The intrusion detection system with Learning Automata

Hamdollah Ghamgin¹, Mohammad Taghi Jafari¹, morteza salari akhgar²

1. Zanjan Branch, Islamic Azad University, Zanjan, Iran.
2. Department of computer, Ghorveh Branch, Islamic Azad University, GHorveh, Iran

Corresponding Author: Hamdollah Ghamgin

ABSTRACT: Intrusion detection, a topic that has evolved heavily due to the rising concern for information technology security, has endured numerous architecture abstractions. All of these architecture abstractions have strengths and weaknesses with regards to various factors like efficiency, security, integrity, durability, and cost-effectiveness, to name a few. In this paper, we will attempt to describe the architecture of intrusion detection that minimizes the weaknesses of this model. Our architecture will heavily build upon the Autonomous Agents For Intrusion Detection (AAFID) architecture, which has already been implemented in the Center for Education and Research in Information Assurance and Security (CERIAS) center in Purdue University. We will, however, design a different functionality for our agents, making them rather intelligent. Such intelligent agents will seek to use tools that the field of artificial intelligence provides in order to maximize their probability of detecting intrusions.

Intrusion Detection
We start our discussion by explaining intrusion detection and why it has become an important need in the information technology field. We will examine several basic definitions that are of fundamental importance to the development of an intrusion detection system.

Intrusion Detection: Fundamentals
In order to comprehend the intricate abstraction of an intrusion detection system, we must first understand the basic definition of one. Intrusion Detection (ID) is defined as “the problem of identifying individuals who are using a computer system without authorization (i.e., ‘crackers’) and those who have legitimate access to the system but are abusing their privileges (i.e., the ‘insider threat’)”. For our purpose, we will use the word “intrusion” to mean both an external and internal threat (Jai Sundar et al., 1998).

It is also important to formally define an attempt as “any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource”. These definitions can let us clearly define the system.

The models of intrusion detection are:

Misuse detection model
The intrusion will be detected by observing specific patterns in the system that are used by known attacks (also known as signatures).

Anomaly detection model
Using a statistical model, the system can observe deviations from known signatures (described in), and present these activities as presumed attacks (David et al., 1995).

An Intrusion Detection System (IDS) is defined as “a computer program that attempts to perform ID by either misuse or anomaly detection, or a combination of techniques”. An IDS is then further categorized as a host-based or network-based. Host-based systems base their analyzation on data retrieved at a single host, while network-based monitor the entire network to which the host is connected.

Note that an IDS aims at preventing network intrusion by warning administrators of malevolent and unusual activities, not by actually mounting an offensive on such activities. Examples of network attacks are either external or internal attacks, such as: IP spoofing and packet sniffing (external), and password attacks and session hi-jacking (internal). Current mechanisms that have been used for illegal behavior detection in security programs
have been listed as: statistical anomaly detection, rule-based detection, and hybrid detection (a combination of rule-based and statistical anomaly). Not surprisingly, these mechanisms have also been used in current ID systems (John et al., 2000).

Why an ID is needed?

Intrusion detection grew out of the need to keep financial audits on the users who accessed the then costly computing time of mainframes. This gave birth to network monitoring in the 1990s, which established a way for administrators to maintain tight control over their networks.

The large impact and growth of the Internet, coupled with the importance of information technology in today’s commerce, has created the increasingly important need to maintain the integrity of data.

Statistics show that the number of people online in January 1993 was 1,313,000 compared to 73,398,092, the number of people online in January 2000. The Computer Security Institute (CSI) reported that IT professionals experienced $256 million in losses due to network attacks in 2001, up 28% from 1999.

What is even more astonishing is that reports from the F.B.I. and the C.I.A. claim that 80% of intrusion problems are caused internally, that is from users that already belong to the system. Even government institutions themselves, such as the CIA, have hacked into the computer of European officials as part of espionage plots. These plots carry the purpose of stealing political and economic secrets to leverage political power. The increase in attacks has made it apparent that the resources spent on implementing an IDS is worth it (Sandeep and Gene, 1994).

Types of ID systems

As previously explained, an IDS may be categorized as a host-based or network-based, and each model has its advantages. A network-based model has the benefits of evidence collection and real-time warnings. Host-based models can detect both external and internal misuse. This is of particular interest because security breaches are more likely to come from an internal user (Stuart and Peter, 2001).

