division of work and coordination amongst IDS components. An effective approach would exploit autonomous local problem resolution coupled with cooperative global problem solving.

> We believe that focusing on intruder's intent (attack strategy) provides a theme that allows distributed IDS agents to work together. We also believe that by analyzing the intruder's attack strategy, IDS agents will be able to perform pro-active look-ahead adaptive auditing.

The first indication of an intrusion or suspicious event provides window of opportunity for

discovery. On the other hand, this is also a time for generating false-positives. The large number of false-positives is a realistic problem in the real-life IDS deployment. To reduce the false-positives, it makes sense to adjust the focus of attention based on the prediction of the intruder's intent to monitor the possible attack directions instead of just generating more alarms.

In a criminal case, detectives must establish both criminal intent and the supporting evidence. Intention analysis often leads the way of criminal investigation and evidence gathering. Likewise, attack strategy analysis facilitates the ID process.

5. Strategy analysis based architecture

We propose an architecture based on the following approach:

1. Local IDS agents monitor and look for suspicious events.
2. When local IDS agents detect suspicious events they determine the substantiation of attack step(s) by mapping from the events within the local context.
3. Local IDS agents report to global IDS agent when an attack step is suspected of being substantiated.
4. A global IDS agent collects substantiated attack step(s) and correlates against pre-existing attack strategy model.
5. A global IDS agent determines possible attack threads, pass to local IDS agents for verification.
6. Local IDS agents examine possible attack threads and determine the relevant attack steps within the threads.
7. Local IDS agents look for evidences supporting relevant attack steps.
8. Local IDS agents report newly substantiated attack steps.
9. Continue global and local IDS collaboration until the investigation is concluded.

An essential feature of this architecture is an analysis that reveals the intruder's attack strategy (or intention). This analysis predicts potential intrusion behaviors based on the sequence of past events. Because the analysis is performed at the strategy level, the result is platform independent.

At this level, attacks are characterized by a sequence of logically related but not necessarily complete *attack steps – attack sub-goals*. Each step represents a state of accomplishment by the intruder in executing an intrusion (the final goal). Intrusion detection thus tries to recognize the attack sub-goal completion and trend analysis, in addition to the glaring violations.

Attack sub-goals usually map into different events in different environments. To determine if a sub-goal has been substantiated by the intruder, autonomous local agents that understand the native event format examine the local events within the local context to make such a decision. In many cases, it may take several events to substantiate that an intruder has executed and accomplished an attack step.

6. Representation of attack strategy

An effective representation format needs to be generic enough so that sharing is not a problem amongst IDS of different implementations. It also makes sense to build on top of upcoming industry standards such as internet engineering task force's (IETF) ID Working Group, ANSI/T4, or DARPA's common ID framework (CIDF) work.

We examined different alternative of representing attack strategy. Petri Nets are powerful in representing parallelism but too complex to manage for large models. We also considered deterministic finite state machines but ruled them out because of the need to represent probability and hierarchy. We chose goal-trees as a simple and nature way to represent attack strategies.

The root of the tree represents the goal of intrusion and lower level nodes represent alternatives or attack sub-goals (ordered or unordered) to achieve the upper level goal. Nodes do not represent security events, but a mapping between security events and nodes. Security events substantiate nodes in the tree.

The standard goal-tree representation does not have the capability to represent temporal sequence. We have augmented goals trees to

represent sequences explicitly. Fig. 3 illustrates the three basic constructs of our goal-tree representation. The OR construct denotes that completion of any sub-goal achieves the upper goal. Similarly, the AND construct denotes that all sub-goals must be complete for the upper goal, and for the ORDER-AND construct, all sub-goals in a particular order for the upper goal. Specifically, in the case of the OR construct branches can be weighted to designate the likelihood of the sub-goals being used. The heavier the weight, the more likely the alternative sub-goal is for the intruder to use to carry out the high level goal.

Fig. 4 illustrates an example attack strategy. Some nodes on this tree have already been filled. A filled node designates that the particular attack sub-goal was substantiated. In another word, somewhere along the line, some IDS agents examined local security events (Fig. 4 local events) and determined that the attacker has achieved that particular attack sub-goal.

Nodes are filled as distributed IDS agents examine their local data. At any given time, a representation could have some filled nodes that can be linked together to form one or more horizontal threads cutting across the structure (Fig. 4). In most cases, these threads are incomplete and have hole(s) in them.

