International Journal of Soft Computing 3 (6): 403-406, 2008
ISSN: 1816-9503

Intrusion Detection Using Autonomous Agents

[1]S. Janakiraman and [2]V. Vasudevan
[1]Anna University, Chennai-600025, India
[2]Arulmigu Kalasalingam College of Engeering

Abstract: The intrusion detection system architectures universally used in commercial and research systems have a number of tribulations that limit their configurability, scalability or efficiency. The most common inadequacy in the existing architectures is that they are built around a single monolithic entity that does most of the data collection and processing. In this study, we review our architecture for a distributed Intrusion Detection System based on multiple sovereign entities working collectively. We call these entities Autonomous Agents. This approach solves some of the problems.

Key words: Intrusion detection, autonomous agents, commercial IDs, architectures

INTRODUCTION

Intrusion detection: Intrusion detection (ID) is defined (Mukherjee *et al*., 1994) as the problem of identifying individuals who are using a computer system without authorization and those who have legitimate access to the system but are abusing their privileges. An intrusion is defined as any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource (Heady *et al*., 1990). The broad categorization of models of intrusion detection described in Mukherjee *et al*. (1994).

An Intrusion Detection System (IDS) is a computer program that attempts to perform intrusion detection by either misuse or anomaly detection, or a combination of techniques. An IDS should preferably perform its task in real time (Mukherjee *et al*., 1994). Intrusion Detection Systems are usually classified (Mukherjee *et al*., 1994) as host based or network based. Host based systems base their verdicts on information obtained from a single host, while network based systems obtain data by monitoring the traffic of information in the network to which the hosts are connected.

The desirable characteristics of an Intrusion detection system (Crosbie and Spafford, 1995) includ:

- It must be fault tolerant in the sense that it must be able to convalesce from system crashes and reinitializations.
- It must defy subversion. The IDS must be able to monitor itself and detect if it has been modified by an attacker.
- It must impose a minimal overhead on the system where it is running.
- It must provide graceful degradation of service in the sense that if some components of the IDS stop working, the rest of them should be affected as little as possible.
- It must allow dynamic reconfiguration, this is, the ability to reconfigure the IDS without having to restart it.

Limitations of existing ID: Many of the existing network and host based IDSs (Heberlein *et al*., 1990; Heady *et al*., 1990) perform data collection and analysis centrally using a cmonolithic architecture. By this we mean that the data is collected by a single host, either from audit trails or by monitoring packets in a network and analyzed by a single module using different techniques. Other IDSs perform disseminated data collection by using modules distributed in the hosts that are being monitored, but the collected data is still shipped to a central location where it is analyzed by a colossal engine. A good review of systems that take both approaches is presented in Mukherjee *et al*. (1994). There are a number of problems with these architectures:

- The central analyzer is a single point of failure. If an intruder can somehow prevent it from working, the whole network is without protection.
- It is difficult to reconfigure the capabilities to the IDS. Changes and additions are usually done by editing a configuration file, adding an entry to a table or installing a new module.
- Analysis of network data can be flawed. As in (Mukherjee *et al*., 1994), performing collection of network data in a host other than the one to which the data is destined can provide the attacker the possibility of performing Insertion and Evasion attacks.

Corresponding Author: S. Janakiraman, Anna University, Chennai-600025, India

Other intrusion detection systems have been designed to do distributed collection and analysis of information. A hierarchical system is described in Staniford-Chen *et al.* (1996) and White *et al.* (1996) describes a cooperative system without a central authority. These systems solve some of the problems mentioned.

Autonomous agents: We define an autonomous agent as a software agent that performs a certain security monitoring function at a host. We term the agents as autonomous because they are independently-running entities. Agents may receive complex control commands from other entities. An agent may perform a single very specific function, or may perform more complex activities.

As agents are independently running entities, they can be added and removed from a system without varying other components. Still, agents may provide mechanisms for reconfiguring themselves without having to restart. Also, agents can be tested on their own before introducing them into a more complex environment. An agent may also be part of a group of agents that perform diverse simple functions but that can exchange information and derive more complex results than any one of them may be able to obtain on their own.

Since, agents can be stopped and started without disturbing the rest of the IDS, agents can be upgraded to new versions and as long as their external interface remains unchanged, other components need not even know that the agent has been upgraded.

