SIDS - A system for enterprise-wide Intrusion Detection

Panagiotis Astithas, Georgios Koutepas, Athanassios Moralis, Basil Maglaris

Network Management and Optimal Design (NETMODE) Lab
Department of Electrical and Computer Engineering
National Technical University of Athens
Iroon Politechniou 9, Zographou, 157 80 Athens, Greece
tel.: (+301) 772.1448 fax: (+301) 772.1452
e-mail: {past, gkoutep, amoral, maglaris}@netmode.ntua.gr

ABSTRACT

In this paper we present the design and implementation of SIDS, the Simple Intrusion Detection System. The main focus in the design of SIDS was to try to overcome the design shortcomings that hinder most currently available intrusion detection systems. Such shortcomings include limited scalability, proprietary intercommunication protocols among the systems nodes and difficult updating procedures. To that effect our system implements a highly scalable distributed architecture that consists of various-level entities, such as sensors, collectors and analyzers. The processing load is spread among the system's nodes and audit information is aggregated as it travels from lower level entities to higher level ones. Furthermore, the system uses SNMPv3 as the default protocol for the communications between collectors and analyzers. This way, it can be easily integrated with traditional Network Management Systems (NMS), and the semantics of its operation can be easily comprehended by most network administrators. It can currently detect UDP or TCP packet flooding, TCP and UDP port scanning, attempts to retrieve various system files containing sensitive information, etc. We are in the process of implementing the ability to detect a wide range of intrusions by fingerprinting and monitoring various system daemons.

I.INTRODUCTION

In the era of e-commerce, security management is a subject of utmost importance for every enterprise. Today, computer networks carry information of substantial financial value, the theft of which could have significant impact on its operation. Consequently, the research in this field is gaining wide attention, in an effort to provide effective response to the threats that have emerged. One of the most promising areas, that of Intrusion Detection, is trying to establish novel methodologies for the monitoring of information flow throughout the enterprise network. Such systems have already been proposed and a number of commercial vendors are providing their own products.

There are two main classifications of intrusion detection systems. The first one divides the techniques of intrusion detection into two main types: anomaly and misuse detection. The anomaly detection model devises a set of statistical metrics that model the behavior of an entity, usually a user, a group of users or a host computer. The profile of a user entity for instance, may include information such as the mean duration of his telnet and FTP sessions, the amount of bytes transmitted in both directions, the time of day or the terminals he usually logs-in from, etc. The profile of a host computer may include the average CPU utilization, the average number of logged-in users, and so on. The IDS monitors the operation of a computer system, and constantly compares the profile of say, a current user session, with the one stored in its database. In case it detects a "large" deviation from the normal behavior it signals an alarm to the system security officer. The magnitude of a "large" deviation is defined as a threshold set by the IDS or the system security officer. Usually the stored profiles are constantly being updated in order to reflect changes in user or system behavior. Since this model works by searching for sessions that are not normal, it is called an anomaly detection model.

The misuse detection model on the other hand, works by searching for a set of known attacks that have been stored in the system's database. The knowledge of the attacks is encoded as a set of attack signatures, which are essentially patterns that occur every time an attack takes place. The way a known attack is represented to the system is an important characteristic of its operation. The variations include various types of graphs, regular expressions, etc. The way this model works is similar to that of an anti-virus program. The implementation of such an IDS usually involves an expert system that performs the matching against the stored rule-base. An obvious difficulty in this architecture is the need for constant updating of the rule-base, as new attack methods become known. Since the model operates by searching for patterns known to represent security attacks, it is referred to as a misuse detection model.