A thread represents the progression of an attack following a logical sequence (*attack thread*). Holes along the path signify data needed to substantiate the attack sub-goal that were either over-looked, not audited, cannot be audited or too weak to support the substantiation. Node X and Y in Fig. 4 represent windows of opportunity to go back and re-examine the archived data to confirm or exonerate the suspicion. A representation of a flooding/spoofing/sniffing attack is shown in Fig. 5.

The difficulty of event correlation in ID is also well illustrated here. With all the permutations of possible attack threads, each attack step in the threads can be further substantiated by different events in different system. Thus, the event level permutation is so large that any kind of comprehensive pattern representation is almost impossible.

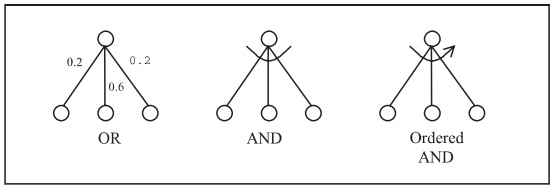

Fig. 3. The OR, AND and Ordered-AND constructs for intrusion intention goal-tree.

7. Analysis of attack strategy

From a tree with partially filled nodes, IDS agent can perform analysis to determine the most plausible attack threads. It is like a field marshal analyzing a partially marked military intelligence map to figure out the enemy's *attack agenda* and next moves.

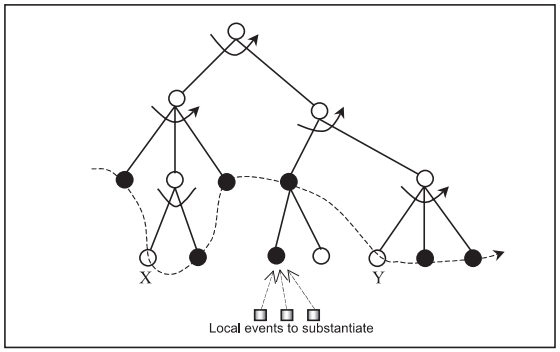

Fig. 4. An incomplete suspicion in development.

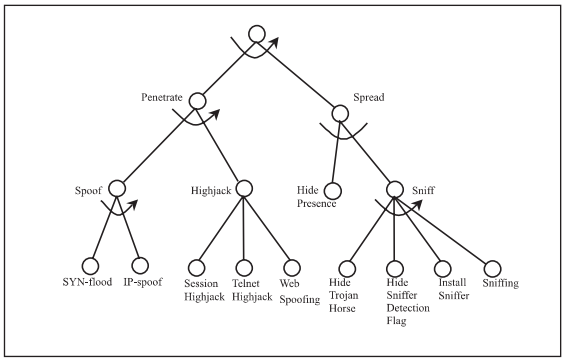

Fig. 5. The sniffing and hiding strategy.

7.1. Length analysis

Threads with the larger number of substantiated nodes deserve more attention. Literally, this translates into intrusion scenarios with a larger amount of supporting evidences (Fig. 6).

In reality, it is unlikely that all the nodes on the thread are substantiated. Under such circumstances, the global IDS agent can request local IDS agents for verification by passing along the entire suspected attack thread(s). When thread(s) are passed in such a manner, they carry the global agent's investigative intention (in terms of the attacker's attack agendas) – similar to the orders given on the battleground.

Holes (unsubstantiated attack steps) on the attack threads can either be verified or exonerated by the local IDS agents. Local IDS agent does this within the local context and passes the results back for global agent's re-evaluation.

7.2. Weight analysis

There are two types of weights associated with an attack thread. First is the branch weight from the OR constructs. Attack threads with nodes hanging from the heavier OR branches deserve more attention. This means the intrusion is taking a more plausible development, from an intruder's perspective.

Second weight is the supporting evidences. When the nodes are substantiated, different degrees of supporting evidences determines the substantiation certainty. In the strategy representation, this certainty factor is represented by the different shades of darkness on the node. Darker nodes are better supported (Fig. 7). Therefore, darker threads are more plausible, from the supporting evidences perspective.

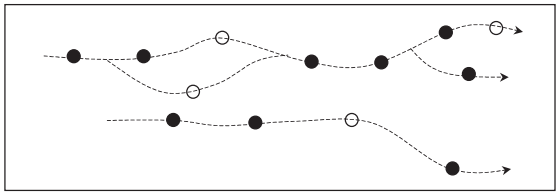

Fig. 6. Different lengths of attack threads.