However, regardless of its implementation, an IDS seeks to have certain desired characteristics:
- The system must run continuously with minimal supervision.
- The system must be fault tolerant, meaning it must be able to recover from crashes to a previous safe and functional state.
- The system must be self-conscious of any activities that attempt to attack and modify itself. This condition is known as resisting subversion.
- Since the IDS aims at being beneficial to the network it protects, it must run with minimal overhead to the network.
- An IDS must be configurable to the security policies administered by its operator.
- The IDS must be able to adapt to non-intrusive changing conditions of the network (e.g. system upgrades, and changing of resources by user).
- Since speed is of extreme importance in the event of an attack, the IDS must be efficiently scalable to monitor large number of hosts while still providing information in a timely manner.
- Graceful degradation of service must be provided. In other words, if some components of the IDS fail, other components should not be affected.
- The IDS must allow dynamic configuration, so that if a group of hosts is being monitored, it becomes unpractical to restart all if an update must be made to a host.
- Another desired characteristic of an IDS architecture is the ability to implement it in a distributed format. In an ideal case where an IDS accurately protects the integrity of the system, a likely target for an attack will be the IDS itself. Therefore, an IDS must not only enforce tight security in itself, but it must not implement a centralized architecture. A centralized structure will allow an attacker to bring down an IDS by attacking a single point, which is undesired (Thomas Ptacek and Timothy Newsham, 1998).

ID Systems Limitations

Existing ID systems have certain limitations that impede some desirable processes of an ID system. Most systems are monolithic in architecture, although they perform distributed operations. The limitations, however, arise out of this monolithic implementation (Biswanath et al., 1997). They consist of:

- Monolithic architectures contain a “central analyzer” which represents a single point of failure if compromised by an attacker.
- If all processing takes place on a single host, scalability is jeopardized, as a limit is placed on the size of the network that can be monitored.
Reconfiguration or upgrading may present a problem, as restarting the host may leave the network vulnerable to attack.

The integrity of network data used for analysis can be compromised. C.E.R.I.A.S. has shown that “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”. Such attacks use mismatched assumptions in the protocol stacks of different hosts to create or hide attacks. C.E.R.I.A.S. also lists systems of monolithic architecture to lack a graceful degradation of service (a desirable characteristic, as previously mentioned) (James, 1998).

Due to these limitations, the ID field has seen a migration towards designing distributed systems in the past few years.

**Intelligent Agents**

An agent is defined as “anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.” Furthermore, an ideal rational agent is an agent that, “For each possible sequence, an ideal rational agent should do whatever action is expected to maximize its performance measure, on the basis of the evidence provided by the percept sequence and whatever built-in knowledge the agent has.” From this definition, we understand how the agent reaches its decision, which is either perceptual evidence or built-in knowledge. It is important to understand that agents play the key role in finding an intrusion. These agents must not only be intelligent enough to discover common intrusions, but must also be able to adapt and learn new attacks. It is for this reason that we will use agents capable of intelligent data analysis as our preferred agents.

**What is an Intelligent Agent?**

In our discussion, we will define an intelligent agent as an agent that uses artificially intelligent methods in order to accomplish its purpose. Our definition will also take advantage of several definitions already defined in the field on IDS. Our agent definition will include autonomous agents, which are “a software agent that performs a certain security monitoring function at a host”. We will also include itinerant agents, which have been defined by IBM as “programs which are dispatched from a source computer and which roam among a set of networked servers until they are able to accomplish their task”.

In this case, intelligent agents will roam the network, and we will design their task to either never expire, or to be relieved by another agent. Also, it is important that our intelligent agent carries the capability of learning in order to become adaptive to new intrusive attacks while being able to detect patterns in previously known attacks.

**Role of an Intelligent Agent**

In order to maximize the role of an intelligent agent, one must first understand the detection process. The agents’ detection strategy will be twofold:

1. It will have to detect known intrusions in a timely manner.
2. It will have to learn to detect new, unknown intrusions and remember them for future reference.

Thus, the role of the intelligent agent in an intrusion detection system will be very similar to the general behavior of the human immune system. Once it learns about a particular viral strain, it will use that knowledge to defend against it in the future.