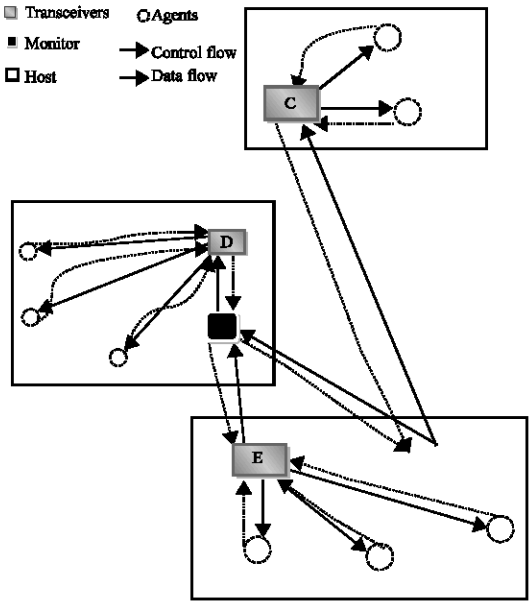

Fig. 1: Physical layout of the components

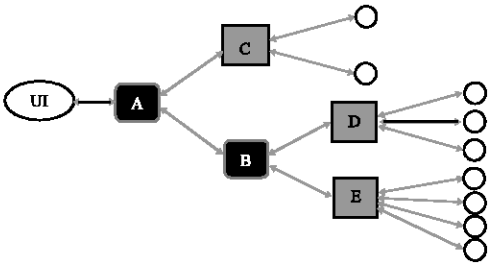

Fig. 2: Logic organization of the system

SYSTEM ARCHITECTRE

We propose architecture for building IDSs that uses agents as their lowest level element for data collection and analysis and employs a hierarchical structure to allow for scalability. A simple example of an intrusion detection system that adheres to our architecture is shown in Fig. 1. Figure 1 shows the three essential components of the architecture: agents, transceivers and monitors.

Our system can be strewn over any number of hosts in a network. Each host can contain any number of agents that observe for interesting events occurring in the host. All the agents in a host report their findings to a single transceiver. Transceivers are per host entities that oversee the maneuver of all the agents running in their host. They exert control over the agents running in that host and they have the ability to start, to stop and to send configuration commands to agents. They may also perform data reduction on the data received from the agents. Finally, the transceivers account their results to one or more monitors. Each monitor oversees the operation of several transceivers. Monitors have access to network wide data, therefore, they are able to perform higher level correlation and perceive intrusions that involve several hosts. Monitors can be organized in a hierarchical fashion such that a monitor may in turn report to a higher-level monitor. Also, a transceiver may report to more than one monitor to provide redundancy and resistance to the failure of one of the monitors. Eventually, a monitor is responsible for providing information and getting control commands from a user interface. This logical organization, which corresponds to the physical distribution portrayed in Fig. 1 and 2.

All the components export an API to communicate with each other and with the user.

Components of the architecture

Agents: An agent is an independently running entity that monitors certain aspects of a host and reports abnormal or interesting behavior to the apt transceiver. The agent

would then engender a report that is sent to the proper transceiver. The agent does not have the authority to directly generate an alarm. Usually, a transceiver or a monitor will generate an alarm based on information received from one or more agents. By coalescing the reports from different agents, transceivers build a picture of the status of their host and monitors build a picture of the status of the network they are monitoring. Agents do not communicate directly with each other in this architecture. Instead, they send all their messages to the transceiver. The transceiver decides what to do with the information based on agent configuration information. As long as the agent produces its output in the appropriate format and sends it to the transceiver, it can be part of this system.

Transceivers: Transceivers are the external communications interface of each host. They have two roles: control and data processing. For a host to be monitored by this system there must be a transceiver running on that host. In its control role, a transceiver performs the following functions:

- Starts and stops agents running in its host. The commands to start and stop agents can come either from configuration information, from a monitor, or as a retort to specific events.
- Keeps track of the agents that are running in its host.
- Responds to commands issued by its monitor by providing the appropriate information or performing the requested actions.

In its data processing role, a transceiver has the following duties:

- Receives reports generated by the agents running in its host.
- Does appropriate processing on the information.
- Distributes information to other agents or to a monitor, as appropriate.

Monitors: Monitors are the highest level entities in this architecture. They also have control and data processing roles that are similar to those of the transceivers. The main difference between monitors and transceivers is that a monitor can control entities that are running in several different hosts whereas transceivers only control local agents. In their data processing role, monitors receive the reduced information from all the transceivers they control and thus can do higher level correlations and detect events that involve several different hosts. Monitors have the capability to detect events that may be unobserved by the transceivers. In their control role, monitors can receive instructions from other monitors and they can control transceivers and other monitors. Furthermore, monitors have the ability to communicate with a user interface and provide the access point for this system.