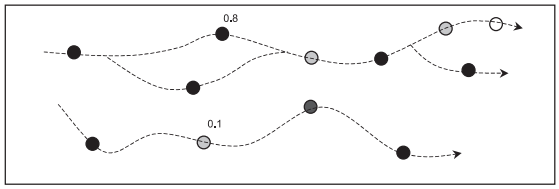

Fig. 7. Threads of different weights.

7.3. Temporal analysis

When a node is substantiated, the supporting evidence carries a timestamp. This marks the time of node substantiation. By examining the node substantiation time along the path, one has a good idea of the temporal aspects of the attack development. More recent timestamps indicate an increased likelihood of the development. Using a half-life-decaying scheme similar to that in NIDES/EMERALD [10], total thread temporal value can be calculated from the individual temporal values of individual nodes.

It is important to note that a temporal analysis needs to be tied with attributes such as machine id, user, or process in the offending event. Temporal analysis makes sense only when it is correlated against specific common attributes.

Individual node temporal values are also used to determine the inclusion to the thread. In other words, if the node is too far out, a more recent attack thread might not include it. However, a longer attack thread of lower probability might still include such a node.

8. Look ahead adaptive auditing and high level recognition

IDS agents communicate through substantiated attack steps and threads that carry an intruder's attack agenda. The threads allow local agents to determine the best method to use in responding. It also provides an opportunity to fine-tune system auditing to pro-actively look for relevant data. The forks of the threads represent such opportunities (Figs. 6 and 7). Later, when meaningful data is observed, local IDS agents report the attack subgoal (attack step) substantiation with supporting

data. With this development, global IDS agents re-evaluate the situation and issue new threads. Helpful local IDS agents again look for data relevant to the development.

Since local IDS agents report back to the centralized agent with attack sub-goal substantiation, the global agent can recognizes attacks at the strategic level without having to figure out all the details embedded in the large amount of local data. Under such circumstances, the high noise ratio and the data volume become less of a problem.

This delegation of responsibility achieves a higher level of efficiency but requires an equally higher level of communication protocol. IDS can never replace human experts. We approach IDS as the first line of defense and recognizing attack intention provides a good alert timing.

9. Integration of paradigms, tools and technologies

One major reason for taking this approach is the integration amongst IDS implemented in different paradigms, products, tools and technologies.

Misuse and anomaly detection techniques use different data to perform different processing. Different computing technologies are suited for different data and problems. Neural networks excel at analyzing data with a high noise-to-signal ratio data. Therefore, it might be more suitable than say, expert system, for some anomaly detection problems. Large-scale ID deployment will likely require varieties of systems and tools. On top of that, attack methods evolve as new systems emerge. It is resource intensive for any single IDS to keep up with all the new attacks patterns. From that practical IDS deployment perspective, it is important to be able to make use of and integrate a variety of IDS.

Under the theme of attack strategy analysis, IDS of different paradigms and technologies can work together and contribute to pursue of suspected attack agendas. Differences in data, operating environments is less important and the total problem space is greatly reduced. Critical issues thus become understanding the attack logic and a high-level communication protocol to support the collaborative process.

10. Agent organization and cooperation

In this architecture, IDS agents are software entities that function continuously and autonomously (i.e., independent of constant human intervention) in an environment containing other processes and IDS agents. Beliefs, capabilities, and commitments form a basis for structuring and describing an agent's state.

An agent-based architecture provides several of advantages over a centralized approach to ID including tailorability, trainability, efficiency, fault tolerance, graceful degradation, extendibility, and scalability [4,5]. In addition to these advantages, our architecture specifically exploits agent cooperation. We categorize IDS agents into local agents (tailored for information gathering and local analysis) and global agents (tailored for strategy analysis, and command and control). Each of these categories can be further specialized (e.g., for specific operating systems, different detection approaches).

The architecture defines an agent arrangement based on a loose hierarchy where local IDS agents monitor and analyze local events. Local monitoring provides these agents a rich context from which to interpret and analyze events and detect probable attack steps. Local agents inform global agents of these steps using a machine and context independent format. This isolates the global agents from having to understand machine specific data or understanding local context (see Fig. 8).

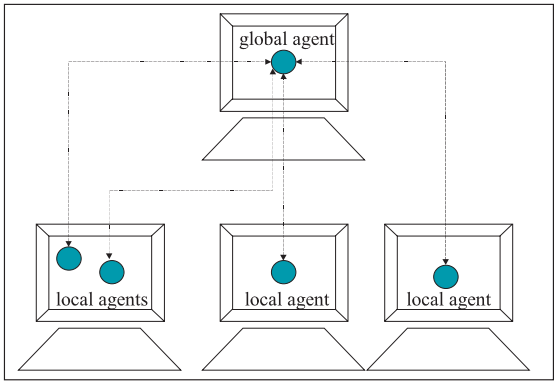

Fig. 8. A typical agent hierarchy.