Once detection has been made, the intelligent agent must communicate and notify other system components. In order to quickly mitigate the damage of the attack, the agents must first notify the system administrator, in detailed form, that an intrusion is taking place. Then, the agent must quickly communicate to other agent(s) the new intrusion behavior it has learned.

Therefore, an important role of an intelligent agent is that it must be able to communicate its knowledge, so that other agents may benefit from it. IBM refers to this when it describes an agent meeting point as: “an abstraction which supports the interaction of agents with each other and server based resources”. We will assume in our architecture that an abstraction that allows intelligent agents to pass information securely from one to another, or broadcast from one to many exists. After an intrusion has been attempted, the agent must immediately and efficiently continue to perform its security monitoring.

**Intelligent Agent Implementation**

Our next concern is the implementation of the intelligent agent (although we shall only concern ourselves now with the design of the agent, and not its domain). A good design scheme has been implemented in...
C.E.R.I.A.S. for autonomous agents, for which they argue the following to be necessary: Agents should be independent running entities, so they can be added, removed or re-configured without restarting the IDS. Agents can be tested on a simple testing environment before introducing them to a more complex domain. Agents may be part of a group of agents that can communicate and derive complex results as a group. If the agents are organized in a mutually independent set and an agent stops working, it should not cause the entire system to cease working.

Organizing agents in a hierarchical structure with multiple layers will reduce the data transferred from level to level, thus making the system scalable.

Since agents are implemented as a separate process (controlled by the O.S.), each agent can be implemented in the programming language that is best suited for.

Intelligent agents must, therefore, adhere to these beneficial characteristics. However, there still exists the question of how to detect an intrusion. As mentioned earlier, intrusions fall into one of two categories: the intrusion is either known (the agent recognizes the pattern), or it is new (agent does not recognize the pattern) and has never been done before in the network. Furthermore, if an intrusion falls into the unknown category it can either be very similar to a known intrusion, or it can be completely different. Thus, at any point during the monitoring we can be sure that if an intrusion occurs, it will be:

- A known or recognized intrusion.
- An intrusion that presents similar characteristics to a known one.
- An intrusion that is completely different from all others that are known.

Our intelligent agents will therefore go through a threefold process when detecting an intrusion. It will first try use a rule based system to see if the intrusion matches one of its rules. If not, it will use a statistical anomaly algorithm to detect if it has any similarities to any known patterns stored in the rule-based system. Lastly, if the intrusion is still not recognized, it will “learn” the patterns of this new intrusion and convert it to a rule for future knowledge. Immediately, it will seek to communicate with other agents and pass its knowledge of the newly detected intrusion.

A rule-based system contains a set of rules that describe malevolent and illegal user behavior patterns. The rules are formed from the analysis of previous attacks. The only disadvantage of this system is that it cannot detect and learn new attacks for which patterns are not described in the rule-based system. Modern rule-based systems rely on expert systems to identify attacks, which permit the access to an extensive amount of human experience in intrusion. These systems utilize the knowledge stored in their expert system to identify intrusions that match defined patterns of attack.

The second attempt to recognize the intrusion will be the statistical anomaly algorithm. This algorithm analyzes audit-log data to detect abnormal behavior in the system from an expected profile. This profile is created from how an organization expects a certain user to behave and access system resources. Thus, once a profile of the user’s expected behavior is created, audit logs typically compare the user’s current patterns to his/her expected patterns. If the current patterns differ immensely from profiled ones (determining the degree of difference that causes the alarm is up to the administrator), then it might become a potential intrusion. The drawback to this method is that it only works if a user profile exists, which more than likely restricts the intrusions to internal intrusions.

Likewise, we can use the statistical anomaly method to create profiles for “intrusions” rather than users. For example, if an activity is very similar to a related attack, we can warn the system administrator that such an action might result in a malevolent act. The system administrator will be able to set the statistical variables that tell the system what degree of tolerance it should perform with this method. This method has been established in Kumar and Spafford’s security enhancement software.

Lastly, our intelligent agent must be able to “learn” new intrusion patterns. The most useful format is to use neural networks to develop an adaptive atmosphere. Neural nets have been discussed as the method of choice to train agents to learn new intrusive patterns. Referring to neural nets and intrusion, Cannady properly stated this scenario: “The constantly changing nature of network attacks requires a flexible defensive system that is capable of analyzing the enormous amount of network traffic in a manner which is less structure than rule based systems. A neural network-based misuse detection system could potentially address many of the problems that are found in rule-based systems”.