The second classification is based on whether the IDS monitors activity on a single host or on multiple hosts interconnected by a network. The original intrusion detection systems used to examine the audit data on a single machine and derive their conclusions based solely on that information. Consequently, they could not detect attacks that were orchestrated by many sources, or attacks that span multiple machines in a network. Furthermore, they rely heavily on the logs provided by the underlying operating system, which renders them architecture-dependent and more vulnerable to Denial of Service attacks against the IDS, since an intruder may manage to delay the logging mechanism, or even turn it off altogether. An efficient solution is provided by the IDS that passively monitor the network for suspicious activity. Since they depend solely on the ubiquitous TCP/IP protocol suite, they are literally architecture-independent and they can monitor heterogeneous networks quite naturally. And with the current trend towards global internetworking, almost every security attack involves more or less the network. Another issue faced by most architectures today is scalability. Monolithic or distributed IDS that collect the audit data and transmit it to a central host for processing are incapable of operating in a large enterprise network, with a vast number of hosts. The solution to this problem involves the construction of the IDS through a layered architecture. Every node in that model operates by aggregating the audit data it receives from the lower layers and passing a summarized form to the upper layer. Thus, the actual detection of an intrusion can occur on any layer, with the simpler ones occurring at a lower layer and the advanced ones at a higher layer.

Today a number of vendors provide ID Systems that belong to one of the categories above. But an important shortcoming of most of these solutions is the lack of the necessary scalability for deployment in large enterprise networks. Our work addresses this issue, in a way consistent with current best practices in the management of large networks. What we have designed and developed is a distributed software system that constantly monitors the network infrastructure of an enterprise for signs of unauthorized and potentially dangerous behavior. It follows the principle that security management is an integral part of network and systems management and therefore, it tries to bridge the gap between them. To that end we have incorporated traditional Network Management System (NMS) design elements into SIDS, such as the SNMPv3 protocol. Our architecture, which we will present extensively in the following paragraphs, contains entities such as monitors, collectors and analyzers as the nodes of a multi-tier computer system, which communicate through various standard or specifically designed protocols. We have already completed the development of a number of sensors, which collect information from both the network and host systems, regarding suspicious activity. We have also completed the development of the collector, which is the entity that controls the operation of the sensors and disseminates the audit data to the analyzers. At this point, our system can detect a number of intrusive actions, including Denial of Service attacks, such as UDP and TCP packet flooding, TCP and UDP port scanning, attempts to retrieve various system files containing sensitive information, etc.

The rest of the paper is organized as follows: section II provides an architectural overview of our system. In section III, we present the architecture of SIDS, describing in length the different entities involved, as well as their interconnection. Section IV describes the IDS-MIB that is used in SIDS. Section V contains a presentation of current and future goals. Finally, section VI concludes with a summary of the benefits of our design.

II.ARCHITECTURAL OVERVIEW

The architecture of SIDS addresses two of the above-mentioned fundamental problems in the field, namely monitoring and scalability. We use techniques found in both host-based and network-based IDS, in order to enable the system to reach a sound conclusion based on the correlation of both types of events. In order to integrate both approaches, the proposed system utilizes data gathered from various agents that scan through the host system audit trails for known intrusion signatures, and from agents that monitor network traffic for suspicious activity. In addition, these agents communicate with traditional network management agents in order to start, stop, be reconfigured or renewed when necessary.

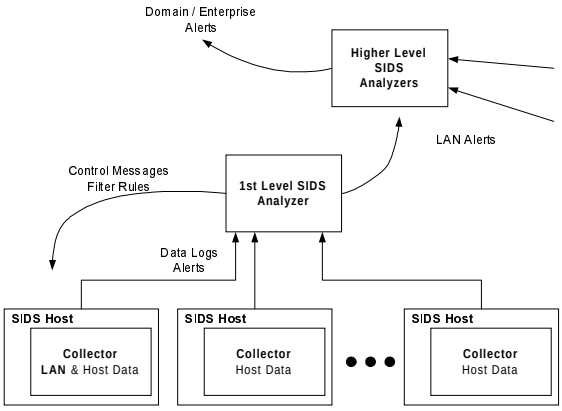

Figure 1: General architecture

In order to filter intrusion detection events in a large inter-network environment, we propose a distributed architecture based on the SNMPv3 protocol. We use a number of sensors to collect audit data in an RDBMS, and provide the data to the higher-level entities through SNMP with an appropriately designed MIB. Local agents perform initial scanning

according to specific instructions passed from a higher-level managing process. The higher-level entities perform more advanced scanning, in order to detect intrusions orchestrated by many sources, and targeting many hosts, such as many forms of Distributed Denial of Service attacks (DDoS).

