

# Network & Agent Based Intrusion Detection Systems

Hakan Albag

TU Munich, Dep. of Computer Science – Exchange Student  
Istanbul Tech. Uni., Dep. Of Comp. Engineering

**Abstract.** The following document is focused on Network Based Intrusion Detection Systems and Agent Based Intrusion Detection Systems. Besides general definitions of these IDS system architectures, it includes an overview of several Network and Agent Based Intrusion Detection implementations.

## 1 Introduction

Intrusion Detection systems can be classified into three categories based on the types of data they examine. These are :

- Host Based IDS
- Network Based IDS
- Application Based IDS

This classification depends on their Data Gathering Components (sensors), which they use to collect data to detect possible attacks against system. In the **host based** approach every host has its own IDS and it collects data in the low level operations like network system calls (monitoring connection attempts to a port, etc.). A **network based** IDS collects data in the network level, transparently to the other hosts. Their sensors are located somewhere in the network and monitor network traffic. And the third type of IDS, the **application based** approach uses data sources from running applications as its input.

Most of the Intrusion Detection Systems used today are not implemented using a single approach, because all of them have their own advantages and disadvantages. Generally they use multiple approaches to gather information to detect malicious behavior in the system. When they are all used together, it also can be thought in a hierarchical order (application - host – network). Output from an IDS can be utilized by other ID systems at the same or higher levels in the hierarchy.

## 2 Network Based IDS

Network based ID systems can be thought as intelligent network hardware, they monitor the network traffic and analyze it according to some signatures. To do this a network interface is set in promiscuous mode and collects all packets through the network. Collected data are considered to be interest of 3 major types of signatures. These are String signatures, Port signatures and Header signatures.

**String signatures.** String signatures look into packet data to find a possible text related with an incident. An example for an UNIX system can be "cat "+ +" > /.rhosts" , which defines an attempt to execute "cat" command to append a new entry to ".rhosts" file which includes list of remote hosts. This might be a severe security leak for a UNIX system.

**Port Signatures.** Port signatures simply monitor traffic to specified ports. They are looking for well-known attack ports like FTP (TCP 21), telnet (TCP 23) and IMAP (TCP 143). If any of these ports are not used by any service in the system, the packets coming to this ports might have been suspicious.

**Header Signatures.** Header signatures is concerned with the illogical and possibly dangerous combinations with packet headers.

The advantage of the network based IDS is that they are generally OS independent and as they work in the network level and not located on every host, they do not have any effect to the existing system. In addition they are very flexible and portable. But they have also disadvantages especially under heavy network loading. In general they are weak in dropping packets in heavy load and fast speeds. In addition there are also problems to apply above defined signatures to unknown protocols, for example something non IP.

### 2.1 GrIDS

Graph Based Intrusion Detection System (GrIDS) is built to detect distributed attacks against large networks. It constructs *activity graphs* to represent connections between hosts (network traffic) in its domain. Every node in the graph represent a single host or a set of hosts (domain) and every edge in the graph represents the network traffic between nodes. GrIDS uses network sniffers and also host based ID systems as its data source. At this point it is said to be a combined approach of host based and network based IDS. Collected data will be the input of the graph engine, which is responsible for creating activity graphs.

Graph engines are built hierarchically, higher level engines gathers data from lower level graph engines and at the lowest level from single hosts. When a very large network with thousands of hosts and tens of subnetworks is considered this approach will provide scalability for the IDS. The whole network is considered as divided into parts and have their own graph engines. This lower level engines create graphs by

collecting data from the host based IDS and also by inspecting traffic between these hosts. The parent engines do not gather information from every host, instead of this the child engines pass information to the upper layer. As the information passes up the hierarchy, the graphs become coarser and coarser with each child node representing a lower domain that may have numerous nodes.

Every kind of graph is responsible for the detection of a certain class of attacks and graphs are build in a flexible way. Each graph may have its own rules and also attributes. The construction of the graphs from the input data is defined by this rule sets. According to the input data provided by the data sources actions are taken as a result of the graph. The rule sets are independent from each other, which provides instead of creating big graphs containing different kinds of activities, smaller and also maintainable graphs are build. Rule sets can also be used as security policies. As subnetworks and also different domains has their own rule sets, inter-domain accesses can be easily restricted (ie.Access restriction between different departments of a company). Since rule sets can be complicated and difficult to write, GrIDS includes a policy specification tool that more easily allows the specification of acceptable and unacceptable behavior.

