



**Rules Based Network Anomaly
Detection using Learning Automata
IKT407
Autumn 2005**

HØGSKOLEN I AGDER
Agder University College
Faculty of Engineering and Science

Authors: Ali Chelli Chen LEIMING Farouk DHAHBI	Supervisor: Prof. Ole-Christoffer GRANMO
Version: 1 Status: Final	Pages: 36 including this page Modified date: 21/11-2005
Keywords: Learning Automata, rule based anomaly detection, personal firewall, anomaly detection, signature detection.	
Abstract: Nowadays more and more companies are doing their business on the internet, this fact increase the threat of network intrusion. According to many surveys the loss due to intrusion is increasing from year to year. Facing this problem many solutions has been developed, but the achieved results can be improved. In this project we evaluate a new Learning Automata scheme for anomaly detection in computer networks. At the hart of the scheme is a game between cooperative learning automata. The purpose of the game is to form a Boolean expression over packet bits, that can be used to decide whether a packet is normal or not.	

Table of contents

1 EXECUTIVE SUMMARY	4
2 INTRODUCTION	5
3 BACKGROUND.....	6
3.1 DENIAL OF SERVICE ATTACKS.....	7
3.1.1 <i>Methods of attack</i>	7
3.1.2 <i>Effects of DoS</i>	9
3.2 APPROACHES	10
3.2.1 <i>Signature detection</i>	10
3.2.2 <i>Anomaly detection</i>	11
3.2.3 <i>Personal Firewall</i>	14
4 PROBLEM STATEMENT.....	14
5 REQUIREMENTS	15
6 APPROACH	17
6.1 LEARNING AUTOMATA PRINCIPAL	17
6.1.1 <i>The T-maze problem</i>	17
6.1.2 <i>Environment Modelling</i>	18
6.1.3 <i>Automaton Modelling</i>	19
6.2 PRINCIPLE OF THE “TWO-ACTION VARIABLE STRUCTURE STOCHASTIC AUTOMATON?”	19
6.3 LEARNING AUTOMATA FOR ANOMALY DETECTION	22
7 IMPLEMENTATION.....	24
8 EVALUATION.....	32
9 DISCUSSION.....	32
9.1 PROJECT OUTCOMES	32
9.2 EVALUATION OF THE OVERALL RESULTS.....	33
10 CONCLUSION	34
APPENDIX	35
LIST OF REFERENCES	35
GLOSSARY AN ABBREVIATIONS	36

Table of figures

FIGURE 1 RULE SPECIFICATION IN PERSONAL FIREWALL BY A HUMAN DECISION MAKER	14
FIGURE 2 LEARNING AUTOMATA FOR ANOMALY DETECTION.....	16
FIGURE 3 T-MAZE PROBLEM.....	18
FIGURE 4 ENVIRONMENT MODELLING	18
FIGURE 5 AUTOMATON MODELLING.....	19
FIGURE 6 RULE PATTERN	23
FIGURE 7 PACKET STRUCTURE	25
FIGURE 8 TRAFFIC GENERATION.....	29
FIGURE 9 LA AND DECISION MAKING.....	30

1 Executive Summary

The network intrusions represent a big threat for user's security. This threat has become a serious problem, since more companies are doing their business on the internet. The static shows a big loss of money for e-business companies. In order to solve this problem many approach have been developed. Mainly we can find two categories of intrusion detection: signature based detection and anomaly based detection. In this last category we find the rule based anomaly detection. The personal firewall is an example of rule based anomaly detectors. The main goal of a personal firewall is to allow non expert to specify rules that are used to decide whether to allow or disallow an incoming traffic. But since the decision maker is a human there is a risk that this person makes mistakes. This risk increases with the size of the network, since in huge networks the number of traffic types is too big.

In order to improve the performance of personal firewall we replace the human decision maker by a collection of Learning Automata. Learning Automata are adaptive decision making devices that can operate in unknown and non deterministic environment. In this project we evaluate a novel Learning Automata scheme for anomaly detection in computer network. At the heart of the scheme is a game between cooperative Learning Automata. The purpose of this game is to form Boolean expression over packet bits that can be used to decide whether a packet is normal or hostile.

We generate network traffic and use Learning Automata in order to find the rules that describe this traffic. Since we have generated the traffic our self we were able to verify the results.

We prove that Learning Automata are able to operate in unknown and non deterministic environment. Learning Automata are also able to find the right rules without human guidance and to handle noisy traffic. Even with the presence of many kind of traffic Learning Automata are able to find the rules that describe every traffic type.

2 Introduction

When a user of an information system takes an action that that user was not legally allowed to take, it is called *intrusion*. The intruder may come from outside, or the intruder may be an insider, who exceeds his limited authority to take action. Whether or not the action is detrimental, it is of concern because it *might* be detrimental to the health of the system, or to the service provided by the system.

Intrusion *detection* involves determining that some entity, an *intruder*, has attempted to gain, or worse, has gained unauthorized access to the system. None of the automated detection approaches of which we are aware seeks to identify an intruder before that intruder initiates interaction with the system. Of course, system administrators routinely take actions to prevent intrusion. These can include requiring passwords to be submitted before a user can gain any access to the system, fixing known vulnerabilities that an intruder might try to exploit in order to gain unauthorized access, blocking some or all network access, as well as restricting physical access. Intrusion detection systems are used in addition to such preventative measures.

The fact that more companies are doing more business on the Internet makes intrusion a very serious problem. According to the FBI survey report on “cyber crime” of the year 2000, ninety percent of respondents detected security breaches over the year 1999. At least 74 percent of respondents reported security breaches including theft of proprietary information, financial fraud, system penetration by outsiders, data or network sabotage, or denial of service attacks. Information theft and financial fraud caused the most severe financial losses, put at \$68 million and \$56 million respectively. The losses from 273 respondents totalled just over \$265 million. Losses traced to denial of service attacks were only \$77,000 in 1998, and by 1999 had risen to just \$116,250. Finally, many companies are experiencing multiple attacks; 19% of respondents reported 10 or more incidents.

Facing these challenges the security field has become a very active domain of research. Many approaches were developed in order to find a solution that can limit the

intrusion threat. Currently there are two basic approaches in intrusion detection. The first approach, called *signature detection*, is based on modelling hostile traffic. The second approach, called *anomaly detection*, is based on modelling normal traffic. The *anomaly detection* contains also two categories:

- The *rule based anomaly detection*, which uses expert rules to describe normal traffic.
- The *statistical anomaly detection*, which uses stochastic model to describe normal traffic.

In anomaly detection the main goal is to model normal traffic. Any received packet that didn't follow the model is considered anomalous. In our project we use a collection of Learning Automata in order to model normal traffic. In this project we use Learning Automata (LA) for anomaly detection in *rule based anomaly detection*. Learning Automata must form rules that describe normal traffic. Many LA will cooperate together in order to form a Boolean expression (rule) over packet bits, that can be used to decide whether a received packet is normal or hostile. Our target is to automatize the rule specification in personal firewall.

