

**AUTOMATED INTRUSION PREVENTION MECHANISM IN
ENHANCING NETWORK SECURITY**

HE XIAO DONG

FACULTY OF COMPUTER SCIENCE

UNIVERSITY OF MALAY

KUALA LUMPUR

MARCH 2008

**AUTOMATED INTRUSION PREVENTION MECHANISM IN
ENHANCING NETWORK SECURITY**

HE XIAO DONG

**DISSERTATION SUBMITTED IN FULFILLMENT
OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE**

FACULTY OF COMPUTER SCIENCE

UNIVERSITY OF MALAY

KUALA LUMPUR

MARCH 2008

ABSTRACT

Firewall, intrusion detection systems (IDS), and intrusion prevention system (IPS) are important tools used to secure networks against hackers' attacks. Ironically, these malicious attacks have brought more adverse impacts on the networks than before. At present, many existing IDS AND IPS work independently without the exchange of information. Hence, this deficit will lower the capability of these tools to protect increasingly vulnerable networks.

In this thesis, an automated intrusion prevention mechanism (AIPM) which comprises the functionalities of IDS, IPS, and network devices is proposed to enhance network security. AIPM is a mechanism that includes automated intrusion prevention function and automated analysis of intrusion messages function. Additionally, the ability of automatically detecting and analyzing network traffic allows AIPM to detect malicious attacks almost in real time. Likewise, the ability of automatically analyzing intrusion messages and network configuration enables AIPM to build a topological view and locate the source of a malicious attack. Results of case studies show that AIPM imposes lower overhead than conventional method, which queries all pre-defined routers to block every interface irrespective of where the attack is launched. On the contrary, AIPM identifies the interface that is nearest to the source of the attack and sends a single query to the associated router to block only that particular interface, only 1 connection per attack is needed. AIPM can block malicious traffic in 2-5 seconds after an attack start because less pre-defined information is needed, the conventional method, on the other hand, needs about 5-10 seconds to finish block processing as more pre-defined information is needed. In summary, AIPM which incorporates the functionalities of IDS AND IPS offers network protection against potential malicious acts without incurring additional overheads as compare to the conventional method.

ACKNOWLEDGEMENT

I wish to express my deepest gratitude to my supervisor, Associate Professor Dr. Ling Teck Chaw, for his enthusiasm, inspirations and guidance throughout my studies and research. This entire research project would not be possible without his continuous support, guidance and advice, and I am grateful for his confidence in me. He has critically challenged my old mode of thinking and provided structure for the development of new ones. He has been crucial to my intellectual development and I have good fortune to work under his supervision.

Second, I am grateful to everyone who has been working hard in the Network Research Lab. All of them have helped me in one way or another during the course of the research. Especially Mr. Lim Wei Chiang, I got many new ideas and suggestions from him when we discussed together.

Finally, I would like to thank my parents; I cannot finish the thesis without their support and love. Thanks.

Table of Content

ABSTRACT	I
ACKNOWLEDGEMENT	II
TABLE OF CONTENT	III
LIST OF FIGURES.....	VII
LIST OF TABLES.....	VIII
ABBREVIATIONS.....	XI
CHAPTER 1 INTRODUCTION.....	1
1.1 MOTIVATIONS.....	2
1.2 OBJECTIVES OF THE RESEARCH	3
1.3 SCOPE	3
1.4 METHODOLOGY	4
1.5 THESIS ORGANIZATIONS.....	5
CHAPTER 2 LITERATURE REVIEW	7
2.1 INTRUSION DETECTION SYSTEM	7
2.2 INTRUSION DETECTION TECHNIQUES.....	9
2.2.1 Anomaly Detection.....	10
2.2.1.1 Genetic Algorithms in Anomaly Detection	12
2.2.1.2 Fuzzy Logic in Anomaly Detection	14
2.2.1.3 Predictive Pattern Generation in Anomaly Detection	18
2.2.1.4 Comparison of Anomaly Detection Approaches.....	19
2.2.2 Misuse Detection	20
2.2.2.1 Pattern/String Matching in Misuse Detection	20

2.2.2.2 State Transition in Misuse Detection	22
2.2.2.3 Model-Based Intrusion Detection in Misuse Detection	24
2.2.2.4 Comparison of Misuse Detection Approaches	25
2.3 ISSUES AND CHALLENGES IN CURRENT INTRUSION DETECTION	26
2.4 INTRUDER PREVENTION SYSTEM	27
2.5 ISSUES AND CHALLENGES IN CURRENT IPS	29
2.6 EXISTING OPEN SOURCE POPULAR IDS AND IPS	29
2.6.1 Snort	29
2.6.2 SnortSam	31
2.6.3 Snort-Inline	33
2.7 COMPARISON OF EXISTING IDS AND IPS	35
2.8 PROBLEMS	36
2.9 SUMMARY	37
CHAPTER 3 ANALYSIS AND DESIGN	38
3.1 ANALYSIS OF AIPM	38
3.2 DESIGN OF AIPM	40
3.2.1 Key Components of AIPM	43
3.3 AIPM	50
3.3.1 Intrusion IP Is Private IP	54
3.3.2 Intrusion IP Is Inside Public IP	54
3.3.3 Intrusion IP Is Outside Public IP	55
3.4 SUMMARY	55
CHAPTER 4 SYSTEM DEVELOPMENT AND TESTING	56
4.1 DEVELOPMENT ENVIRONMENT	56
4.2 UNIT TEST	57
4.2.1 Unit Test 1 - Initial Information Processing	57

4.2.1.1 CISCONATAACLInit, ciscoconfparsefile and ciscoconfparseline	57
4.2.1.2 CISCONATAACLParse	60
4.2.2 Unit Test 2 - Running-Config File Processing	61
4.2.2.1 CISCONATAACLRunningDownload.....	62
4.2.2.2 CISCONATAACLRunningParse and CISCONATAACLRunningParseline	63
4.2.3 Unit Test 3 - Routing-Table File Processing.....	64
4.2.3.1 CISCONATAACLRouteDownload.....	64
4.2.3.2 CISCONATAACLRouteparse and CISCONATAACLRouteparseline	65
4.2.4 Unit Test 4 - NAT/PAT Pool File Processing	66
4.2.4.1 CISCONATAACLPoolDownload	67
4.2.4.2 CISCONATAACLPoolParse and CISCONATAACLPoolParseLine	68
4.2.5 Unit Test 5 - “Add ACL entry processing”	69
4.2.5.1 CISCONATAACLCreateACL.....	69
4.2.5.2 CISCONATAACLRunningModify	71
4.2.6 Unit Test 6 - Connection Processing	72
4.2.7 Unit Test 7 - ACL Checking.....	73
4.3 INTEGRATION TESTING	74
4.3.1 Integration Testing Case 1 - IP of Intruder Is Private IP	76
4.3.1.1 Situation One - No ACL Exists In Target Interface	76
4.3.1.2 Situation Two - An ACL Exists In Target Interface	78
4.3.2 Integration Testing Case 2--IP of Intruder Is Outside Public IP	80
4.3.2.1 Situation One-- No ACL Exists In Target Interface.....	81
4.3.2.2 Situation Two-- ACL Exist In Target Interface.....	82
4.3.3 Integration Testing Case Three--IP of Intruder Is Inside Public IP.....	83
4.3.3.1 Situation One - No ACL Exist In Target Interface.....	84
4.3.3.2 Situation Two - ACL Exist In Target Interface.....	85

4.4 RESPONDING TIME OF INTEGRATION TEST	86
4.5 CHAPTER SUMMARY	87
CHAPTER 5 CASE STUDY	88
5.1 CASE STUDY 1	88
5.1.1 Case Study 1.1—Attack From Inside	89
5.1.2 Case Study 1.2—Attack From Outside	90
5.2 CASE STUDY 2	92
5.2.1 Case Study 2.1—Attack From Inside To Inside	93
5.2.2 Case Study 2.2—Attack From Inside To Outside	94
5.2.3 Case Study 2.3—Attack from Outside to Inside	95
5.3 COMPARISON BETWEEN EXISTING IDS AND IPS AND AIPM	97
5.3.1 Advantages of AIPM	97
5.3.2 Features Comparison	98
5.4 SUMMARY	100
CHAPTER 6 CONCLUSION AND FUTURE WORK	101
6.1 THESIS SUMMARY	101
6.2 PROBLEMS FACED IN DEVELOPMENT	102
6.3 THESIS CONTRIBUTION	105
6.4 SUGGESTION FOR FUTURE WORK	106

LIST OF FIGURES

Figure 2-1 System Design for NIDS	12
Figure 2-2 System Architecture In the model of Aly et al.	16
Figure 2-3 Components of Snort	30
Figure 2-4 Process of Snort-Inline	34
Figure 3-1 Use Case Diagram of AIPM	39
Figure 3-2 Component Diagram of AIPM	40
Figure 3-3 Architecture of AIPM	41
Figure 3-4 Sample of Network Environment	43
Figure 3-5 Key Components in AIPM	46
Figure 3-6 Layers of AIPM	51
Figure 3-7 Active Diagram of CiscoNATAACL Plug-In	52
Figure 4-1 Testing Environment of Unit Test 1	58
Figure 4-2 Testing Environment of Unit Test 2	61
Figure 4-3 Testing Environment of Unit Test 3	64
Figure 4-4 Testing Environment of Unit Test 4	67
Figure 4-5 Topology Used for the Test Cases in the Integration Testing	75
Figure 5-1 Network Environment of Case Study 1	88
Figure 5-2 Network Environment of Case Study 2	92
Figure 6-1 Key Components of AIPM	102
Figure 6-2 Sample of Command Tree of Cisco Router 2600	103

LIST OF TABLES

Table 2-1 Sample Attributes.....	17
Table 2-2 Comparison of Anomaly Detection Approaches	19
Table 2-3 Comparison of Misuse Detection Approaches.....	25
Table 2-4 Comparison of some Existing IDS and IPS	35
Table 4-1 Environment of Development	56
Table 4-2 Expected Output of Unit Test 1.1	59
Table 4-3 Testing Result of Unit 1.1	59
Table 4-4 Expected Output of Unit Test 1.2	60
Table 4-5 Testing Result of Unit Test 1.2	61
Table 4-6 Expected Result of Unit Test 2.1	62
Table 4-7 Testing Result of Unit Test 2.1	62
Table 4-8 Expected Result of Unit Test 2.2	63
Table 4-9 Testing Results of Unit Test 2.2.....	63
Table 4-10 Expected Result of Unit Test 3.1	65
Table 4-11 Testing Result of Unit Test 3.1	65
Table 4-12 Expected Result of Unit Test 3.2	66
Table 4-13 Testing Result of Unit Test 3.2	66
Table 4-14 Expected Result of Unit Test 4.1	67
Table 4-15 Testing Result of Unit Test 4.1	68
Table 4-16 Expect Result of Unit Test 4.2	68
Table 4-17 Testing Result of Unit Test 4.2	69
Table 4-18 Feeding Data for Unit Test 5.1.....	69
Table 4-19 Expected Result of Unit Test 5.1	69
Table 4-20 Testing Result of Unit Test 5.1	70
Table 4-21 Feeding Data for Unit Test 4.3.....	71

Table 4-22 Expected Result of Unit Test 4.3	71
Table 4-23 Testing Result of Unit Test 4.3	71
Table 4-24 Feeding Data for Unit Test 6.....	72
Table 4-25 Expected Result of Unit Test 6	72
Table 4-26 Testing Result of Unit Test 6	73
Table 4-27 Feeding Data for Unit Test 7.....	73
Table 4-28 Expected Result of Unit Test 7	74
Table 4-29 Testing Result of Unit Test 7	74
Table 4-30 General Configurations For Integration Testing.....	75
Table 4-31 Expected Result of Integration Test Case 1.1	77
Table 4-32 Testing Result of Integration Testing Of Case1.1.....	77
Table 4-33 Result of Ping Command	78
Table 4-34 Types of ACL.....	79
Table 4-35 Expected Result of Integration Test Case 1.2	79
Table 4-36 Testing Result of Integration Testing Case1.2.....	80
Table 4-37 Result of Ping Command	80
Table 4-38 Expected Result of Integration Testing Case 2.1	81
Table 4-39 Testing Result of Integration Testing Of Case 2.1.....	81
Table 4-40 Result of Ping Command	81
Table 4-41 Expected Result of Integration Testing Case 2.2	82
Table 4-42 Testing Result of Integration Testing Of Case 2.2.....	82
Table 4-43 Result of Ping Command	83
Table 4-44 Expected Result of Integration Testing Case 3.1	84
Table 4-45 Testing Result of Integration Testing 3.1.....	84
Table 4-46 Result of Ping Command	85
Table 4-47 Expected Result of Integration Testing 3.2.....	85

Table 4-48 Testing Result of Integration Testing Case 3.2	85
Table 4-49 Result of Ping Command	86
Table 4-50 Responding Time	87
Table 5-1 Expected Result of Case Study 1.1	89
Table 5-2 Testing Result of Case Study 1.1	89
Table 5-3 Result of Ping Command	90
Table 5-4 Expected Result of Case Study 1.2	90
Table 5-5 Testing Result of Case Study 1.2	91
Table 5-6 Result of Ping Command	91
Table 5-7 Expected Result of Case Study 2.1	93
Table 5-8 Testing Result of Case Study 2.1	93
Table 5-9 Result of Ping Command	94
Table 5-10 Expected Result of Case Study 2.2	94
Table 5-11 Testing Result of Case Study 2.2	95
Table 5-12 Result of Ping Command	95
Table 5-13 Expected Result of Case Study 2.3	95
Table 5-14 Testing Result of Case Study 2.3	96
Table 5-15 Result of Ping Command	96
Table 5-16 Comparison of Existing IDS AND IPS and AIPM.....	97
Table 5-17 Features Comparison of Different IDS AND IPS.....	99
Table 5-18 Comparison of Overhead	99
Table 6-1 Sample Routing Table.....	104

ABBREVIATIONS

ACL	Access Control List
AIPM	Automated Intrusion Prevention Mechanism
ANN	Artificial Neural Networks
ANNIDS	Artificial Neural Networks Intrusion Detection System
DDoS	Distributed Deny of Service
ID	Intrusion Detection
IDS	Intrusion Detection System
IDT	Intrusion Detection Technology
IPS	Intrusion Prevention System
IPT	Intrusion Prevention Technology
LAN	Local Area Network
NIDS	Network Based Intrusion Detection System
HIDS	Host Based Intrusion Detection System
NIPS	Network Based Intrusion Prevention System
HIPS	Host Based Intrusion Prevention System
IP	Internet Protocol
NAT	Network Address Translation
PAT	Network Address Port Translation
GA	Genetic algorithms
SSCR	Snort+SnortSam+Cisco Router
WAN	Wide Area Network

Chapter 1 Introduction

This is an information-explosion era. Information can be accessed in various ways, such as Internet, SMS, MMS, WAP, cable TV and etc. Additionally, the retrieving, processing, disseminating and storing of information have become more complex than before. This difficulty is because many enterprises and/or government institutes requires huge information to make important decisions. Although the computer industry is still a young industry, computers have made great progress in just a short period and are now used in all aspects of life such as education, entertainment, retails and etc. As a result of rapid technological progress, computer network is used wildly in almost every industry and organizations, such as in business, education, government and etc. Computer networks play more and more important role in current society and it completely change human's life.

Today, the enterprise networks often open to mobile employees, suppliers, partners, and customers. Thus, the accessibility, integrity, and confidentiality of the network have become critical factors in supporting the enterprise business. The computer and network security become more important than ever. Nevertheless, various threats and vulnerability can easily arise from mis-configured hardware or software, poor network design, inherent technology weaknesses, or end-user carelessness. For instance, on February 7, 2000, Yahoo's Internet portal was inaccessible for three hours because of a distributed deny of service (DDoS) attack. Analysts estimated that during the three hours, Yahoo suffered a loss of e-commerce and advertising revenue that amounted to about \$500,000 USD (BBC, 2000). In addition, On February 8, 2000, Amazon, CNN, and eBay were also attacked by DDoS attacks. (BBC, 2000) These attacks caused them to either stop functioning completely or slowed them down significantly. All these have led to a huge profit loss. Thus, how to protect the network from

security threats is imperative to make secure communication possible.

1.1 Motivations

The development of the Internet has created an environment in which millions of computers across the world are connected to each other. Furthermore, access to this network is fairly ubiquitous and cheap, enabling any cyber-thieves in the world to target any computers, regardless of their physical location. With rapidly growth of unauthorized activities on network, network security is now more important than before. Traditional way to protect access to internal networks is to deploy a firewall at there network perimeter to limit external access. Unfortunately, attack threats have matured rapidly and are now sophisticated enough to deceit firewall. As a result, important technologies in defense-in-deep, intrusion detection technology (IDT) and intrusion prevention technology (IPT) are developed as traditional firewall technology can no longer provide complete protection against intrusion.

Most of the current enterprise networks often use network address translation (NAT) and port address translation (PAT) technologies to keep pace with the network growth. As a result, many intrusion detection systems (IDSs) and intrusion prevention systems (IPSs) are unable to identify attackers' real IP address. Furthermore, many existing IDS AND IPS are standalone systems which do not communicate with other network devices, such as routers and switches. A standalone IDS AND IPS may not be able to protect an increasingly vulnerable network. Based on (FBI, 2006) report, there are about 70% attacks originated from internal, which means IDS AND IPS should pay more attentions to detect and prevent internal attacks. However, many current IDS AND IPS only focus on how to detect and prevent intrusion (e.g. Email spams, DDoS etc.) from the external. Hence, an integrated IDS AND IPS that is able to indentify whether NAT/PAT is deployed or the attacks stem from internal/external networks, will be an

important tool to help secure the network.

1.2 Objectives of the Research

The objectives of the thesis are aligned as follows:

1. To review and study how existing IDS AND IPS and network devices enhance network security
2. To compare and identify the strengths and shortages of existing IDS AND IPS.
3. To propose and develop an integrated IDS AND IPS mechanism to enhance network security.
4. To set up pseudo real-time environments to validate and fine tune the functionalities of the proposed mechanism.

1.3 Scope

In light of the objectives defined, the scope of the thesis is specified as below:

1. Review important roles and basic concepts of IDS and IPS as well as their mechanisms that detect/prevent malicious traffic.
2. Review and study the access control list (ACL) technology in routers.
3. Review and study the NAT/PAT operation and implementation in routers.
4. Identify problems in the existing IDS and IPS.
5. Investigate and analyze configuration command sets in routers of different model.
6. Review the technologies and basic concepts of router security
7. Analyze and understand some of the open source IDS AND IPS products.

8. Analyze the routing table format of different type of routers.
9. Analyze the NAT/PAT pool format and update mechanisms.
10. Review and study the way to communicate with network devices in network programming.
11. Design and develop the proposed mechanism while ensuring that the proposed mechanism does fully satisfied the design objectives.
12. Set up testing environments, which includes unit test and integrated test.
13. Verify and validate the written code using several case studies.
14. Make sure the proposed mechanism has ability of automated analyze malicious traffic and automated intrusion prevention. Ability of blocking interior attack and exterior attack are also need to be tested.

1.4 Methodology

This research will begin with a literature review on the functionality of existing IDS AND IPS. The intrusion detection technology (IDT) and IDS will be reviewed first, followed by reviewing intrusion prevention technology (IPT) and IPS. Many new and traditional technology and approaches of IDT/IPT will be studied in this part. Subsequently, a study of strengths and weaknesses of current IDS AND IPS will then be performed.

In the analysis, design and development phase, a new prevention mechanism will be proposed and developed to overcome some of the weaknesses Unified modeling language (UML) will be used in this analysis and design phase, together with the use case diagram, the activity diagram, and the component diagram. In development phase, each functions will be carefully designed according result of analysis and design phase and written in ANSI C language.

After finished analysis, design and development, a testing environment to validate the functionalities of the proposed mechanism is then set up. Every small function will be divided and be tested individually in unit testing. An integrated testing will be implemented after successful unit testing. At the meanwhile, fine-tuning may be necessary throughout the testing process in order to achieve optimal performance.

1.5 Thesis Organizations

This thesis report covers a total of six chapters.

Chapter one defines a general overview on the goals this research aims to achieve, and describes the research methodology used in this work.

Chapter two provides general background on the concepts and strategies of IDS AND IPS. Current intrusion detection technology and intrusion prevention technology are discussed and analyzed; this is followed by the discussion of conventional intrusion detection system and intrusion prevention system. The disadvantages of existing IDS AND IPS systems are also discussed. After finished literature review, some major problems of current IDS AND IPS are identified. The purpose of this research is to minimize or solve those problems.

Chapter three analyzes the purpose and main conception of this research. Object oriented analyzes method and UML are used in this chapter. Then, the architecture and design of the proposed system is presented. An analysis of proposed system and proposed mechanism is discussed as well.

Chapter four provides an overview of system development. Unit and integration testing

deployed in this work is also reported.

Chapter five describes two case studies which evaluate the proposed system in different networks. Next, a comparison between existing IDS AND IPS and the proposed system are presented. Results of case studies are analyzed and portrayed in tables to ease the evaluation of the mechanisms concerned. Finally, advantages of the system and existing IDS AND IPS systems are discussed in the end of this chapter.

Chapter six ends this thesis by summarizing the efforts and contributions made and by discussing problems faced in the development phased. Additionally, some suggestions for improvement are outlined to allow possible development of new ideas and realms on this research.

Chapter 2 Literature Review

Covering in this chapter are some background information on intrusion detection and intrusion prevention. Current intrusion detection technology/system (IDT/IDS) are discussed first, followed by current intrusion prevention technology/system (IPT/IPS). The issues and challenges of current intrusion detection/prevention are also elaborated. Then, several mechanisms used by existing popular IDS AND IPS together with their disadvantages are pointed out. Finally, the chapter ends with some problems that this thesis attempts to solve.

2.1 Intrusion Detection System

McHugh et al. (2000) defined that intrusion detection system or IDS as software, hardware, or combination of both for detecting intruder activities by using intrusion detection techniques. IDS look for attack activities and signatures, which are specific patterns that usually indicate malicious or suspicious intention.

Anderson (1980) grouped IDS into two basic categories based on the technology used. The first category is signature-based or misused-based intrusion detection systems. Each type of intrusion has unique signatures, like computer viruses, which can be detected by software. In other words, signature is any distinctive characteristic that can be used to identify something. Based upon a set of signatures and rules, the detection system is able to find and log suspicious activity and generate alerts. The second category is anomaly detection systems. Anomaly-based intrusion detection usually depends on packet anomalies present in protocol header parts.

In 2000, Caberera etc. divided IDS into another two primary categories. The first category is

network-based intrusion detection systems (NIDS), another one is host-based intrusion Detection Systems (HIDS). This is one common classification of IDS. This classification relies on the way that data is collected by IDS, not how or where it is processed.

NIDS uses raw network packets as the data source, and network adapter in promiscuous mode that listens and analyses all traffic in real-time as it travels across the network. Using NIDS has following advantages: first, the ability of real time detection and response to intrusion; second, the ability to detect unsuccessful attack attempts; third, operating system (OS) independence, which means that NIDS can monitor intrusions on multiple platform (YueBin et al., 2003a).

NIDS involves looking at the packets on the network when network traffics pass by some sensors. The sensors can only see the packets that happen to be carried on the network segment it is attached to. Packets are considered to be of interest if they match a signature. Three primary types of signatures are string signatures, port signatures, and header condition signatures. Well-known, network-based IDS include AXENT, CyberSafe, ISS, Snort, and Shadow.

HIDS uses the software that continuously monitors system specific log in real-time, compared to other solutions that run processes that check the logs periodically for new information and changes. Some HIDS can also listen to port activity and be alert when specific ports are accessed; this allows for some network type attack detection (Aurobindo, 1996). Using HIDS has following advantages: first, cost saving without the need to buy new hardware; second, the ability to detect and response almost in real time detection; third, the deployment of HIDS in encrypting and switching environment (YueBin et al., 2003a).

