A Distributed Intrusion Detection Prototype using Security Agents

V. Chatzigiannakis, G. Androulidakis, M. Grammatikou, B. Maglaris
Network Management & Optimal Design Lab (NETMODE),
ECE Department – National Technical University of Athens (NTUA)
9 Iroon Polytechniou str. Zografou, Athens, Greece
{vhatzi, gandr, mary, maglaris}@netmode.ntua.gr

Keywords*: Distributed Intrusion Detection, Central Management, Multicast Communication, IDMEF data model*

Abstract

Intrusion Detection is the problem of identifying unauthorized use, misuse, and abuse of computer systems by both system insiders and external intruders. Intrusion Detection Systems provide in depth packet analysis and application awareness and can be deployed for discovering network attacks. In this scenario a system that gives intelligence about the traffic on your network is necessary. This paper describes a prototype for Distributed Intrusion Detection considering a large-scale network environment in order to monitor multiple hosts connected via a network as well as the network itself. The design and implementation of our Distributed Intrusion Detection prototype relies on Security Agents which monitor network traffic and report intrusion alerts to a central management node. The Intrusion Detection Prototype is comprised of sensor and management elements. Distributed operation is handled through the introduction of multiple sensors and the use of Security Agents that are responsible for incident reporting and message propagation control.

1. Introduction

An Intrusion Detection System (IDS) has been traditionally categorized according to the way it collects data and the detection method used on the data collected [1]. If the data processed originates from one or more hosts, then the IDS is called host-based. This methodology is mostly based on examining system logs and has become obsolete. However, if the IDS monitors a network of interconnected hosts for malicious traffic it is called network-based. Network-based intrusion detection systems are more efficient because of their ability to combine network traffic data with audit data from individual hosts.

The second categorization divides IDS into anomaly detection and misuse detection systems. Anomaly detection systems monitor the system and try to decide whether its behavior is normal or not. This is achieved by keeping user group and host profiles. Usually a group profile is determined by the kind of programs used, the current time and duration of the user session. A host profile is determined by checking resources such as cpu and memory usage, the number of processes and users logged in. Such systems have to be continuously updated and adapted to changes in system and host behavior.

Misuse detection systems on the other hand search for known attack signatures. A signature is a trail of a known attack. For example, it may be a specific series of bits in the header of an IP packet. Such systems resemble in their function to anti-virus programs. A weakness of this architecture is that they have to be updated on day to day basis, by downloading new attack signatures.

Besides these classic approaches to the problem of Intrusion detection, there exists one more category: the Distributed Intrusion Detection Systems. These systems have recently started to evolve, because of the expansion of large-scaled networks. Traffic monitoring in such networks has been usually done in the past by installing sensors on the border routers. Nowadays, this schema seems obsolete, because of the

burst in network capacity it is impossible for an IDS to monitor a gigabit link without experiencing packet loss, no matter the hardware used. So, instead of using one central IDS, one should install a distributed system of sensors spread in the network to be monitored.

However distributed systems introduce new barriers to be surpassed. First, the various sensors must share a common protocol for communicating with each other. Second, their intercommunication has to be secure, reliable and efficient; the data sent to the administrator must not be compromised and must not produce too much extra traffic in the network. Moreover, a large number of sensors drive to an inefficient management. This problem may become more intense in case of a denial of service attack towards one of the sub-networks monitored. The alert messages from the sensors that detected the attack could be large enough to result in extra flooding of the network. Another problem is the correct placement of the intrusion detection sensors. This is a matter of great consideration and depends on the network topology and special characteristics such as the number of routers and especially border routers, the bandwidth of the links connecting routers and switches and the number of hosts. Finally, a major problem of a Distributed Intrusion Detection System is the efficient manipulation of alerts from its sensors. The central node of the system has to combine the heterogeneous data it receives and decide whether there is an attack or not.

2. Architecture

2.1. Setting our goal

The main goal in our design is the intrusion detection in large-scaled networks. To address the problems discussed previously, we propose the creation of a Distributed System that comprises of heterogeneous Intrusion Detection Systems. Every sub-system is an autonomous Security Agent and is installed on a designated host. If a system is attacked and disabled, another may transparently host take its place. This scheme is efficient because of its ability to combine information from many sources. The management of this distributed system consists of one or more central IDS nodes.