This report consists of several sections. We start by presenting some literature about intrusion detection, then we state the problem of our project, afterwards we present the requirements. Then we explain how we have implemented our approach. In a next step we evaluate our solution. Finally we give the project outcomes and evaluate the overall results.

3 Background

The spread of the internet and the tendency to the e-commerce has made the intrusion threat a very serious problem. Many researches were carried out in order to solve this problem. In this section we start by giving examples of Denial of Service (DoS) attacks to illustrate how intrusions happen in real world and to see their effects. Then we give an overview of the different approaches that are developed in order to prevent the network from intrusions.

3.1 Denial of Service Attacks

A denial-of-service attack (DoS attack) is an attack on a computer system or network that causes a loss of service to users, typically the loss of network connectivity and services by consuming the bandwidth of the victim network or overloading the computational resources of the victim system.

3.1.1 Methods of attack

A DoS attack can be committed in a number of ways. There are three basic types of attack:

- consumption of computational resources, such as bandwidth, disk space, or CPU time
- disruption of configuration information, such as routing information
- disruption of physical network components

A nuke attack sends a packet, usually ICMP, which is malformed or fragmented in an invalid way, triggering a bug in the operating system and crashing the targeted computer. This is known as the ping of death, but this bug exists only in old operating system.

WinNuke is a similar kind of attack, exploiting the vulnerability in the NetBIOS handler in Windows 95. A string of out-of-band data is sent to TCP port 139 of the victim machine, causing it to lock up and display a Blue Screen of Death. This attack was very popular between the IRC-dwelling¹ script kiddies², due to easy availability of a user-friendly click-and-crash WinNuke program.

Various DoS-causing exploits³ can cause server-running software to get confused and fill the disk space or consume all available memory or CPU time.

Other kinds of DoS rely primarily on brute force, flooding the target with overwhelming flux of packets, over saturating its connection bandwidth or depleting

¹ Internet Relay Chat

² is a derogatory term for inexperienced crackers who use scripts and programs developed by others to launch attacks

³ An **exploit** is a common term in the computer security community to refer to a piece of software that takes advantage of a bug or vulnerability, leading to privilege escalation or denial of service on a computer system.

target's system resources. Bandwidth-saturating floods rely on the attacker having higher bandwidth available than the victim; common way of achieving this today is via Distributed Denial of Service¹. Other floods may use specific packet types or connection requests to saturate finite resources by, for example, occupying the maximum number of open connections or filling the victim's disk space with logs.

An attacker with access to a victim computer can bring it to a crawl or even to a crash by using a fork bomb².

Ping flood is based on sending the victim an overwhelming number of ping packets, usually using the "ping -f" command.

SYN flood sends a flood of TCP/SYN packets, often with a forged sender address. Each of these packets is handled like a connection request, causing the server to spawn a half-open connection, by sending back a TCP/SYN-ACK packet, and waiting for a TCP/ACK packet in response from the sender address. However, because the sender address is forged, the response never comes. These half-open connections saturate the number of available connections the server is able to make, keeping it from responding to legitimate requests until after the attack ends.

A smurf attack is one particular variant of a flooding DoS attack on the public Internet. It relies on mis-configured network devices that allow packets to be sent to all computer hosts on a particular network via the broadcast address of the network, rather than a specific machine. The network then serves as a smurf amplifier. In such an attack, the perpetrators will send large numbers of IP packets with a faked source address, which is set to the address of the intended victim. To combat Denial of Service attacks on the Internet, services like the Smurf Amplifier Registry have given network service providers the ability to identify misconfigured networks and to take appropriate action such as filtering.

¹ The attack is committed by many computers controlled by the cracker who gain control of those machines using viruses and Trojan horses.

² A fork bomb works by creating a large number of processes very quickly in order to saturate the available space in the list of processes kept by the computer's operating system. If the process table becomes saturated, no new programs may be started until another terminates.

A "banana attack" is another particular type of DoS. It involves redirecting outgoing messages from the client back onto the client, preventing outside access, as well as flooding the client with the sent packets.

Attempts to "flood" a network with bogus packets, thereby preventing legitimate network traffic, are the most common form of attack, often conducted by disrupting network connectivity with the use of multiple hosts in a distributed denial-of-service attack. Specific means of attack include: a smurf attack, in which excessive ICMP requests are broadcast to an entire network; bogus HTTP requests on the World Wide Web; incorrectly formed packets; and random traffic. The source addresses of this traffic are usually spoofed¹ in order to hide the true origin of the attack. Due to this and the many vectors of attack, there are not comprehensive rules that can be implemented on network hosts in order to protect against denial-of-service attacks, and it is a difficult feat to determine the source of the attack and the identity of the attacker. This is especially true with distributed attacks.

Attacks can be directed at any network device, including attacks on routing devices and Web, electronic mail, or Domain Name System servers.

3.1.2 Effects of DoS

Denial of Service attacks can lead to problems in the network 'branches' around the actual computer being attacked. For example, the bandwidth of a router between the Internet and a LAN may be consumed by a DoS, meaning not only will the intended computer be compromised, but the entire network will also be disrupted.

If the DoS is conducted in a sufficiently large scale, entire geographical regions of Internet connectivity can also be compromised by incorrectly configured equipment without the attacker's knowledge or intent. For this reason, most, if not all Internet service providers ban the DoS.

¹ the creation of IP packets with a forged (spoofed) source IP address

3.2 Approaches

Many approaches were developed in order to find a solution to protect networks from intrusion threat. Currently there are two basic approaches for intrusion detection. The first approach, called *signature detection*, is based on modelling hostile traffic. The second approach, called *anomaly detection*, is based on modelling normal traffic.

3.2.1 Signature detection

A *signature* detector, such as SNORT or Bro examines traffic for known attacks using rules written by security experts. Based on these rules the detector tries to find out malicious packets. When a new type of attack is discovered, new rules must be written and added to the *signature* detector.

Signatures are specific byte sequences which appear in the attack traffic, such as a specific string in the application payload, or suspicious behaviour, such as server requests to unused ports. The signatures are manually identified by human experts through careful analysis of the byte sequence from captured attack traffic. Then experts write the appropriate rules to detect those signatures. A good signature should be one that consistently shows up in attack traffic but rarely appears in normal traffic.

The signature-based systems have advantage over the anomaly-based systems due to their simplicity and the ability of operating online in real time. The problem is that they can only detect known attacks with identified signatures that are produced by experts. Automated signature generation for new attacks is extremely difficult due to three reasons:

- First, in order to create an attack signature, we must identify and isolate attack traffic from legitimate traffic.
- Second, the signature generation must be general enough to capture all attack traffic of certain type while at the same time specific enough to avoid overlapping with the content of normal traffic in order to reduce false alarms.