**Design Issues**

The intelligent agents will, therefore, perform intrusion detection by:

First, comparing the patterns with a rule-based expert system.
If no intrusion patterns match the ones in the expert system, it should try a statistical anomaly algorithm for the user, and test for similarity to known signatures. If there is still no success in the detection (and there exists an intrusive action), it should “learn” this intrusion via neural nets.

Although the incorporation of neural nets takes care of any disadvantages with an expert system, there are several issues with neural nets that may present problems. There are two major reasons why neural network implementations are difficult for an intrusion system. Neural nets are dependent on training time, and in order for an intrusion to be detected, the neural network must have training in the attack. For example, the training of neural nets for intrusion detection purposes will require a large amount of attack sequences and signatures. Such sequences will require a large amount of time to obtain. Also, qualitative data that accurately represents a new attack may not be available.

Second, neural networks also have an inability to show the basis for their successful decisions. This problem has plagued neural net implementations in the past, and is often know as the “black box” theory. Cannady states: “Unlike expert systems which have hard-coded rules for the analysis of events, neural networks adapt their analysis of data in response to the training which is conducted on the network” Furthermore, Cannady adds: “The connection weights and transfer functions of the various network nodes are usually frozen after the network has achieved an acceptable level of success in the identification of events. While the network analysis is achieving a sufficient probability of success, the basis for this level of accuracy is not often known”.

A challenge is therefore presented, because although the actual agent will be now be skilled at detection for a particular attack, it will have difficulties communicating to other agents what attack pattern symbolizes this intrusion. However, by designing an intelligent agent that stores learned signatures in a rule-based architecture, transceivers can be used to communicate newly learned patterns to the other agents.

System Architecture

An architecture proposed by C.E.R.I.A.S. called Autonomous Agents For Intrusion Detection (AAFID) uses autonomous agents to perform intrusion detection. We shall build on this architecture by adding intelligent agents, which will allow our IDS to be able to detect and learn new intrusions.

Overview

A simple diagram which follows an AAFID architecture is displayed in Figure 1. This basic configuration shows the three essential components: agents, transceivers, and monitors. In this hierarchical design, each level will minimize the overhead by performing data reduction per level. Although the components above are based on the AAFID architecture, we shall add to their model by adding intelligent agents. Our defined intelligent agent will make it possible to learn and detect new attacks.

Following the distributed architecture provided by the AAFID system, we will allow each host to have a number of agents which will be responsible for the monitoring of that host. Per host, the agents will report their activities to a transceiver, which oversee the operations of their agents in that host. Transceivers will have the ability to start, stop, and configure the agents.

Transceivers then report their results to one or more monitors, which oversee the operation of transceivers. Since monitors are in charge of per-host entities, they can oversee a wider view of the network, and as such, can detect intrusions on several hosts. A monitor will not only perform a higher level of data correlation, but it may be organized in a hierarchical fashion. The monitor will be responsible for providing intrusion information and receiving commands from the user interface. Application programmatic interfaces (API) will be used to communicate between components.

Components

We shall provide an in-depth description of the architecture components in order to understand the design. Agents are the most essential part of the system. Since we have already described their internal design, we will focus on describing their interaction with the external environment. First, it is important to realize that the agent does not have the authority to directly generate an alarm. The results gathered by the agents are sent to transceivers and then monitors, which do possess the authority to generate alarms. To provide inter-agent communication, agents must communicate to a transceiver, which in turn will route the messages to another agent. Several interesting possibilities are mentioned as part of the AAFID system which we will implement; Using genetic programming techniques, agents may evolve. In order to detect long term attacks, agents must employ a mechanism which remembers a state (Eugene and Diego, 2000).
Agents will be allowed to migrate, using iterant agents (or mobile agents) architecture

Language implementation for all agents will depend on different factors (operating environment, speed, ease of use) and thus will be left to the developers choice.

Transceivers will be responsible for two functions: control and data processing. During its control responsibility, transceivers will:
- Start and stop agents, which will be encapsulated as an order from monitors or another agent.
- Be responsible for keeping track of all agents active on the host.
- Respond to different requests by monitors.

**During its data processing responsibility, transceivers will**
- Receive reports generated by its agents.
Process the data (analyze, minimize) being brought by the agents.
Route the appropriate results to the monitors.