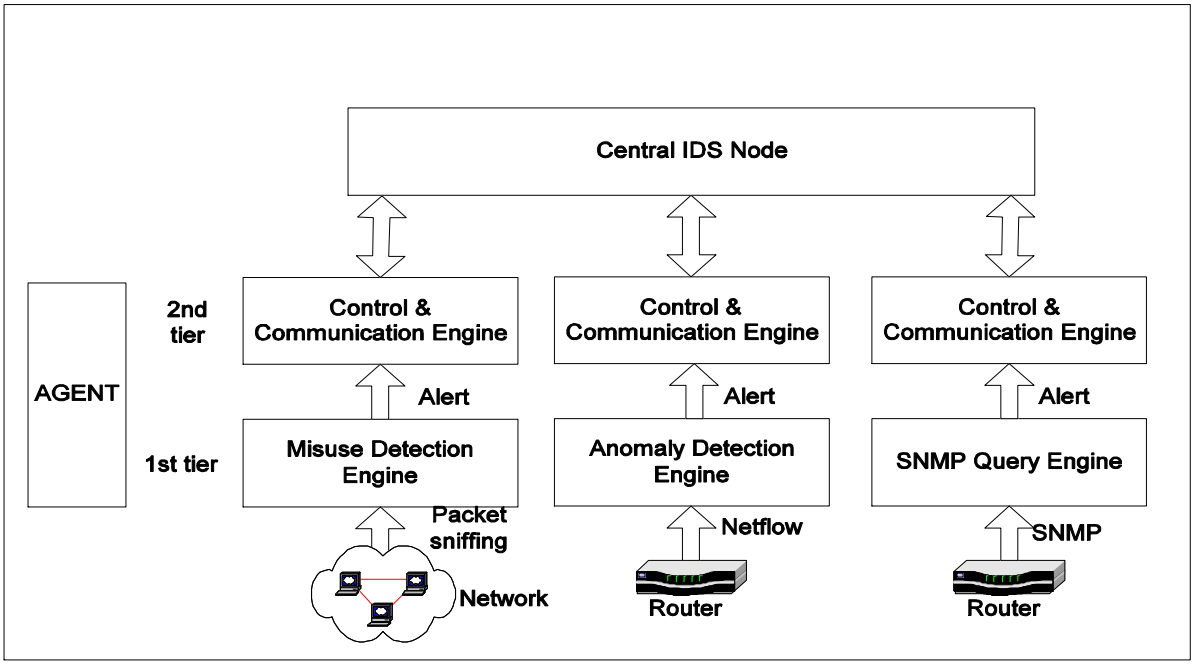

Figure 1. Agent Architecture

2.2. Agent Structure

The Agent performs a series of operations concerning network monitoring and simultaneously listens for new directions from the central IDS node. Figure 1 sketches the outline of the agent architecture we propose. We have designed and implemented three types of agents: a misuse detection agent, an anomaly detection agent and an agent that conducts SNMP queries. Each Agent comprises of two main tiers. In the lower tier, the Agent conducts network monitoring, collecting information from the sensor and passing it to the next level. The upper tier is responsible for communicating with the Central node and for controlling the lower tier's operation. Communication with the central node is bidirectional, the agent sends messages for announcing new attacks and for notifying that is operational whereas the central node sends new configuration and policy rules.

2.3. System's intercommunication

Figure 2 presents the system's intercommunication which is achieved through secure and reliable multicast messages. There are three main reasons for choosing multicast. The first reason is that by using multicast the central IDS node may transparently and simultaneously control all agents. The system is more scalable and easy to manage centrally. Moreover, the use of UDP in the transport layer is more preferable in the case of a denial of service attack on the network, because TCP traffic is almost impossible to pass through a flooded link.