Finally, GrIDS is met as a scalable solution in the network based intrusion detection, since it solves a major disadvantage (scalability) of network based IDS systems.

## 2.2 Bro

Bro is developed as a research tool at the Lawrence Livermore National Laboratory and focuses on the protection of IDS itself against attacks. Also Bro provides high speed, large volume monitoring of the networks without dropping packets. It has a modular structure and it is easy to make distinction between different system modules. There are three main layers in Bro:

*Packet Capture Unit* - This unit can be thought as detection unit. It uses libpcap to capture packets from the network. The use of libpcap isolates Bro from the underlying network technology and makes it portable.

*Event Engine* - It analyzes packet streams captured by the Package Capture Unit, verifies their integration and sends them to the appropriate handler. Handlers are provided by the policy script interpreter.

*Policy Script Interpreter* - Policy Script Interpreter runs scripts written in Bro language and associated with a handler. Whenever an event arrives, it executes related handler script. This script may execute other arbitrary commands to log events, modify state or record a data.

As Bro is designed to deal with attacks against itself, it defends itself against 3 kinds of different attacks. The first one, *overload attack* tries to overload the IDS monitor by sending so many packets that exceeds the processing capacity of the IDS.

Bro deals with this kind of attack by performing *shed loading*, which means protocols that cause enormous traffic are not monitored any more to provide to focus on important streams. *Crash attacks*, a second kind of IDS attacks have goal to put the monitor out of action by exploiting program errors. Bro defends itself against this attack by a *watchdog*, which controls the state of the event engine periodically and restarts it if needed. And the last type of attacks, *subterfuge attack* does not leave any traces when successfully performed, which makes IDS completely unaware of this activity.

Currently Bro can monitor traffic for finger, ftp, portmapper and telnet protocols. As it is easy to extend, by writing new event engines in Bro language the scope of Bro can be enlarged.

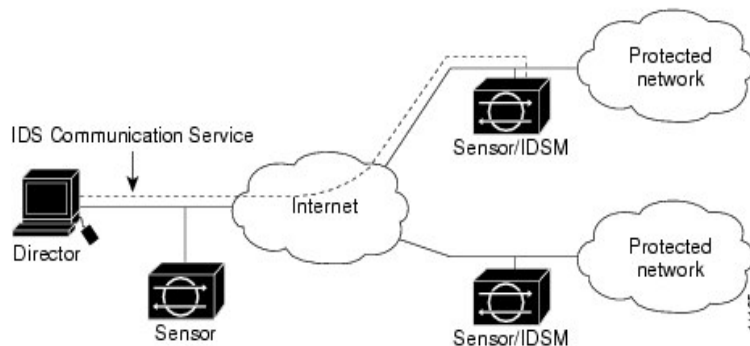
### **2.3 NetRanger**

NetRanger is a network based IDS initially developed by WheelGroup, but today it is integrated to Secure Intrusion Detection System by Cisco. It monitors network traffic with specialized hardware integrated in Cisco network products (routers, switches, etc.). It consists of *Sensors*, *Directors* and *PostOffice* components.

A *Sensor* consist of at least two network interfaces. One to communicate with the directors, perform maintenance operations and others to monitor network traffic, capture packets. When a sensor decides to report an incident, it send collected data to its director.

*Directors* provide centralized control on sensors connected to them. A director monitors and controls sensors, collects data and also loads new attack signatures to sensors. To prevent DoS (denial-of-service) attacks to performed, directors do not analyze data themselves. It writes collected data to log files and these files can be read by other third party databases (NSDB, network security database) and can be analyzed separately from IDS. A Director can also handle user defined responses against reported attacks. It can only inform the security staff by showing an alert or also perform start of execution of other tools.

*PostOffice* is responsible for the organization of sensors and directors. It allows point-to-point connection between sensors and related directors to prevent send reports to broadcast. It can address much larger domains than IPv4 and also finds alternate domains between hosts, which provides a fault-tolerant connectivity.



**Netranger System Overview**

## 2.4 NetSTAT

NetSTAT is a real time, distributed network based intrusion detection system. Unlike other network based intrusion detection systems that monitor a single subnetwork for patterns representing malicious activity, NetSTAT is oriented towards the detection of attacks in complex networks composed of several subnetworks. It detects intrusions in real time and monitors events where they are observable. NetSTAT is a distributed application composed of the following components: the network fact base, the state transition scenario database, a collection of general purpose probes, and the analyzer.

*The network fact base* component stores and manages the security relevant information about a network. The fact base is a stand-alone application that is used by the Network Security Officer to construct, insert, and browse the data about the network being protected. It contains information about the network topology and the network services provided.