In general, NIDS is the core, and is more important; HIDS is just a supplement. An efficient intrusion detective capability will use both host- and network-based systems.

2.2 Intrusion Detection Techniques

Intrusion detection is a set of mechanisms that aim at scanning system resources and detecting the activity of intrusion (Cowan et al., 1998). The intrusion detection methods are used to collect useful information from known attacks, and to find out if someone is trying to attack the network or particular networks.

Amoroso (1999) define intrusion detection (ID) as:

“Process of identifying and responding malicious activities targeted at computing and network resources”

More specifically it involves “monitoring and analyzing both user and system activities, analyzing system configurations and vulnerabilities, assessing system and file integrity, being able to recognize patterns typical of attacks, analyzing abnormal activity patterns, and tracking user policy violations (Nilsen,2002).”

Anderson (1980), Denning and Neumann (1985), Denning (1987), and Sebring et al. (1988) identified two major types of intrusion detection strategies. They are anomaly detection (e.g. NIDES/STAT [Javits and Valdes 1993]) and misuse detection (e.g. NetSTAT [Vigna and Kemmerer 1999]). Anomaly detection is based on the normal behavior of a user or a system; any actions that significantly deviate from the normal behavior are considered intrusive. Misuse detection detects attacks based on the characteristics of known attacks or system vulnerabilities; any actions that conforms to the pattern of a known attack or vulnerability is considered intrusive.

2.2.1 Anomaly Detection

An Anomaly Detection is a method for detecting computer intrusions and misuse by monitoring system activities and classifying them as either normal or anomalous. The classification is based on heuristics or rules, rather than patterns or signatures. Anomaly Detection will detect any types of misuse that fall out with normal system behaviors (Jagannathan et al., 1993). It is different with signature-based systems, which can only detect attacks for which a signature has been created previously. Overall, the primary strength of anomaly detection is its ability to recognize unknown attack method (Mchugh J. et al., 2000).

Steven et al. (1997) stated that there are two stages to the anomaly detection. In the first stage, profiles or databases of normal behavior (this processing is called as training phase for a learning system) are created; in the second stage, databases are used to monitor system behavior for significant deviations from normality (this processing is called as test phase or detecting phase).

As discussed previously, in order to identify malicious traffic, the system must be taught to recognize normal system behaviors. This can be accomplished in several ways; the main approaches are described below.

YueBin, Kobayashi et al. (2003b) stated that the first and popular approach is by using artificial intelligence. Some IDS uses the artificial neural networks (ANN) learning technology; this kind of IDS are called artificial neural networks intrusion detection system (ANNIDS). The idea is to train the neural network to learn user's action and command. A program can then tell the network how to response to an external activity or can take corresponding activity on its own, such as blocking intrusion IP, dropping packets, etc. After the training phase, the network tries to match actual behaviors with the user profile, which already be created in the network. Any

unmatched events imply the deviation of the user from the established profile (Aurobindo, 1996). ANNIDS does not need expert knowledge and it can find unknown intrusions because of the advantage of neural network (Aly et al., 2005). However, this approach also has disadvantages inherited from neural network. A major disadvantage is its complexity and difficulty to define an appropriate knowledge representation.

Neural networks use several principles to make decision; these principles include gradient-based training, genetic algorithms (GA), fuzzy logic, and Bayesian methods. (Verwoerd et al, 2001).

University of Malaysia

2.2.1.1 Genetic Algorithms in Anomaly Detection

Some approaches that apply the GA technology to IDS have been introduced in recent years. For instance, Pillai et al. (2004) describes an approach to implement a NIDS using GA. The network traffic sniffed is analyzed to create a data set first. The next step is to automatically generate the rule set. The NIDS explained in this paper uses the data set to classify the anomalous connections. This data set serves as the basis for the NIDS to detect intrusions.

Pillai et al. (2004) use the following system design to build a new NIDS.

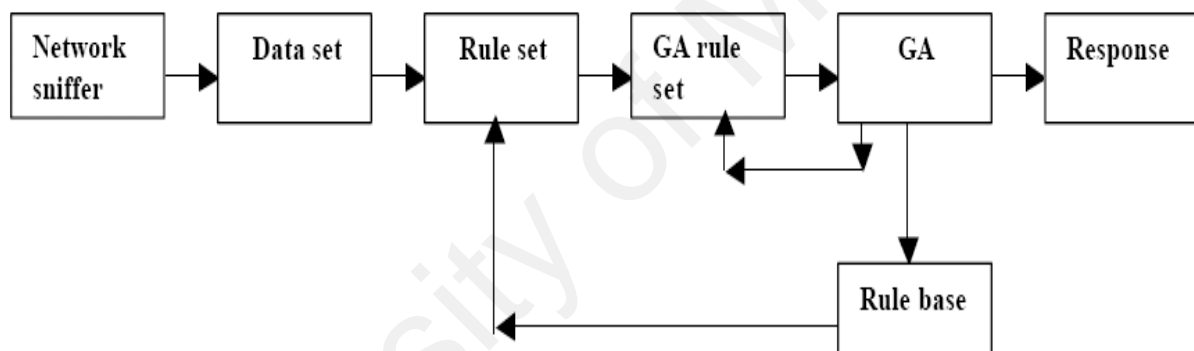


Figure 2-1 System Design for NIDS

The network traffic used in the GA component (see Figure 2-1) is a pre-classified data set that differentiates normal network connections from anomalous ones. This pre-classified data set is manually created by analyzing the data that are captured by network sniffer. Based on the traffic from a sniffer software (e.g. WinDump), the data set is created. The data set then covers the necessary information to generate rules. These information include the source IP address, the destination IP address, the source port, the destination port, the protocol used, and a field that indicates whether the specific connection related to an intrusion or not. The data set will include both normal and anomalous network connections. When a connection refers to an entry in the dataset, the connection will be marked by the value true or false. If the connection is an

intrusion, the value is truth, or false otherwise. These network connections in the dataset are manually created by network administrators. The dataset is used to generate rules for the rule set prior to training the GA. Once the GA is trained with the rules, more network connections can be added to the dataset. This means that administrator have to update the dataset to add a new connection or to discard a connection. As soon as the initial dataset is created, the next action is to create the rule set. Rules will be generated in the rule set by analyzing the dataset.

The first part of the GA will act as a search algorithm. The purpose is to help the rules acquire values that will be used in the fitness function later. The search algorithm will match the rules with any anomalous connections that occur on the network to detect an intrusion. Each rule will carry values for the intrusions that they have detected, and a value for a false alarm that the rule produces. The initial values for the rule will be initialized to zero. The rules will acquire these values when the search algorithm is executed. Once the rules have acquired the values, the complete GA which includes the fitness function and mutation will be executed.

The second part of the GA is the fitness function. The fitness function determines whether a rule is 'good', in other words, it detects intrusions, or whether the rule is 'bad', in other words, it does not detect intrusions. The fitness function is calculated for each rule.

When an intrusion is indicated, the network sniffer will be executed to determine whether it is an intrusion or a false alarm. The fitness function will assign a fitness value for each rule. The fitness value will be a predefined value. The rules, which have acquired the required fitness value, will be selected to form a new generation to produce new rules. When a rule acquires the predefined value of "1", the rule will be reproduced for producing a new generation of rules. After a successive number of generations, if the rule still has not gained the value "1", which means it is unfit, the rule will be discarded.

Subsequently, the GA will regenerate a new rule according the crossover of chromosomes. The GA will provide the rule in codified form to the GA rule set. The NIDS using the GA, which explained in paper of Pillai et al. (2004), produces new rules for each connection through mutation. For instance, all connections, which are established from source to destination using the specific protocol, are treated as an intrusion regardless of source ports it uses. This mechanism increases the effectiveness of detecting intrusions for that specific connection.

The GA then continues to detect intrusions and produce new rules, storing 'good' rules in the rule base and discarding 'bad' rules. Once in a while, the dataset will have to be updated for new connections, and the rule set will also be updated. Additionally, the human factor is important because a human input is still required to the dataset.

The rule set will initially contain the new generation of rules that the GA produces. Once the GA has been trained, i.e. when it has supplied a 'good' set of rules for the corresponding dataset to the rule base, the rules in the rule base will be supplied to the rule set. The rule set can then start using these new produced rules to detect intrusions. Pillai et al. (2004) shows using GA in this way can improve efficiency of detecting intrusion.

2.2.1.2 Fuzzy Logic in Anomaly Detection

Fuzzy logic is suitable for solving the intrusion detection problem because of two major reasons. First, intrusion detection includes many quantitative features. The second reason is that security itself includes fuzziness. After given a quantitative measurement, an interval can be used to present a normal value. Then, any values falling outside the interval will be indicated as anomalous to the same degree regardless of their distance to the interval. (Susan M. Bridges et

al. 2000). Many researchers have already done some works in this field. Norbik et al. (2005) proposed a dynamic model intelligent IDS which utilizes fuzzy logic along with the data mining technique. Additionally, a new framework for implementing intrusion detection systems using fuzzy logic is also proposed by Aly et al. (2005)

The dynamic model of intelligent IDS that is proposed by Norbik et al. (2005) is based on specific AI approach for intrusion detection. The authors combined fuzzy logic with data mining to provide efficient technique for anomaly-based intrusion detection. Neural networks and fuzzy logic with network profiling, and simple data mining techniques to process network data are investigated in this paper. The proposed system in this paper is a hybrid system, which combines anomaly, misuse, and host-based detection technologies. Simple fuzzy rules construct if-then rules that reflect common ways of describing security attacks. Suspicious malicious traffic can be traced back to the source, and any package from that particular source will be redirected back to them in future. Network traffic and system audit data are used as inputs for both.

In the model of Norbik et al., the data processor and classifier summarize and tabulate the data into categories. A kind of data mining is performed on the collected data in this stage. In the next stage, the current data and historical-mined data is compared to create values, which reflect how new data differ from the past observed data. According to the facts from the analyzer, the decision that whether to activate the detection phase or not will be taken. If the detection phase is activated, then an alert will be issued and tracer phase will be implemented. This phase will trace back to the intruders original source location. The authors proposed a framework for tracing the abnormal packets back to its original source. Once the intruder's original location has been identified and verified, all traffics from that particular host will be redirected to their source in future.

There are four steps in this model. Prior to any data analysis, attributes representing relevant features of the input data (packets) must be established. The set of attributes that send to the Data Analyzer is a subset of all possible attributes that contained in packet headers, packet payloads, as well as aggregate information, such as statistics on the number and type of packets or established TCP connections. After identifying data source and defining relevant attributes, a Data Analyzer is implemented to compute configuration parameters that regulate operation of the IDS. This module analyzes packets and computes aggregate information by grouping packets. Then rules will be automatically generated. When trace back is requested, a query message, which consists the packet, egress point and the time of receipt, is sent to all the Local Data Managers (LDM). Subsequently, LDM responds with the partial attack graph and the packet when it entered the region. The attack graph either terminates within the region managed by the LDM, in which a source has been identified, or it contains nodes to the edges of the other LDM network region.

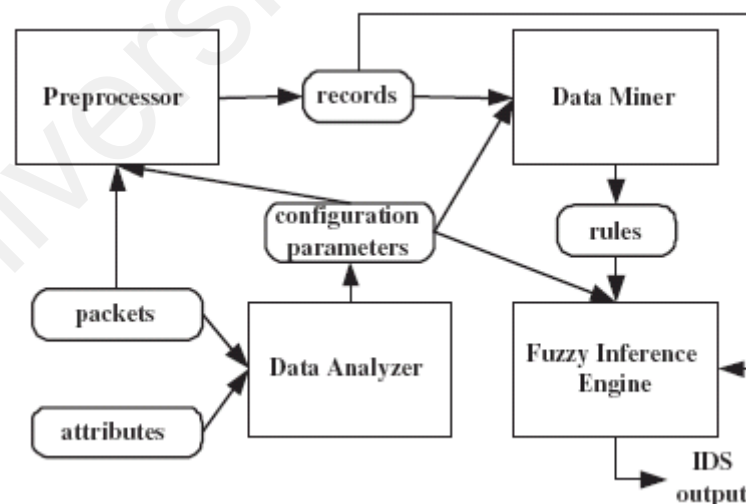


Figure 2-2 System architecture in the model of Aly et al.

Aly et al. (2005) proposed a fuzzy data-mining algorithm to generate fuzzy rules for the inference engine. The modular architecture is implemented using the Java Expert System Shell

(Jess) and the FuzzyJess toolkit. The proposed system architecture (Figure 2-2) has two modes of operation: rule-generation and detection.

When operating in the rule-generation mode, the system processes network data and uses a fuzzy data mining algorithm to generate rules. The data mining algorithm produces a subset of the rules and the rules are used as a model for the input data. In next stage, the detection mode uses this rule subset for intrusion detection. The different components of the architecture are described in the following.

The initial input of IDS includes packets and attributes. Attributes are represented by names, which will be used as linguistic variables by Data Miner and the Fuzzy Inference Engine. Table 2-1 shows a typical sample of attributes.

Name	Description
ICMP	Number of ICMP packets received within a time frame
UDP	Number of UDP packets received within a time frame
TCP	Number of TCP packets received within a time frame
SYN	Number of TCP SYN packets received within a time frame
FIN	Number of TCP FIN packets received within a time frame

Table 2-1 Sample Attributes

Attributes name and describe the parts of the packets for which some analysis must be made. They are stored in the following format: $VARs = \{attribute_1, attribute_2, \dots, attribute_n\}$ where $VARs = \{ICMP, UDP, TCP, SYN, FIN\}$ is a typical example of useful attributes. Data analyzer module analyzes packets and generates aggregate information by grouping packets. Packets can be placed in fixed size groups (n -group) or in groups of packets captured in a fixed amount of time (t -group). Upon analyzing data and attributes, the Data Analyzer creates a file where all configuration parameters are stored. The configuration Parameter values stored in the file regulates operation of the Preprocessor, Data Mining algorithm, and Fuzzy Inference Engine. The configuration file specifies how to normalize attribute values, associate attributes

with a term set, and how to describe functions corresponding to the fuzzy membership functions of each term. The Preprocessor is responsible for accepting raw packet data and producing records for each n -group or t -group as specified by the configuration file. Both rule-generation mode and detection mode will use this component. The data miner allows for efficient, single-pass, record processing by partitioning data into hierarchical file. The procedure is a deterministic algorithm that produces a set of output rules. The Rules are expressed as a logic implication $p \rightarrow q$, where p is called the antecedent of the rule and q is called the consequence of the rule. p and q are both assumed to be in conjunctive normal form.

The main scientific merit of this architecture is its demonstrated ability. It works as a hybrid-system using fuzzy logic rules, which produced by a data mining algorithm. Data preprocessing and the use of fuzzy logic to describe attributes of interest can greatly reduce the amount of data to be analyzed by the inference engine as the great amount of data to be analyzed is the main bottleneck of many IDS in high-bandwidth network.

The advantages of using the ANN technology include three. First, this kind of IDS copes well with noisy data. Second, the technology does not depend on any statistical assumption about the nature of the underlying data. Third, it is easier to modify for new user communication. However, this technology still faces some problems, such as false positives and false negatives, and the opportunity exploited by the intruder to train the network during its learning phase. (Aurobindo, 1996)

2.2.1.3 Predictive Pattern Generation in Anomaly Detection

Another approach of anomaly detection is predictive pattern generation detection. The idea of this approach is to define what normal usage of the system comprises by using a strict

mathematical model, and to mark any deviation from this as intrusion (Aurobindo, 1996). This is also called as strict anomaly detection. Rule-based sequential patterns can detect anomalous activities that were difficult with traditional methods. The built systems using this model are highly adaptive to changes because not good patterns are continuously eliminated. This technology also faces the problem of false positives and false negatives which stem from strict rules that may affect normal action.

2.2.1.4 Comparison of Anomaly Detection Approaches

	Genetic Algorithms	Fuzzy Logic	Predictive Pattern Generation
Rate of false positives and false negatives	Lower	Low	High
Need Training?	Yes	Yes	No
Efficiency	Low	Low	High
Complexity	High	High	Lower
Latency	Long	Long	Short

Table 2-2 Comparison of Anomaly Detection Approaches

Table 2-2 shows the comparison of the aforementioned anomaly detection approaches. Predictive pattern generation has higher rate of false positives and false negatives because it does not include a training phase as in genetic algorithms and fuzzy logic. The lack of the training phase, however, leads to better efficiency, less complexity, and shorter latency compared to other anomaly detection approaches. On the other hand, genetic algorithms and fuzzy logic which include a training phase to improve its capability of detecting intrusion generates lower rate of false positives and false negatives. This advantage comes at the cost of lower efficiency, higher complexity, and longer latency as compared to predictive pattern generation.

2.2.2 Misuse Detection

A Misuse Detection is a method for detecting attacks based on patterns/signature or weak spots of system so that even variations of the same attack can be detected. This means that these methods can only detect many or even all known attack patterns, but they are of little use for as yet unknown attack methods.

The main problems that misuse detection system are how to write a good signature that encompasses all possible variations of the pertinent attack, and how to write signatures that do not also match non-intrusive activity (Aurobindo, 1996). Several methods of misuse detection, including a new pattern/string matching model are used in this kind of IDS. Those major approaches based on misuse IDS will be discussed below.

2.2.2.1 Pattern/String Matching in Misuse Detection

A popular technology is based on pattern or string matching. This approach encodes known intrusion signatures as patterns that are then matched against the data. It attempts to match incoming events to the patterns that represent intrusions scenarios (Aurobindo, 1996). For example, the presence of "scripts/iisadmin" in a packet going to a web server may indicate an intruder activity. Signatures may be present in different parts of a data packet depending upon the type of the attack. For example, signatures can be found in the IP header, transport layer header (TCP or UDP header) and/or application layer header, or payload. The problem of this approach is that it can only detect attacks based on known unauthorized behaviors.

Lower efficient string matching algorithm may make IDS to become a performance bottleneck in network (YueBin et al., 2003b). Many researchers are trying to find/create higher efficient string matching algorithm to increase the performance of IDS. Some new string matching algorithms are already developed in recent years. YueBin et al. (2003b) proposed a new String Matching Technology in their paper. The new algorithm is based on simplified Boyer-Moore-Horspool algorithm. In this paper, Array NEXT in preprocessing stage is redesigned. Novel generated rules are presented. Using these rules, a simple NEXT is generated. These characteristics will be useful in all these applications.

The new string matching algorithm is described as follows;

```

Begin
1  For (each char  $\in$  t_str ) do begin
2      If (each char  $\in$  p_str)
3          do next[char] $\leftarrow$  n-j ;
4      If (each char  $\notin$  p_str)
5          do next[char]  $\leftarrow$  n+1;
        end for
6  If (n>m) do exit;
7  while ( p<m) do begin
8      j $\leftarrow$ n-1;
9      i $\leftarrow$ p;
10     while (j $\geq$ 0) and (t_str[i]=p_str[j]) do begin
11         j $\leftarrow$ j-1;
12         i $\leftarrow$ i-1;
        end while
13     if (m-p>n)
14         do p $\leftarrow$ p+next[t_str[p+1]];
15 end while
end.

```

Where, t_str is the text string of length m . P_str is the pattern string of length n . $next$ is an array. i and j are pointers of text string and pattern string respectively. p is a work pointer of text string.

This string matching algorithm can be divided into two parts: preprocessing stage and search stage. Step 1 to 5 in the above algorithm is called as preprocessing stage. Step 6 to 14 is called

as searching stage.

There are some distinctive points in the algorithm. First, the string matching algorithm is very simple and easy to implement. The key part of the algorithm has only 14 steps. Second, the algorithm is more effective than Boyer-Moore and Boyer-Moore-Horspool algorithm even it is based on Boyer-Moore and Boyer-Moore-Horspool algorithm. Third, the algorithm designed a novel and simple array NEXT. It makes the times of shift during string matching decreases;

2.2.2.2 State Transition in Misuse Detection

Another approach of misuse detection is based on state transition. The monitored system is described as a state transition diagram. When data is analyzed, the system makes transitions from one state to another. A transition takes place on some boolean condition beginning true. In this approach, the initial node is in legal states, and the terminal node is in the final compromised state. This approach is more robust in detecting unknown vulnerabilities because the tradeoff of monitored system states is emphasized (YueBin et al. 2003a). However, this approach also has several weaknesses. First, the attack patterns can only be specified on a sequence of events, rather than more complex forms. Second, some intrusions behaviors may not be detected because they are either not recorded or cannot be described by state transition diagrams. Currently, state transition analysis technique is a new approach applied to intrusion detection.

Brian et al. (2004) proposed a new state transition model for intrusion detection. The approach proposed in the paper is based on characterizing intrusion attacks into sequences of computation and communication processes. The state transition model proposed includes three layers. These layers are the physical layer, the communication sequencing layer, and the state

transition layer. The physical layer defines the components and devices which are necessary for an attack include source/victim computers, software, ports, and routers. The communication sequencing layer provides abstraction of all the computation and communication processes that are executed through an attack scenario. The state transition layer describes the attack profile in terms of states and transitions between the states. The start state is the inactive state on the source machine, and there are three final states.

This paper demonstrates through a case study the applicability of a state transition model approach for modeling both network and host information pertaining to intrusive activities. The proposed approach requires profiling attack activities and victim responses as states and state transitions. As such, the proposed approach is suitable for deployment in IDS. The approach allows for matching the detected suspicious activity to victim states; thereby it permits determination of the state transitions both forward (in the future) and reverse (into the past). The IDS could then scan past network or host logs for evidence of specific activities that would match with past states; and heighten scanning for expected attack vectors.

Brian et al. (2004) proposed a case study include source and victim machine in the paper. For source machine, initially, the source machine is switched off and as a result, it is in the start state, 'Inactive'. When the source machine is switched on it goes to the 'Active' state. As the LANguard program on the source machine is initiated, the source goes to the 'Probe Active' state. The attacker, then, inputs the victim's IP address of to LANguard along with the port type and port to be scanned; as a result, the source enters the 'Probing' state. Initially, the victim's machine is switched off in this part of the experiment before this probe attack is launched to realize the 'Probe Fail' state on source. The authors use the 'Probe Rcvd' and 'Probe Fail' states to indicate that the victim's machine is turned off and is not active. Now, the victim's machine is turned on and the above TCP port scan actions are repeated. The 'Probe Rcvd' and 'Probe

Successful' state indicates that the victim has been successfully probed and the probe information has been received. The source machine then goes to the 'Decision' state based upon the probe information that has been received from victim's machine. For victim machine, the victim's machine is initially either in the switched off state 'Inactive' or the switched on state 'Active'. The victim machine then goes to the 'Probing' State after a TCP SYN scan signal from the source to the victim arrived. After the TCP port has been probed, the results are sent back to source machine leading to the 'Probe Snt' state. Since TCP port on the victim's machine is open, it could be abused and is vulnerable which leads the victim to the 'Insecured' state.

2.2.2.3 Model-Based Intrusion Detection in Misuse Detection

The third approach of misuse-based IDS is model-based intrusion detection. It states that certain scenarios are inferred by certain other observable activities. Once these activities are monitored, it is possible to find attack by analyzing activities that infer a certain intrusion scenario. This kind of system can predict the intruder's next step based on the intrusion model. These predictions can be used to verify an intrusion hypothesis, to take preventive measures, or to determine which kind of data to look for next. The problem of such system is that the patterns for intrusion scenarios must be easily recognized. Additionally, the patterns must not be associated with other normal behavior. Which means it is difficult to define good patterns (Aurobindo, 1996).