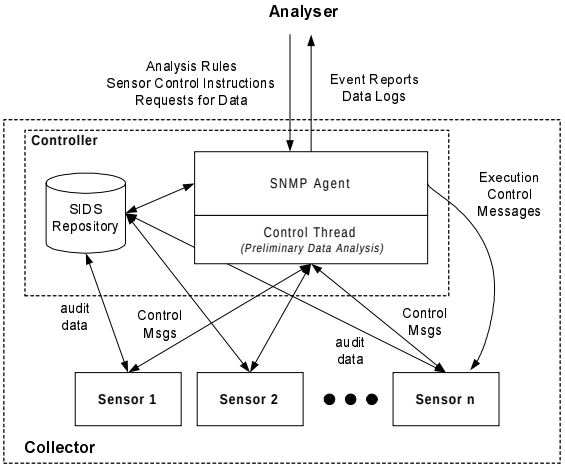

Figure 2: The collector components

The main design requirements for SIDS were a distributed architecture, conformance to current standards, extensibility and scalability. These characteristics provide it with a number of advantages, namely reliability, distribution of intelligence, easy deployment and extension, fitness to a wide scale of networks:

- *Reliability:* since the architecture of the system consists of a number of loosely coupled entities distributed across a network, the IDS has no single point of failure. In some trivial configurations, of course, it may posses a single point of failure. For instance, consider a small company network consisting of a single LAN with ten hosts in an Ethernet segment. In such a case, it could be reasonable to deploy a single analyzer and up to three collectors on the servers, but then again in such a setup the size of the problem's solution is far too small to make a difference.
- *Distribution of intelligence:* every entity in the architecture, that is collectors and analyzers have the intelligence to reach conclusions – albeit of a different level. Therefore the load on existing computer equipment is not substantially affected, something that satisfies one of the basic design principles of an IDS. Furthermore collectors could operate under little or no surveillance, and analyzers could perform intrusion searches with historical data, rending network outage a non-critical problem.
- *Easy deployment and extension:* network administrators can take up the responsibilities of security officers without having to comprehend new protocols or operational semantics. The interconnection protocol between the system's entities is SNMPv3, something that most network administrators are familiar with. Also, additional sensors and detection rules are fairly straightforward to implement, should someone wish to collect and analyze data from a novel source.
- *Fitness to a wide scale of networks:* many intrusion detection systems suffer from the inability to scale well. The reason for this is that their design dictates that data analysis is separated from data collection, with the former being conducted in a central workstation. Our design follows closely the architecture of the Network's Management System, which can scale along the company's network size and complexity.

The design of SIDS borrows concepts from many older Intrusion Detection Systems, but especially from autonomous agents, GrIDS and the IETF's Intrusion Detection Working Group IDMEP (Intrusion Detection Message Exchange Protocol), which is a standardization effort that in a sense has succeeded the Common Intrusion Detection Framework (CIDF).

III.THE MAIN COMPONENTS

SIDS consists of two main types of entities: collectors that gather the audit data and analyzers that scan the data for intrusive behavior. Collectors are responsible for performing a preliminary analysis of the audit data, before submitting them for further review by the analyzers. They are subdivided to the controller and a number of sensors. The sensors, which are the entities performing the data collection tasks - each one in its own specific audit domain - are lightweight processes that constantly load new information into the data store. They can be started, stopped, reconfigured or updated by the controller. The data store, which is a relational DBMS, is also managed by the controller. Besides that, the collector also contains an SNMP agent that provides a standard interface to the analyzers, for the data in the data store. The SNMP agent provides data with respect to a custom MIB called IDS MIB, which groups together both the information collected from the sensors as well as the data necessary for the management of the sensors and the collector. The upper level entities have the ability to customize the behavior of the collector in real time, therefore adapting to potential irregular conditions, caused by an attack (i.e. increased network traffic or CPU load, etc.). The DBMS as a data store provides for a robust yet flexible solution, with the additional benefit of SQL queries that are used by the controller and possibly the sensors, for sophisticated attack signature matching. The design provides additional flexibility by essentially

decoupling the operation of the sensors from the operation of the controller. Their sole means of communication is the data store, besides the signals used by the controller to start, stop and reinitialize the sensors. This gives the sensor's developer the ability to implement fast and without required conformance to new APIs.