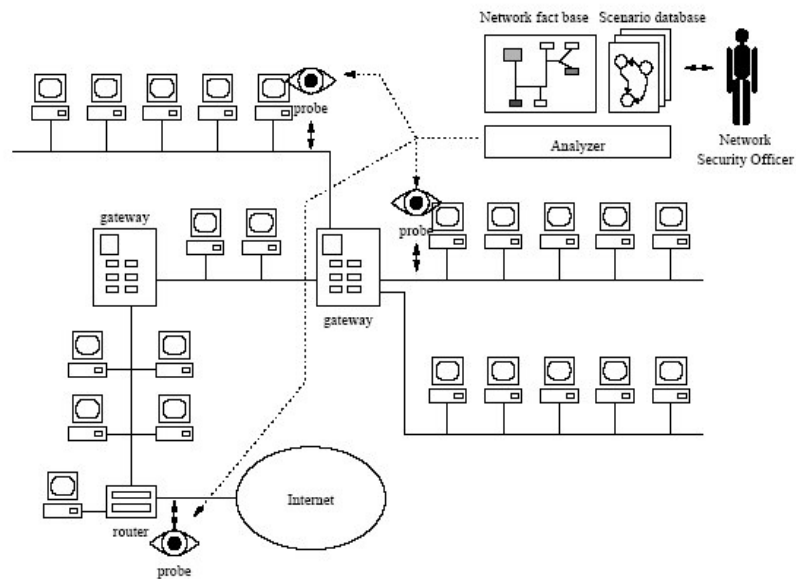
*The state transition scenario database* is the component that manages the set of state transition representations of the intrusion scenarios to be detected. The state transition scenario database can be executed as a stand-alone application that allows the Network Security Officer to browse and edit state transition diagrams using a user friendly graphic interface.

*The probes* are the active intrusion detection components. They monitor the network traffic in specific parts of the network, following the configuration they receive at startup from the analyzer, which is described in the following section. Probes are general purpose intrusion detection systems that can be configured remotely and dynamically following any changes in the modeled attacks or in the implemented security policy.

*The analyzer* is a stand-alone application used by the Network Security Officer to analyze and instrument a network for the detection of a number of selected attacks. It

takes as input the network fact base and the state transition scenario database and determines, which events have to be monitored, where the events need to be monitored, what information must be maintained about the state of the network in order to be able to verify state assertions, etc.

NetSTAT is able to automate security issues using a formal model for network and possible attacks.



**NetSTAT System Architecture**

### 3 Agent Based IDS

The deficiency of centralized intrusion detection systems leads the idea of mobile agents. In an agent based IDS idea, there is no central station, therefore no central point of failure. In addition since agents behave independently, there is no hierarchy between them. A centralized ID system approach is not scalable, because under heavy network load the system suffers from the poor capacity of central analyzer. Also, reconfiguration of sensors is usually difficult. Agents are autonomous softwares that can act independent from other agents and perform different tasks.

In order to use mobile agents, all the hosts in the networks must have an agent platform installed, where the agents are going to be executed. Instead of using static components in a IDS, mobile agent based systems has the following advantages.

**Overcoming Network Latency** - Since agents operate directly on the host, where an action has to be taken, their response is faster than hierarchical systems, where the actions are taken by central coordinator.

**Reducing Network Load** - Instead of sending audit data from sensors to central stations, sending the code of the agent may cause little network load, because audit data may become huge amounts.

**Autonomous Execution** - In order to prevent letting the whole network undefended, when a part of the IDS fails, agents can work autonomously even if their creators don't operate anymore.

**Platform Independence** - Where the agents run on the agent platform, they are independent from the platform of the host.

**Dynamic Adaption** - The system can be reconfigured at run-time because of the agents dynamic behavior.

**Static Adaption** - When a new attack signature has to be added to the IDS, the algorithm of the agents can be updated without restarting the whole system.

**Scalability** - Mobile agents reduce the computational load on the system by dividing it different hosts.

Besides this advantages, mobile agent based IDS have some troubles in security, code size and performance issues. There are several types of security threads against agent based IDS like execution of exploits of different agents, malicious activities against running platform, etc. This security problems are solved by signed agents, encryption and authorization methods. When the complete IDS is considered, it consists of several lines of code and also agents may have so. It means distributing the code of agents over the network may take some time, but in general it is done only once. To overcome this drawback, the common routines of agents are put to the agent platform. Another drawback, performance issue is caused by the languages that the agents are implemented. They are generally scripting languages, which can not even be compared to native code in the meaning of performance.