Communication between local and global agents goes beyond the simple reporting of attack steps. Global agents focus local agent monitoring by communicating attack agendas. Communication is peer-to-peer rather than simply by client/server.

As the diagram shows, it is possible to have more than one local agent on the same machine. This would allow multiple agents using different technologies (e.g., expert systems, goal trees, Petri nets, and neural networks) to monitor local events.

11. IDS resource allocation

Competition for system resources limits the ID effectiveness. Obviously, devoting 100% of systems resources to ID is not acceptable. Efficient allocation of system resources, both global and local, is key to effective ID. Agent cooperation plays an important role in our strategy for resource allocation. Local agents utilize a variety of pattern-matching techniques to detect attack steps in confirming attack agendas. Different detection methods have different costs. Typically, more costly methods yield more information or result in higher confidence information.

In the absence of additional information, local agents will make a trade-off. Global agents may redirect local agents to use higher cost methods based on the global situation and evidence gathered from other agents.

This is similar to battleground management. From a behavioral perspective, global agents (e.g., field marshal) perform strategy analysis and predict trends based on local observations made by local agents (e.g., platoons). This analysis provides the basis to make strategic decisions. Platoons execute orders and look for data to confirm/exonerate conjectures made by global agents. This gives the local agents autonomy in determining the mapping from local events to attack steps. Local agents live in a context-rich environment in which to interpret local data. Global agents can "encourage" local agents to adjust auditing behavior based on strategic information and predictions.

12. IDS agent communication

IDS agents require a well-defined, reliable, and extensible method of communication. We have developed a layered communications approach composed of the following independent layers (in increasing levels of abstractness): *transport*, *content*, *message*, and *conversation*. This is consistent with other agent communication schemes (e.g., KQML, KAoS [3], FIPA) in which messages are independent of the transport layer and opaque with respect to content.

The transport layer provides the most basic communication services between IDS agents. Commercial standards (themselves layered on more primitive standards) provide a number of mechanisms for basic communication (e.g., TCP/IP, SNMP, CORBA, Java RMI). The choice of transport layer standard for agent communication has little impact on the architecture.

The content layer is specific to ID and provides mechanisms for communicating the details of attack steps and attack agendas. A bulk of the work necessary to support ID occurs in this layer. The IETF IDWG is currently working on content standards. Ontologies provide a mechanism for defining terms, relationships, and meaning a field of discourse (e.g., ID). As an example, we would expect the terms *user*, "*file-access-violation*", and *file* to appear in ID ontology used to analyze the following content:

User x generated '*file-access-violation*' on *file* y.

ID agents will agree to these terms prior to engaging in conversations. Agreement, in this context, implies that the agents agreed upon the definition and intent of the terms in the ID ontology. These ID ontologies can uniquely define protocols for the content of messages between agents.

Speech acts form the basis of the message layer. Agents communicate via a finite, but extensible, set of performative messages (or simply messages). For example, "I hereby *request* you to report any activities pertinent to the following attack agenda..." Each peformative (e.g., request, query, inform) has well-defined semantics to assure communicative acts between different agents have the same meaning. These messages are indepen-

dent of the transport service and content. A bulk of the communications will involve inform, acknowledge, request, query, and accept messages. Other messages, supporting negotiation include offer, counter, withdraw, promise, and renege.

Conversations are sequences of messages between agents to accomplish some purpose (e.g., to inform another agent about a hacker's intention). For a specific conversation, one IDS agent plays the role of initiator and the other responder. Conversations policies regulate legal message sequences between agents. Each policy defines a (potentially infinite) set of legal sequences between IDS agents. This frees the agents from having to know which message types are legal in a particular context. Like performatives, the conversation policies are extensible.

UML [8] statecharts provide a convenient method for representing conversation policies. Fig. 9 shows a statechart for the *inform conversation policy*. In this diagram, contours represent states and arrows transitions. Events trigger transitions; the labeling

agent-1 → *agent*-2 : *message*

denotes the event of *agent-1* sending *agent-2 message*, *silence* is a temporal event triggered by the passage of time.