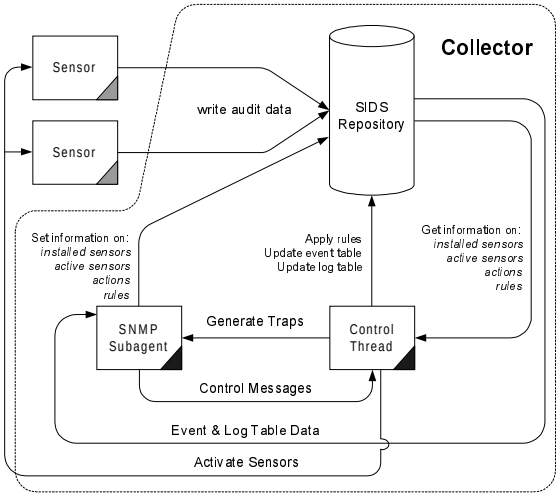

Figure 3: The collector implementation

The analyzers consist of a communications module, a data repository, a rule base, a network topology base, an inference engine and a visualization module. The communications module is used for the communication with collectors and other analyzers in a multi-level configuration. The data repository stores historical data collected from collectors or subordinate analyzers. The rule base contains the rules that are applied against the audit data and trigger the intrusion alerts. The network topology base contains information provided by the network administrator that describes the specific installation. The inference engine performs the actual matching of rules against audit data. Finally, the visualization module provides the network manager with a user-friendly and content-rich view of the current security status of the network. Large enterprise networks with multiple subdomains can use a multi-level configuration, in which the analyzers of a higher level will perform their analysis on data provided to them by the lower level analyzers.

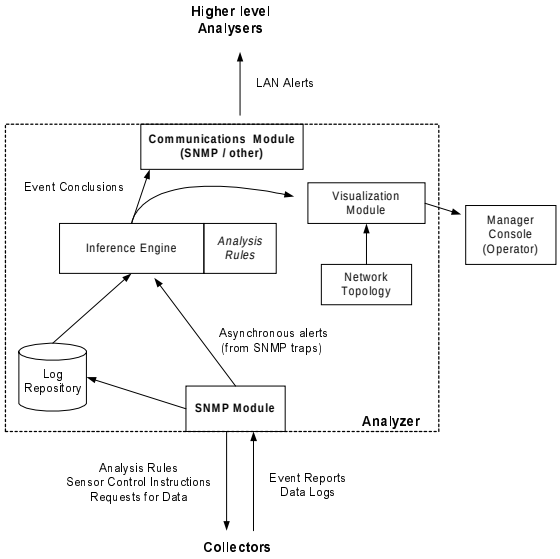

Figure 4: The analyzer

The system's architecture is flexible enough to allow other protocols to be used for the intercommunication between the collectors and the analyzers, such as LDAP (with the necessary schema) and RMI/IIOP. We are currently investigating these alternatives and intend to provide prototype implementations, in order to perform a comparison on the efficiency, scalability and ease of development of each one.

IV. IDS MIB

The information that the collector provides to the analyzers through the SNMP interface is contained in the Intrusion Detection System Management Information Base (IDS MIB). This MIB conforms to the SMIv2 conventions for the structure of the managed objects and follows standard MIB design practices. An effort has been made to keep the design sufficiently generic, in order to incorporate data generated from many different sources. The managed objects are divided into the following object groups: sysGroup, installedAgentsGroup, configurationGroup, logGroup, rulesGroup and notificationGroup. Let’s examine them in more detail.

sysGroup: this is a group that contains information about the computer system that hosts the collector. We can find information about the collector’s condition in it, and also define some global configuration options. The contained objects are *systemType*, *currentSystemLoad*, *currentCollectorLoad*, *maxCollectorLoad*, *currentCollectorDiskUsage*, *maxCollectorDiskUsage*, *currentCollectorMemUsage*, *maxCollectorMemUsage*, *actionOverMax*, *collectorsConfigurationChangeCheckPeriod*, *configurationChangeFlag*.

installedAgentsGroup: here we find information about the sensors that are installed on the system. The

collector can interoperate only with sensors that are properly installed and registered with it. This group contains essentially a table (*agentTable*) with a row for each installed sensor. The attributes that exist in each row are *agentIndex, agentPath, agentVersion, agentDescription* and *agentRowStatus*.