Communication mechanisms: The transmission of messages between entities is a central part of the functionality of this system. Although this architecture does not specify which communication mechanisms are to be used, there is a minimum set of characteristics that we consider desirable.

We consider the following to be some important points about the communication mechanisms used this system:

- Appropriate mechanisms should be used for different communication needs.
- The communication mechanisms should be efficient and reliable in the sense that they should not add significantly to the communications load imposed by regular host activities and provide reasonable expectations of messages getting to their destination quickly and without alterations.
- The communication mechanisms should be secure in the sense that they should be resistant to attempts of rendering it unusable by flooding or overloading and provide some kind of authentication and confidentiality mechanism.
- The topics of secure communications, secure distributed computation and security in autonomous agents have been already studied (Staniford-Chen *et al*., 1996).

IMPLEMENTATIONS

We have developed two models based on this architecture and we are currently in the process of improving those implementations as well as developing new ones.

The first model was programmed in a combination of Perl (Wall *et al*., 1996) and C (Kernighan and Ritchie, 1998) and was intended as a proof of notion for the architecture. In this implementation, much of the behavior of the components was hard coded and it was not extremely configurable. It used UDP as the inter host communication mechanism and Solaris message queues as the intra-host communication mechanism. This prototype allowed to:

- Identify some design issues that had to be improved.
- This architecture could work for doing distributed detection of anomalous events.

- Gain some experience in writing agents that allowed us to identify important functionality that is needed for all agents.

The second model is written exclusively in Perl, which has the advantage of making it easy to port to other architectures at the expense of some performance loss. The main aim of this implementation is to allow for widespread testing of the architecture; therefore, emphasis has been made in its ease of use, configurability and extensibility.

Some of the major contributions of this new implementation are:

- Increased portability because it is written completely in Perl.
- Implementation of an infrastructure that provides all the base services necessary for developing new entities.
- Definition of an internal API for developing new agents.

These are some of the specific points that we have identified as relevant for future work are Developing agents, Low-level implementations, Communication mechanisms, Developing transceivers and monitors, Semantics of the communication and Data reduction.

CONCLUSION

We propose architecture for Intrusion Detection Systems, which is based on independent entities called Autonomous Agents for performing distributed data collection and analysis. Centralized analysis is done on a per-host and per-network basis by higher level entities called Transceivers and Monitors. The architecture allows for computation to be performed at any point where enough information is available.

This architecture allows data to be collected from multiple sources, thus being able to combine the best characteristics of traditional host based and network based IDSs. It apparently also allows building IDSs that are more resistant to insertion and evasion attacks (Mukherjee *et al*., 1994) than existing architectures, although no tests have been performed to support this claim. Furthermore, the modular characteristics of the architecture allow it to be easily extended, configured and modified, either by adding new components, or by replacing components when they need to be updated.

User interface is a big issue for future work. Most of the work that has been done in Intrusion Detection focuses on how to perform the detections, but very little has been done in the way of presenting the information to the user.

REFERENCES

Crosbie, M. and G. Spafford, 1995. Active defense of a computer system using autonomous agents. Technical Report 95-008, COAST Group, Purdue University. West Lafayette, IN: 47907-1398.

Heady, R., G. Luger, A. Maccabe and M. Servilla, 1990. The architecture of a network level intrusion detection system. Technical report, University of New Mexico.

Heberlein, L., G. Dias, K. Levitt, B. Mukherjee, J. Wood and D. Wolber, 1990. A network security monitor. In: Proc. IEEE. Symp. Res. Security Privacy, pp: 296-304.

Kernighan, B.W. and D.M. Ritchie, 1988. The C Programming Language. Prentice-Hall. 2nd Edn. Englewood Cliffs, NJ 07632, USA.

Mukherjee, B., T.L. Heberlein and K.N. Levitt, 1994. Network intrusion detection. IEEE. Network, 8 (3): 26-41.

Ptacek, T.H. and T.N. Newsham, 1998. Insertion, evasion and denial of service: Eluding network intrusion detection. Technical report, Secure Networks, Inc.

Staniford-Chen, S., S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip and D. Zerkle, 1996. GrIDS: A graph based intrusion detection system for large networks. In: Proc. 19th National Information Systems Security Conference. National Institute of Standards and Technology, 1: 361-370.

Wall, L., T. Christiansen and R.L. Schwartz, 1996. Programming Perl. 2nd Edn. O'Reilly and Associates, Inc.

White, G.B., E.A. Fisch and U.W. Pooch, 1996. Cooperating security managers: A peer-based intrusion detection system. IEEE Network, pp: 20-23.