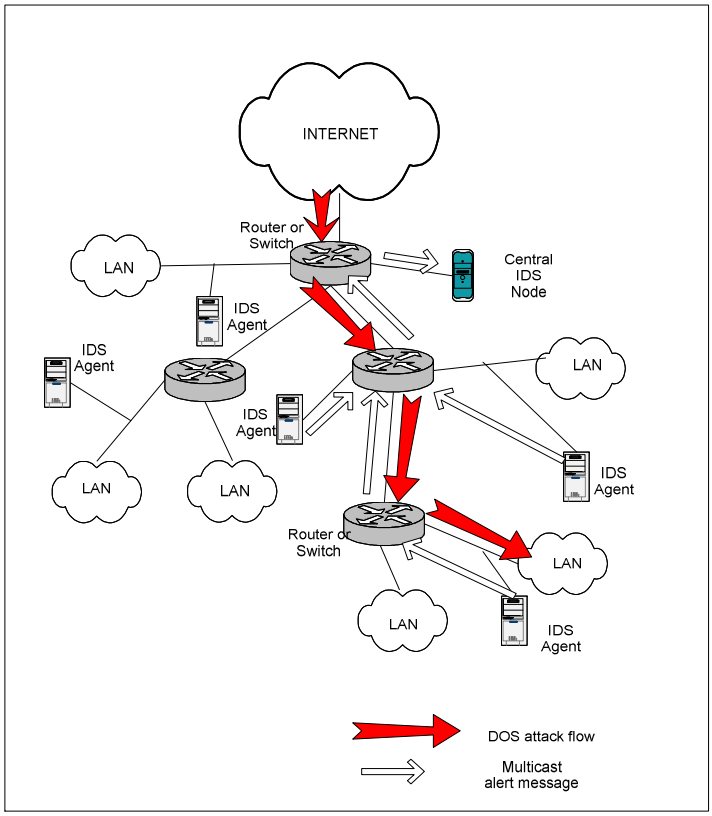

Figure 2. Network Architectural concept

Consider a LAN experiencing a DoS attack. If the LAN's intrusion detection system tried to communicate with the central node using TCP through the LAN's upstream link, the connection would probably fail to be established. Although the packets transmitted by the IDS would reach their

destination, the ACK packets would likely be dropped. The other reason for choosing such a schema is portability, ease of use and the possibility of using more than one central node for collecting messages. While the main node reports the site administrator about the security events detected by the various sensors, the secondary node passively and obscurely listens to the messages sent and is ready to take over in case of hardware failure or a DoS attack targeting the primary system.

Moreover, it is desirable to use more than one IDS in every sub-network. This way we achieve load balancing in detection. The wide variety of known attacks usually results in a great load for a signature based IDS. To solve this problem, every system may focus in a narrow range of attacks so as be more efficient.

The messages sent by the Agents to the central node are based on IDMEF (Intrusion Detection Message Exchange Format) [2]. It is a draft of the IETF working group which researches intrusion detection. IDMEF is an XML Document Type Definition for the exchange of messages between Intrusion Detection Systems. It supports two kinds of messages: Alerts and Heartbeats. Heartbeats are periodic messages. They mainly inform the central node that the system sending the message is operational. Besides that, they carry general information about the Agent. Alerts carry information about detected attacks, such as the analyzer that produced it, the identification of the attack, the source and target ports, the IP of the target(s) and other optional data.

In order to solve the problem of security in Multicast we extended the IDMEF data model to support digital signatures. We propose an extended IDMEF DTD which includes a digital signature of the message. This way the receiver can verify the integrity and authentication of the message by using the public key of the sender, stored in its database. The threat of replicated messages is solved by the timestamp carried in a special tag of the IDMEF DTD called “Creation time”. Multiple alerts that have the same creation time are rejected.

To address the problem of unreliability in multicast transmission, various protocols have been introduced, each with varying success in different operating requirements and environments. There are two main categories of reliable multicast protocols. The first category includes protocols that employ receiver acknowledgements to inform the sender for the loss of packets and request a retransmission. These protocols are further divided into two categories based on whether receivers generate positive (ACKs) or negative acknowledgments (NAKs). Protocol examples using NAKs are PGM[3] and NORM [4]. The use of NAKs is preferred because it produces less traffic, but in this case the protocol has to be very sensitive to a NAK loss since it is the only mechanism for providing reliability. Also, the notification mechanism proves very inefficient for large client numbers, or when packet loss varies greatly among receivers. Under these circumstances, the source is flooded by ACKs/NAKs or it has to retransmit large numbers of different groups of packets at the same time, resulting in degrading overall performance and low scalability.