### 3.1 AAFID

The Architecture for Intrusion Detection Using Autonomous Agents (AAFID) implements a hosts based hierarchical design to deal with disadvantages of centralized systems. The system consists of three layers, each layer calls methods of the layer below. At the base level *agents* collect information, search for suspicious packets and forward collected data to *transceiver*, which exists in the upper layer and once in every host. Transceivers are like agent managers, they control and configure agents on the host, and channel information, most probably exclude unnecessary parts of the

collected information and forward it to *monitors*. In the higher level, each monitor collects data from one or more transceivers and analyzes their input. As each host has one transceiver, monitors collect data from multiple hosts. Monitors can be set hierarchically, having root monitor at the top, which provides user interface to control the IDS.

In order to make agents to communicate with each other, audit information are delivered to agents by the Audit Router. Audit Router holds a database of running agents and their interests. Agents register to this database and get a handle to read the entries of other agents.

To sum up, in the AAFID architecture, agents are used to collect and preprocess information which is needed to detect intrusions in the system. They are not intelligent programs, but due to genetic programming they inherit their experience. In addition transceivers and monitors is aimed to make the system scalable, which allow detection of distributed attacks. The major disadvantage of AAFID architectures is the delay in the detection of the intrusion caused by the layers between agents and the monitor and also monitors are the single point of the failure.

### **3.2 IDA – Intrusion Detection Agent System**

IDA is an implementation of the mobile agent approach in intrusion detection. According to the system designer, attackers follow four common steps to gain unprivileged access to remote machines. In the first step they scan for machines and ports, in the second stage they try to use vulnerabilities of common services. Following stage is the mark stage where the attacker gets actually access to the system and later in the masquerade stage they try to hide their actions by trying to erase logs, etc.

IDA is aimed to detect intrusions by scanning marks left by the intruder, who is actually in the mark stage. The MLSI, which stands for Mark Left by Suspected Intruder, is defined by suspicious activities like opening a root shell, trying to change important files, etc.

The IDA has a simple structure, it consists of a central manager on each network segment, sensors and multiple kind of agents. The central manager is responsible for collecting information from sensors and analyzing it. The sensors, as usual monitor network traffic according to MLSIs. The first kind of agent, *Information Gathering Agent* (IDA), which exists once for every MLSI, is responsible for examining logs in detail. Another type of agent, *Tracing Agent* finds the source of the attack. It works in coordination with IDAs. Providing agents to communicate with each other and the manager, is the *Communication Boards* duty.

IDA uses agents to collect information about the intrusion and the intruder, but the management is still centralized. The main disadvantage of the architecture is scalability, because managers can deal with only a limited amount of sensors and agents.

### 3.3 Micael

Micael is another agent based approach, where decision making, distributed, autonomous agents have the role of investigating intrusions. Its architecture consists of the following elements.

*Sentinels* are static agents existing on every host. They do not have any knowledge about different attacks, but they are against distributed attacks.(DoS etc.).

*Detachments* are mobile agents to investigate intrusions. When a possible attack is detected, they are moved to the related host and start to examine log files in detail. In case they decide that there is an attack, they can perform various actions from disconnecting the host to counterattacking to the intruder.

*Headquarters* are centralized agents collecting data from sentinels and creating new detachments if needed.

In an example scenario, the intrusion is detected as it follows. First sentinels look into data for a suspicious activity, when they find, they request for a detachment from the headquarters. The detachment is created by headquarters and sent to the host. After it arrives to the host, it analyzes information more in detail and if decides to an attack, it take actions and then returns the result to headquarters.

Micael is a successful implementation of the agent based approach with several advantages. Its agents are not used only collect data, they can also react to incidents. In addition the whole system is written in Java, therefore it is very portable, can run on any platform supporting Java.

### References

1. Krügel Christopher, Toth Thomas: A Survey on Intrusion Detection Systems . TU Vienna , Austria (2000) 7, 22–33
2. Jones Anita K, Sielken Robert S.: Computer System Intrusion Detection: A Survey. University of Virginia, USA 19–20
3. Cisco Systems : Cisco Secure Network Intrusion Detection System Overview. <http://www.cisco.com/univercd/cc/td/doc/product/iaabu/csids/csids1/csidsug/overview>.
4. Vigna G., Kemmerer R.: NetSTAT: A Network-based Intrusion Detection Approach University of California, Santa Barbara, USA