- Third, the intrusion detection system must be flexible enough to deal with the polymorphism in the attack traffic. Otherwise, worms may be programmed to deliberately modify themselves each time they replicate and thus trick the intrusion detection system.

3.2.2 Anomaly detection

Anomaly-based systems such as SPADE, ADAM, and NIDES profile the statistical features of normal traffic. Any deviation from the profile will be treated as suspicious. Although these systems can detect previously unknown attacks, they have the tendency to generate false alarm when the normal activities are diverse and unpredictable.

In anomaly detection, the system administrator defines the baseline, or normal, state of the network's traffic load, breakdown, protocol, and typical packet size. The anomaly detector monitors network segments to compare their state to the normal baseline and look for anomalies.

An Anomaly-Based Intrusion Detection System is a system for detecting computer intrusions by monitoring system activity and classifying it as either Normal or Anomalous. The classification is based on statistics or rules, rather than signatures, and will detect any type of anomaly that falls out with normal system operation. This is as opposed to signature based systems which can only detect attacks for which a signature has previously been created.

Mahoney and Chan identifies five types of deviations in hostile traffic

User Behaviour: Hostile traffic may have a novel source address because it comes from an unauthorized user of a restricted (password protected) service. Also, probes such as *ipsweep* and *portsweep* may attempt to access nonexistent hosts and services, generating anomalies in the destination addresses and port numbers.

Bug Exploits: Attacks often exploit errors in the target software, for example, buffer overflow vulnerability. Such errors are most likely to be found in the least-used features

of the program, because otherwise the error is likely to have been discovered during normal use and fixed in a later version. Thus, any remaining errors are invoked only with unusual inputs (e.g. a very long argument to a seldom used command), which are not likely to occur during normal use.

Response Anomalies: Sometimes a target will generate anomalous outgoing traffic in response to a successful attack, for example, a victimized mail server passing root shell command responses back to an attacker. This is analogous to Forrest's host based detection method, where a server compromise can be detected because it makes unusual sequences of system calls.

Bugs in the attack: Attackers typically must implement client protocols themselves, and will fail to duplicate the target environment either out of carelessness or because it is not necessary. For example, many text based protocols such as FTP, SMTP and HTTP allow either uppercase or lowercase commands. An attacker may use lowercase for convenience, even though normal clients might always use uppercase.

Evasion: Attackers may deliberately manipulate network protocols to hide an attack from an improperly coded intrusion detection system (IDS) monitoring the application layer. Such methods include IP fragmentation, overlapping TCP segments that do not match, deliberate use of bad checksums, short TTL values, and so on. Such events must be rare in normal traffic, or else the IDS would have been written to handle them properly.

There are mainly two categories of anomaly detection:

- The first category is the *Rule Based Anomaly Detection* which uses 'expert rules' to describe normal traffic. For example if a sub network has the right to access an FTP server. And if this sub network has as an IP address 123.2.4.* and the server has as an IP address 123.2.4.7. The rule in that case will be:

IF IP SRC==123.2.4.* ***AND*** IP DST==123.2.4.* ***AND*** DST PRT==21
THEN Normal Traffic.

It is known that the FTP protocol use the port 21 as a destination port.

- The second category is the *Statistical Anomaly Detection* which uses a stochastic model to describe normal traffic. The models are usually based on the distribution of source and destination addresses and ports per transaction (TCP connections, and sometimes UDP and ICMP packets). For example, SPADE offers four probability models (estimated by average frequency) of incoming TCP connections:

- $P(\text{destination-address, destination-port})$
- $P(\text{source-address, destination-address, destination-port})$
- $P(\text{source-address, source-port, destination-address, destination-port})$
- Bayes network approximation of the above.

Lower probabilities result in higher anomaly scores, since these are presumably more likely to be hostile.

To explain how we can use statistical model to decide whether traffic is normal or hostile we give the following example:

$$P(IP_SRC = x, TCP_DST = y) > \gamma \implies \text{NormalTraffic}$$

This example shows that we can define a threshold γ and then compare the probability of occurrence of an IP source address and an incoming TCP connexion to decide whether the traffic is normal or not.

Compared to the *rule based models*, the *stochastic models of normal traffic* are more robust to noise. This means that they are more efficient in detecting hostile packet in a non deterministic environment. They also provide a better support for online learning and a better adaptation to the changes that can affect normal traffic. But it is difficult for humans to interpret and to verify results obtained with *stochastic models*.

On the other side the Rule based models of normal traffic are easier to interpret and to verify by humans, but less robust to noise and provide less support for on-line learning and adaptation to changes that affect normal traffic.

3.2.3 Personal Firewall

The personal firewall is an example of rule based anomaly detector. In order to determine hostile traffic, the anomaly based intrusion detector must be taught to recognize normal network activity. In the personal firewall the human user must provide the firewall with rules that describe normal traffic. A main goal of personal firewalls is to allow *non-experts* to specify which traffic is accepted and which traffic must be blocked.

The personal firewall must determine the characteristics of the incoming traffic, such as the traffic type, the IP source address and the IP destination address. Then the firewall provides the user with these informations. Afterward the user must decide whether to accept or to ban this traffic. Based on the user decision new rules are added to the firewall.

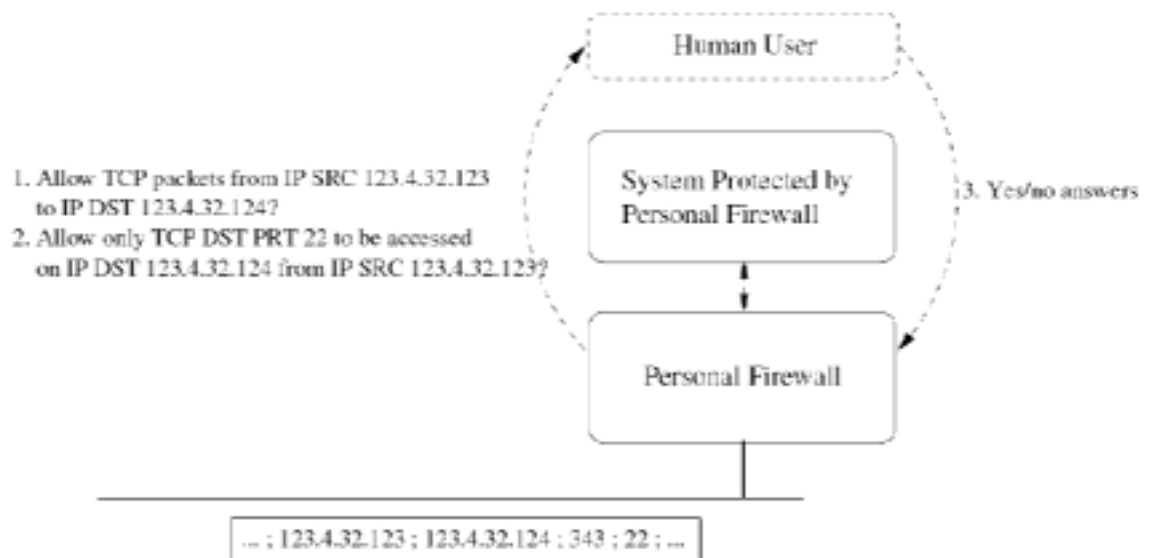


Figure 1 Rule specification in personal firewall by a human decision maker

4 Problem statement

The increasing threat of intrusion detection has made the security field an active domain of research in order to find solutions for this problem. One of the developed

solutions is the personal firewall. In this kind of firewall, the main goal is to permit non experts to specify which traffic is allowed and which traffic is not allowed. The anomaly detector provide the human user with a description of the current traffic by specifying the IP source address, the IP destination address, the source port, the destination port etc...Then this user must decide whether to accept or to block this traffic.