An alternative that avoids the back channel from the receiver to the sender is the use of Forward Error Correction (FEC) [5]. This is the second category of reliable multicast protocols where the sender generates FEC values for selected packets of the content stream that allows detection and repair of data loss at the receiver end. However, in this case reliability is not absolutely ensured.

The control messages between the nodes are exchanged in a NAK-based reliable multicast protocol (NORM) whereas the alert messages are sent using FEC coding. So, we ensure that the configuration of the Agents is reliable and that in case of a DoS attack, alerts will reach one or more central nodes, provided that they are correctly placed in the network.

3. Security Agents

3.1. Misuse Detection Agent

As we previously mentioned, each security agent consists of two tiers. The lower tier comprises of the process that handles the misuse detection within our network. Snort [6] has been chosen as the misuse

IDS software for our system. Snort is a libpcap-based [7] software that can be used as a sniffer, packet logger or network intrusion detection system. In our case, we used Snort as a misuse intrusion detection tool. The detection of malicious packets is based on known attack signatures. Snort is able to detect a variety of attacks such as DoS/DDoS attacks, Portscans, HTTP, DNS, SMTP, IMAP, POP3 attacks and Virus/Worm attacks. Alerts generated from Snort are passed to the upper tier of our agent.

The upper tier of the Misuse Detection Agent receives alert messages from the lower tier and stores them for a defined period of time in a buffer. For every different case of attack, that is, source IP address and port, target IP address and port and known attack signature, the upper tier process uses a unique alert identification. Rate limiting is achieved independently for different types of attacks, sending the alert message only once in the specified period of time. Agent's upper tier process is also responsible for sending the heartbeat messages to the Central IDS Node informing that the misuse IDS system is operational. The messages sent to the Central IDS Node are formatted using the extended signed IDMEF format. In addition, the upper tier process listens for commands from the Central IDS Node. It receives parameters for the rate limiting of alert messages, configuration for the Snort process and new attack signatures.

3.2. Anomaly Detection Agent

For the lower tier of the Anomaly Detection Agent we developed a prototype anomaly detection tool [8] that currently focuses on DoS Attacks. The prototype tool consists of two main modules: the collector and the detector. The collector module is responsible for asynchronously receiving flow data from the Netflow-enabled [9] router; information is analyzed, mean values and adaptive thresholds are calculated and stored in a local data structure. The tool extracts and stores packet and flow counters per destination IP address, as well as total counters and mean values for each pair of input-output interfaces. The detector process is responsible for calculating the metrics for the interface pairs stored by the collector, and comparing the results to detection thresholds. It is periodically activated, implements extensive logging of detection events and generates notifications with security alerts which are sent to the upper tier.

The upper tier process receives the alerts and sends them to the Central IDS Node using the signed-IDMEF Format. Moreover, the Central IDS Node adjusts Anomaly Detection Agent's parameters (metrics and thresholds for the DoS attack detection algorithm).

3.3. SNMP Query Agent

As the other two agents, the SNMP Query Agent is comprised of two tiers. The lower tier process is a custom SNMP client that performs SNMP queries at the routers of the network. Values like CPU and memory usage, active and inactive flows are polled from routers at specific intervals.

The upper tier accepts the values from the SNMP queries and forwards them to the Central IDS Node after formatting them using the signed-IDMEF data model. The upper tier process is also responsible for sending heartbeat messages to inform the Central IDS Node that the SNMP client is operational. Instructions from the Central IDS Node are sent to the SNMP Query Agent, giving information about the router and the SNMP objects to be polled.

4. Central IDS node

The central IDS node does not monitor the network like the other nodes. Its purpose of existence is the control and fine tuning of the Agents, the gathering and storage of messages, and the notification of the administrator when necessary. Attack detection in the central node is done by combining alerts from different Agents. The central IDS node acts like a manager for the Agents. Through this manager one can

check the alert messages produced in every node, keep the system updated by downloading new configuration and manage the policy of attack detection per agent.