For this policy, IDS agent ***A*** sends ***B*** an inform message, ***B*** can either send an acknowledge message to ***A*** or remain silent; at some point the temporal event silence occurs. Time plays an important role in this architecture. Responses to some requests and queries must occur within a specific time to be useful. Message parameters capture the notion of acceptable response times.

Fig. 10 shows a more complex example, the request conversation policy, followed by a typical conversation between agents.

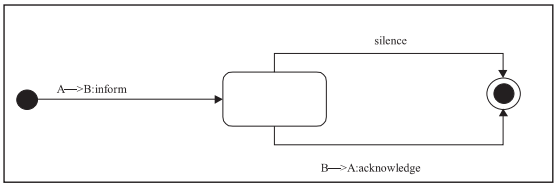

Fig. 9. The inform conversation policy.

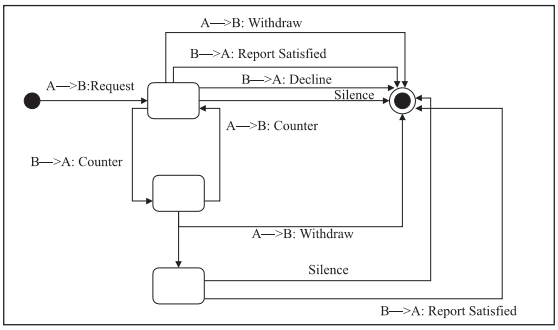

Fig. 10. Request conversation policy.

1. Global agent ***A*** sends a request for local agent ***B*** to report any activities pertinent to the following attack agenda...
2. While agent ***B*** counters by saying that it can only report on one of the activities in the attack agenda.
3. Agent ***A*** accepts agent ***B***'s counter offer.
4. Agent ***B*** does not respond, but accepts the counter request silently.

13. Issues

Evasive actions. In the real-life ID situations, intruders may take evasive actions to prevent the real attack direction from being discovered. Various decoys are also used to generate large volume of misleading data. It becomes more difficult to analyze strategies when one of goals is to mislead the detection. However, with mathematical modeling and other statistical analysis techniques, computation on best-judgment can possibly be exercised. Without strategy analysis, trying to determine decoys at the event level is almost impossible.

Strategy models. In this design, we have not taken into the consideration of machine generated attack strategy models. It is foreseeable difficult to capture methods of intrusion even for a human expert. In a sense, attack strategy model is a platform independent generalization of attack patterns commonly used in misuse detection IDS. Furthermore, attack strategy covers heavily the distributed aspects of coordinated attacks. Current misuse detection patterns are typically local machine/network limited while attack strategy covers

many distributed aspects of coordinated attacks. Due to the difficulty of generalization and the expanded scope of pattern coverage, human build attack strategy model seems to be a reasonable first step. However, it is foreseeable in the future to employ machine learning and data mining to facilitate automatic construction the attack strategy models.

System management integration. Using augmented goal-tree representation for intrusion intention is similar to that of using fault-tree for the system/network diagnosis. The concept of an intrusion intention is equivalent to that of a fault in a fault-tree failure model. The failure model describes the logic behind system/network faults. With this model analysis approach, ID will take place in parallel with system/network diagnosis. The combined analysis could potentially reduce false positives.

14. Related works

The model-based approach was used by DEC in building the PLOYCENTER (ESSENSE) [11] host-based IDS. POLYCENTER uses s-expressions to represent high-level attack patterns, not sequence of events. It then compares current events with these patterns to adjust auditing and performs limited look-ahead adaptive auditing [7,13,14]. This approach greatly reduces the search space in VMS and Ultrix misuse detection. However, the high-level patterns were not modeled after intentions and it does not integrate with Anomaly Detection IDS. Also, the model-based approach has not been generalized to address large-scale network IDS problem.

Kumar and Spafford used colored petri automaton (CPA) condition analysis as a generic pattern matching model for misuse ID [9]. The signature patterns represented by the automaton focus on system/network event level. Bishop, Wee and Frank examined various models for representing security policies [2]. By working with a system model, the criteria and goals for auditing can be determined.

EMERALD at SRI [6] by Porras and Neumann provides a distributed "service monitor" agent framework. Following the lineage of IDES and NIDES, EMERALD IDS monitor agents are capable of performing both signature analysis and statistical profiling. Both analysis engines work through a "resolver" to achieve internal and external agent coordination. AAFID [1] at Purdue University by Zamboni concentrates on providing a distributed IDS agent communication substrate and system architecture. Our effort focuses on the semantics of the IDS agent collaboration in achieving distributed ID.