The decision maker must base his decision on a number of parameters that characterise normal traffic such as IP source address, IP destination address, source port, destination port, etc...The fact that a lot of parameters type are used to decide if a traffic must be allowed or not increase the possibility of mistake. The risk of mistake increases with the size of the network, because the amount of parameters increases with the size of network. We can think that the solution to this problem is to increase the abstraction level, in other words drop some type of parameters used to describe normal traffic. This can make the decision process easier. But in that case the specification is not accurate enough to separate hostile traffic from normal traffic. Then there is a risk to allow hostile traffic or disallow normal traffic.

In addition when a human user makes a decision, his decision can be influenced by his physical state (tired, not concentrated...) and also by his feelings (nervous, angry...). So we cannot insure that he will make the same decision even if the parameters take the same values because a human being is always influenced by external factors.

All these presented facts shows that the personal firewall can become a threat it self to the network security if the rule specification is given to a human user. That is why it would be better to make the decision making process an automatic task.

5 Requirements

In the existing personal firewall a human user makes the decision to accept or to ban incoming traffic to the network. The fact that the decision maker is a human being can increase the threat of intrusion. To overcome this problem we replace the human decision maker by a collection of Learning Automata. Many Learning automaton will cooperate together in order to find a Boolean expression over packet bits that can be used to decide whether the traffic is normal or hostile.

We plan to address the following challenges: How can we use the packet bits to specify rules without online guidance from human user? And how can we handle noisy traffic when identifying packets or specifying rules? Learning Automata will form the basis to overcome the above challenges.

In order to apply Learning Automata for intrusion detection in rule based anomaly detection we will use the following architecture.

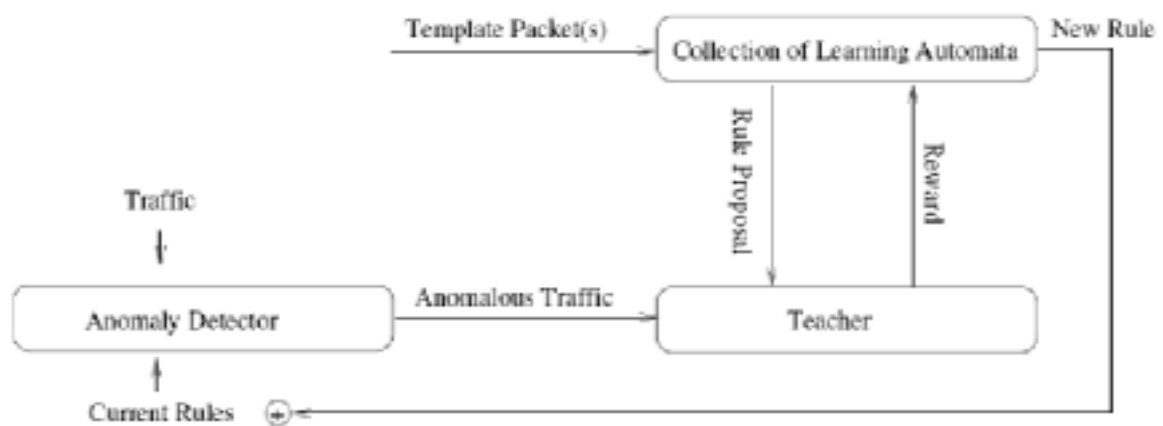


Figure 2 Learning Automata for anomaly detection

The collection of Learning Automata receives a template packet and tries to find the relevant bits and the irrelevant ones. The relevant bits represent a set of parameters that distinguish one traffic type from another. In order to know the relevant bits in the packet template Learning Automata treat many packets that belong to the same traffic type as the template packet but also to other types. After every iteration, the collection of Learning Automata gives a rule proposal. Afterward every Learning Automaton receives a reward or a penalty based on their decision. This process continues until all the automata converge. When this state is reached the rule is found and can be added to the current rules of the anomaly detector. Based on these rules the anomaly detector decides whether to accept or to ban incoming traffic.

In order to evaluate the performance of Learning Automata and their ability to handle noisy traffic when specifying rules we will make a simulation of network traffic and let

Learning Automata find the rules of this traffic. The verification of result will be an easy task since we generate the traffic our self, this mean that we know the rules that correspond to this traffic.

6 Approach

In this project we are going to evaluate a recent learning automata scheme for intrusion detection in computer network. Many automata will cooperate together in order to form a Boolean expression over packet bits that can be used to decide whether the packet is normal or not.

Since the Learning Automata constitute the foundation of our approach we start by giving an idea about Learning Automata and then illustrate the principle of Learning Automata by giving an example of a game that use Learning Automata. Finally we explain how to use Learning Automata for anomaly detection.

6.1 Learning Automata Principal

Learning Automata (LA) are adaptive decision making devices that can operate in unknown and non-deterministic environments. An environment is qualified as unknown for the LA because LA has no information about the effect of their actions at the beginning of the whole process. This environment is also non-deterministic i.e. that a given action doesn't necessarily produce the same response each time it is performed. A powerful property of LA is that they progressively improve their performance by the meaning of a learning process. They combine rapid and accurate convergence with low computational complexity.

6.1.1 The T-maze problem

Many inventions in robotics, automatic, artificial intelligence, etc... have their origins in animal intelligence. Scientists are interested in studying the behaviour of animals by submitting them to different kind of experiments. Then they try to model the animal's behaviour, in order to find solution for some common problems in different fields. The LA is one of the outcomes of this approach. The LA has its origin in the T-maze problem.

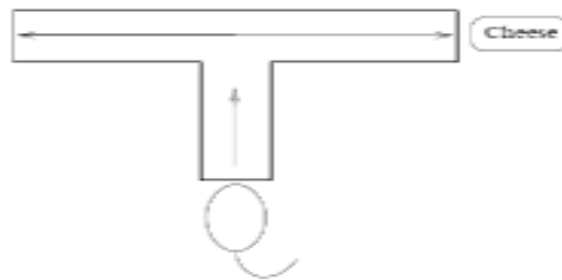


Figure 3 T-maze problem

In the T-maze problem a hungry rat is placed at the lower end of the middle limb of a T-maze. The rat can move along the limb and turn to the right or to the left. The cheese is putted at the end of the right arm with probability 0.7 and at the end of the left arm with probability 0.3. In other words if the experiment is repeated 10 times, the cheese will be putted 7 times on the right side and 3 times on the left side. This experiment is carried out in order to study the behaviour of the rat over successive trials. Then try to model the behaviour of the rat and exploit this mathematical model to find solution for problems in many fields.