We define a central policy scheme for the configuration of all the agents. This scheme uses XML and includes generic configuration for each type of agent. Specific configuration for a particular agent may also be applied. Once again, to ensure integrity these XML messages contain digital signatures.

```
<Policy>
  <MisuseDetection id="all">
    <Action type="Add">
      <signature id="3445" description="Sasser Worm">alert tcp $HOME_NET any -> any 9996 (
msg:"Sasser ftp script to transfer up.exe"; content:"|5F75702E657865|"; depth:250; flags:A+;
classtype: misc-activity; sid:1000000; rev:3;)
      </signature>
    </Action>
  </MisuseDetection>
  <AnomalyDetection id="sensor-4">
    <Action type="Update">
      <FlowThresholdDeviation>1.20</FlowThresholdDeviation>
      <PacketThresholdDeviation>1.35</PacketThresholdDeviation>
    </Action>
  </AnomalyDetection>
  <SNMPQuery id="all">
    <Action type="Remove">
      <object>MemoryUsage</object>
    </Action>
  </SNMPQuery>
</Policy>
```

Figure 3. A sample Policy Configuration

In Figure 3 we present a sample Policy Configuration which is sent to the agents. In this specific message all Misuse Detection Agents are receiving the signature of the "Sasser Worm" [10] in order to raise an alert when a packet matches this particular signature. In addition, the Anomaly Detection Agent named "sensor-4" adapts its thresholds to the values that are sent to it. Finally, all SNMP Query Agents stop polling "MemoryUsage" object from the corresponding router.

The central node correlates information from the Agents and tries to decide whether there is an incoming or an outgoing attack and whether it affects one or more subnetworks. The decision is not based on an expert system, it is based on scenarios. A scenario must contain one or more policy rules installed in the Agents. When a new alert message arrives, the central IDS node checks whether it has already received alerts contained in the same scenarios. The certainty of a reported attack is proportional to the percentage of the policy rules matched.

Scenarios that describe "Misuse Detection" attacks usually contain one policy rule. However, for every simple scenario described by one policy rule, there is one more that checks if the attack has multiple targets. This scenario's alert is triggered if the same rule is matched in more than one sub-network.

For the anomaly detection engine some of the scenarios used are the following:

- Alarms from the SNMP Query Agent stating increase in the CPU usage of a router are combined with alarms from the equivalent Anomaly Detection Agent concerning increase in traffic. If both increases exceed the thresholds set by the administrator, the central IDS node detects a DoS attack.
- The system checks if a sub-network participates in a DoS attack by combining alerts from the Misuse Detection Agents for 'rootkit' signatures in packets originating from compromised hosts and alerts from Anomaly Detection Agents for increase of outgoing traffic.
- If more than one sub-networks experience "Anomaly Detection" attacks the system decides whether they belong to the same attack by checking network traffic characteristics such as source and target ports, number of flows or packets.

Until now, we have successfully tested each type of Agent independently. By experimenting on the parameter values of every Agent we have developed efficient metrics and configuration. Our experiments have led us to believe that by combining much information it is difficult to create efficient policy rules and train the system to make correct decisions. So the target of our research is the discovery of efficient scenarios rather than a complicated expert system.

5. Related Work

Over the past years many Distributed Intrusion Detection Systems have been developed. These schemes were a result of research on methods of aggregating data generated by individual intrusion detection systems placed across large-scale networks.

The GrIDS (Graph Based Intrusion Detection System) [11] which is developed at the University of California, Davis is a hierarchical graph-based IDS capable of doing distributed data collection and analysis in large networks. Data source modules running at hosts across the network report information to graph engines resulting in building a graph representation of network activity. Data sources can be either network sniffers or an IDS that works on a single host.

The EMERALD project [12] which is developed at the Computer Science Laboratory at SRI International proposes a distributed intrusion detection scheme that besides the act of detecting attacks has also the ability to automatically respond to them. EMERALD works both as an anomaly detection system, using the profiler engine to perform statistical profile-based anomaly detection and a misuse detection system, using the signature engine which employs a rule-coding scheme. Response to suspected attacks is done with the use of the resolver which is an expert system that implements the response policy based on the intrusion reports produced by the profiler and signature engines.