Timothy et al. (2005) argued in favor of the explicit inclusion of suspicion as a concrete concept to be used in the analysis of audit data in order to guide the search for evidence of misuse. The explicit concept of suspicion shows promise in assisting model-based IDS. The approach is similar to that of a human forensic analyst, who first notices details that seem slightly odd, and then investigates further and cross checks information in an attempt to build a consistent and

clear explanation for the observed details. This paper uses deductive reasoning combined with expert knowledge about system behavior, potential attacks and evidence, and patterns of suspicion to link individual clues together in an automated way. A prototype implementation that is based on these considerations is presented in the paper; it includes details of how to represent suspicions and deductions, and how these structures are updated when new evidence is discovered. Finally, this paper describes how the algorithm performs on a realistic example, which automatically brought five discrete pieces of evidence together to create a unified and coherent description of what is believed to have occurred.

The authors of this paper brought two new ideas: first, a deductive point of view, where clues are identified, their consequences investigated, and supporting evidence may or may not be found, and second, the inclusion of suspicion as an explicit concept that guides the search. The authors have developed these ideas into a working prototype system. Additionally, this paper also discusses deduction and modeling, including how uncertainty and suspicion figure into the approach.

2.2.2.4 Comparison of Misuse Detection Approaches

	Pattern/String Matching	State Transition	Model-Based Intrusion Detection
Efficiency	Depends on pattern/string matching algorithm	Low	Low
Rate of false positives and false negatives	Medium	High	High
Complexity	Low	High	High
Latency	Short	Long	Long

Table 2-3 Comparison of Misuse Detection Approaches

Table 2-3 depicts the comparison of misuse detection approaches. State transition and

model-based intrusion detection approach has lower efficiency due to high complexity. The efficiency of pattern/string matching depends solely on the algorithm. In comparison to pattern/string matching, state transition and model-based intrusion detection approach has higher rate of false positives and false negatives. Finally, latency of pattern/string matching intrusion detection approach is the shortest among these approaches.

2.3 Issues and Challenges in Current Intrusion Detection

The goals of all IDS are higher detection rate and lower false positive rate. However, they are serious challenges to the current IDS technologies. The main reason is the lack of sufficient and efficient information available to IDS. Many current IDS use raw packets to analyze. However, only using these raw data sources is not efficient for IDS to perform further analysis (Xinzhou et al., 2002). IDS can improve the detection efficiency by integrating multiple and diverse sources of information. This method can also help an IDS cross-check the analysis results and improve accuracy. Additionally, many IDS, such as Snort, rely on signature matching techniques, which can only detect those known attacks. As the network is growing, many new attack methods have appeared. Therefore, using only misuse detection technique is insufficient to resolve the problem; anomaly detection technology should be more widely applied. Another challenge to current IDS is the sophistication of attack strategies and attack tools. Latest attacks always try to evade being detected. For example, multi-staged attacks, such as DDoS, have become one of the most difficult intrusions to detect, and they are also one of the most dangerous threats to networks. These attacks can be distributed and coordinated by using attack relays to achieve the end-goals. However, current IDS lack the ability to analyze the related security events in multiple domains, hence cannot detect attacks effectively (Xinzhou et al., 2002).

2.4 Intruder Prevention System

Intrusion prevention technologies (IPT) are differentiated from intrusion detection technologies (IDT) as IPT can respond to a detected threat by attempting to prevent it from succeeding (Jakubet et al., 2006). IPT includes several active response techniques, which can be divided into the following groups. The first group is based on the fact that IPS stops the attack itself, i.e. the IPS can terminate the network connection or user session that is being used for the attack, and block access to the target from the offending user account, IP address, or other attacker's attribute. IPS can also block all access to the targeted host, service, application, or other resource. The second group is based on the fact that IPS can change the security environment (Jakubet et al.2006). The IPS could change the configuration of other security controls to disrupt an attack. Common examples are reconfiguring a network device (e.g. firewall, router, switch) to block access from the attacker or to the target, and to alter a host-based firewall on a target to block incoming attacks. The last group is that IPS can change the content of attack. Some IPS technologies can remove or replace malicious portions of an attack to make it benign. An example is an IPS removing an infected file attachment from an e-mail and then permitting the cleaned email to reach its recipient. Another example is an IPS that acts as a proxy and normalizes incoming requests, which means that the proxy repackages the payloads of the requests, discarding header information. This might cause certain attacks to be discarded as part of the normalization process.

Nick et al. (2005) gave a definition of IPS as:

“An Intrusion Prevention System (IPS) is any device (hardware or software) that has the ability to detect attacks, both known and unknown, and prevent the attack from being successful. ”

Like IDS, the IPS can also be divided to two categories: The first category is a host-based IPS (HIPS); it provides a component that effectively integrates into host system. Another category is a network-based IPS (NIPS); it provides a component that effectively integrates into the overall network security framework.

Aurobindo (1996) stated some limitations of HIPS and NIPS. HIPS lacks complete coverage and is subject to end-user tempering, and NIPS is difficult to be deployed to protect the network effectively.

Many IPSs use packet filtering and IP blocking to block intrusion behavior. Packet filtering is a technology that drops traffic containing malicious information. IP blocking uses access control mechanism to block malicious host IP.

Detecting intrusions do not attempt to automatically prevent intrusion in the first place. If a vulnerable system is successfully intruded by a malicious host, then IDS may detect and send an alert about the intrusion but take no further steps to block packets from the attacker. Hence the attacker can have full access and control of the target system until an administrator can manually intervene. The time lag between successful compromise and such intervention may be quite long (Xinyou et al., 2004). That is the reason of why automatically blocking attacks can be an attractive capability if it could be done effectively. Active response is such technology. It is the core idea of IPS. Active response is a mechanism that can dynamically reconfigure or modify network access, sessions, and individual packets based on alerts generated from IDS. The goal of active response is to automatically respond to a detected attack and minimize (or ideally nullify) the damaging effects of attempted computer intrusions without requiring an administrator.

2.5 Issues and Challenges in Current IPS

The challenge of NIPS is its immense effort up-front to implement and refine. Refinement is challenging with this type of tool since events and alerts have to be individually analyzed and in many instances investigated prior to implementation of a white-list policy. On occasion, a lab environment supported by multiple resources has to be set up. These resources include internal network, security, OS, and application team members; not to mention external support resources and consultants to collaborate on the intent of one event type, which may have happened once or thousands of times.

2.6 Existing Open Source Popular IDS AND IPS

Many open-source IDS and IPS have been developed to secure the network infrastructure and communication over the Internet. This section discusses three mechanisms of Snort, SnortSam, and Snort-Inline – for their wide deployment.

2.6.1 Snort

Snort is perhaps the best known signature-based NIDS because Snort is open source and is reasonably easy to modify or extend. Snort operates three main functions: first, it can serve as a packet sniffer; second, it can serve as a packet logger; finally, it can serve as a NIDS. There are also many add-on programs to Snort to provide different ways of recording and managing Snort log files, fetching and maintaining current Snort rule sets, and alerting to let administrators know when potentially malicious traffic has been found.

Figure 2-3 shows how these components are arranged.

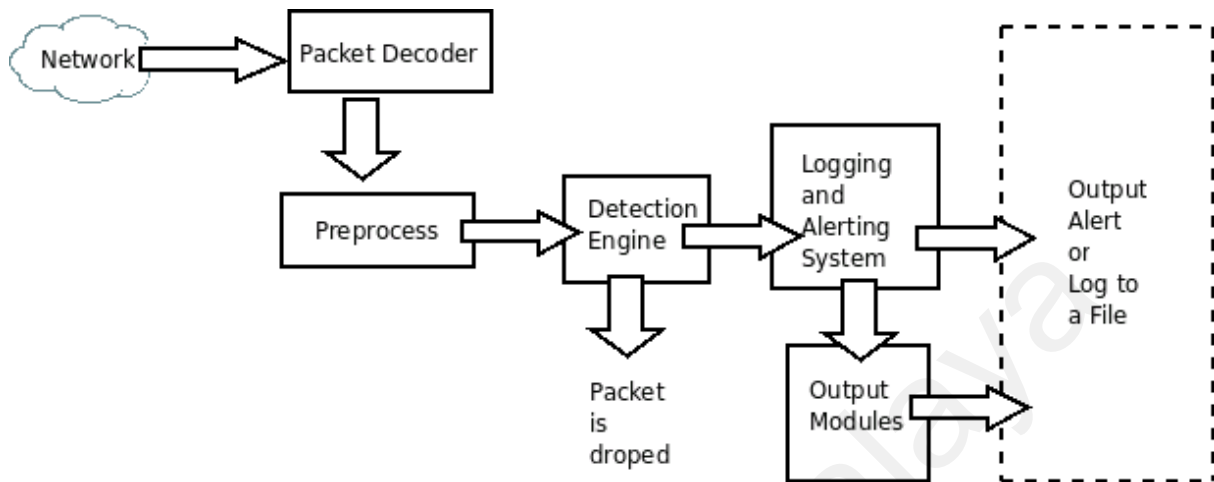


Figure 2-3 Components of Snort

Snort can logically be divided into many components. These components work together to detect attacks and to generate output in required format. Major components of Snort are Packet Decoder, Preprocessors, Detection Engine, Logging and Alerting System, and Output Modules.

First, traffic is acquired from the network link via the libpcap library. Once packets from different network interface are received, the packet decoder module determines which protocol is in use for a given packet and matches the data against allowable behavior for packets of their protocol.

Packets are then sent through a set of preprocessors. Preprocessors are components that can be used with Snort to arrange or modify data packets before the detection engine does some operation to identify whether the packet is an intrusion. Packets are examined and manipulated before being handed to the next module - detection engine. Functions of preprocessors in Snort include defragment packets, decode HTTP URI, and re-assemble TCP streams, and etc.

After the Preprocessors module finishes processing packets, the packets will be sent to the detection engine module. The detection engine module is the most important component of Snort. Its responsibility is to detect if any intrusion activity exists in a packet. The detection engine employs Snort rules for this purpose. The rules are read into internal data structures or chains where they are matched against all packets. If a packet matches any rule, appropriate action is taken; otherwise the packet will be dropped.

Finally, after the rules have been matched against the data, the packet may be used to log the activity or generate an alert according what the detection engine finds inside a packet

Output modules or plug-ins can do different operations depending on how an administrator wants to save output generated by the logging and alerting system. Basically these modules control the type of output generated by the logging and alerting system.

2.6.2 SnortSam

SnortSam (Frank, 2006) is an active response system that interacts with both commercial and open source firewalls to block IP address at direction of a patched version of the Snort IDS (Beale, 2004). ANSI C language is used in developing this IPS. The purpose of using SnortSam is to create a firewall/IDS combined solution. In this solution, the firewall can be automatically configured to block malicious traffic and intruder's IP addresses when an intruder activity is detected. SnortSam supports flexible time specification for blocked addresses, which means that an IP address can be blocked for period of seconds, minutes, hours, days, weeks, or even years. Additionally, SnortSam can run as a daemon on the firewall host and accept commands from a special output plug-in for the Snort IDS over an encrypted TCP session.

SnortSam consists of two main parts. The first part is a Snort output plug-in that is installed on the Snort sensor. It responds send block request to SnortSam intelligent agent. Another part is an intelligent agent that runs on the firewall, or a host near the firewall. The intelligent agent which is responsible for interaction with the firewall or other network device dynamically blocks IP addresses that Snort has detected an attack. Snort communicates to the agent using the output plug-in in a secure way.

The first part that running with Snort is a Snort output plug-in, named fwsam. In order to cause Snort to send a block request to SnortSam agent, that agent has to be listed in Snort.conf file.

The statement for that is:

```
output alert_fwsam: <SnortSam Station>:<port>/<password>
```

The rules must be modified for triggering by alert and sending block request to the SnortSam agent. This is done by adding the following statement to the rules file.

```
Fwsam: who[how], time
```

who: Can be: src, source, dst, dest, destination

how: Optional. Can be: In, out, src, dest, either, both, this, conn,

time: Duration of block in seconds. (Accepts 'days', 'months', 'weeks', 'years', 'minutes', 'seconds', 'hours')

The second part is the SnortSam intelligent agent. When the SnortSam agent that runs on the firewall or a host near the firewall receives a blocking request packet from Snort, it first verifies that the request came from an authorized source. It then decrypts the request packet using predefined key. If successful, which means that if the passwords or keys of the Snort sensor and the SnortSam agent match, the agent accepts it as a valid request, or discards otherwise? Subsequently, SnortSam will figure out the IP address of the host against the Snort rule.

SnortSam then checks if this IP address is in a white-list. A white-list is a list of IP addresses that will never be blocked. Then it checks if the duration of the block that the Snort sensor requested should be overridden with default duration. Finally it sends a blocking request to the firewall host it resides on or a network device (such as a router). This block can be performed either by sending a packet to the OPSEC port for SAM (Suspicious Activity Monitor, port 18183), or by launching the SnortSam plug-in. (SnortSam, 2007)

2.6.3 Snort-Inline

Snort-Inline IPS is basically a modified version of Snort that accepts packets from IPTable and IPFW via libipq (Linux) or diverts sockets (FreeBSD), instead of libpcap. It then uses new rule types (drop, sdrop, reject) to tell IPTable/IPFW whether the packet should be dropped, rejected, modified, or allowed to pass based on a Snort rule set. Think of this as an IPS that uses existing IDS signatures to make decisions on packets that traverse Snort-Inline.

Figure 2-4 shows the mechanism of how packets are processed in Snort-Inline. First, Netfilter queues packets to Snort-Inline in the user space with the help of the ip_queue kernel module and libipq. Then, if a packet matches a Snort_Inline attack signature, it is tagged by libipq and comes back to Netfilter where it is dropped.

NetFilter is a Linux kernel module available since the kernel version 2.4. It provides three main functionalities, the first function is packet filtering - accepts or drops packets, the second is NAT - changes the source or destination IP address of network packets and the last is packet mangling - modifies packets.

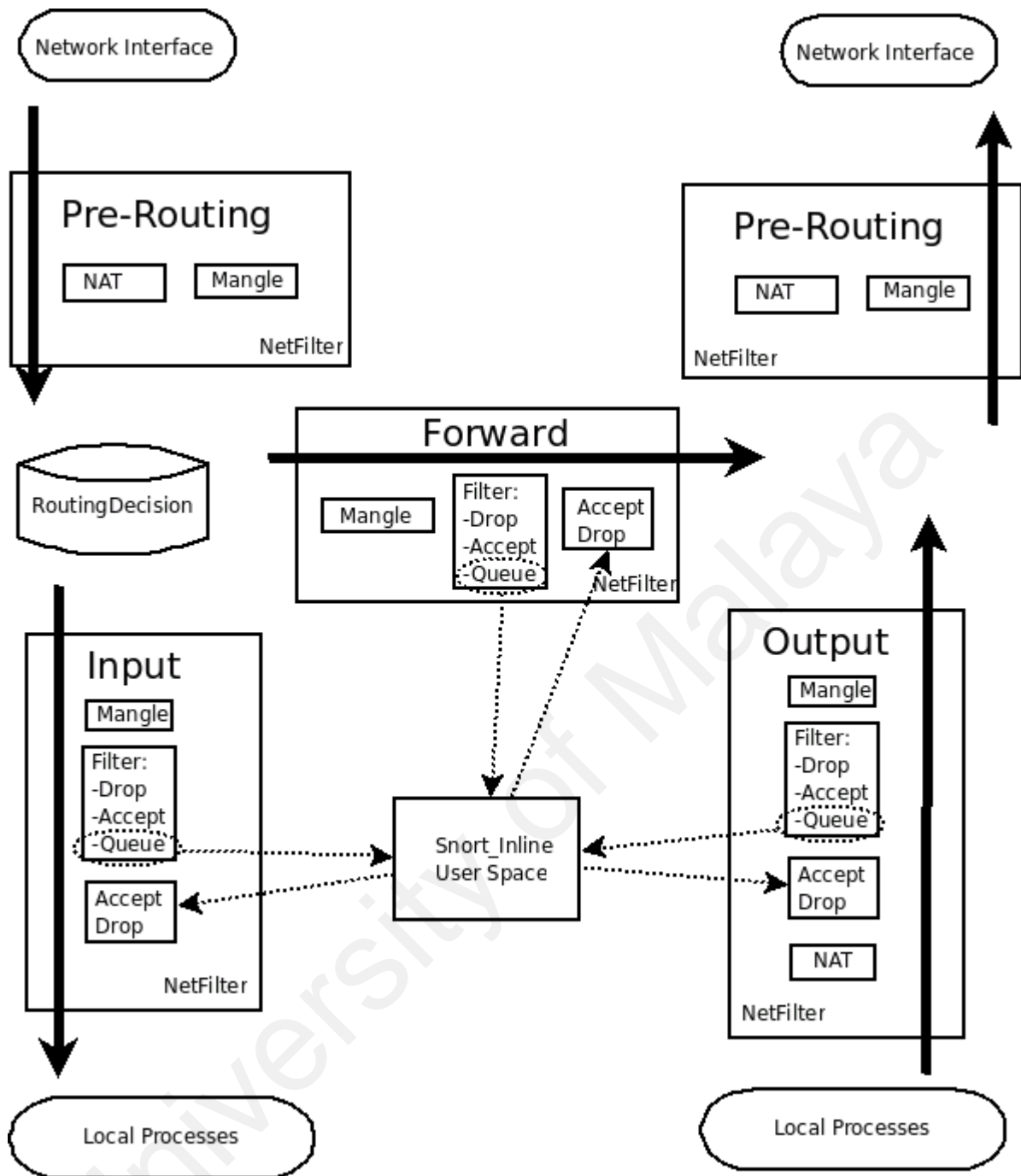


Figure 2-4 Process of Snort-Inline

There are three major modes of Snort-Inline; the first mode is drop mode, which a packet is dropped if it matches an attack signature. Three options are available in this mode. The first is drop, which drops a packet, sends a reset back to the host, and logs the event. The second mode is sdrops, which drops a packet without sending a reset back to the host. The last mode is ignore, which drops a packet, sends a reset back to the host, and does not log the event. Additionally, there is a minor mode - replacing mode, in which a packet is modified if it matches an attack

signature.

2.7 Comparison of Existing IDS and IPS

Table 2-4 depicts the comparison of existing IDS and IPS.

	Snort	Snort-Inline	Snort+SnortSam+Cisco Router (SSCR)
Software Type	IDS	IPS	IPS
Need runs in Gateway?	No	Yes	No
Able to communicate with other network security hardware?	No	No	Yes
Intelligent analyze captured packages?	No	No	No
Support firewall?	No	Only IPTable and IPFW	Yes
Intelligent block intruder's IP?	No	No	No
Support NAT/PAT?	No	No	No
Need predefined ACL file?	---	---	1 file/router
Need predefined interface name?	---	---	Yes
Need predefined ACL name?	---	---	Yes
Connection when implement block	---	---	1 connection / router

Table 2-4 Comparison of some Existing IDS and IPS

For Snort, it is a passive system that detects a potential security breach, logs the information, and signals an alert on the console and or the owner. However, Snort is just an intrusion detector; it can only detect intruder's action and trigger an alert. Snort has no any active response features because Snort is not designed to sit inline with traffic flows and prevent attacks in real-time, i.e. Snort is just an IDS, which lacks the active response feature. It cannot be used as an IPS. System administrators must monitor the alerts and block attack manually.

For Snort-Inline, this IPS must run in gateway for performing block request since it can only communicate with local firewall applications. Additionally, Snort-Inline only supports two kinds of firewalls - IPTable and IPFW.

For SnortSam, the first disadvantage is that although SnortSam supports many firewalls and

adds ACL to Cisco routers; it can only apply ACL to predefined interface. Secondly, SnortSam cannot automatically decide what kind of ACL should be applied to which router or which interface. SnortSam will try to add the block ACL to all predefined routers even if it is not necessary. Additionally, SnortSam will send block request to every predefined firewall even the attack packets are impossible to reach those firewalls. Finally, SnortSam cannot be applied to a network with NAT/PAT technology as it cannot find real attacker's IP in such NAT/PAT environment.

2.8 Problems

This section identifies major problems found in existing IDS AND IPS.

The most serious problem of IDS AND IPS is false positive (treats normal traffic as intrusion) and false negative (fails to indicate intrusion action). Some approaches are already applied to make detection more accurate, such as continuously rule-update, more accurate pattern/string matching, multiple detection method applied, and applying neural network and fuzzy logic.

For IDS, detection efficiency is another problem. If IDS cannot process all data in almost real-time, it is useless. Some researchers already provide many approaches/mechanisms to increase efficiency in recent years. These include creating a more effective detection engine and finding more effective detection method.

In addition, as many existing IDS AND IPS run with traditional software-based firewalls, they almost never or seldom communicate with kernel network devices, such as switches and routers. It is the main factor that affects efficiency. Because switches and routers are kernel devices in the network environment, heavy useless malicious traffics can be decreased if

intrusion traffics are blocked in the switches or routers.

The last problem of current IDS AND IPS is the lack of ability to detect and block intrusion behaviors coming from internal. There are about 70% attacks originated from inside (FBI, 2006), which means IDS AND IPS should pay more attentions to detect and prevent internal attacks.

So, this thesis will propose a new automated intrusion prevention mechanism to solve some of the problems mentioned.

2.9 Summary

This chapter offers an understanding of IDT/IDS, and IPT/IPS. After reviewing current IDT/IDS and IPT/IPS, some issues and challenges are found. Several problems and disadvantages of existing IDS AND IPS are also discussed. Most IDS AND IPS in market is running with traditional software-based firewalls, many of them almost never interact with network devices such as switches and routers. This research will try to integrate software and hardware to enhance network security and add intelligent features.

CHAPTER 3 Analysis And Design

This chapter includes three sections. The first section analyzes the purpose and main conception of this research and describes the architecture of the proposed automated intrusion prevention mechanism (AIPM). The second section discusses the design and key components of AIPM. The third section describes the mechanism.

3.1 Analysis of AIPM

The task of an intrusion prevention system (IPS) is to detect malicious traffic and block it. Obviously, a typical IPS includes two main components. One is a network traffic sniffer, also called IDS, which captures network traffic packets and identify whether it is malicious traffic. Another is prevention processing, which makes decision and block malicious traffics.

The general conception on AIPM is described in the following. The target of this proposed mechanism is to automatically monitor network and block malicious traffic in Cisco routers on different network environments. This mechanism can be roughly divided to 3 steps. Firstly, the network sniffer monitors network traffic and identifies whether it is an intrusion. An alarm will be triggered when any malicious traffic is found and a block request to the prevention processing component is sent. Additionally, information of the malicious traffic will also be sent to the prevention processing component. Secondly, the prevention processing component analyzes the information of malicious traffic and determines the first router that will establish connection. This component then analyzes downloaded router's configuration after establishing a connection and determining the correct router and interface. Finally, correspondent access list will be added to the interface. Figure 3-1 depicts the use case diagram of AIPM.