The T-maze problem constitutes the origin of the Learning Automata. In this problem we have two main actors the rat and the environment. In the following we give a general modelling of this problem. We replace the rat by an automaton, and try to see the interaction between the environment and the automaton. We also focus on how LA can learn from their previous trials.

6.1.2 Environment Modelling

The model in figure 4 is a general and it represents a large class of unknown media in which an automaton or a group of automata can operate.

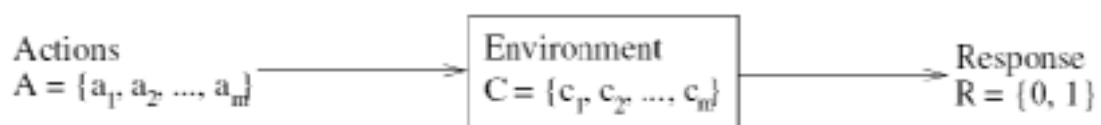


Figure 4 Environment Modelling

This model shows the environment as a box characterised by a set of penalty probabilities $\{c_1, c_2, \dots, c_m\}$. The actions $\{a_1, a_2, \dots, a_m\}$ constitute the input for the environment. Each action has a penalty probability $P(\text{Penalty}(i) | \text{Action}(i) = a_j) = c_j$. The output of the environment is binary; it can be a reward or a penalty. The mapping between the actions and the response is based on the penalty probability.

The environment can be a stationary (static) i.e. the penalty probabilities do not change. Or it can be non stationary (dynamic) penalty probabilities can change. In our case the environment is non stationary.

6.1.3 Automaton Modelling

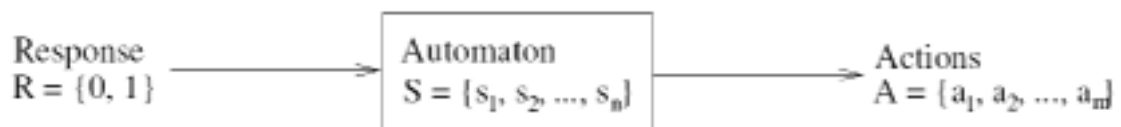


Figure 5 Automaton modelling

The automaton takes a response from the environment as input and maps it to an action which is given as output. The internal automaton state defines the mapping. The state is typically updated based on each (action, response) pair.

There are 3 classes of Learning Automaton:

- Two-Action Tsetlin Automaton (Fixed Structure Deterministic Automaton)
- Two-Action Discretized Variable Structure Stochastic Automaton
- Two-Action Variable Structure Stochastic Automaton \rightarrow Action Probability Vector

In our project we use the “Two-Action Variable Structure Stochastic Automaton”.

6.2 Principle of the “Two-Action Variable Structure Stochastic Automaton”

For this class of Automaton it has only two actions as an output, action 1 and action 2. The probability to choose action 1 is equal to p_1 and the probability to choose action 2

is equal to p_2 . Actions are chosen by random draw according to their probabilities. In the beginning of the learning process action 1 and action 2 have the same probabilities since the Automaton doesn't know the right action that they must choose (unknown environment), then $p_1 = p_2 = 0.5$. Afterwards the action probabilities are updated based on the chosen action and the environment response.

If action 1 is chosen and the environment gives a reward to the automaton then the automaton update the probabilities p_1 and p_2 as follow:

$$*p_1(n+1) = p_1(n) + a \times p_2(n)$$

$$*p_2(n+1) = p_2(n) - a \times p_2(n)$$

$$*0 < a < 1$$

$$*p_1(n) + p_2(n) = 1, \forall n$$

n is an integer that denotes the number of trials carried out by the automaton. $p_1(n)$ is the probability of action 1 in the n^{th} iteration and $p_2(n)$ is the probability of action 2 in the n^{th} iteration.

In this case the automaton has chosen action 1 and the environment gives a reward to the automaton, in other words action 1 is probably the good choice that's why the automaton increases p_1 . When the automaton chooses his action, this choice is random and depends on the action probability. Then if p_1 is bigger than p_2 , the automaton will probably choose action 1 is. So we can say that the automaton start to converge towards the right solution. In the next trial the automaton will probably choose action 1 and will get a new reward if the environment is stationary. This process continues until p_1 become very close to 1, if this state is reached we can say that the automaton has converged to the solution of the problem. The parameter a has in influence on the speed of convergence and also the accuracy of the final solution. If a is big the automaton converge very fast but we are not sure if the final solution is right or not. But if a is small then the automaton needs more time to converge, but we are more sure about the accuracy of the solution. The choice of the value of a can change from a problem to another because it depends mainly on the environment and his stochastic characteristics.

If action 1 is chosen and the environment gives a penalty to the automaton then the automaton update the probabilities p_1 and p_2 as follow:

$$*p_1(n+1) = p_1(n) - b \times p_1(n)$$

$$*p_2(n+1) = p_2(n) + b \times p_1(n)$$

$$*0 < b < 1$$

$$*p_1(n) + p_2(n) = 1, \forall n$$

In order to illustrate the “Two-Action Variable Structure Stochastic Automaton” and how they work, we have used this class of automaton to find the optimal solution of a cooperative game. In this game we have two players; every one can choose 1 or 2. If the two players choose 1 both of them will get a reward otherwise they didn't get any reward.

We replace the two players by two Learning Automata. At this step the environment is considered unknown and deterministic for the Learning Automata. The LA doesn't have any knowledge about the rules of the game but by receiving feedback of their decision from the environment they can learn and find the right solution. In order to test the ability of Learning Automata to operate in non deterministic environment, we make some changes to this game. The Learning Automata will get a reward with a probability equal to 0.5 if both of them choose 1 and they also get a reward for other choices with a probability equal to 0.4. Although the environment is unknown and non deterministic for the two Automata, they were able to find the solution of this game. We have implemented this game in order to evaluate the performance of the Learning Automata. Using this program we were able to see the influence of the value of a (parameter used to update action probabilities). We have first chosen $a=0.01$ in this case the Learning Automata need 6857 iteration to converge to the right solution. Then we choose $a=0.1$ in this case Learning Automata need 361 iteration to converge but the found solution was not right. We can conclude that if a is small the speed of convergence is slow; we need more iteration and more time to converge but we are sure about the accuracy of the solution. However if we choose a big we converge faster but the solution found by Learning Automata can be wrong. The obtained results are illustrated in the two tables 1 and 2.