The AAFID (Autonomous Agents For Intrusion Detection) architecture [13] developed at the Purdue University is a distributed intrusion detection system that consists of multiple independent entities, called autonomous Agents. These Agents perform a very specific function or more complex activities and can be added and removed dynamically from the system, without having to restart the whole IDS system.

The Indra (Intrusion Detection and Rapid Action) scheme [14] proposes a distributed and reactive intrusion detection system based on individual daemons running at hosts across the network. These daemons distribute the intrusion attempt information that each one gathers, among all other daemons forming a trusted network. The communication between the daemons is based on peer-to-peer systems and is achieved through a cryptographic mechanism that provides support for sending encrypted and digitally signed messages. The reaction to suspected intrusion attempts is implemented through controlled access to resources.

6. Future Work

We intend to use the NTUA campus as a test bed for our framework. Up until now we have only installed and tested our distributed system in a closed environment in our laboratory and emulated attacks for our experiments. NTUA has a class B network with almost 150 active LANs and 5000 active hosts. Its size makes it ideal for our experiments. We plan to install the Misuse Detection Agent in some of the LANs, configure the border routers to feed the Anomaly Detection Agent with Netflow data and SNMP data to the SNMP Query Agent. The central node will be able to fuse real information and the policy rules will be tested in real security incidents. Finally, we intend to extend our research by focusing on the discovery and testing of new scenarios.

7. Conclusions

We presented a Distributed Intrusion Detection System for a large scale network environment. The framework implements a distributed scheme via a scalable and secure distributed architecture that provides efficient and transparent control to the central monitoring system. The system we outlined relies on Security Agents that monitor the network and report security alerts to central IDS nodes via multicast messages. We have design and implemented three types of Agents: a Misuse Detection Agent, an Anomaly Detection Agent and a SNMP Query Agent. System's intercommunication is based on reliable multicast and the integrity of exchanged messages is assured by the use of digital signatures. The system provides ease of management as almost all administrative operations can be conducted centrally. Information sent from the Agents is gathered in the central node, and attack detection is based on predefined scenarios. Finally, we plan to validate this architecture and produce quantitative results via experiments on the NTUA campus.

8. References

[1] Biswanath Mukherjee, Todd L. Heberlein, and Karl N. Levitt, "Network Intrusion Detection", IEEE Network, May/June 1994

[2] D.Curry and H. Debar, "Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition", Internet Draft, November 2002.

[3] RFC 3208 - PGM Reliable Transport Protocol Specification

[4] NORM Internet Draft: http://ietf.org/internet-drafts/draft-ietf-rmt-pi-norm-09.txt

[5] RFC 3453 - The Use of Forward Error Correction (FEC) in Reliable Multicast

[6] Snort project - http://www.snort.org

[7] Libpcap library – http://www.tcpdump.org

[8] G. Androulidakis, V. Chatzigiannakis, M.Grammatikou, F.Stamatelopoulos, "Network Flow-Based Anomaly Detection of DDoS Attacks", Terena Networking Conference 2004, Rhodes, Greece, June 2004.

[9] Cisco Netflow, http://www.cisco.com/warp/public/732/Tech/nmp/netflow/index.shtml.

[10] Sasser Worm - http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html.

[11] Stuart Staniford-Chen, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, D. Zerkle. GrIDS: A Graph-Based Intrusion Detection System for Large Networks. Proceedings of the 19th National Information Systems Security Conference, 1996.

[12] Phillip A. Porras and Peter G. Neumann. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In 1997 National Information Systems Security Conference, October 1997.

[13] Jai Sundar Balasubramaniyan, Jose Omar Farcia-Fernandez, David Isacoff, Eugene Spafford, and Diego Zamboni. An Architecture for Intrusion Detection using Autonomous Agents. Technical report 98/05, Purdue University, 1998.

[14] Q. Zhang and R. Janakiraman, "Indra: A Distributed Approach to Network Intrusion Detection and Prevention", Washington University Technical Report # WUCS-01-30, 2001.