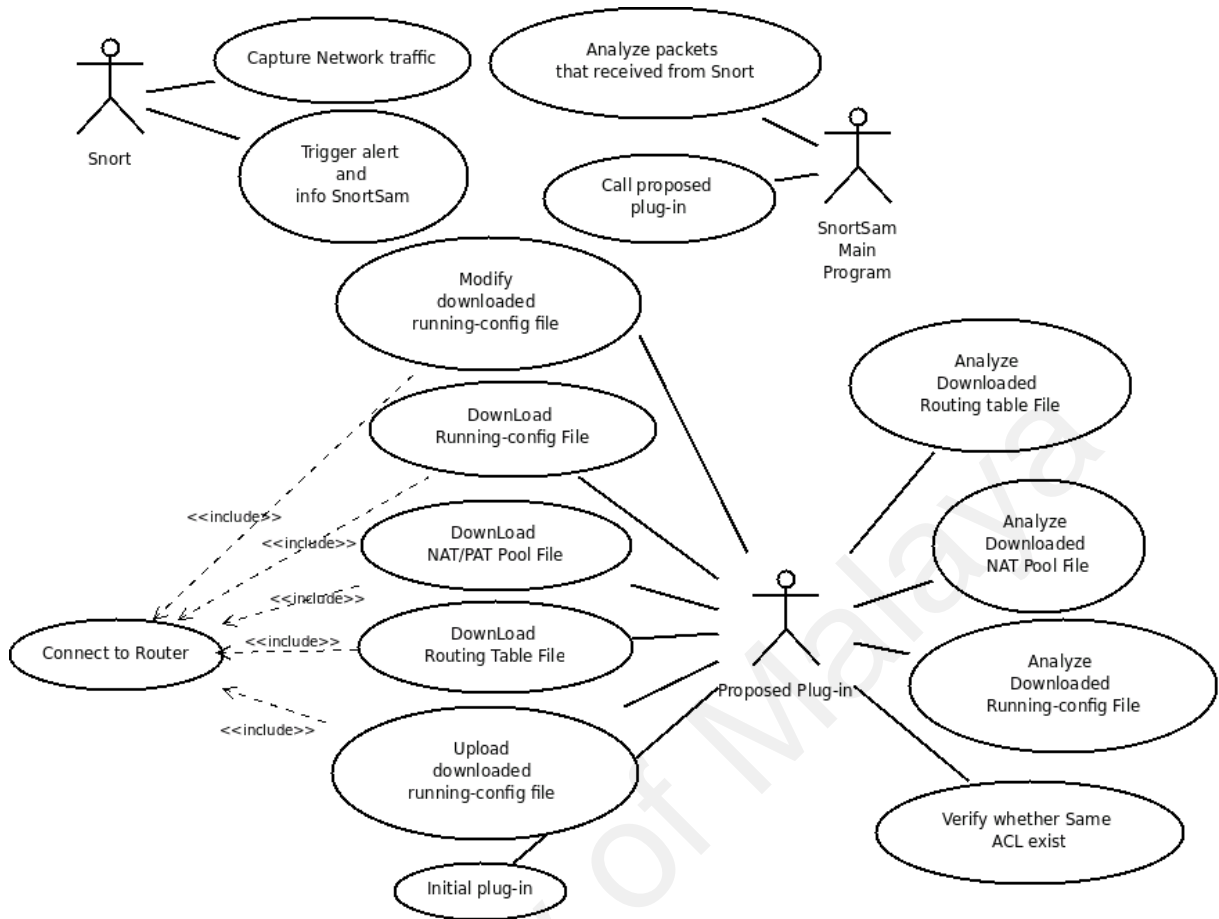


Figure 3-1 Use Case Diagram of AIPM

AIPM can be logically divided into three main components. The first component is the network traffic sniffer which is in charge of monitoring network and identifying malicious behaviors. The second component is prevention processing which analyzes intrusion message, determines location that should be secured and implements prevention. The last component is connection processing which establishes connections to target routers. Figure 3-2 depicts the three components of AIPM.

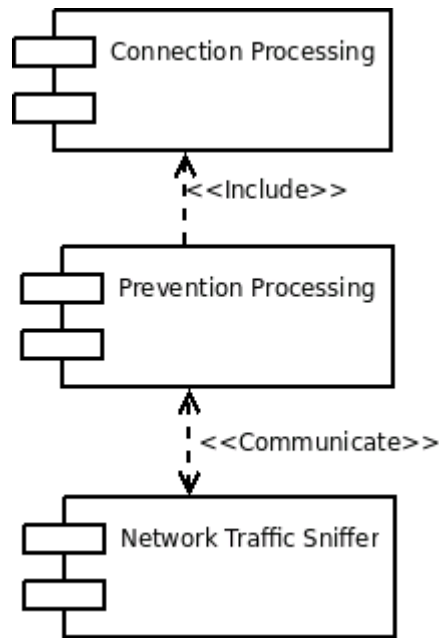


Figure 3-2 Component Diagram of AIPM

3.2 Design of AIPM

According to previous analysis, AIPM must include three functions. The first function is to capture network packets. Snort is chosen to satisfy this function. The second function is to receive block request. SnortSam is chosen to receive block request. The last function is to implement prevention. A new plug-in will be developed in this work to satisfied this function.

Snort is selected to work as a packet sniffer for these four reasons. Firstly, Snort is perhaps the best known open source intrusion detection system available. Snort is currently used in many IDS situations, from small office and home networks to corporate and IT offices worldwide. Secondly, Snort has been ported to a variety of platforms. Thirdly, many third-party applications have been engineered around its use. Finally, Snort is actively maintained.

SnortSam is chosen in this work as an intelligent agent to receive block request because SnortSam has the ability of communicating with Snort and receiving information of captured

malicious traffic. Figure 3-3 shows the architecture of AIPM.

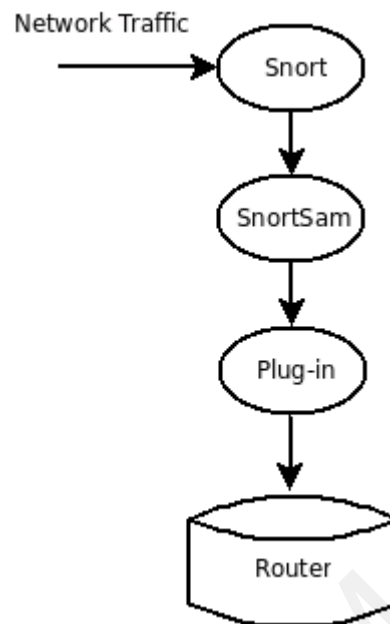


Figure 3-3 Architecture of AIPM

Apart from employing Snort and SnortSam, a new plug-in is developed to implement prevention. The new plug-in is in charge of analysis, connection, and prevention. The plug-in includes five functions. Firstly, the new plug-in downloads running-configure file and routing table file from Cisco routers and intelligently analyzes it. Secondly, the new plug-in can run well in almost all environment (include NAT/PAT and static/dynamic route or others environment). Thirdly, the new plug-in minimizes the synchronization between IPS and Security devices (Cisco routers), only less predefined information are needed. Fourthly, it supports managing multiple interfaces (include sub-interfaces) in Cisco routers without any predefined information. And fifthly, the new plug-in has the capability of automatically determining the target router and target interface that ACL entry should be applied.

The new plug-in implements all prevention functions after receiving information of captured malicious traffic packet from SnortSam. The main functions of this plug-in include

1. Initialize according configuration file

- a) Initialize according snortsam.conf file
 - b) Initialize according /etc/cisconatacl.conf file
2. Communicate with Cisco routers
 - a) Download running-config file
 - b) Download routing table file
 - c) Download NAT/PAT pool file
 - d) Apply new ACL to the target router by uploading modified running-config or sending config commands to the router
3. Analyze downloaded configuration file
 - a) Analyze downloaded running-config file
 - b) Analyzed download routing table file
 - c) Analyzed download NAT/PAT pool file
4. Create new ACL entry
 5. Check whether ACL exists in the target router
 6. Modify downloaded running-config file
 7. Implement block intruder action

Every function should be achieved in one or more functions in the development phase.

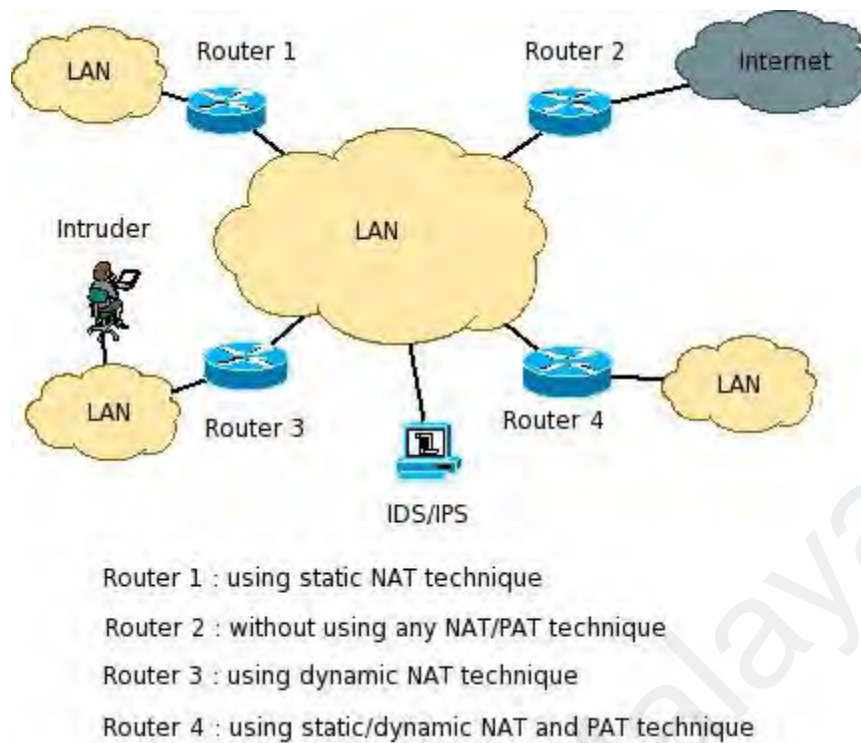


Figure 3-4 Sample of Network Environment

Figure 3-4 shows a sample of network environment that AIPM runs. In particular, one intruder starts the attack from an inside LAN, then the IDS catches the attack packets and finds its match with an intrusion signature. If an attack is found in the network, an alert will be triggered to ask the IPS to block the intruder's IP in one of the routers.

There are three questions that need to be addressed, however. First, what is the real IP of the attacker? Second, which router should be secured? Third, which interface should be secured? Solving these three problems can be challenging due to different network environment.

3.2.1 Key Components of AIPM

The new plug-in, namely CiscoNATAACL, includes these components:

1. CiscoNATAACLInit:
 - A component that initializes some arguments and parameters

2. CiscoNATACLParse:
 - A component that parses the CiscoNATAACL statement in SnortSam.conf file
 - Gets IP, telnet/enable password, name of ACL configuration file, and more information of the router
3. CiscoNATACLBlock:
 - A component that will be performed after SnortSam receives a block request and tries to make a block IP action
 - Enter point of the plug-in
4. CISCONATACLRRunningDownload:
 - A component that downloads running-configure file and routing table from routers to the local TFTP direction
5. CiscoNATACLRRunningParse:
 - A component that opens downloaded running-configure file and routing table, then reads the file line by line to memory
6. CiscoNATACLRRunningParseLine:
 - A component that parses a line content from downloaded running-configure file
 - Analyzes the running-configure file and gets all information needed.
7. CiscoNATACLPoolDownload:
 - A component that downloads the NAT pool from a router to the TFTP direction.
8. CiscoNATACLPoolParse:
 - A component that opens downloaded pool file and reads data line by line for analyzing purposes
9. CiscoNATACLPoolParseLine:
 - A component that parses a line content of downloaded pool file
 - Compares the block IP with data from the NAT pool and decides the real block

IP

10. CiscoNATAACLRouteDownload:

- A component that downloads the IP route table record file from routers

11. CiscoNATAACLRouteParse:

- A component that opens downloaded IP route table record file and reads data line by line for analyzing purposes

12. CiscoNATAACLRouteParseLine:

- A component that parses a line content of downloaded IP route table record file
- Compares the block IP with IP route records and finds the correct interface or correct routers

13. CiscoNATAACLCreateACL:

- A component that creates ACL entry and applies it to router if there is no any ACLs exist in the correct interface

14. CiscoNATAACLCheck:

- A component that checks whether the new ACL has already existed or applied to the router before.

15. CiscoNATAACLRunningModify:

- A component that handles modifying downloaded running-configure file by adding new ACLs to the file

16. CiscoNATAACLsendreceive:

- A component that handles communication with routers, such as downloading/uploading the running-configure file, telnet and perform commands

17. GetIPFromPrefix

- A routine that gets IP addresses from the prefix form.

18. cisonatconfparsefile

- A routine that parses /etc/cisconatacl.conf file.
- Gets gateway router's information and tftp server information

19. cisonatconfparseline

- A routine that parses a line of the /etc/cisconatacl.conf file
- Gets gateway router's information and TFTP server information

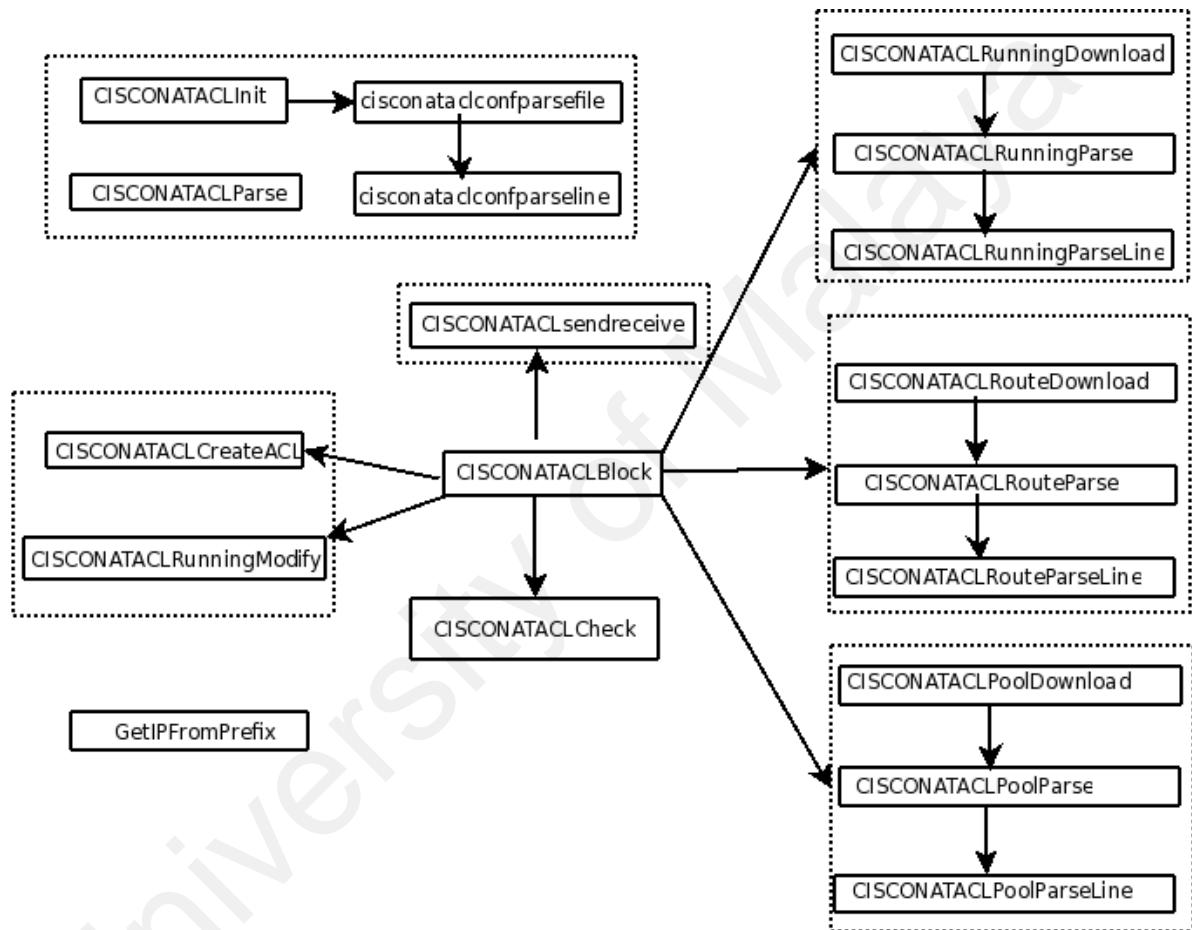


Figure 3-5 Key Components in AIPM

Figure 3-5 depicts the key components of AIPM and describes their relationships. These key components can be roughly grouped into seven main parts.

The first main part is “Running-config file processing”. This part includes 3 components - CISCONATACLRunningDownload, CISCONATACLRunningParse, and

CISCONATACLRRunningParseLine. This part downloads and analyzes the running-config file from Cisco routers. The downloading purposes are achieved by sending this command: “show running-config | tee <tftp_ip_address>”, where <tftp_ip_address> is the IP address of the TFTP server. Following is the sample code:

```
snprintf(msg, sizeof(msg)-1, " show running-config | tee  %s", tftpAddressRun);
if( CISCONATACLsendreceive(cisconataclsocket, msg, "#") )
{
    closesocket(cisconataclsocket);
    ErrorFlag=1;
    return;
}
```

The second main part is “Routing-table file processing”. This part is composed with CISCONATACLRRouteDownload, CISCONATACLRRouteParse, GetIPFromPrefix and CISCONATACLRRouteParseLine. This part downloads and analyzes the routing table file from the Cisco router. The routing table is retrieved by this command: “show ip route | tee <tftp_ip_address>”, where <tftp_ip_address> is the IP address of the TFTP server. GetIPFromPrefix is a routine that gets the IP address from the prefix form. The sample code is showed bellow.

```
snprintf(msg, sizeof(msg)-1, "show ip route | tee %s", tftpAddressRouting);
if( CISCONATACLsendreceive(cisconataclsocket, msg, "#") )
{
    closesocket(cisconataclsocket);
    ErrorFlag=1;
    return;
}
```

The third main part is “NAT/PAT Pool file processing”, which is composed of CISCONATACLPoolDownload, CISCONATACLPoolParse and CISCONATACLPoolParseLine. This part downloads and analyzes the NAT/PAT pool file from the Cisco router. The NAT/PAT pool is downloaded from the Cisco router via this

command: “show ip nat translate | tee <tftp_ip_address>”, where <tftp_ip_address> is the IP address of the TFTP server.

```
snprintf(msg, sizeof(msg)-1, "show ip nat translate | tee %s", tftpAddressPool);
if( CISCONATAACLsendreceive(cisconataaclsocket, msg, "#") )
{
    closesocket(cisconataaclsocket);
    ErrorFlag=1;
    return;
}
```

The fourth main part is “Initial information processing”, which includes CISCONATAACLInit, CISCONATAACLParse, ciscoconfparsefile, and ciscoconfparseline to read and analyze AIPM configurations. Initially, ciscoconfparsefile and ciscoconfparseline was not created in the design phase; all information are predefined and stored in a MYSQL database. However, any changes in the router’s configuration during running time will render the information in the database to be outdated. Hence, ciscoconfparsefile and ciscoconfparseline are designed to process less predefined information. This mechanism helps minimize the synchronization between IDS AND IPS and Cisco routers.

The CISCONATAACLParse function parses the IP address, telnet/enable password, ACL configure file, etc. That information will be stored in a structure – CISCOACLDATA, which is defined in plug-in.

```
typedef struct _ciscoacldata          /* List of ciscoacl routers */
{
    struct in_addr    ip;
    charusername[CISCOACLPWLEN+2];
    chartelnetpw[CISCOACLPWLEN+2];
    charenablepw[CISCOACLPWLEN+2];
    characlfile[CISCOACLFILELEN+2];
} CISCOACLDATA;
```

At the same time, the information is stored in a linked list

```
typedef struct _ciscoaclrouter /*list of router information*/
```

```

    {
        struct in_addr ip;
        char username[CISCONATACLPWLEN+2];
        char telnetpw[CISCONATACLPWLEN+2];
        char enablepw[CISCONATACLPWLEN+2];
        struct _ciscoaclrouter *next;
    } CISCOACLROUTER;

```

At the initialization phase, configure file “/etc/cisconatacl.conf” is read and parsed. Predefined gateway router's information (IP address, telnet password etc.) and the TFTP server information (Running-config file, pool file and routing file) are loaded and analyzed.

The fifth main part is “Add ACL processing”. This part includes CISCONATACLCreateACL and CISCONATACLRRunningModify. It is in charge of adding ACL entries to Cisco routers with two methods. CISCONATACLCreateACL connects to Cisco router and sends a serial config commands to create new ACLs to be applied to target router. CISCONATACLRRunningModify modifies the downloaded running-config file and adds new ACLs in appropriate positions.

The sixth main part is “connection processing”, which consists of CISCONATACLsendreceive to communicate with Cisco routers. All communication between IPS and Cisco routers must call this function. Below is the sample code to establish a connection to a Cisco router.

```

/* do simple authentication with password */
printf("\ndo simple authentication with password");
if( CISCONATACLsendreceive(cisconataclsocket, "", "Password: ") )
    {
        closesocket(cisconataclsocket);
        ErrorFlag=1;
        return;
    }
if( CISCONATACLsendreceive(cisconataclsocket, cisconataclptr_temp->telnetpw, ">") )
    {
        closesocket(cisconataclsocket);
        ErrorFlag=1;
        return;
    }

```

```

    }
    if( CISCONATACLsendreceive(cisconataclsocket, "enable", "Password: ") )
    {
        closesocket(cisconataclsocket);
        ErrorFlag=1;
        return;
    }
    if( CISCONATACLsendreceive(cisconataclsocket, cisconataclptr_temp->enablepw, "#")
)
    {
        closesocket(cisconataclsocket);
        ErrorFlag=1;
        return;
    }
    /* A Telnet connection has been established */
    printf("\nalready connected to cisco router\n");

```

The last part is “prevention process”. This part includes CISCONATACLBlock and CISCONATACLCheck. CISCONATACLBlock is in charge of preventing malicious traffic; it is the core component in AIPM. CISCONATACLCheck is responsible for checking whether the same ACL exists in the target interface.

3.3 AIPM

This section attempts to address these questions: “how to decide which router should be secured?”, and “how to decide which interface should be secured?”

Essentially, AIPM is divided into 5 layers. Figure 3-6 depicts these layers. The first layer is Initialize Layer, in which all necessary data are initialized and structured. IDS and IPS also establish connections in this layer. The second layer is Capturing Layer, in which IDS (Snort in AIPM) captures attacker’s packets. The third layer is Analysis and Determination Layer – the most important layer in the whole system. This layer analyzes data retrieved from security devices and determines the target router, the target interface, an ACL name, and so on. The fourth layer is Blocking Layer. This layer implements a block request to target network security

device. Connect Layer, the last layer, implements all network connections.