Iteration	LA1-action	LA2-action	Environment Response
6853	1	1	No reward
6854	1	1	No reward
6855	1	1	No reward
6856	1	1	Reward
6857	1	1	Reward

Table 1 Convergence to the right solution in 6857 iterations ($\alpha = 0.01$)

Iteration	LA1-action	LA2-action	Environment Response
357	1	1	No reward
358	2	1	No reward
359	2	2	Reward
360	2	1	No Reward
361	2	2	Reward

Table 2 Convergence to a wrong solution in 361 iterations ($\alpha = 0.1$)

6.3 Learning Automata for anomaly detection

In this project we are going to evaluate a recent novel learning automata scheme for intrusion detection in computer network. Many automaton will cooperate together in order to form a Boolean expression over packet bits that can be used to decide whether the received packet is normal or not.

Our idea of applying Learning Automata in rule-based anomaly detection has its basis in the personal *firewall* rule configuration. In essence, we replace the human decision maker by a collection of Learning Automata. The goal of the Learning Automata is to learn, in some sense, the optimal decisions when specifying the rules that describe normal traffic. Those rules are used afterward by the anomaly detector to decide whether to accept or to ban incoming traffic.

Normal traffic patterns are useful to detect attacks. LA construct rules that represent normal traffic and then the anomaly detector use current data to detect a possible mismatch with rules and recognize possible attack attempts.

In order to learn the pattern of normal traffic LA needs training data. The training data consist of many packets that belong to different kind of normal traffic exchanged between the network nodes in a period of time, this data is attack free. The LA must go through the training data and find the rules that describe this set of packets. We assume that the data set contain n traffic types. First LA has to choose randomly one packet from the training data; we assume that this packet belongs to the traffic m and we call it filter packet. In this step the collection of LA has to find the rule of traffic m. We use a Learning Automaton for every bit. We select then another packet from the data set. By comparing the bits in the target packet with those in the selected packet LA can learn if the bit is changing or if it is static. By comparing the filter packet with all packets LA can find the relevant bits and the irrelevant ones. If we take a set of packet that belongs to the same traffic type, we can remark that some fields that are the same in all packets and the remaining fields are changing from one packet to another. The static bits characterise the traffic, hence they are relevant. The relevant bits can correspond for instance to the source, and the destination IP address. The bits that changes from packet to packet are irrelevant, these bits can correspond for instance to the sequence number. LA must decide for every bit if it is relevant or not.

1	0	*	1	*	1
----------	----------	----------	----------	--------------	----------	----------

Figure 6 Rule pattern

The figure 6 gives an example of rule. In this figure the irrelevant bits are shown as a star and the relevant bits are 1 or 0. Since we work with rule based anomaly detector, the IDS must verify if the received packet belongs to any type of normal traffic. To verify if the packet belongs to the traffic m, the anomaly detector has to compare the relevant bits in the rule that specify the traffic m, with the same bits in the received packet. If the relevant bits in the rule and in the packet match together, then the packet belongs to the

traffic m. if they don't match the IDS must do the same comparison with the other rules. And if the relevant bits don't match when they are compared with all rules then the packet is anomalous.

7 Implementation

In this project instead of using real traffic we are going to simulate this traffic and try to evaluate the performance of learning automata. In fact it is possible to get an attack free training data set from the DARPA web site. But the problem in that case is that we don't know exactly normal traffic in this network and it will be impossible then to verify the results we found. In addition we don't have any idea about the number of rules that we must find, also the size of the training data is very big since it belongs to a huge network and the period of information gathering is a whole week. Then the time needed to treat this data is very long and this will constitute a problem.

In other hand the simulation leads to the same results, since we can test our solution in almost the same conditions. It is true that the amount of data and the number of traffic is lower in the case of the simulation. But this doesn't deny that we can test the ability of LA to form rules in the presence of noisy information. In addition since we generate the traffic our self it is then possible then to verify the obtained results. Moreover the fact that the amount of treated data is not too big makes the processing time reasonable. In our simulation we generate 3 traffic types; this will make the program easier since we have to specify for every kind of traffic the value of all bits. And at the same time the result will be the same with 3 traffic types or more because we can test even with 3 traffic types the ability of LA to hold noisy traffic. This means that when LA is trying to find the rule of traffic 1 and then treat a packet that belongs to traffic 2 this event will not influence the finale results. Even in presence of other kind of traffic LA must be able to specify the right rule.

In our programme LA has to find the rules of 3 traffic types. The first traffic is HTTP (Hyper Text Transfer Protocol), for this traffic the source port is 80 and the destination port is 80. The second traffic is FTP (File Transfer Protocol), for this traffic the source port is 1027 and the destination port is 21. The third traffic is SMTP (Simple Mail Transfer Protocol), for this traffic the source port is 578 and the destination port is 25.

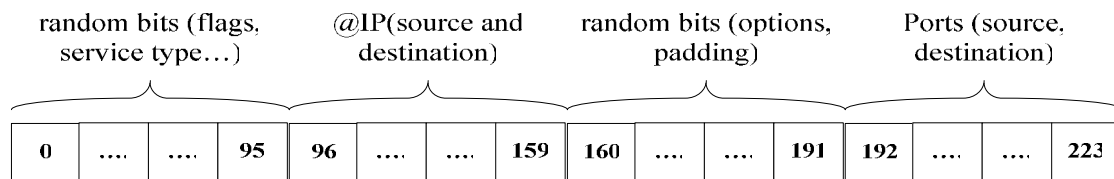


Figure 7 Packet structure

The figure 7 represents the packet structure. The first 96 bits are random and they correspond to the first bits in the IP header. These bits represent the service type, flags, the Time To Live,... they can change from one packet to another. The source and the destination IP address constitute a static block that stretch from the bit 96 to the bit 159. The next 32 bits are random and correspond to the options and the padding. The last bloc is static and it represents the source and the destination port. We choose to make the parameters source IP address, destination IP address, source port and destination port static fields because they can be considered as the characteristic fields for any traffic. It is true that other parameters can be static for a traffic type but in our simulation we just suppose 4 parameters as static.

In our implementation we use oriented object programming. We create a class called Learning Automata and a class called generate-packet.