Acknowledgements

Without the supporting environment provided by The Boeing Company, Kjell Carlsen and Ken Neves, this effort would have been impossible. We would also like to acknowledge Roberto Altschul, Jeff Bradshaw, Larry Bugbee, Jai Choi, Mark Greaves, Anne Kao, Dave Levine, Steve Poteet and Tom Tosch for their technical insights and stimulating idea exchanges that greatly contributed to the formation of the concept behind this architectural paradigm.

References

[1] D. Zamboni, E.H. Spafford, A prototype for a distributed intrusion detection system, Purdue University, Coast Lab, Coast TR 98-06, 1998.
[2] M. Bishop, C. Wee, J. Frank, Goal-oriented auditing and logging, IEEE Trans. Comput. Sys. http://seclab.cs.ucdavis.edu/papers.html.
[3] J.M. Bradshaw et al., KAoS: Toward an industrial-strength open agent architecture, in: J. Bradshaw (Ed.), Software Agents, MIT Press, Cambridge, 1997.
[4] M. Crosbie, E.H. Spafford, Active defense of computer systems using autonomous agents, in: Proceedings of the 18th National Information Security Conference, October 1995.
[5] M. Crosbie, E.H. Spafford, Defending a computer system using autonomous agents, Purdue University, COAST Lab, CSD-TR-95-022, Coast TR 95-02, 1995.
[6] EMERALD: Event monitoring enabling responses to anomalous live disturbances, in: P.A. Porras, P.G. Neumann (Eds.), National Information Systems Security Conference, 1997.
[7] G.W. Hoglund, E.M. Valcarce, The ESSENSE of intrusion detection: A knowledge-based approach to security monitoring and control.

[8] M. Fowler, K. Scott, Uml Distilled: Applying the Standard Object Modeling Language, Addison-Wesley, New York, 1997.
[9] S. Kumar, E.H. Spafford, A pattern matching model for misuse intrusion detection, a pattern-matching model for instrusion detection, in: Proceedings of the National Computer Security Conference, October 1994, pp. 11–21.
[10] D. Anderson, T. Lunt, H. Javitz, A. Tamaru, A. Valdes, Detecting unusual program behavior using the statistical component of the next-generation intrusion detection expert system (NIDES), SRI-CSL-95-06, May 1995.
[11] Digital Equipment Corporation, Polycenter Software Product Description. http://www.digital.com/info/security/id_spds.htm.
[12] A. Sundaram, An introduction to intrusion detection, http://www.cs.purdue.edu/coast/archive/data/author_index.html.
[13] H.S. Teng, J. Chen. Adaptive real-time anomaly detection using inductively generated sequential patterns.
[14] E.M. Valcarce, G.W. Hoglund, L. Jansen, L. Baillie, ESSENSE: An experiment in knowledge-based security monitoring and control.

Ming-Yuh Huang worked at DEC's Artificial Intelligence Technology Center between 1985 and 1990 as a Principal Engineer and Project Lead for the Expert System For Service Network Security (ESSENSE) expert system research project in host-based ID. ESSENSE led to one of world's first commercial ID product – DEC PLOYCENTER ID. Currently, Huang is the project manager for Boeing Applied Research and Technology's Distributed System Management and Security project. He is also the Principal Investigator for DARPA's ID research project GIANT – Global Intrusion Assessment by Distributed Decision-Making. Huang received B.S. in Physics from Fu-Jen University at Taiwan in 1979 and M.S. in Computer Science from University of Oregon in 1984.

Robert J. Jasper developed knowledge development language (KDLAN) for Boeing Commercial Airplane's knowledge base product definition (KBPD) project. He has also been actively involved in agent-based collaborative design technology. Currently working in Boeing Applied Research and Technology as a Senior Advanced Computing Technologist, Rob is also an Adjunct Faculty Member of Seattle University's Master of Software Engineering program. Rob Jasper received his B.A. from Pacific Luthern University and M.S.E. in software engineering from Seattle University.

Thomas M. Wicks worked for Los Alamos National Lab and later taught at University of Missouri as a Professor and Fellow. Since joining Boeing Applied Research and Technology, he has managed the High Performance Computing program for several years. Currently, Tom is the program manager for Boeing's Distributed System Integration Technology Program responsible for large-scale infrastructure salability, management and integration technology issues. Tom Wicks received his B.S. (1972) from Auburn University, M.S. (1975) from University of Texas and Ph.D. (1982) from North Western University all in the area of Engineering and Applied Sciences.