Further, monitors are seen as the highest level entity in the AAFID architecture. Monitors are the ones that have interaction with the user, and may detect an attack that is unforeseen by transceivers. Monitors are different from transceivers in the sense that they can control entities running in separate hosts, whereas transceivers only control local agents. Monitors, like transceivers, have a data processing and control role. In the data processing role, monitors receive data correlated by transceivers that contain the results collected by agent groups per host. In their control role, monitors can control transceivers and agents, and may communicate with other monitors. Also, monitors allow for an API which allows user interaction, which allows the results to be retrieved from monitors or other lower level components.

In the AAFID architecture, C.E.R.I.A.S quotes: “The most complex and feature-full IDS can be useless if it does not have good mechanisms to allow users to interact with and control it”. Therefore, it is of major importance to build a comprehensive API in the monitor that allows the user to gather any results desired. We will not give such an implementation, as it will be left up to the developer (Dorothy Denning, 1987).

**Inter-component communication**

A crucial part of our system is to provide secure and accurate communication between all components. Any security compromise in the communication of the system can potentially become malevolent and bring the IDS down. Specifically, since our intelligent agents have the ability to communicate information, it is important that the information passed be accurate. If an intelligent agent is somehow compromised, erroneous data can cause the system to be open for an attack.

Additionally, the following are desirable characteristics:
- Different mechanisms should be used for different communication needs.
- Mechanisms should not add a large overhead or latency to the network they protect.
- Communications must be reliable to arrive at their destination.
- Security and confidentiality should be supported by a message mechanism.

Topics in secure communication and security in autonomous agents have been discussed in the AAFID architecture.

**Implementation**

Although implementation details are left up to a development team, we will look at examples of the AAFID architecture developed in Purdue University which serve as a proof-of-concept. Although the implementations do not take advantage of the intelligent agents defined, such agent implementations are feasible due to advancements in the artificial intelligence field.

**Real World Examples**

The first prototype created was built using a combination of low-level languages and scripting languages; languages used included Perl, Tcl/Tk, and C. Since these languages are very system native and unportable, the system did not allow for a flexible configuration. Much of the work done in this system was hard-coded, and it used UDP as the communication protocol (which is unreliable). Agents were developed according to the AAFID architecture, but they did not implement the intelligent features discussed earlier. Besides serving as a proof-of-concept, this implementation helped by giving experience in writing agents, and identifying other design issues, such as the importance of a GUI for the monitor (William et al., 1996).

The second prototype was written mostly in Perl, increasing its portability to different operating systems. Drawing from experiences in the first prototype, the second prototype allows testing of the architecture, ease of use, and configurability. The major contribution of this implementation are: Perl implementation allows for portability.

A complete infrastructure implementation that allows development of new abstractions.
A clear separation of abstractions and communications was established.
Communication details were established, including encapsulation and security for inter-component communication.
This implementation allows for the current testing of new abstractions, and the exploring of new communication mechanisms.
Design Issues

The development of these prototypes have allow developers at C.E.R.I.A.S. to experience some design issues. It is of paramount importance to understand these issues, so they can be resolved in future systems.

The main issues gathered from the second prototype have been categorized as: communications of components, impact of an IDS on the performance of its network, data processing and reduction, and user interface design. The communications of components have been categorized as intra-host and inter-host communication. Their design must provide not only a secure and reliable transfer, but must maintain a small (if any) overhead on the system. Since most of the information analyzed will require a context switch from kernel space to user space, this change will create an undesirable overhead, which needs to be addressed. Network traffic will also increase due to all the data processing, as agents will have to process and forward data to transceivers, which will do the same for monitors. Lastly, the system will suffer greatly if there is not a coherent user interface which allows the administrator to retrieve any data he/she needs (William et al., 1996).

CONCLUSION

An IDS has grown to be an important defense in an information technology firm. Advances in the fields of security, networking, and artificial intelligence have contributed largely to an IDS which will be ready to tackle modern and spontaneous attacks. The architecture described, coupled with the intelligent agents defined, provides the design needed to handle such attacks.

REFERENCES


James C. 1998. “Artificial Neural Networks for Misuse Detection.” School of Computer and Information Sciences, Nova Southeastern University, Fort Lauderdale, FL 33314, October 5.