configurationGroup: this group contains a table (*activeAgentTable*) with the configured sensors. Sensors that are installed in the system, have a corresponding entry in the *installedAgentsGroup*, but do not have an entry in this table, are not automatically executed. A sensor can have multiple entries in this table, with different invocation parameters. This way a sensor can be instructed to operate in different ways in each of its instances, in order for example to collect different data. Invocation parameters are passed to the sensor as command-line options during its initial execution. The objects that describe the configuration options are *agentActIndex, agentName, activationTime, formatOfLog, currentAgentLoad, maxAgentLoad, actionWhenAgentLoadExceeded, currentlyActive, activeRowStatus* and *agentExecutionParameters*.

logGroup: this is where we find the audit data collected by the sensors. This group is composed of two tables, the *logTable* where the sensors append their data, and the *eventTable*, which contains the data that the analyzers are interested in. The *logTable* is all that the sensors have access to, whereas analyzers can, if they so wish, browse through both tables. The information in the *logTable* is parsed by the controller, according to the directions in the *rulesGroup*, in order to reduce the magnitude of information that is passed on to the analyzer, and also to spread the CPU load among the nodes of the system. There is also an object (*maxLogTable*) that contains the upper limit on the table's size. The objects in the *logTable* are *logIndex, loggingAgentIndex, loggingTime, loggingSourceIP, loggingSourcePort, loggingDestIP, loggingDestPort, loggingData, extraInfo, checkedBit, certaintyBit* and *logRowStatus*. The objects in the *eventTable* are *eventIndex, eventType, discoveryRuleIndex, eventDiscoveryTime, references* and *eventRowStatus*.

rulesGroup: in this group we store the rules that are used by the controller to perform a preliminary analysis of the audit data, stored by the sensors. This analysis can spot intrusion attempts that are easy to identify, and it helps to minimize the magnitude of information that the analyzer has to work with. Besides the table with the rules (*rulesTable*), the group also contains a table with directions for the action that has to be taken in response to the triggered event (*actionTable*). The objects in the *rulesTable* are *rulesIndex, rulesObject, attackType, actionTaken, loggingCharacteristics* and *rulesRowStatus*. The objects that are contained in the *actionTable* are *actionIndex, actionObject, actionExternalProgram, actionWhichAction* and *actionRowStatus*.

notificationGroup: this group contains the SNMP traps that the collector sends to the analyzer, as a notification for an identified intrusion attempt or another event that requires the immediate attention of the manager. These traps are generated when intrusive behavior has been identified with high degree of certainty, based on the preliminary analysis performed by the controller on the audit data. The traps contained in this group are *denialOfServiceTrap, portScanningTrap* and *otherEventTrap*.

V. CURRENT STATUS AND FUTURE WORK

We have finished the design of the MIB and we have implemented the collector with one sensor that gathers data from the network, packages it in a summary format, and stores them in the MIB. Its operation is similar to a network sniffer, but the rules that we have constructed aim primarily to detect a port scan, including UDP connections, TCP connect() scans and TCP SYN, FIN, etc. stealth scans. We have also developed sensors that search through the system's log files, such as the ones used by apache (and other HTTP daemons that use CLF), wu-ftpd (and others FTP daemons), sendmail (and other MTAs), etc. They look for suspicious activity, such as very long URLs, attempts to transfer sensitive files, attempts to exploit known vulnerabilities, and so on. The development is done on Solaris and we have used MySQL as the data store, and SNMP Research's EMANATE tool to develop the SNMP agent. Parts of the system are written in C, C++ and Perl.

We are currently in the process of implementing the analyzer, and a sensor that monitors UNIX daemons for abnormal behavior using process fingerprints. The fingerprints are a series of system calls that the daemon makes during its operation. We construct these fingerprints by operating the daemon in a provably safe environment, and then monitor the system calls it makes - through truss(1) - trying to detect unusual activity. We are evaluating various metrics for the comparison of the current call sequence against the fingerprint, in order to minimize the false alarms. We are also evaluating different fingerprint sizes, in order to attain the optimum balance between memory space and execution speed.