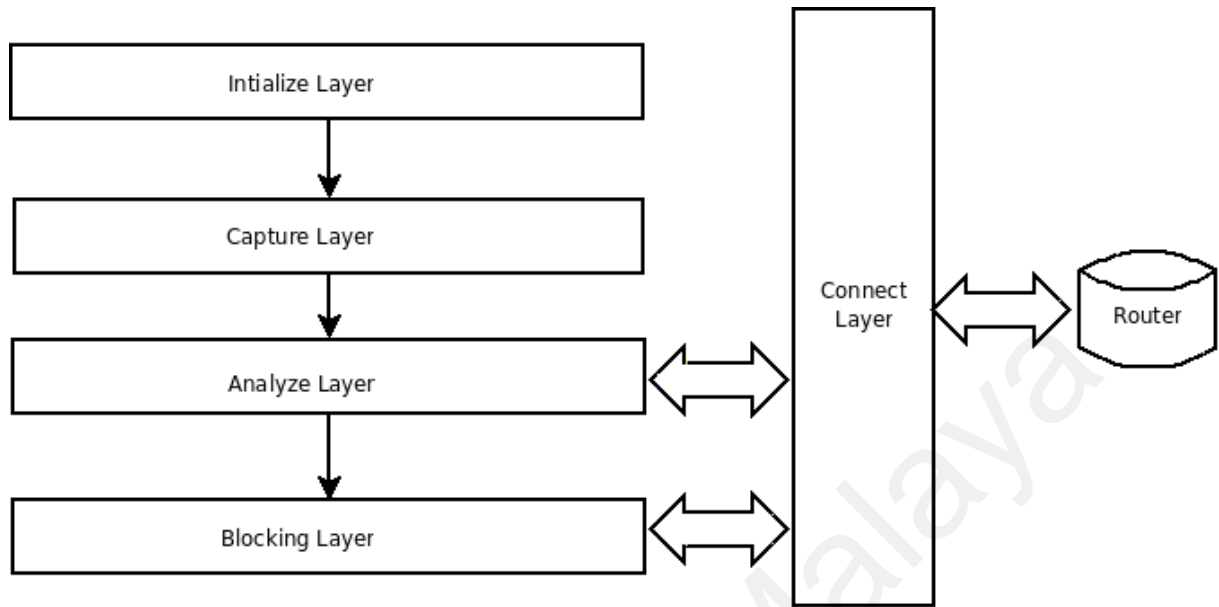


Figure 3-6 Layers of AIPM

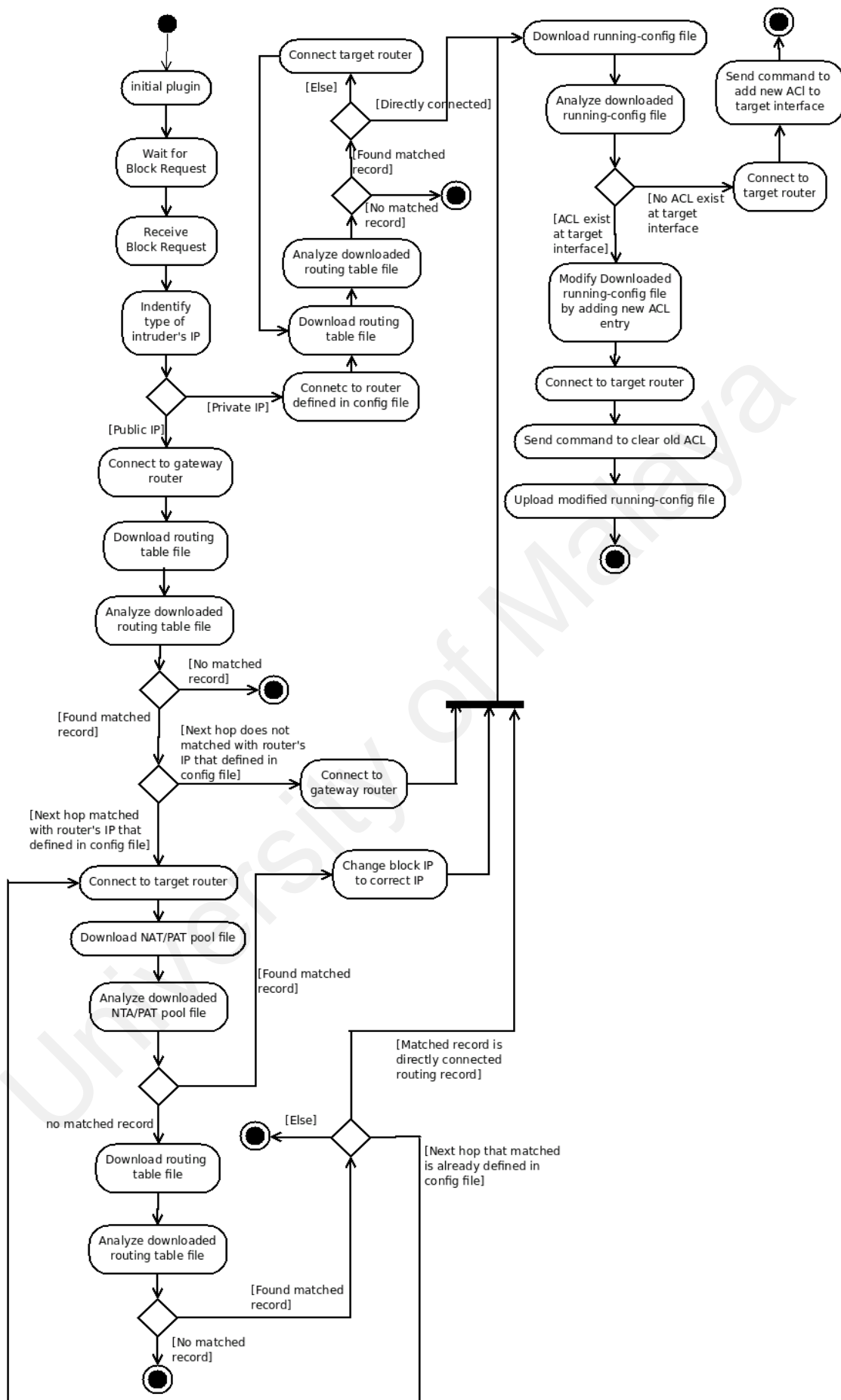


Figure 3-7 Active Diagram of CiscoNATAACL Plug-In

Figure 3-7 shows the active diagram of the CiscoNATAACL plug-in. When SnortSam is started, SnortSam opens and parses the configuration file—“SnortSam.conf”. As soon as the configure line of CiscoNATAACL is parsed, the CiscoNATAACLParse function of the plug-in will be called.

At the initialization phase, configure file “/etc/cisconataacl.conf” is read and parsed. Predefined gateway router's information (IP address, telnet password, etc.) and TFTP server information (Running-config file, pool file and routing file) are loaded and analyzed.

After all information are read and initialized, the plug-in is ready to receive blocking requests. When SnortSam receives a block request from a valid Snort sensor, the function of CiscoNATAACLBlock will be called. A structure about block packet is passed to the plug-in at the same time. The structure is defined in SnortSam.h file.

```
Typedef struct _blockinfo /* Block info structure */
{
    unsigned long sig_id; /* Snort Signature ID (for logging/presentation) */
    unsigned long blockip; /* IP to be blocked */
    unsigned long peerip; /* Peer IP (if connection) */
    time_t duration; /* Duration of block */
    time_t blocktime; /* Time when block started */
    unsigned short port; /* Port (if connection) */
    unsigned short proto; /* Protocol (if connection) */
    unsigned short mode; /* Blocking mode(src, dst, connection) */
    short block; /* block or unblock flag--this flag is dynamically changed */
} BLOCKINFO;
```

Intrusion IP can be divided into three types: the first type is a private IP used in a LAN, the second type is inside public IP which is used in a LAN, and the last type is outside public IP which is used in the Internet. Following sections will discuss AIPM according to the type of intrusion IP.

3.3.1 Intrusion IP Is Private IP

When intrusion IP is a private IP, AIPM will connect to the router according to the configuration in `/etc/snortsam.conf` and will download the routing table file from the connected router. The downloaded routing table will be analyzed by AIPM to try to find a record of directly connected router by comparing with the intrusion IP. If no such record exists, AIPM will terminate. If a routing record matches the intrusion IP but it does not correspond to a directly connected router, AIPM will try to connect to the next router and repeat the same process until a match of directly connected router is found. When this match is found from a router, AIPM will determine how to block intrusion by downloading the running-config file and analyzing it. If there is no ACL exists at the target interface, AIPM will establish a connection and send a command to add new ACLs. Otherwise, AIPM will modify the downloaded running-config file and add an ACL entry. AIPM then establishes a connection to the target router and send a command to clean the old ACL. Finally, AIPM uploads the modified running-config file back to the target router.

3.3.2 Intrusion IP Is Inside Public IP

When intrusion IP is an inside public IP, AIPM will connect to the gateway router that is already defined in `/etc/cisconatacl.conf`. This piece of information needs to be pre-defined as AIPM does not know whether the IP is used in a LAN or WAN. AIPM will download the routing table file from the gateway router and analyze it. If there is no record matched with intrusion IP, AIPM is stopped. Otherwise, if the IP of the next hop is found and it matches with the IP of a router that is defined in `/etc/snortsam.conf`, AIPM will connect to the target router according the value of next hop. The NAT/PAT pool will be downloaded from the target router for analysis purposes. If a matched IP is found from the pool, which means the intrusion IP

belongs to NAT, then the intrusion IP is changed to a real IP. The subsequent process will continue as that of the process when the intrusion IP is a private IP. If there is no matched record in the NAT/PAT pool, AIPM will download routing the table file from the target router and analyze it. If there is no matched record found, AIPM exits. Otherwise, if the matched record corresponds to directly connected router, following processing is same with that when the intrusion IP is a private IP, or AIPM will connect to another router according to the next hop value. The previous process is repeated until a directly connected router is found.

3.3.3 Intrusion IP Is Outside Public IP

When the intrusion IP is an outside public IP, AIPM will connect to the gateway router that is already defined in `/etc/cisconatacl.conf`. Again, AIPM will download the routing table file from the gateway router and analyze it. For a matched record found, the value of next hop is not defined in the configuration file since the intrusion IP is outside public IP. AIPM will establish a new connection to the gateway router. Subsequent process will continue as that when the intrusion IP is a private IP.

3.4 Summary

This chapter provides an overview of how the system is analyzed and how the system runs; a detailed analysis of AIPM is covered too. The whole system is analyzed and developed in real network environment. Building on top of the SnortSam and Snort, a new plug-in is added to improve functions. AIPM integrates Snort, SnortSam, and network security devices to become an IDS AND IPS.

Chapter 4 System Development and Testing

This chapter describes key components of AIPM, followed by functions testing. Function testing is important to ensure that the code and protocol requirements are implemented as expected. The testing carried out herein is classified mainly into unit testing and integrating testing. Unit testing verified the correctness of individual modular functions, while the integration testing combines all functions to test the system as a whole.

4.1 Development Environment

Table 4-1 depicts development environment.

	Development Environment
Operating System	GNU/Linux Slackware11.0
Programming Language	ANSI C
Compiler	GCC v4.12
Model of Cisco Router	Cisco 2600 serials
Model of Cisco Switch	Cisco 2950 serials
Computer	Dell 2600 serials
RAM	512MB DDR

Table 4-1 Environment of Development

AIPM is developed by using ANSI C language in GNU/Linux operating system. A GNU/Linux distribution - Slackware is chosen as the development platform. Five Cisco 2600 routers and one Cisco switch are used in developing phase. All computers are Dell 2600 serials.

4.2 Unit Test

The key components of AIPM are classified into seven main parts (see Section 3.2.1). These parts were tested individually. All units run in individual programs are fed with necessary data to test and compare with expected results. If the testing results match with expected results, the components pass the unit test.

4.2.1 Unit Test 1 - Initial Information Processing

Unit Test 1 tests “Initial information processing” which includes four components—CISCONATACLInit, CISCONATACLParse, ciscoconfparsefile, and ciscoconfparseline. The responsibility of these components is to read and analyze information in the system configuration file.

4.2.1.1 CISCONATACLInit, ciscoconfparsefile and ciscoconfparseline

CISCONATACLInit function is the first function called by the main() function of SnortSam when it encounters a plug-in in the configuration file. The function returns either TRUE or FALSE, indicating a successful or unsuccessful initialization. If the function returns FALSE, SnortSam will disable the plug-in. The parameter is a pointer to the first element in the device/parameter list configuration file. This function has two missions; the first mission is to allocate memory for four link lists. The codes are showed below:

```
RouterData=malloc(sizeof(CISCOACLROUTER));
RouterData->next=NULL;
CurrentRouter=malloc(sizeof(CISCOACLROUTER));
CurrentRouter=RouterData;
ACLData=malloc(sizeof(CISCOACL));
ACLData->next=NULL;
```

```
CurrentACL=malloc(sizeof(CISCOACL));  
CurrentACL=ACLData;
```

The second is to process `/etc/cisconatacl.conf` file by calling function `cisconatconfparsefile()`.

The `cisconatacl.conf` file includes some predefined information of the gateway router.

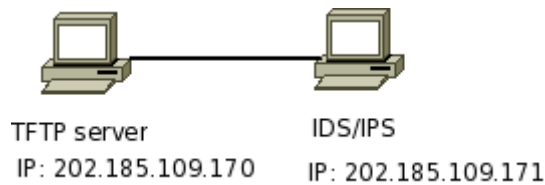


Figure 4-1 Testing Environment of Unit Test 1

Figure 4-1 shows the testing environment of Unit Test 1. In this test, `CISCONATACLInit` takes responsibility for initializing data and calling `cisconatconfparsefile` to read `/etc/cisconatacl.conf` file. `Cisconatconfparseline` take responsibility for analyzing `/etc/cisconatacl.conf` file. A sample of the config file is showed below.

```
#Gateway router's information  
#sample:  
GatewayRouterIP 192.168.0.210  
GatewayRouterTelnetPW cisco  
GatewayRouterEnablePW cisco  
OutsidePublicInterfaceName f0.101  
  
#Tftp server information  
#Sample:  
tftpAddressRun tftp://202.185.109.170/run  
tftpAddressPool tftp://202.185.109.170/pool  
tftpAddressRouting tftp://202.185.109.170/routing
```

Table 4-2 depicts the expected output in this unit test.

<p>Expected result</p> <ol style="list-style-type: none"> 1. Successfully open and read config file 2. Successfully analyze config file and assign values to arguments 3. Print all information that receive from /etc/cisconatacl.conf, output should be same with config file

Table 4-2 Expected Output of Unit Test 1.1

```

Testing result:
Initializing plugin 'cisconatacl'...
Parsing config file /etc/cisconatacl.conf...
the 1 loop
  buf=
  cfgfile=/etc/cisconatacl.conf
  line=1

the 2 loop
  buf=
  cfgfile=/etc/cisconatacl.conf
  line=2

the 3 loop
  buf=GatewayRouterIP 202.185.109.173
  cfgfile=/etc/cisconatacl.conf
  line=3
start call parseline function
Start parse line.....
the hole arg is: GatewayRouterIP 202.185.109.173
finishe 1 line parse
(.....overleap.....)
>>>>>>>>>>Start Gateway router's Information<<<<<<<<<<<<<<<<<<<<<<<<<
the Gateway's ip is 202.185.109.173
the GatewayRouterTelnetPW is cisco
the GatewayRouterEnablePW is cisco
the OutsidePublicInterfaceName is f0.101
the tftfAddressRun is tftp://202.185.109.170/run
the tftfAddressPool is tftp://202.185.109.170/pool
the tftfAddressRouting is tftp://202.185.109.170/routing
>>>>>>>>>>End Gateway router's Information<<<<<<<<<<<<<<<<<<<<<<

```

Table 4-3 Testing Result of Unit 1.1

Table 4-3 shows the testing result, which verifies that the components has successfully opened/read config file and analyzed the file, and successfully assigned values to arguments. Since the output is same with that defined in config file, the unit test passed.

4.2.1.2 CISCONATACLParse

The CISCONATACLParse component of “Initial Information Processing” parses CiscoNATAACL statements in the snortsam.conf file. One example is showed below.

```
cisconatacl 202.185.109.171 cisco1 cisco2
cisconatacl 202.185.109.172 cisco1 cisco2
cisconatacl 202.185.109.173 cisco1 cisco2
```

“cisconatacl” tells SnortSam to use the plug-in; 202.185.109.171 is the IP of the target router; the string cisco1 is the telnet password; cisco2 is the enable password. This unit test used the same environment as the previous unit test. The expected output is showed in Table 4-4.

Expected result:

1. Successfully link cisconatacl plug-in
2. Successfully read data from snortsam.conf
3. Output should be same with defined in snortsam.conf

Table 4-4 Expected Output of Unit Test 1.2

Testing result:

```
Parsing config file /etc/snortsam.conf...
Linking plugin 'cisconatacl'...
(.....overleap.....)
Plugin Parsing...
the cisconataclip.s_addr arg is: -1418872374
the IP arg is: 202.185.109.171
the RouterData->telnetpw arg is: cisco1
the RouterData->enablepw arg is: cisco2
the RouterCounter is 1
Plugin Parsing...

the cisconataclip.s_addr arg is: -1402095158
the IP arg is: 202.185.109.172
the RouterData->telnetpw arg is: cisco1
the RouterData->enablepw arg is: cisco2
the RouterCounter is 2
Plugin Parsing...

the cisconataclip.s_addr arg is: -1385317942
the IP arg is: 202.185.109.173
the RouterData->telnetpw arg is: cisco1
```

```
the RouterData->enablepw arg is: cisco2
the RouterCounter is 3
```

Table 4-5 Testing Result of Unit Test 1.2

Table 4-5 shows the test result of Unit Test 2. This result shows that the component has successfully linked cisconatacl plug-in and read data from snortsam.conf. The output of the unit test is the same with that defined in snortsam.conf; the unit test passed.

4.2.2 Unit Test 2 - Running-Config File Processing

“Running-config file processing” includes three functions: CISCONATACLRunningDownload, CISCONATACLRunningParse, and CISCONATACLRunningParseLine. Testing of “Running-config file processing” will be divided into two unit tests. One is to download running-config file - CISCONATACLRunningDownload testing, another is to parse downloaded running-config file — CISCONATACLRunningParse and CISCONATACLRunningParseLine testing. Figure 4-2 shows the testing network environment.

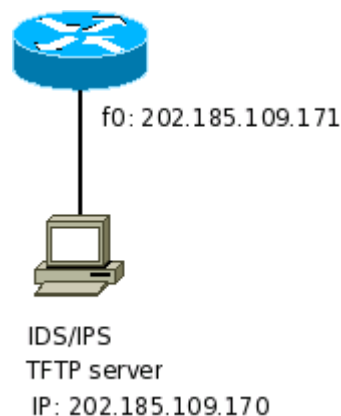


Figure 4-2 Testing Environment of Unit Test 2

4.2.2.1 CISCONATACLRRunningDownload

This function establishes connections to target routers and downloads the respective running-config file. The expected output of this unit test is showed in Table 4-6.

Expected result

1. Successfully connect to target router
2. Successfully download running-config file and save to target TFTP server

Table 4-6 Expected Result of Unit Test 2.1

Testing output:

```
=====Start download Running- config file=====
Finished creat and bind socket, then try to connect
Connected to=202.185.109.171
Connected to ciskonatacl at 202.185.109.171.
do simple authentication with password

Receiving: --Password: --

Sending:cisco1

Receiving: -->--

Sending:enable

Receiving: --Password: --

Sending:cisco2

Receiving: --#--

already connected to cisco router

Sending:show running-config | tee tftp:// 202.185.109.170/run

Receiving: --#--
```

Table 4-7 Testing Result of Unit Test 2.1

The result of this unit testing output is showed in Table 4-7. It is shown that the function has successfully connected to the target router and entered into the TFTP folder. A file named “run” is found, in which the content of this file is identical to the running-config file in the target router. This means that the function has successfully downloaded running-config file and saved

the file to the target TFTP server. This unit test passed.

4.2.2.2 CISCONATACLRRunningParse and CISCONATACLRRunningParseLine

These two functions open/read information from the downloaded running-config file and analyze it. The testing environment is the same with the CISCONATACLRRunningDownload test. Table 4-8 shows the expected result. The feeding data is the downloaded running-config file in the previous unit test.

Expected result:

- 1.Successfully open and read downloaded running- config file
- 2.Correctly analyze downloaded running-config file and assign values to arguments

Table 4-8 Expected Result of Unit Test 2.2

Testing result:

```
=====Parsing running-config file /tftpboot/run...=====
(.....overleap.....)
This is 10 loop
  buf=interface FastEthernet0
  cfgfile=/tftpboot/run
  line=39
start call parseline function
Start parse line.....
the whole arg is: interface FastEthernet0
finish 10 line parse
The correct interface name is : FastEthernet0
This is 11 loop
  buf=ip address 202.185.109.171 255.255.255.0
  cfgfile=/tftpboot/run
  line=41
start call parseline function
Start parse line.....
the whole arg is: ip address 202.185.109.171 255.255.255.0
finish 11 line parse
(.....overleap.....)
```

Table 4-9 Testing Results of Unit Test 2.2

Table 4-9 shows the testing results - these functions have successfully opened/read downloaded running-config file, analyzed data, and has assigned to arguments. This unit test passed.

4.2.3 Unit Test 3 - Routing-Table File Processing

As in Unit Test 2, Unit Test 3 is also divided into two unit tests: one is to download the routing table file (CISCONATACLRoutDownload), and another is to analyze the downloaded routing table file (CISCONATACLRouteparse and CISCONATACLRouteparseline). The testing environment used in this unit test is shown in Figure 4-3.

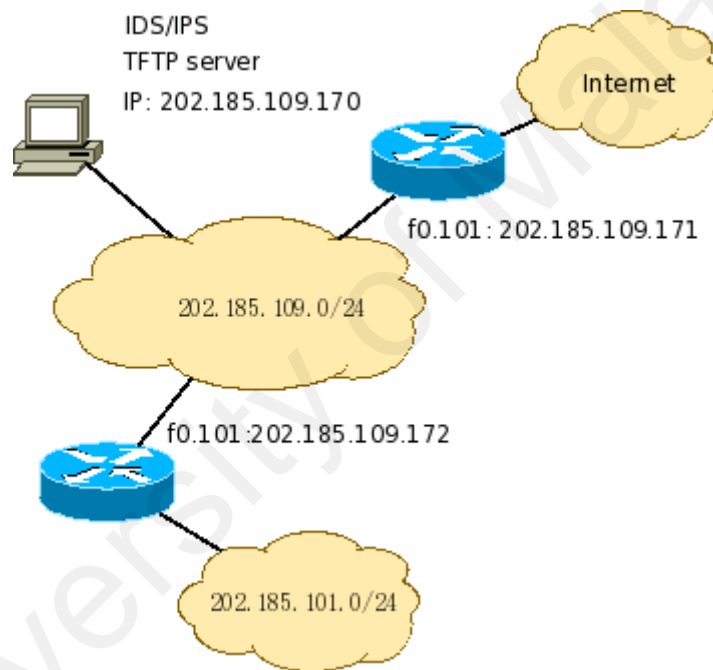


Figure 4-3 Testing Environment of Unit Test 3

4.2.3.1 CISCONATACLRoutDownload

This function takes the responsibility for downloading the routing table file and saving it to a TFTP server. The expected result is shown in Table 4-10.

Expected Result:

1. Successfully connect to target router

2.Successfully download routing table file and save to target TFTP server

Table 4-10 Expected Result of Unit Test 3.1

Testing Result:

```
=====Start download Routing file=====
Connected to cisconatacl at 202.185.109.171.
do simple authentication with password
Receiving: --Password: --

Sending:cisco

Receiving: -->--

Sending:enable

Receiving: --Password: --

Sending:cisco

Receiving: --#--

already connected to cisco router

Sending:show ip route | tee tftp://202.185.109.170/routing

Receiving: --#--
```

Table 4-11 Testing Result of Unit Test 3.1

Table 4-11 shows the testing result of the CISCONATACLRouteDownload unit test, in which the function has successfully connected to the target router, and has entered into the TFTP folder to find a file named “routing”. The content of this file is the same as the output of the “show ip route” command. This signifies that the unit test passed.

4.2.3.2 CISCONATACLRouteparse and CISCONATACLRouteparseline

CISCONATACLRouteparse and CISCONATACLRouteparseline open/read downloaded the routing table file and analyze it. Table 4-12 shows the expected result of this unit test. The feeding data is the downloaded routing table file in the previous unit test.

Expect Result:

- 1.Successfully open and read downloaded running- config file
- 2.Correctly analyze downloaded running- config file and assign values to arguments

Table 4-12 Expected Result of Unit Test 3.2

Testing result:

```

=====Parsing Routing Table file /tftpboot/routing...=====
(.....overleap.....)
the 4 Line
  buf=
  cfgfile=/tftpboot/routing
  line=4

the 5 Line
  buf=R          202.185.101.0/24 [120/1] via 202.185.109.172, 00:00:01,
FastEthernet0.101
  cfgfile=/tftpboot/routing
  line=5
start call parseline function
Start parse line.....
the whole arg is: R    202.185.101.0/24 [120/1] via 202.185.109.172, 00:00:01,
FastEthernet0.101
the ip is 202.185.101.0

submask is 255.255.255.0
Blockip is 167772151
NextHop is 202.185.109.171
Found is 1
finished 5 line parse
(.....overleap.....)