The LA class: This class contain three attributes and four methods. The first attribute “a” is used to update action probabilities. The second attributes p denotes the probability of action 1; since we have two actions then the probability of action 2 is equal to (1-p). For the LA action 1 corresponds to the decision “the bit is irrelevant” and action 2 corresponds to the decision “the bit is relevant”.

```
class LA
{
    public:
    float a;
    float p;
    int action;
```

```
void initialisation(float);
void UpdateActionProbabilities(float);
bool converged();
int MakeDecision();
};
```

The method initialisation: this method initialize the attributes a, action and p. p is initialised to 0.5 since in the beginning LA doesn't have any knowledge then action 1 and action 2 must have the same probabilities.

```
void LA::initialisation(float a)
{
    this->a = a;
    this->action = 0;
    this->p = 0.5;
}
```

The method Update Action Probabilities: this method permits to update the action probabilities. The probabilities of action 1 and 2 will be updated based on the decision of LA and the value of reward.

```
void LA::UpdateActionProbabilities(float reward)
{
    if(action==1)
        p = p + a * reward * (1-p);
    else
        p = p - a * reward * p;
}
```

The method converged: this method verifies if LA has converged or not. LA converges if $(p < 0.001)$ or $(p > 0.999)$. If $(p < 0.001)$ then $(1-p)$ is very close to 1 and LA converge in

this case to the decision “the bit is relevant”. If ($p > 0.999$) then p is very close to 1 and LA converge in this case to the decision “the bit is irrelevant”.

```
bool LA::converged()
{
    if ((p < 0.001) || (p > 0.999))
        return (true);
    else
        return (false);
}
```

The method Make Decision: this method is used by LA to decide whether the bit is irrelevant (action 1) or relevant (action 2). We use a Learning Automaton for every bit. The decision is made randomly according to action probabilities.

```
int LA::MakeDecision()
{
    if (getrandom <= p)
        action = 1;
    else
        action = 2;
    return action;
}
```

The class generate-packet: this class generates the packet needed for the simulation. It has two attributes and one method. The first attribute denotes the total number of bits in the generated packet which is equal to 224. The second attribute denotes the number of random bits which is equal to 128. The method “*generatepacket*” generates the packets one by one.

```
class generate_packet
{
    public:
```

```
int numberofbits;
int numberofrandombits;
public:
int *generatepacket(int,int **);
};
```

We have 3 traffic types, for every type the static fields doesn't change but the random ones change even for two packets that belong to the same traffic type. First the method chooses randomly the traffic type to which the generated packet belongs. If this step is performed the static fields are already defined. The remaining bits are random so we have just to fill in the corresponding elements of the array. The structure of the packet is given in the figure 7. The current packet can belong to one of the three traffic types.

```
int *generate_packet::generatepacket(int numberofrandombits, int **rawpacket)
{
    int packetnumber;

    packetnumber = int(getrandom * 3.0);

    for(int i = 0 ;i < numberofrandombits-32; i++)
    {
        rawpacket[packetnumber][i] = int(getrandom * 2);
    }

    for(i = numberofrandombits+32 ;i < numberofrandombits+64; i++)
    {
        rawpacket[packetnumber][i] = int(getrandom * 2);
    }

    return(rawpacket[packetnumber]);
}
```

Using the method “generatepacket”, we can generate packets that belong to the three traffic types. Since we have to find the rules of the three traffic types, we do this step by step and find the rule for HTTP traffic then for FTP traffic and finally for SMTP traffic. In order to find the rule of traffic 1 the filter packet must belong to traffic 1. We call the traffic for which we want to specify the rule, target traffic.

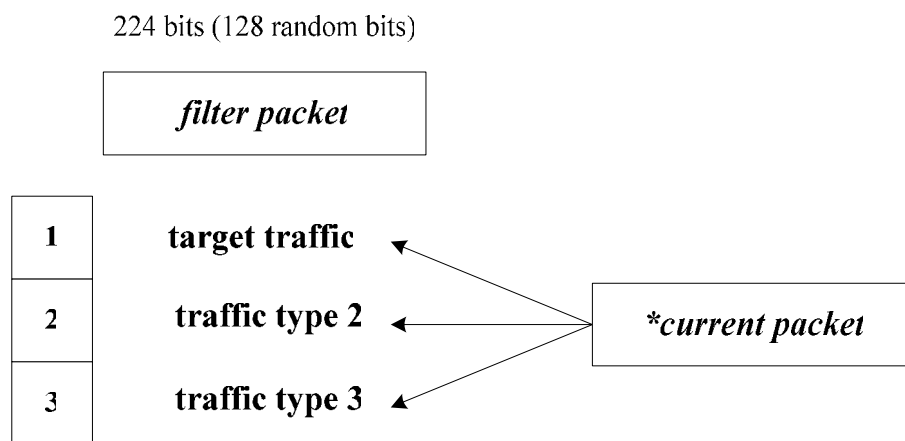


Figure 8 Traffic generation

After choosing the filter packet we create an array of LA using the LA class. The number of LA is equal to the number of packet bits.

The following processing is repeated to find the rule of traffic 1, 2 and 3.

We have to iterate until all LA converge.

Set converged to false

While not converged

1-Generate a packet randomly using the method “generate packet”

2-In order to verify whether the packet pass the filter or not

Set Pass equal to true

Go over different bits and let the 224 LA make decision. And update the value of pass. LA decides if the bit is relevant or not.

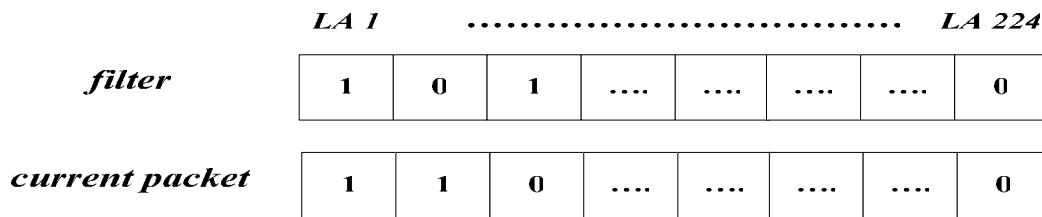


Figure 9 LA and decision making

If one of the LA decides that a bit is relevant and if the value of the corresponding bit in the filter packet doesn't match with its value in the filter packet then the current packet will not pass the filter. The variable Pass will be set to false. In the figure 9 for example if LA 2 decides that the second bit is relevant the packet will not pass.

3- Update action probabilities of the LA array

*If (action=2) i.e. the automaton decides that the field is relevant and if the packet pass the filter → the automaton is given a full reward.

We update the probability of action 2 for this automaton as follow

$$p_2(n + 1) = p_2(n) + a \times reward \times p_1(n) \quad \text{with (reward=1)}$$

*if (action=1) the automaton decides that the field is irrelevant → the automaton is given a partial reward

We update the probability of action 1 as follow

$$p_1(n + 1) = p_1(n) + a \times reward \times p_2(n) \quad \text{with (reward=0.05)}$$

If the automaton decides that the bit is relevant then there are only two chances from four possibilities that the bit in the current packet and the filter packet match. If only one automaton decides that the bit is relevant and then there is a mismatch, the packet will not pass. In the beginning, for every automaton there is a probability equal to 0.5 to decide that the bit is relevant, and the total number of automata is 224. Then there is a very small probability that the packet pass the filter. And if the packet passes the filter and the automaton decided that the field is relevant there is a very big probability that this decision was right, that's why we give the automaton a full reward in this case. But if the automaton decides that the bit is irrelevant this decision will not affect the variable Pass

then this decision have only a small probability to be a right decision that's why the automaton is given a partial reward in this case.