Future work includes the development of more sensors with more complex tasks and the evaluation of our architecture in terms of efficiency, actual intrusion detection and ease of administration. We also intend to investigate possible enhancements in the architecture, such as the use of other protocols for the intercommunication between the collectors and the analyzers, such as LDAP and RMI/IIOP. The use

of LDAP will provide us with a global schema that will facilitate the retrieval of intrusion information on a regional, national or even global scale. For that purpose, we are considering a custom schema, derived from the IDS MIB. The use of RMI/IIOP is a consequence of an effort to port the system to Java. This port will give us the ability to implement dynamic sensor upgrade in a more straightforward manner, platform neutrality and code reuse.

VI.CONCLUSIONS

We presented the architecture of an Intrusion Detection System that features a high degree of scalability and ease of integration with standard network management platforms. Its distributed architecture consists of two main types of entities: collectors that gather the audit data and analyzers that scan them for intrusive behavior. Collectors contain a controller and a number of sensors. The sensors are lightweight processes that perform the actual data collection. They can be started, stopped, reconfigured or updated by the controller. The audit data is stored in the data store, which is an RDBMS. The controller contains an SNMP agent that presents the data to the analyzer through a standard protocol. The respective MIB, namely IDS MIB, contains the audit data as well as the sensors' configuration information. The controller itself performs a preliminary analysis of the data, based on elementary rules specified in the MIB. The analyzers consist of a communications module, a data repository, a rule base, a network topology base, an inference engine and a visualization module. The communications module is used for communicating with collectors and other analyzers in a multi-level configuration. The data repository stores historical data collected from collectors or subordinate analyzers. The rule base contains the rules that are applied against the audit data and trigger the intrusion alerts. The network topology base contains information provided by the network administrator that describes the specific installation. The inference engine performs the actual matching of rules against audit data. Finally, the visualization module provides the network manager with a user-friendly and content-rich view of the current security status of the network. Every node in that model operates by aggregating the audit data it receives from the lower layers and passing a summarized form to the upper layer. Thus, the actual detection of an intrusion can occur on any layer, with the simpler ones occurring at a lower layer and the advanced ones at a higher layer. Large enterprise networks with multiple subdomains can use a multi-level configuration, in which the analyzers of a higher level will perform their analysis on data provided to them by the lower level analyzers. Furthermore, the use of an industry-standard protocol facilitates the integration with traditional Network Management Systems.

REFERENCES

[1] **Dorothy E. Denning**, "An intrusion-detection model", *IEEE Transactions on Software Engineering, vol. SE-13, pp. 222-232, February 1987*

[2] **Teresa Lunt et al.**, "IDES: The enhanced prototype", *Technical report, SRI International, Computer Science Lab, October 1988.*

[3] **J. Case, R. Mundy, D. Partain, B. Stewart.**, "Introduction to Version 3 of the Internet-standard Network Management Framework", *The Internet Society, RFC 2570, April 1999.*

[4] **L. T. Heberlein et al.**, "A network security monitor", *Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA., May 1990.*

[5] **S. R. Snapp et al.**, "A system for distributed intrusion detection", *Proceedings of the IEEE COMPCON 91, San Francisco, CA., February 1991.*

[6] **Mark Crosbie, Gene Spafford.**, "Defending a Computer System using Autonomous Agents", *Technical report No. 95-022, COAST Laboratory, Department of Computer Sciences, Purdue University, March 1994.*

[7] **S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, D. Zerkle**, "GrIDS -- A Graph-Based Intrusion Detection System for Large Networks", *The 19th National Information Systems Security Conference, Baltimore, MD., October 1996.*

[8] Common Intrusion Detection Framework (CIDF), http://olympus.cs.ucdavis.edu/cidf/.

[9] **Tim Bass, David Gruber,** "A glimpse into the future of ID", *;login:, Special Issue on Intrusion Detection, September 1999.*

[10] **Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji,** "Computer Immunology", *Communications of the ACM, October 1997.*

[11] **Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, Thomas A. Longstaff,** "A Sense of Self for Unix Processes", *Proceedings of the 1996 IEEE Symposium on Security and Privacy, 1996.*