```

Table 4-13 Testing Result of Unit Test 3.2

Table 4-13 shows the testing result, that is, the two functions has successfully opened/read and analyzed the downloaded routing table file. The testing result is the same with the expected result, which means the unit test passed.

4.2.4 Unit Test 4 - NAT/PAT Pool File Processing

Unit Test 4 is similar to the previous unit test of “Running-config processing”. This unit test is divided into to two unit tests, one is to download the NAT/PAT pool file

(CISCONATACLPoolDownload), while another is to analyze the downloaded routing table file (CISCONATACLPoolparse and CISCONATACLPoolparseline). The test environment is showed below.

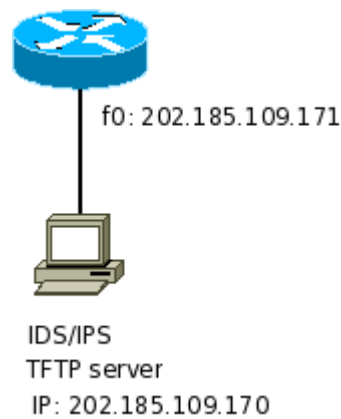


Figure 4-4 Testing Environment of Unit Test 4

4.2.4.1 CISCONATACLPoolDownload

CISCONATACLPoolDownload take the responsibility for downloading the NAT/PAT pool file from the target router. Table 4-14 shows expected result.

Expected result:

- 1.Successfully connect to target router
- 2.Successfully download NAT/PAT pool file and save to target TFTP server

Table 4-14 Expected Result of Unit Test 4.1

Testing result:

```
Finished creat and bind socket, then try to connect
Connected to=202.185.109.171
Connected to ciskonatacl at 202.185.109.171.
do simple authentication with password
Receiving: --Password: --

Sending:cisco1

Receiving: -->--

Sending:enable

Receiving: --Password: --
```



```

Sending:cisco2
Receiving: --#--
already connected to cisco router
Sending:show ip nat translate | tee tftp:// 202.185.109.170/pool
Receiving: --#--

```

Table 4-15 Testing Result of Unit Test 4.1

Table 4-15 shows the testing result: the function has successfully connected to the target router, and has entered into the TFTP folder to look for a file named “pool”. As the content of the file is the same with the output command “show ip nat translate” command, the unit test succeeded.

4.2.4.2 CISCONATACLPoolParse and CISCONATACLPoolParseLine

These two functions handle the operation of opening/reading the downloaded NAT/PAT pool file and analyzing it. Table 4-16 shows the expected result of these two functions. The feeding data is taken from the downloaded NAT/PAT pool file in the previous unit test.

```

Expect Result:
1.Successfully open and read downloaded NAT/PAT pool file
2.Correctly analyze downloaded NAT/PAT pool file and assign values to arguments

```

Table 4-16 Expect Result of Unit Test 4.2

```

Testing result:
=====Parsing running-config file /tftpboot/pool...=====
(.....overleap.....)
This is 2 loop
  buf= --- 202.185.110.1 202.185.109.170 --- ---
  cfgfile=/tftpboot/run
  line=2
start call parseline function
Start parse line.....
the whole arg is: --- 202.185.110.1 202.185.109.170 --- ---
the inside global ip is 202.185.110.1
the inside local ip is 202.185.109.170

```

finish 2 line parse (.....overleap.....)

Table 4-17 Testing Result of Unit Test 4.2

As can be seen in Table 4-17 which shows the testing result of this unit test, the unit test succeeded; these two functions has successfully opened/read downloaded NAT/PAT pool file and has correctly analyzed it.

4.2.5 Unit Test 5 - “Add ACL entry processing”

This unit includes two functions testing of CISCONATAACLCreateACL and CISCONATAACLRunningModify. The unit test will use the testing environment shows in Figure 4-4.

4.2.5.1 CISCONATAACLCreateACL

This function creates a new ACL at the target interface when there is no any ACL at the target interface. Table 4-18 shows the data fed to this function, whereas Table 4-19 shows the expected result.

Feeding Data:

Intruder's IP: 192.168.1.2

Target Interface: FastEthernet0

Table 4-18 Feeding Data for Unit Test 5.1

Expected result

- | |
|---|
| <ol style="list-style-type: none">1. Successfully connect to target router2. Correctly add ACL entry in target interface |
|---|

Table 4-19 Expected Result of Unit Test 5.1

Testing result:

Connected to ciscoatacl at 202.185.109.171
do simple authentication with password

Receiving: --Password: --

Sending:cisco

Receiving: -->--

Sending:enable

Receiving: --Password: --

Sending:cisco

Receiving: --#--

Sending:config t

Receiving: --#--

Sending:ip access-list extended CiscoNATAACLFastEthernet0

Receiving: --#--

Sending:deny ip host 192.168.1.2 any

Receiving: --#--

Sending:permit ip any any

Receiving: --#--

Sending:exit

Receiving: --#--

Sending:int FastEthernet0

Receiving: --#--

Sending:ip access-group CiscoNATAACLFastEthernet0 in

Receiving: --#--

Sending:exit

Receiving: --#--

Table 4-20 Testing Result of Unit Test 5.1

be

```
ip access-list extended denytest
deny ip host 192.168.1.2 any
deny ip host 192.168.1.5 any
permit ip any any
```

Obviously, the testing result matches with the expected result. This function has successfully opened/read the downloaded running-config file, and intelligently added new ACL entries to the correct position. This unit test is successful.

4.2.6 Unit Test 6 - Connection Processing

This unit test includes only one function - CISCONATACLsendreceive. This function sets up connections between IPS and Cisco routers. The testing environment is showed in Figure 4-6.

Table 4-24 shows the corresponding feeding data.

Feeding Data:

```
IP of Target router: 202.185.109.171
telnet password:cisco1
enable password: cisco2
tftp server:202.185.109.170
sent Command: show running-config | tee tftp://202.185.109.170/run
```

Table 4-24 Feeding Data for Unit Test 6

Expected Result:

1. Successfully connect to target router
2. Successfully run command in router
3. Successfully receive data send from router

Table 4-25 Expected Result of Unit Test 6

Testing result:

```
Connected to cisonatacl at 202.185.109.171

do simple authentication with password
Receiving: --Password: --
```

```

Sending:cisco1
Receiving: -->--
Sending:enable
Receiving: --Password: --
Sending:cisco2
Receiving: --#--
Sending:config t
Receiving: --#--
already connected to cisco router
Sending:show running-config | tee tftp://202.185.109.170/run
Receiving: --#--

```

Table 4-26 Testing Result of Unit Test 6

The testing result is showed in Table 4-26. After running this function, a downloaded file named “run” is found in the TFTP directory. The file “run” is sent by a Cisco router and is received via this function. The testing result shows that this function has successfully connected to the Cisco router by using the Telnet protocol, and has successfully received data sent from the Cisco router by the run command. This unit test is successful.

4.2.7 Unit Test 7 - ACL Checking

Unit Test 7 tests CISCONATAACLCheck, with the feeding data showed in Table 4-27 and the expected result showed in Table 4-28.

```

Feeding Data
Checked ACL entry: deny ip host 192.168.0.2 any
Running- config-1 include this ACL entry
Running- config-1 does not include this ACL entry

```

Table 4-27 Feeding Data for Unit Test 7

Excepted result

1. return Present=0 when check running-config-1 file
2. return Present=1 when checn running-config-2 file

Table 4-28 Expected Result of Unit Test 7

Testing result of checking running-config-1

```
>>>>>>>>>>>>>>Start checking<<<<<<<<<<<<<<<<<<<<<<
CheckedACL=deny ip host 192.168.0.2 any
Checkedfile= running-config-1
Present =0
>>>>>>>>>>>>>>End checking<<<<<<<<<<<<<<<<<<<<<<
```

Testing result of checking running-config-2

```
>>>>>>>>>>>>>>Start checking<<<<<<<<<<<<<<<<<<<<<<
CheckedACL=deny ip host 192.168.0.2 any
Checkedfile= running-config-2
Present =1
>>>>>>>>>>>>>>End checking<<<<<<<<<<<<<<<<<<<<<<
```

Table 4-29 Testing Result of Unit Test 7

The testing result of this unit test 7 is showed in Table 4-29. The match of the testing results and the expected result signifies that the unit test is successful.

4.3 Integration Testing

Following the unit testing, integration testing is of paramount importance. If the interaction among most of individual functions is not controlled properly, unexpected results behind the scenes could easily lead to catastrophes in the running phase. In this section, the integration test cases embrace three situations: (i) the IP address of the intruder belongs to a private IP, (ii) the IP belongs to an outside public IP, (iii) the IP belongs to an inside public IP. Figure 4-5 depicts the topology used in each of the test case.

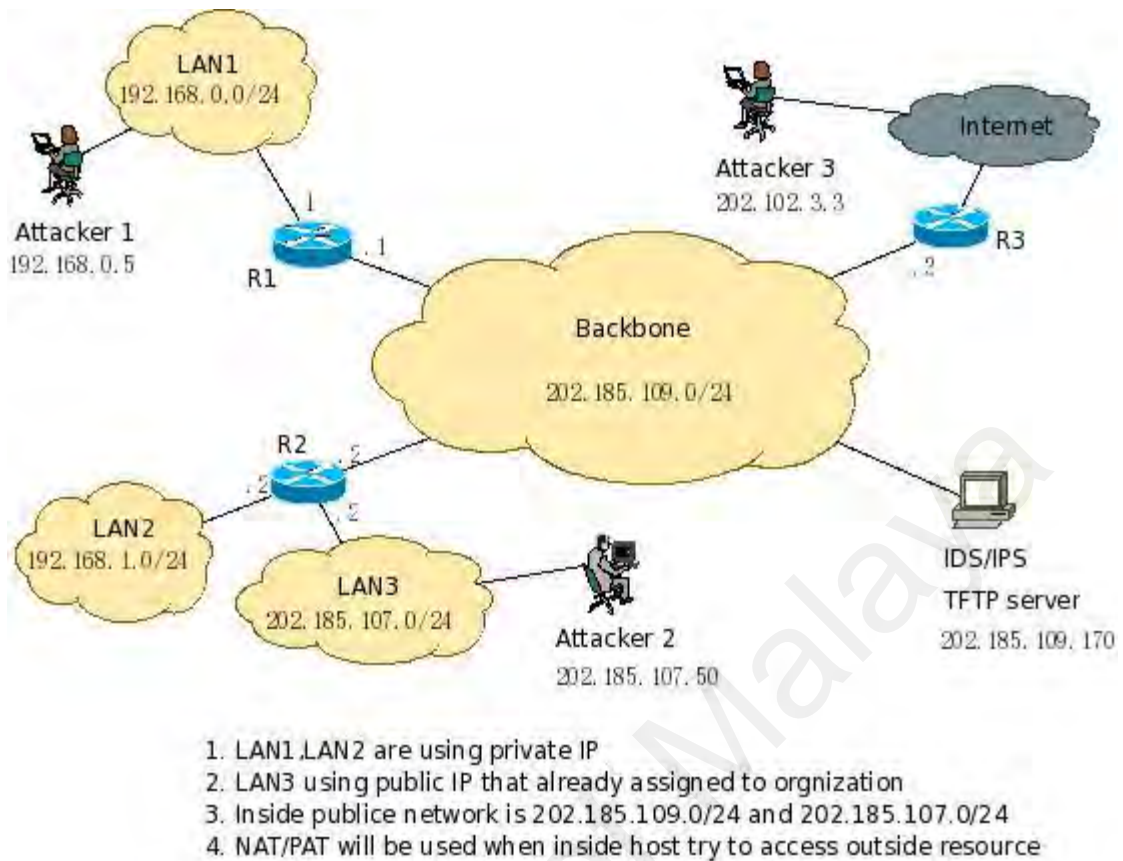


Figure 4-5 Topology Used for the Test Cases in the Integration Testing

Table 4-30 shows the general configuration for the test cases.

<pre>Snortsam.conf #cisconatacl <ip> <telentpw> <enablepw> cisconatacl 202.185.109.1 cisco cisco cisconatacl 202.185.109.2 cisco cisco cisconatacl 202.185.109.3 cisco cisco</pre>	<pre>/etc/cisconatacl.conf #Gateway router's information GatewayRouterIP 202.185.109.3 GatewayRouterTelnetPW cisco GatewayRouterEnablePW cisco OutsidePublicInterfaceName f0.101 #Tftp server information tftpAddressRun tftp://202.185.109.170/run tftpAddressPool tftp://202.185.109.170/pool tftpAddressRouting tftp://202.185.109.170/routing</pre>
---	--

Table 4-30 General Configurations For Integration Testing.

4.3.1 Integration Testing Case 1 - IP of Intruder Is Private IP

In Case 1, the attacker (IP is 192.168.0.5) starts an attack from an inside private IP. The IDS (Snort) detects the attack signature and triggers an alert which sends a block request to the IPS (SnortSam). The IPS then calls the block plug-in to process this block request. In this testing, only the CISCONATAACL plug-in is used. The plug-in will identify the type of attacker's IP. Because it is private IP in this case, the routines about processing private IP will be called.

At this point, there are two possible situations. First, there is no ACL exists at the target interface; second, the ACL that the direction is “in” exists in the target interface but the attacker's IP is not blocked.

These situations will be discussed in detail in following sub-sections.

4.3.1.1 Situation One - No ACL Exists In Target Interface

In this situation, there is no ACL exists at the target interface of the target router.

After receiving a block request, AIPM will try to find the correct router by analyzing the downloaded routing table file and try to find the correct interface by analyzing the downloaded running-config file. Then, a new ACL will be created since there is no ACL with the direction “in” exists at the target interface; the name of the new ACL is “CiscoNATAACL+<interface name>”. For example, if the target interface is “FastEthernet0/0.101”, then the name of the ACL will be CiscoNATAACLFastethEthernet0/0.101.

After connecting to the correct router, a serial command will be sent to the router to implement this block. Table 4-31 depicts the expected result; Table 4-32 shows the change of running-config file in the target router.

Expected result

1. ACL named CiscoNATACLFastethEthernet0/0.101 will be created
2. “deny ip host 192.168.0.5 any” will be added to the ACL

Table 4-31 Expected Result of Integration Test Case 1.1

Before blocked	After blocked
<pre> (.....overleap.....) ! interface FastEthernet0/0 ip address 192.168.0.1 255.255.255.0 duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.109.1 255.255.255.0 duplex auto speed auto ! ! ! ! ! ! line con 0 transport preferred all transport output all (.....overleap.....) end </pre>	<pre> (.....overleap.....) ! interface FastEthernet0/0 ip address 192.168.0.1 255.255.255.0 ip access-group CiscoNATACLFastEthernet0/0 in duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.109.1 255.255.255.0 duplex auto speed auto ! ! ! ! ! ! ip access-list extended CiscoNATACLFastEthernet0/0 deny ip host 192.168.0.5 any permit ip any any ! (.....overleap.....) End </pre>

Table 4-32 Testing Result of Integration Testing Of Case1.1

Table 4-33 depicts the result of the ping command.

```
[root:/]ping 202.185.109.170
PING 202.185.109.170(202.185.109.170) 56(84) bytes of data.
64 bytes from 202.185.109.170: icmp_seq=1 ttl=64 time=0.045ms
64 bytes from 202.185.109.170: icmp_seq=2 ttl=64 time=0.046ms
From 192.168.0.1 icmp_seq=3 Destination Host Unreachable
From 192.168.0.1 icmp_seq=4 Destination Host Unreachable
```

Table 4-33 Result of Ping Command

Obviously, the attacker's IP has been blocked successfully at the target router. Since the testing result matches with the expected result, this integration test is successful.

4.3.1.2 Situation Two - An ACL Exists In Target Interface

In this situation, an ACL with the direction “in” exists at the target interface in the target router.

Again, AIPM will try to find an appropriate router. In this case, however, the block request cannot be implemented by simply sending an “add new ACL” command to the router. More steps should be taken. The first step is to get the existing ACL’s name when parsing the downloaded running-config file. The second step is to read the ACL content from the running-config file. Note that there are two types of ACLs in Cisco routers - extend ACL and standard ACL. AIPM has to process these two types of ACL because the format is not the same.

Table 4-34 demonstrates the two examples.

Type	Contents
Extended ACL	ip access-list extended what deny ip host 1.1.1.1 any deny ip host 1.1.1.2 any permit ip any any

Standard ACL	access-list 102 deny ip host 10.0.0.1 any access-list 12 deny host 3.3.3.3
--------------	---

Table 4-34 Types of ACL

After parsing the ACL content, the line number that correspond to an ACL matched with the previously found ACL name is recorded for further process. After retrieving the information of the correct router, correct interface, ACL name, and the location in running-config file, the next step is to modify the downloaded running-config file by adding new ACLs in the right location and uploading it to the router. Before uploading the file, however, the current ACL in the router must be erased first to avoid unexpected behavior. AIPM achieves this goal by sending some commands to router. After uploading the modified running-config file to the router, the block request should be implemented successfully. Table 4-35 depicts the expected result and Table 4-36 shows the change of running-config file in the target router.

Expected Result:

1. Add ACL entry “deny ip host 192.168.0.5 any” to running-config file

Table 4-35 Expected Result of Integration Test Case 1.2

Before blocked	After blocked
<pre>overleap.....) interface FastEthernet0/0 ip address 192.168.0.1 255.255.255.0 ip access-group TEST in duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.109.3 255.255.255.0 duplex auto speed auto ! (.....overleap.....) ! ip access-list extended TEST deny ip host 192.168.0.15 any deny ip host 192.168.0.25 any permit ip any any ! (.....overleap.....) </pre>	<pre> (.....overleap.....) interface FastEthernet0/0 ip address 192.168.0.1 255.255.255.0 ip access-group TEST in duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.109.1 255.255.255.0 duplex auto speed auto ! (.....overleap.....) ! ip access-list extended TEST deny ip host 192.168.0.5 any ← deny ip host 192.168.0.15 any deny ip host 192.168.0.25 any permit ip any any ! </pre>

End	(.....overleap.....) end
-----	-----------------------------

Table 4-36 Testing Result of Integration Testing Casel.2

Table 4-37 depicts the result of the ping command.

```
[root:/]ping 202.185.109.170
PING 202.185.109.170(202.185.109.170) 56(84) bytes of data.
64 bytes from 202.185.109.170: icmp_sep=1 ttl=64 time=0.047ms
64 bytes from 202.185.109.170: icmp_sep=2 ttl=64 time=0.045ms
64 bytes from 202.185.109.170: icmp_sep=3 ttl=64 time=0.045ms
From 192.168.0.1 icmp_seq=4 Destination Host Unreachable
From 192.168.0.1 icmp_seq=5 Destination Host Unreachable
```

Table 4-37 Result of Ping Command

Obviously, the block request has been applied successfully at the target router. Thus, this integration test is successful.

4.3.2 Integration Testing Case 2--IP of Intruder Is Outside Public IP

In case three, the attacker (IP is 202.102.3.3) initiates an attack from an outside public IP. Once AIPM finds out that the captured attacker's IP belongs to a public IP, it directly connects to the gateway router as pre-defined in /etc/cisconatacl.conf. After identifying that the IP is outside public IP, a new block of ACL is applied to the outside public interface. If there is an ACL with direction "in" exists at the outside public interface, the system will download the running-config file, add a new ACL entry, and upload it to the gateway router. If there is no such ACL exists, the system will connect to the gateway router and create a new ACL based on the name of the outside public interface. For instance, if the interface name is FastEthernet0/1, a new ACL named CiscoACLFastEthernet0 is then created and applied to the outside public interface of the gateway router.

As in the previous test case, this integration testing case also embraces two situations.

4.3.2.1 Situation One-- No ACL Exists In Target Interface

Table 4-38 depicts the expected result in this situation.

<p>Expected result</p> <ol style="list-style-type: none"> 1. ACL named CiscoNATACLFastethEthernet0/0.101 will be created 2. “deny ip host 202.102.3.3 any” will be added to the ACL
--

Table 4-38 Expected Result of Integration Testing Case 2.1

Before blocked	After blocked
<pre>(.....overleap.....) interface FastEthernet0/0 ip address 202.185.110.2 255.255.255.0 duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.109.3 255.255.255.0 duplex auto speed auto ! (.....overleap.....) !</pre>	<pre>(.....overleap.....) interface FastEthernet0/0 ip address 202.185.110.2 255.255.255.0 ip access-group ← CiscoNATACLFastEthernet0/0 in duplex auto speed auto ! iend nterface FastEthernet0/1 ip address 202.185.109.3 255.255.255.0 duplex auto speed auto (.....overleap.....) ip access-list extended CiscoNATACLFastEthernet0/0 deny ip host 202.102.3.3 any ← permit ip any any !</pre>

Table 4-39 Testing Result of Integration Testing Of Case 2.1

Table 4-40 depicts the result of the ping command.

<pre>[root:/]ping 202.185.109.170 PING 202.185.109.170(202.185.109.170) 56(84) bytes of data. 64 bytes from 202.185.109.170: icmp_seq=1 ttl=64 time=0.047ms 64 bytes from 202.185.109.170: icmp_seq=2 ttl=64 time=0.045ms From 202.1585.110.2 icmp_seq=3 Destination Host Unreachable From 202.1585.110.2 icmp_seq=4 Destination Host Unreachable</pre>

Table 4-40 Result of Ping Command

As can be observed in Table 4-40, the block request has been applied successfully at the target router. So, this integration test is successful.

4.3.2.2 Situation Two-- ACL Exist In Target Interface

Table 4-41 shows the expected result in this integration testing.

Expected result

1. Add ACL entry “deny ip host 192.168.0.5 any” to running config file

Table 4-41 Expected Result of Integration Testing Case 2.2

Table 4-42 shows a sample that changes the running-config file in the target router. In this sample, an extended ACL named TEST has already existed at the target interface.

Before block	After Block
<pre>(.....overleap.....) interface FastEthernet0/0 ip address 202.185.110.2 255.255.255.0 ip access-group TEST in duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.109.3 255.255.255.0 duplex auto speed auto (.....overleap.....) ! ip access-list extended TEST deny ip host 205.10.25.8 any deny ip host 207.231.25.9 any permit ip any any ! (.....overleap.....) ! End</pre>	<pre>(.....overleap.....) interface FastEthernet0/0 ip address 202.185.110.2 255.255.255.0 ip access-group TEST in duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.109.3 255.255.255.0 duplex auto speed auto (.....overleap.....) ! ip access-list extended TEST deny ip host 202.102.3.3 any ← deny ip host 205.10.25.8 any deny ip host 207.231.25.9 any permit ip any any ! (.....overleap.....) ! End</pre>

Table 4-42 Testing Result of Integration Testing Of Case 2.2

Table 4-43 depicts the result of the ping command.

```
[root:/]ping 202.185.109.170
PING 202.185.109.170(202.185.109.170) 56(84) bytes of data.
64 bytes from 202.185.109.170: icmp_seq=1 ttl=64 time=0.048ms
64 bytes from 202.185.109.170: icmp_seq=2 ttl=64 time=0.046ms
64 bytes from 202.185.109.170: icmp_seq=3 ttl=64 time=0.045ms
From 202.1585.110.2 icmp_seq=4 Destination Host Unreachable
From 202.1585.110.2 icmp_seq=5 Destination Host Unreachable
```

Table 4-43 Result of Ping Command

The testing result reveals that the block request has been applied successfully at the target router; this integration test is successful.

4.3.3 Integration Testing Case Three--IP of Intruder Is Inside Public IP

In case two, the attacker (IP is 202.185.107.50) starts an attack from an inside public IP. Once AIPM has identified that the captured attacker's IP belongs to a public IP, it will first connect to the gateway router. Obviously, the target router (R2) is not the first router to connect; CISCONATAACL plug-in has to find the target router by analyzing the downloaded routing table file of the gateway router (R3 in this testing environment).