- 4 Test if all automata have converged or not. If yes we stop iteration. The automaton converge if the probability of action 1 or 2 is >0.999 or <0.001 . The value of reward in case that automaton decides "the bit is relevant" will influence the speed of convergence and the accuracy of the decision. If this value is closer to 1 the convergence is very rapid but the results are totally wrong. But if this value is closer to 0 the convergence is very slow but the result is very accurate. In our case we have three traffic types and we want to find the rules of one of the traffic in every loop until getting the rule of the three types. When trying to find the rule of traffic 1 LA treat also packets from traffic 2 and 3. Then the probability to generate a packet that belongs for instance to traffic 1 is equal to $1/3$ since the choice of traffic type when we generate a packet is done randomly. We remark that if the value of reward is smaller than $1/3$ we are sure to find the right rules and then we have found the optimal value that permit to get the best speed convergence and to insure also accuracy in the found results. There is a relation between the amount of packets that belong to one traffic type and the value of reward. This relation can be generalised. If we have n traffic type the optimal value of reward that permit to find the rule of one traffic type among the n ones is x , with $(x < 1/n)$.

In this program we know the position and the number of relevant and irrelevant bits, since we generate the packets our self. Then it was simple for us to verify the results. We have printed the probability p for all automata and we have verified that results are right. For the relevant bits the value of p is close to 0, and for the irrelevant bits the value of p is close to 1. We have found the rules for the three types of traffic by doing the same process. Then we have proved the ability of LA to find rules in unknown environment without human guidance. We have proved also that Learning Automata are able to handle noisy traffic.

8 Evaluation

Since we generate the traffic our self it is easy to verify the accuracy of the obtained results. After verifying the results we find that LA is able to identify the relevant and the irrelevant bits and so to construct the right rules. In the first test we generate three different types of traffic and try to find the rule of one of them. This test was successfully concluded. When LA are trying to find the rule of traffic 1 they receive packets that belong to traffic 2 and 3, but this doesn't influence the accuracy of the founded results.

In another test we generate 3 types of traffic, and we succeed to find the rule of every traffic type.

Through the carried test we have seen the influence of the variable reward on the speed of convergence and the accuracy of results. If the reward, given to the automaton when it decides that the bit is irrelevant, is close to 0 the speed of convergence is slow but the results are accurate. But if we assign to the reward a value close to 1 the speed of convergence is fast but the obtained rule is not accurate. We find that the optimal solution is to choose the value of reward inferior to $1/n$ with n is the number of generated traffic types.

The fact that the testing is done in the presence of three different traffic type when specifying the rule of one of them prove the ability of LA to handle noisy traffic. In addition LA find the rules using packet bits and without any online guidance from human and this prove the ability of LA to operate in unknown environment.

9 Discussion

9.1 Project outcomes

Our purpose in this project is to replace the human decision maker in the personal firewall by a collection of Learning Automata in order to improve the accuracy of decisions. When choosing this solution we have some challenges to overcome. How can the packet bits used to specify rules without online guidance from human and how to handle noisy traffic when specifying rules? LA constitutes a solution for these challenges. We prove through many tests the ability of LA to operate in unknown and non deterministic environment.

In the testing we generate the traffic our self by the means of a simulation. The tests can be improved by using real traffic instead of simulated one. But we need in that case

to know previously the rules of normal traffic such that we can verify the obtained results afterwards. We can also improve the testing by using data that contains attacks in order to verify the ability of LA to detect hostile traffic.

9.2 Evaluation of the overall results

The found results show that Learning Automata are able to operate in unknown and non deterministic environment. In the beginning LA doesn't have any previous knowledge about the normal traffic. But they succeed to find the rules that models normal traffic. When LA is specifying the rule of traffic 1 for instance the can receive packets that belong to traffic 2, these packets constitute a noise for LA. We proved that LA can handle noisy information and succeed to find the appropriate rule for every kind of traffic.

The treated data in our case is attack free, but this fact didn't influence the relevance of results. In fact in order to specify the rule of one traffic type LA need a very big amount of packets that belongs to the specified traffic type. The number of packets that belongs to one attack can't reach this amount then LA will not consider them as normal traffic.

The testing we have done is based on a simulation of network traffic. This doesn't affect the relevance of results, since we have tried to conserve the structure of a real packet. Moreover if we perform the tests with real traffic, then we can't verify the results since we don't know the normal traffic of the network. But when we generate the traffic our self we overcome this problem.

An important feature of the LA approach is that it combines the benefits of stochastic traffic models (i.e. the ability to handle noisy traffic and to operate without online guidance) with the benefits of rule based models (facilitate human interpretation), since LA has as an output rules that can be easily understood and interpreted by human. In the stochastic traffic models the results are just scores that are used to decide whether to accept or not incoming traffic. These score can't be easily understood and interpreted by human.

10 Conclusion

The personal firewall allow non expert to specify which traffic is allowed and which traffic is banned. Since the decision maker is a human then the risk of mistake increase with the number of traffic exchanged in the network. In this project we propose to replace the human decision maker by a collection of Learning Automata. In order to evaluate the performance of a new Learning Automata scheme for anomaly detection we generate three traffic type and we use LA to the corresponding rules for the generated traffic types. We prove that LA is able to form Boolean expression (the rules) over packet bits that can be used to decide whether a received packet is normal or hostile. We prove that LA is able to operate in unknown and non deterministic environment without human guidance. We prove also that LA are able to handle noisy traffic since even in presence of three kind of traffic LA can find the rule of the three traffic types.

Appendix

List of references

- [1] Rule Based Network Anomaly Detection using Learning Automata:
Course given by Ole-Christoffer Granmo, Arne Wiklund, and Edward Fjellskål in
Agder University College
- [2] Security Seminar—Finite Action Learning Automata (FALA):
presented by Ole-Christoffer Granmo
- [3] Network Traffic Anomaly Detection Based on Packet Bytes:
Matthew V. Mahoney/ Florida Institute of Technology, Melbourne, Florida
- [4] Computer System Intrusion Detection: A Survey by Anita K. Jones and Robert S.
Sielken Department of Computer Science/ University of Virginia
- [5] Intrusion Detection Systems (IDS) - Classification; methods; techniques:
http://www.windowsecurity.com/articles_tutorials/intrusion_detection/
- [6] Intrusion detection:
<http://en.wikipedia.org/wiki/>
- [7] IDS: Signature versus anomaly detection:
<http://searchsecurity.techtarget.com>
- [8] "Cybercrime" April 21, 2000:
www.fbi.gov

Glossary an abbreviations

LA	Learning Automata
DoS	Denial of Service
ICMP	Internet Control Message Protocol
FTP	File Transfer Protocol
HTTP	Hyper Text Transfer Protocol
SMTP	Simple Mail Transfer Protocol
DARPA	Defence Advanced Research Project Agency
TCP	Transmission Control Protocol
LAN	Local Area Network
IP	Internet Protocol
TTL	Time To Live
UDP	User Datagram Protocol