Because AIPM does not know which public IP is inside public IP (there is no any predefined), two possibilities emerge: one is inside public IP, and another is outside public IP. AIPM must know how to identify types of the IP. This problem can be solved by comparing the routing table since NAT/PAT is only applied when the inside network is trying to access the outside network. If a match is found in the routing table downloaded from the gateway router, the attack comes from inside; otherwise, the attack comes from outside.

After finding the target router, the proposed plug-in checks whether it is defined in

SnortSam.conf file. If it is already defined, then the corresponding arguments (telnet/enable password) are read. Next, AIPM will connect to the target router according to this information. Since there are two possibilities, AIPM must identify if the captured IP belongs to a host or a pool. The pool is checked fist; if a matched record is found, the record is replace with the block IP; otherwise, the system will check the routing table. Subsequent process is similar to that in Case One.

4.3.3.1 Situation One - No ACL Exist In Target Interface

Table 4-44 depicts the expected result in this situation.

Expected Result

1. ACL named CiscoNATACLFastethEthernet0/0 will be created
2. deny ip host 202.185.107.50 any” will be added to the ACL

Table 4-44 Expected Result of Integration Testing Case 3.1

Table 4-45 shows testing result of integration testing case 3.1.

Before block	After Block
<pre> .(.....overleap.....) interface FastEthernet0/0 ip address 202.185.107.2 255.255.255.0 duplex auto speed auto ! (.....overleap.....) ! end </pre>	<pre> (.....overleap.....) interface FastEthernet0/0 ip address 202.185.107.2 255.255.255.0 ip access-group CiscoNATACLFastethEthernet0/0 in duplex auto speed auto (.....overleap.....) ! ip access-list extended CiscoNATACLFastethEthernet0/0 deny ip host 202.185.107.50 any ← permit ip any any ! (.....overleap.....) end </pre>

Table 4-45 Testing Result of Integration Testing 3.1

Table 4-46 depicts the result of the ping command.

```
[root:/]ping 202.185.109.170
PING 202.185.109.170(202.185.109.170) 56(84) bytes of data.
64 bytes from 202.185.109.170: icmp_seq=1 ttl=64 time=0.048ms
64 bytes from 202.185.109.170: icmp_seq=2 ttl=64 time=0.049ms
From 202.185.107.2 icmp_seq=3 Destination Host Unreachable
From 202.185.107.2 icmp_seq=4 Destination Host Unreachable
```

Table 4-46 Result of Ping Command

The ping result suggests that the block request has been applied successfully at the target router.

Therefore, this integration test is successful.

4.3.3.2 Situation Two - ACL Exist In Target Interface

Table 4-47 shows the expected result, and Table 4-48 shows the changes made in the running-config file of the target router. In this case, the extended ACL named TEST has already exist at the target interface.

Expected result:

1. Add ACL entry “deny ip host 202.185.107.50 any” to TEST access-list

Table 4-47 Expected Result of Integration Testing 3.2

Before block	After Block
<pre>(.....overleap.....) interface FastEthernet0/0 ip address 202.185.107.2 255.255.255.0 ip access-group TEST in duplex auto speed auto (.....overleap.....) ! ip access-list extended TEST deny ip host 192.168.0.15 any deny ip host 192.168.0.25 any permit ip any any ! (.....overleap.....)</pre>	<pre>(.....overleap.....) interface FastEthernet0/0 ip address 202.185.107.2 255.255.255.0 ip access-group TEST in duplex auto speed auto (.....overleap.....) ! ip access-list extended TEST deny ip host 202.185.107.50 any ← deny ip host 192.168.0.15 any deny ip host 192.168.0.25 any permit ip any any (.....overleap.....)</pre>

Table 4-48 Testing Result of Integration Testing Case 3.2

Table 4-49 depicts the result of the ping command.

```
[root:/]ping 202.185.109.170
PING 202.185.109.170(202.185.109.170) 56(84) bytes of data.
64 bytes from 202.185.109.170: icmp_seq=1 ttl=64 time=0.048ms
64 bytes from 202.185.109.170: icmp_seq=2 ttl=64 time=0.046ms
64 bytes from 202.185.109.170: icmp_seq=3 ttl=64 time=0.049ms
From 202.185.107.2 icmp_seq=4 Destination Host Unreachable
From 202.185.107.2 icmp_seq=5 Destination Host Unreachable
```

Table 4-49 Result of Ping Command

Again, the ping result shows that the block request has been applied successfully at the target router; this integration test is successful.

4.4 Responding Time of Integration Test

All testing are successful in this integration testing environment. Table 4-50 shows the responding time after an attack is fired.

Testing Number	Testing number of times	Condition 1	Condition 2	Private IP	Inside Public IP	Outside Public IP
1	20	Without ACL Exist	Without NAT/PAT	Average 2sec (after 2 packages transmitted)	Average 3sec (after 3 packages transmitted)	Average 3sec (after 3 packages transmitted)
2	20	Without ACL Exist	With NAT/PAT	Average 3sec (after 3 packages transmitted)	Average 4sec (after 4 packages transmitted)	Average 3sec (after 3 packages transmitted)
3	20	With ACL Exist	Without NAT/PAT	Average 6sec (after 6 packages transmitted)	Average 9sec (after 10 packages transmitted)	Average 8sec (after 8 packages transmitted)
4	20	With ACL Exist	With NAT/PAT	Average 6sec (after 7 packages transmitted)	Average 10sec (after 11 packages transmitted)	Average 9sec (after 10 packages transmitted)

				transmitted)	transmitted)	transmitted)
--	--	--	--	--------------	--------------	--------------

Table 4-50 Responding Time

Table 4-50 shows that the responding time increases when an ACL exists at a target interface or when the NAT/PAT technology is applied in a target router. That is because AIPM has additional workload: erasing old ACL entries, modifying downloaded running-config files, uploading modified running-config file to the target router when an ACL has already existed at the target interface, and downloading and analyzing the NAT pool file when the NAT/PAT technology is applied at the target router.

In this test, the processes of cleaning old ACL entries, and downloading and modifying the running-config file are fairly fast, but uploading the modified running-config file to the target router to apply the new running-config file takes longer time (i.e. 4-6 seconds). On the other hand, downloading and analyzing the NAT pool file takes about 1-2 seconds. Future work will focus on solving reducing the time needed.

4.5 Chapter Summary

This chapter provides an overview of system and development. Key components of AIPM have been discussed. Unit and integration testing are also discussed in this chapter. All tests are successful. System testing of AIPM will be discussed in the next chapter.

Chapter 5 Case Study

This chapter describes two case studies, which AIPM runs on different typical business networks. The first case involves a network environment of small business, whilst the second is in an environment of a big company or college/university.

5.1 Case Study 1

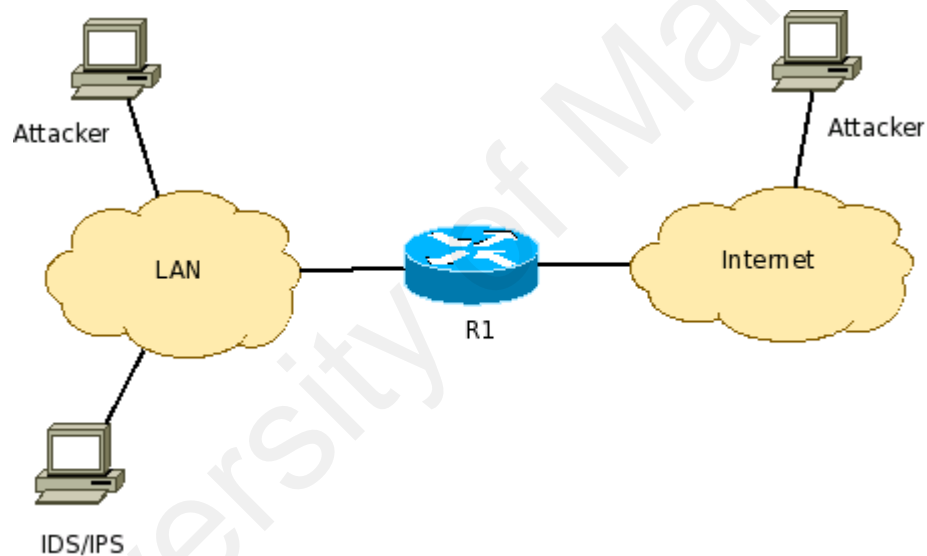


Figure 5-1 Network Environment of Case Study 1

Figure 5-1 shows the network environment of Case Study 1. In this environment, LAN connects to the Internet through a router (R1). There are two possible intruders: one coming from inside, and another one from outside. The IP of an inside intruder is set as 192.168.0.10/24; the IP of an outside intruder is set as 222.200.200.2/24. This network environment is a typical network design mostly used by small business.

5.1.1 Case Study 1.1—Attack From Inside

One possibility is that the inside intruder fired an attack. Table 5-1 depicts the expected result of this case study.

Expected Result:

1. Inside intruder's IP is blocked in Router R1.

Table 5-1 Expected Result of Case Study 1.1

Table 5-2 shows comparison of running-config file before and after the block request is implemented.

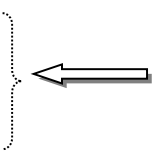
Before blocked	After blocked
<pre>(.....overleap.....) interface FastEthernet0/0 ip address 192.168.0.1 255.255.255.0 duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.109.3 255.255.255.0 duplex auto speed auto ! ! ! (.....overleap.....) end</pre>	<pre>(.....overleap.....) interface FastEthernet0/0 ip address 192.168.0.1 255.255.255.0 ip access-group CiscoNATACLFastEthernet0/0 in duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.109.1 255.255.255.0 duplex auto speed auto ! (.....overleap.....) ! ip access-list extended CiscoNATACLFastEthernet0/0 deny ip host 192.168.0.10 any permit ip any any ! (.....overleap.....) end</pre> <div style="text-align: right; margin-right: 20px;">  </div>

Table 5-2 Testing Result of Case Study 1.1

Table 5-3 depicts the result of the ping command. Note that IP of the IDS AND IPS server is 192.168.0.200.

```
[root:/]ping 192.168.0.200
PING 192.168.0.200 (192.168.0.200) 56(84) bytes of data.
64 bytes from 192.168.0.200: icmp_seq=1 ttl=64 time=0.048ms
64 bytes from 192.168.0.200: icmp_seq=2 ttl=64 time=0.046ms
From 192.168.0.1 icmp_seq=3 Destination Host Unreachable
From 192.168.0.1 icmp_seq=4 Destination Host Unreachable
```

Table 5-3 Result of Ping Command

Hence, it can be inferred that the block request has been applied successfully at the target router.

5.1.2 Case Study 1.2—Attack From Outside

Since most networks of small businesses connect to the Internet,, the ability of defend outside attack is very important. As such, Case Study 1.2 will test AIPM on this ability. Table 5-4 is the expected result of this case study.

```
Expected Result:
1. Outside intruder’s IP is blocked at interface that connect to internet
```

Table 5-4 Expected Result of Case Study 1.2

Table 5-5 shows comparison of running-config file before and after the block request is implemented.

Before blocked	After blocked
<pre>(.....overleap.....) interface FastEthernet0/0 ip address 192.168.0.1 255.255.255.0 duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.109.3 255.255.255.0 duplex auto speed auto !</pre>	<pre>(.....overleap.....) interface FastEthernet0/0 ip address 192.168.0.1 255.255.255.0 duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.109.1 255.255.255.0 ip access-group CiscoNATACLFastEthernet0/1 in duplex auto speed auto !</pre>

<pre>! ! (.....overleap.....) end</pre>	<pre>(.....overleap.....) ! ip access-list extended CiscoNATACLFastEthernet0/1 ← deny ip host 222.200.200.2 any permit ip any any ! (.....overleap.....) end</pre>
---	--

Table 5-5 Testing Result of Case Study 1.2

Table 5-6 depicts the result of the ping command checking the accessibility to the IDS AND IPS server.

```
[root:/]ping 192.168.0.200
PING 192.168.0.200 (192.168.0.200) 56(84) bytes of data.
64 bytes from 192.168.0.200: icmp_seq=1 ttl=64 time=0.046ms
64 bytes from 192.168.0.200: icmp_seq=2 ttl=64 time=0.049ms
From 202.185.109.1 icmp_seq=3 Destination Host Unreachable
From 202.185.109.1 icmp_seq=4 Destination Host Unreachable
```

Table 5-6 Result of Ping Command

The testing result shows that the malicious traffics from the intruder have been blocked. So, the result matches with the expected result.

5.2 Case Study 2

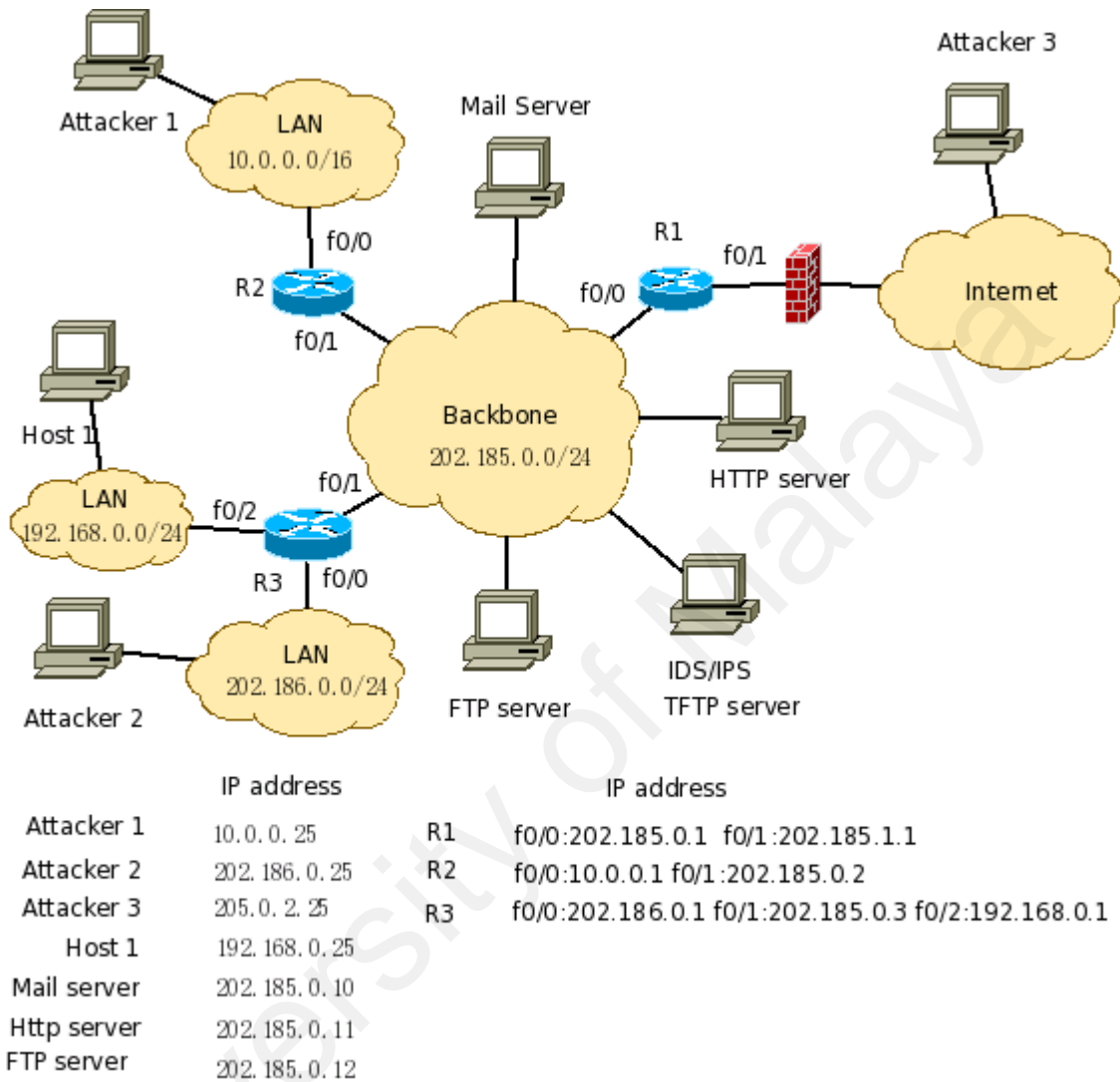


Figure 5-2 Network Environment of Case Study 2

In Case Study 2, the network environment used is typical for a big company or college/university (See Figure 5-2). There are several application servers running in the company network. A firewall to protect the network against malicious traffic from the Internet is also deployed. Besides, NAT/PAT is configured in R2 and R3 and is only applied when the traffic destination is the Internet. Further, both static routes and dynamic routing protocols are set. Three kinds of attack are modeled in this case study, that is, attack coming from inside to inside, inside to outside, and outside to inside.

5.2.1 Case Study 2.1—Attack From Inside To Inside

In this case study, an intrusion of attack comes from inside to inside: Attacker 1 trying to attack Host 1. Table 5-7 shows the expected result.

Expected Result

1. ACL CiscoNATAACLFastEthernet0/0 is created in R2
2. 10.0.0.25 is blocked at f0/0 in R2

Table 5-7 Expected Result of Case Study 2.1

Table 5-8 shows the comparison of running-config file before and after the block request is sent.

Before blocked	After blocked
<pre>(.....overleap.....) interface FastEthernet0/0 ip address 10.0.0.1 255.255.0.0 ip nat inside duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.0.2 255.255.255.0 ip nat outside duplex auto speed auto ! ! ! (.....overleap.....) ! end</pre>	<pre>(.....overleap.....) interface FastEthernet0/0 ip address 10.0.0.1 255.255.0.0 ip nat inside ip access-group CiscoNATAACLFastEthernet0/0 in duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.0.2 255.255.255.0 ip nat outside duplex auto speed auto ! (.....overleap.....) ! ip access-list extended CiscoNATAACLFastEthernet0/0 deny ip host 10.0.0.25 any ← permit ip any any ! (.....overleap.....) end</pre>

Table 5-8 Testing Result of Case Study 2.1

Table 5-9 depicts the result of the ping command. Note that the IP of IDS AND IPS server is

202.185.0.170. As can be seen in this table, the block request has been successfully applied at the target router (R2), and the malicious traffics from Attacker 1 have been blocked.

```
[root:/]ping 202.185.0.170
PING 202.185.0.170 (202.185.0.170) 56(84) bytes of data.
64 bytes from 202.185.0.170: icmp_seq=1 ttl=64 time=0.046ms
64 bytes from 202.185.0.170: icmp_seq=2 ttl=64 time=0.047ms
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
From 10.0.0.1 icmp_seq=4 Destination Host Unreachable
```

Table 5-9 Result of Ping Command

5.2.2 Case Study 2.2—Attack From Inside To Outside

This case study involves an intrusion of attack coming from inside to outside when Host 1 attempts to attack an Internet host (e.g. a HTTP server - www.google.com). Table 5-10 shows the expected result.

Expected Result
1. ACL CiscoNATACLFastEthernet0/2 be created in R3
2. 192.168.0.25 is blocked at f0/2 in R3

Table 5-10 Expected Result of Case Study 2.2

Table 5-11 shows comparison of running-config file before and after the block request is implemented.

Before blocked	After blocked
(.....overleap.....) interface FastEthernet0/0 ip address 202.185.0.1 255.255.0.0 ip nat inside duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.0.3 255.255.255.0 ip nat outside duplex auto speed auto !	(.....overleap.....) interface FastEthernet0/0 ip address 202.185.0.1 255.255.0.0 ip nat inside duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.0.3 255.255.255.0 ip nat outside duplex auto speed auto !

<pre> interface FastEthernet0/2 ip address 192.168.0.1 255.255.0.0 ip nat inside duplex auto speed auto ! ! (.....overleap.....) ! end </pre>	<pre> interface FastEthernet0/2 ip address 192.168.0.1 255.255.0.0 ip nat inside ip access-group CiscoNATACLFastEthernet0/2 in duplex auto speed auto ! (.....overleap.....) ! ip access-list extended CiscoNATACLFastEthernet0/2 deny ip host 192.168.0.25 any ← permit ip any any ! (.....overleap.....) end </pre>
--	---

Table 5-11 Testing Result of Case Study 2.2

As in Case Study 2.1, the block request has been applied successfully at the target router (R3);

Table 5-12 depicts the result of the ping command.

```

[root:/]ping www.google.com
PING www.1.google.com (72.14.205.104) 56(84) bytes of data.
64 bytes from qb-in-f104.google.com (72.14.205.104): icmp_seq=1 ttl=64 time=264ms
64 bytes from qb-in-f104.google.com (72.14.205.104): icmp_seq=2 ttl=64 time=267ms
From 192.168.0.1 icmp_seq=3 Destination Host Unreachable
From 192.168.0.1 icmp_seq=4 Destination Host Unreachable

```

Table 5-12 Result of Ping Command

5.2.3 Case Study 2.3—Attack from Outside to Inside

Lastly, in this case study, AIPM tries to protect the company’s FTP server against an intrusion of attack coming from outside (Attacker 3). Table 5-13 shows the expected result.

<p>Expected Result</p> <ol style="list-style-type: none"> 1. ACL CiscoNATACLFastEthernet0/1 be created in R1 2. 205.0.2.25 is blocked at f0/1 in R1
--

Table 5-13 Expected Result of Case Study 2.3

Table 5-14 shows comparison of running-config file before and after the block request is issued.

Before blocked	After blocked
<pre>(.....overleap.....) interface FastEthernet0/0 ip address 202.185.0.1 255.255.0.0 duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.1.1 255.255.255.0 duplex auto speed auto ! (.....overleap.....) ! end</pre>	<pre>(.....overleap.....) interface FastEthernet0/0 ip address 202.185.0.1 255.255.0.0 duplex auto speed auto ! interface FastEthernet0/1 ip address 202.185.1.1 255.255.255.0 ip access-group CiscoNATACLFastEthernet0/1 in duplex auto speed auto ! (.....overleap.....) ip access-list extended CiscoNATACLFastEthernet0/1 ← deny ip host 205.0.2.25 any permit ip any any (.....overleap.....) end</pre>

Table 5-14 Testing Result of Case Study 2.3

Likewise, the ping result in Table 5-15 shows that the block request has been applied successfully at the target router (R1); any type of traffic coming from Attack 3 will be dropped by R1.

```
[root:/]ping 202.185.109.170
PING 202.185.109.170 (202.185.109.170) 56(84) bytes of data.
64 bytes 202.185.109.170: icmp_seq=1 ttl=64 time=0.088ms
64 bytes 202.185.109.170: icmp_seq=2 ttl=64 time=0.089ms
From 202.185.109.170 icmp_seq=3 Destination Host Unreachable
From 202.185.109.170 icmp_seq=4 Destination Host Unreachable
```

Table 5-15 Result of Ping Command

5.3 Comparison between Existing IDS AND IPS and AIPM

	Snort	Snort-Inline	SnortSam	AIPM
Platform based	Windows/Linux	Linux/Unix based	Windows/Linux/Unix based	Linux/Unix based
Language	Gnu C	Gnu C	Gnu C	Gnu C
Ability to communicate with security hardware	Not support	Not support	Partly support	Support
Prevention method	--	--	Simple	Complex and intelligent
Ability to communicate with Firewall	Not support	IPTable/IPFW	yes	no
License catalog	GUN license	GUN license	GUN license	GUN license
System catalog	IDS	IPS	IPS	IPS

Table 5-16 Comparison of Existing IDS AND IPS and AIPM

Shown in Table 5-16 is a brief overview of comparison between some existing IDS AND IPS and AIPM. In this work, SnortSam is almost comparable with AIPM except that SnortSam only partly supports communication with security hardware. SnortSam requires much predefined information for communicating with security hardware and only supports simple prevention processing. On the other hand, AIPM can support communication with security hardware and requires only few predefined information. Additionally, AIPM comprises the prevention feature that can be adopted in an NAT/PAT environment.

5.3.1 Advantages of AIPM

Different IDS AND IPS products have different advantages. While inheriting all advantages of Snort and SnortSam, AIPM comprises other new features. The most important advantage of AIPM is the ability to automatically analyze the network environment information without any manual intervention and the ability to defend intrusion accordingly. This proposed IPS system obtains the network information by downloading running-config files, route table files, and

NAT pool files from Cisco routers. After analyzing these files, AIPM determines the IP of the intruder, and makes a decision on which router's interface to block, and applies ACL if necessary.

The second important advantage of AIPM is the ability to find the real intruder's IP even if the NAT/PAT technology has been deployed. Many IDS AND IPS systems can only capture intruder's packages without identifying whether the IP is real IP or not. AIPM overcomes this weakness by analyzing the downloaded NAT/PAT pool file.

Last but not least, AIPM integrates advantages of Snort and SnortSam into a single IPS, while possesses the ability to decide which Cisco router to send a block request. In other words, AIPM works as both network security software and hardware. Meanwhile, since these software and hardware are independent, an IPS administrator can maintain and/or upgrade these software/hardware separately without affecting the network operation.

5.3.2 Features Comparison

	Snort	Snort-Inline	Snort+SnortSam+Cisco Router (SSCR)	AIPM
Software Type	IDS	IPS	IPS	IPS
Need runs in Gateway?	No	Yes	No	No
Able to communicate with other network security hardware?	No	No	Yes	Yes
Intelligent analyze captured packages?	No	No	No	Yes
Support firewall?	No	Yes	Yes	Yes
Intelligent block intruder's IP?	No	No	No	Yes
Support NAT/PAT?	No	No	No	Yes
Need predefined ACL file?	---	---	1 file/router	No need
Need predefined interface name?	---	---	Yes	No need

Need predefined ACL name?	---	---	Yes	No need
Connection when implement block	---	---	1 connection / router	1 connection only

Table 5-17 Features Comparison of Different IDS AND IPS

Table 5-17 shows a comparison among different IDS AND IPS including Snort, Snort-Inline, “Snort+SnortSam+Cisco Router” (SSCR) and AIPM. The most similar system with the proposed AIPM is SSCR. In comparison to AIPM, SSCR does not have the ability to analyze captured packages. Thus, SSCR fails to get the real IP of an intruder when the NAT/PAT technology is used. In addition, SSCR requires the information of predefined ACL configuration, which means that all ACL configuration of the router must be known before establishing a connection to the router. If the ACL configuration is changed by an administrator, the predefined ACL file must be manually edited again. Also, SSCR requires the predefine interface’s name which is difficult to be changed during running time, whereas AIPM does not need the predefined interface’s name; all arguments can be retrieved during running time.

The most interesting feature of AIPM is its ability to decide which router’s interface to block. On the other hand, SSCR will block the IP in every router even if the malicious traffic does not pass through the router. In fact, if there are predefined M routers, SSCR will need to establish M connections. $M-1$ connections are considered unnecessary. Evidently, additional overhead as much as $(M-1)/M\%$ will incur when SSCR has to communicate with all routers. To sum up, Table 5-13 shows the comparison of overhead of these two systems.

	SSCR	AIPM
Number of Connections	M for M routers	1
Total overhead	M for M routers	1 for M routers
Connections Wasted	$(M-1)/M\%$	0%

Table 5-18 Comparison of Overhead

5.4 Summary

This chapter offers two system testing cases of AIPM. These successful system testing prove that AIPM can run well in various typical business network. Next, the advantages of AIPM are discussed by comparing AIPM with others IDS AND IPS systems.

University of Malaya

Chapter 6 Conclusion and Future Work

This final chapter summarized the research undertaking. A brief discussion on the thesis contribution is then presented. Some suggestions for future research end the thesis to allow possible development of new ideas and realms.

6.1 Thesis Summary

IDS AND IPS is popular network security tools. Many new technologies such as artificial intelligence have been applied to IDS AND IPS. Nevertheless, some problem may still occur. In particular, false positive and false negatives may happen in IDS, and IPS may not be able to block an attack after being detected by an IDS.

This thesis attempts to propose a method to minimize or avoid the problem of current IDS AND IPS. Many IDS AND IPS are single applications, which means that the systems are developed in one program. AIPM, however, is divided into two parts; one part is IDS, which is Snort, while another is IPS which is built on top of SnortSam. This architecture gains advantages of both Snort and SnortSam.

After some current IDS AND IPS are reviewed, a system, namely, AIPM, to enhance network security by integrating IDS AND IPS and routers is proposed and developed. AIPM has been tested in several network environments; all testing are successful.

Concluding the work accomplished, the objectives (as defined in Chapter 1) have been achieved. Several intruder patterns in network security are studied, and many mechanisms of

IDS AND IPS and network devices enhance network security are analyzed. Targeting on some of the weaknesses, another achieved objective is the development of CISCONATAACL to make blocking decision and perform blocking as accordance.

As a final remark, the implementation of AIPM is a success of this research. The final testing results further imply that the accomplishment is worthwhile in making a contribution to the research community.

6.2 Problems Faced In Development

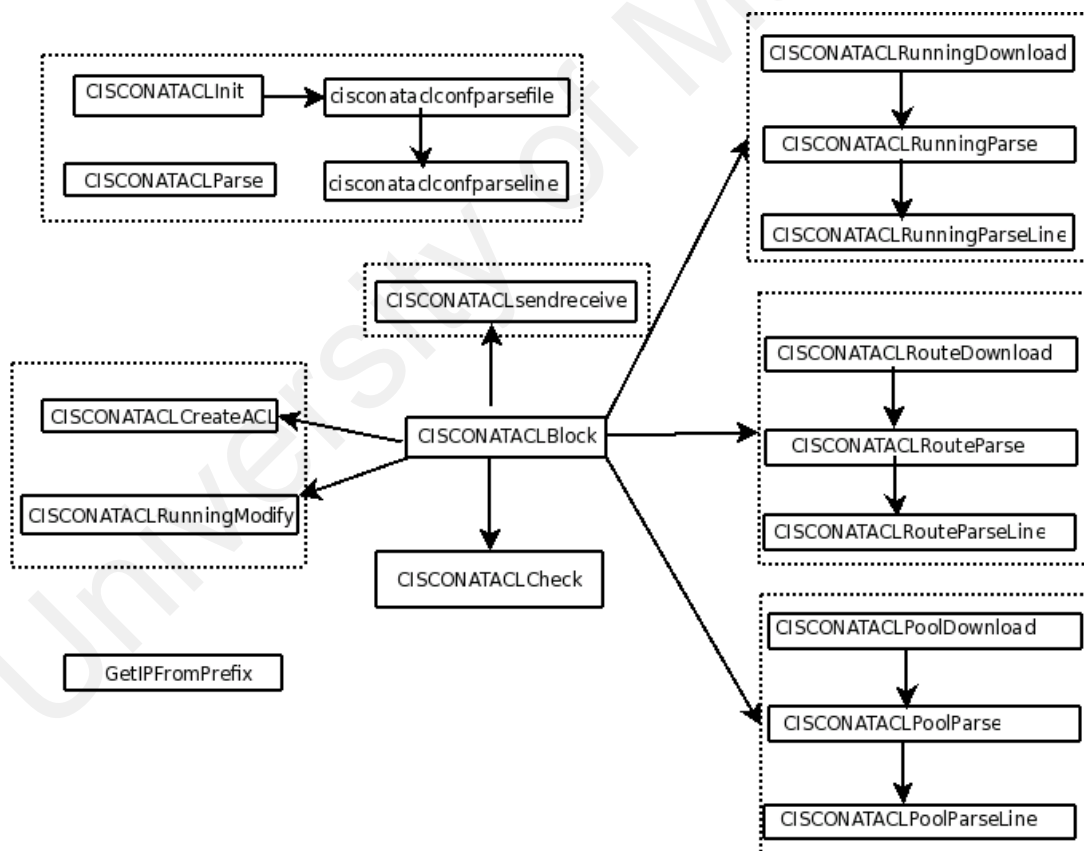


Figure 6-1 Key Components of AIPM

Figure 6-1 shows the key components of AIPM and describes the relationships among these components. 19 key components have been designed and written in the development phase.

These key components are described in Chapter 4. There are many problems faced when analyzing, designing and developing these components.

The first problem faced is when developing part one - “Running-config file processing”. Because there are different format of the running-config file in different routers, the designed part must be able to analyze almost all kinds of running-config files. For example, when the program reads the word “IP” from a downloaded running-config file, it must identify whether the next word is an IP address since the possible words can be DHCP, pool, or A.B.C.D. Figure 6-2 depicts a sample of command tree of Cisco router 2600. In order to embrace as much commands set as possible, this work has collected and analyzed about 30 running-config file samples.

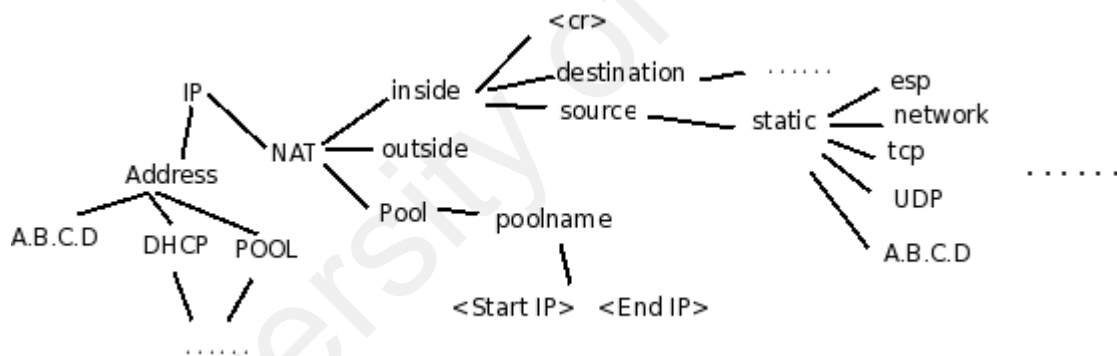


Figure 6-2 Sample of Command Tree of Cisco Router 2600

The second problem faced is the development of part two -“Routing-table file processing”. Similar to the previous problem mentioned, this part also encounters the difficulty of reading different format of routing tables form different routers. Table 6-1 shows a sample routing table. From the table, we can see that there are different format even for same type of routing records.

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
 E1 - OSPF external type 1, E2 - OSPF external type 2
 i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
 ia - IS-IS inter area, * - candidate default, U - per-user static route
 o - ODR, P - periodic downloaded static route

Gateway of last resort is 202.185.109.171 to network 0.0.0.0

C 202.185.109.0/24 is directly connected, Loopback0
 R 192.168.3.0/24 [120/2] via 192.168.1.2, 00:00:17, FastEthernet0/0
 O 6.0.0.0/8 [110/74] via 5.0.0.2, 00:00:49, Serial2
 O E2 192.168.10.0/24 [110/100] via 192.168.1.4, 00:26:50, Fddi0/0
 O 192.168.33.3/32 [110/2] via 192.168.1.3, 00:26:50, Fddi0/0
 R 192.168.66.0/24 [120/10] via 192.168.1.3, 00:00:06, Fddi0/0
 R* 0.0.0.0/0 [120/10] via 192.168.1.3, 00:00:06, Fddi0/0

I 192.168.90.0/24 [100/1110] via 192.168.1.5, 00:01:08, Fddi0/0

i L1 192.168.22.0/24 [115/20] via 192.168.1.2, Fddi0/0
 i L1 194.10.1.0/24 via 55.55.55.1
 i UD 12.2.41.23/32 via 55.55.55.1

D 192.168.33.0/24 [90/156160] via 192.168.1.3, 01:26:38, Fddi0
 D*EX 0.0.0.0/0 [170/2198016] via 192.168.1.3, 01:26:36, Fddi0

S 200.1.1.0/24 via 126.0.0.1
 S* 200.1.1.0/24 via 126.0.0.1

Table 6-1 Sample Routing Table

Another problem faced is when developing part five - “Add ACL processing”. In the early development phase, new ACL can be added to target router but the sequence is incorrect. For example, if the original ACL in a downloaded running-config file is

```
ip access-list extended TESTACL
deny ip host 192.168.0.1 any
permit ip any any
```

then after adding a new ACL entry –“deny ip host 192.168.0.2 any” to the correct position of the downloaded running-config file and uploading it to the target router, the whole ACL in the target router becomes

```
ip access-list extended TESTACL
deny ip host 192.168.0.1 any
```

```
permit ip any any
deny ip host 192.168.0.2 any
```

Obviously, the new ACL entry will never be matched. Since the Cisco router will read the uploaded running-config file and apply the configuration, the sequence of the ACL in the uploaded file should have been correct. However, it was not the case. After checking the Cisco website and testing many times, the solution is found: the clean old ACL command must be run before the uploading the modified running-config file back to the router.

6.3 Thesis Contribution

The most important contribution of this thesis is the proposal of mechanism integrating software (Snort and SnortSam) and network devices (Cisco routers) to enhance network security. This new mechanism minimizes synchronization between software and hardware since almost all information except the predefined information of gateway can be obtained in running time.

The new mechanism can be applied in various network environments even the NAT/PAT technology is running. The main aim of IDS is to detect intrusion and identify intruder. However, if the NAT/PAT technology is applied in a network, typical IDS will not be able to get the real IP of the intruder. This is because the IP address retrieved belongs to the NAT/PAT pool, causing the IDS to trigger a false alert. AIPM solves this problem by parsing the content of downloaded NAT/PAT pool file and by comparing the IP block with the data from the NAT/PAT pool to decide the real IP block.

The last contribution is that this work proves the possibility of integrating IDS AND IPS with network devices to enhance network security. AIPM includes three parts: IDS (Snort chosen in

this thesis) that detects intrusion actions; IPS (SnortSam chosen) that blocks malicious intrusion, and network devices (Cisco routers) which resides closest to the attacker. AIPM integrates them and runs as a single IPS.

6.4 Suggestion for Future Work

As indicated in Chapter 5, although AIPM has some added functionalities, there are still some drawbacks that need to be solved.

First, AIPM only blocks malicious traffic at the network layer (i.e. blocking IP addresses in routers). Therefore, future work should include the ability to block malicious traffic at the data link layer (i.e. blocking MAC addresses in switches).

Second, AIPM uses the TELNET (TELEcommunication NETwork) protocol to establish connections between IPS and network devices. As TELNET does not encrypt any data sent over the connection (including passwords), future work may consider deploying other security protocol, such as the SSH (Secure Shell) protocol, to establish secure connections.

Third, the gateway information must be predefined in AIPM for blocking outside intrusion. To increase flexibility, this predefinition should be removed without affecting the system's overall functionalities.

Finally, current AIPM does not remove blocked IP automatically; future work will try to add the feature of maximum blocking time in order to remove the blocked entry after a defined expired duration.

REFERENCE:

1. Aly El-Semary, Janica Edmonds, Jes'us Gonzalez and Mauricio Papa,, 2005, "Annids: Intrusion Detection System Based On Artificial Neural Network", In Proceedings of international Conference on Fuzzy Systems
2. Amoroso E., Intrusion Detection, 1999, "An Introduction To Intrusion Detection. Correlation. Traps, Trac Back, And Response"
3. "An Introduction to Intrusion Detection" ,2007, <http://www.acm.org/crossroads/xrds2-4/intrus.html>
4. Anderson, J. P., 1980, "Computer security threat monitoring and surveillance"
5. Andrew Mason, 2004, "Network Security and Virtual Private Network Technologies".
6. Aurobindo Sundaram, Dorothy Denning, 1996, "An Intrusion Detection Model,"
7. Balajinath, B., Raghavan, S.V., 2000, "Intrusion Detection Through Learning Behaviour Model"
8. BBC news, 2007, <http://news.bbc.co.uk>
9. Bojarczuk C.E., Lopes H.S. y Freitas A.A., 1999, "Discovering comprehensible classification rules using genetic programming: a case study in medical domain". Proceedings Genetic and Evolutionary Computation Conference GECCO99, 1999.
10. Brian J. d'Auriol and Kishore Surapaneni, 2004, "A State Transition Model Case Study for Intrusion Detection Systems", In Proceedings Of The 2004 International Conference On Security And Management (SAM'04), 2004, Pages: 186-192
11. Chittur, A., 2001, "Model Generation For An Intrusion Detection System Using Genetic Algorithms"
12. Cowan C. et al., 1998, "Stackguard: automatic adaptive detection and prevention of buffer-overflow attacks", In USENIX Security Symposium, pages 63–77, 1998.
13. Crosbie,M., And Spafford, G., 1995, "Applying Genetic Programming to Intrusion Detection"
14. Chittur, A., 2001, "Model Generation for an Intrusion Detection System Using Genetic Algorithms"
15. Caberera, J.B.D., Ravichandran, B., and Mebra, R.K., 2000, "Statistical Traffic Modeling for Network Intrusion Detection", In Proceeding of Modeling, Analysis and Simulation of Computer and Telecommunication Systems conference, 2000
16. Denning, D. E., 1987, "An intrusion detection model. IEEE Trans"
17. Denning, D. E. and Eumann, P. G., 1985. "Requirements and model for IDIS: A real-time intrusion detection system"

18. Earl Carter, Jonathan Hogue, 2006, "Intrusion Prevention Fundamentals", Published by Cisco Press
19. Edward Kern, 2004, "Artificial Intelligence In Network Security"
20. FBI 2006 IC3 Annual Report, 2006,
http://www.ic3.gov/media/annualreport/2006_IC3Report.pdf
21. Fogel, D.B. 1998, "Evolutionary Computation; Second Edition", IEEE Press, 1998.
22. Frank Knobbe., 2006, <http://www.SnortSam.net>
23. Gomez J., Gonzalez F., and Dasgupta D., 2002, "Complete Expression Trees for Evolving Fuzzy Classifier Systems with Genetic Algorithms", In Proceedings of the Evolutionary Computation Conference GECCO02, 2002
24. Intrusion Prevention Systems (IPS), 2004
http://www.nss.co.uk/WhitePapers/intrusion_prevention_systems.htm
25. Ishibuchi H. y Nakashima T., 2000, "Linguistic rule extraction by genetic-based machine learning". In Proceedings of Genetic and Evolutionary Computation Conference GECCO00, 2000.
26. Jonatan G., Dipankar D., 2002, "Evolving Fuzzy Classifiers for Intrusion Detection", In Proceedings of the 2002 IEEE Workshop on Information Assurance
27. Jakub Botwicz, Piotr Buciak, and Piotr Sapiecha, 2006, "Building Dependable Intrusion Prevention Systems"
28. Jonathon T. Giffin Somesh Jha Barton P. Miller, 2004, "On Effective Model-Based Intrusion Detection"
29. Jagannathan R., Teresa Lunt, Debra Anderson, Chris Dodd, Fred Gilham, Caveh Jalali, Hal Javitz, Perter Nrumann, Ann Tamaru, and Alfonso Valdes, 1993, "System design document: Next-generation intrusion detection expert system (NIDES)"
30. Karen Kent, Peter Mell, 2006, "Guide to Intrusion Detection and Prevention (IDP) Systems "
31. Lianying Zhou, Feogyu Liu, 2003, "Research on Computer Network Security Based on Pattern Recognition"
32. Liu J. y Kwok J., 2000, "An extended genetic rule induction algorithm". In Proceedings of the Congress on Evolutionary Computation Conference, 2000.
33. Liu Yan-Heng, Tian Da-Xin and Wang Ai-Min WANG, 2003, "ANNIDS: Intrusion Detection System Based On Artificial Neural Network", In Proceedings of the Second International Conference on Machine Learning and Cybernetics, 2003
34. Matt Bishop., 2003, "What is computer security", Published by THE IEEE COMPUTER SOCIETY

35. Mukkamale S., Hanoshi G. Sung A., 2002, "Intrusion Detection Using Neural Networks And Support Vector Machines, Appears In: Neural Networks", In Proceedings of the 2002 International Joint Conference on Search Security on TechTarget.com
36. McHugh, J. Christie, A. Allen, J. , 2000, "Defending Yourself: The Role Of Intrusion Detection Systems"
37. Mohamad R. Neilforoshan, 2004, "Network Security Architecture", In Proceedings Of Cesc: South Central Conference, 2004
38. Nick Ierace, Cesar Urrutia, and Richard Bassett, 2005, "Intrusion Prevention Systems",
39. Norbik B. I., Bharanidhran S., 2005, "Artificial Intelligence Techniques Applied To Intrusion Detection", In Proceedings Of IEEE Indicon 2005 conference, 2005
40. Odd Nilsen, 2002, "Protection Of Information Assets"
41. Peng N., Yun C., Douglas S. Reeves, and Ddingbang X, 2004, "Techniques And Tools For Analyzing Intrusion Alerts", Published in ACM Transactions On Information And System Security, Vol. 7, No. 2, May 2004, Pages 274–318.
42. Pillai M. M., Eloff J. H.P. and Venter H. S., 2004, "An Approach to Implement a Network Intrusion Detection System using Genetic Algorithms", In Proceedings of SAICSIT 2004, Pages 221 – 228
43. Rehman, R.U., 2003, "Intrusion Detection Systems with Snort: Advanced IDS Techniques with Snort, Apache, MySQL, PHP, and ACID", Published by Prentice Hall PTR, Inc.
44. Robert Drum, 2006, IDS AND IPS PLACEMENT FOR NETWORK PROTECTION
45. Sebring, M. M., Shellhouse, E., Hanna, M. E., AND Whitehurst, R. A., 1988, "Expert systems in intrusion detection: A case study". In Proceedings of the 11th National Computer Security Conference, Pages: 74–81.
46. Sinclair, C., Pierce, L., And Matzner; S., 2004, "An application of machine learning to Network Intrusion Detection"
47. Shan Zheng, Chen Peng, Xu Ying, Xu Ke, 2001, "A Network State Based Intrusion Detection Model"
48. Steven A. Hofmeyr, Stephanie Forrest, Anil Somayaji, 1997, "Lightweight Intrusion Detection for Networked Operating Systems"
49. Snort project term, 2007. "Snort manual", <http://www.Snort.org/docs>
50. Susan M. Bridges, Rayford B. Vaughn, 2000, "Intrusion detection via fuzzy data mining", In Proceedings of The 12th Annual Canadian Information Technology Security Symposium 2005.
51. Susan M. Bridges, 2000, "fuzzy data mining and genetic algorithms applied to intrusion detection", Proceedings of the 23rd National Information Systems Security Conference.

52. Timothy Hollebeck and Rand Waltzman, 2005, "The Role of Suspicion in Model-based Intrusion Detection", In Proceedings of the 2005 workshop on New security paradigms,
53. Verwoerd T., and Hunt, R., 2001, "Intrusion Detection Techniques and Approaches"
54. Vidar Ajaxon Grønland , 2006, "Building IDS rules by means of a honeypot"
55. Wang Jing-xin, Wang Zhi-ying, Dai Kui, 2003, "A Network Intrusion Detection System based on the Artificial Neural Networks"
56. Wei Li, 2003, "Using Genetic Algorithm for Network Intrusion Detection", <http://www.security.cse.msstate.edu/docs/Publications/wli/DOECSG2004.pdf>
57. Xinyou Zhang, Chengzhong Li, and Weibin Zhen, 2004, "Intrusion prevention system design"
58. Xinzhou Q, Wenke L., Lundy L. Jocio B.D. Cabrera , 2002, "Integrating Intrusion Detection and Network Management"
59. Yuebin Bai, Hidetsune Kobayashi, , 2003a, "intrusion detection system: technology and development", Proceedings of the 17th International Conference on Advanced Information Networking and Applications, page(s): 710-715
60. Yuebin Bai, Hidetsune Kobayashi, 2003b," New String Matching Technology for Network Security", Proceedings of the 17th International Conference on Advanced Information Networking and Applications, page(s): 198 – 201