NETWORK ACCESS AND ADMISSION RESTRICTION USING TRAFFIC MONITORING AND VULNERABILITY DETECTION

A Thesis submitted to Gujarat Technological University

for the Award of

Doctor of Philosophy

in

Computer Science

by

Shalvi Darshan Dave
Enrollment No: 119997493014

under supervision of

Dr. Bhushan Trivedi

GUJARAT TECHNOLOGICAL UNIVERSITY
AHMEDABAD

December 2015

NETWORK ACCESS AND ADMISSION RESTRICTION USING TRAFFIC MONITORING AND VULNERABILITY DETECTION

A Thesis submitted to Gujarat Technological University

for the Award of

Doctor of Philosophy
in
Computer Science

By
Shalvi Darshan Dave
Enrollment No. 119997493014

under supervision of

Dr. Bhushan Trivedi

GUJARAT TECHNOLOGICAL UNIVERSITY
AHMEDABAD

December 2015

DECLARATION

I declare that the thesis entitled Network Access and admission restriction using traffic monitoring and vulnerability detection submitted by me for the degree of Doctor of Philosophy is the record of research work carried out by me during the period from July 2011 to January 2015 under the supervision of Dr. Bhushan Trivedi and this has not formed the basis for the award of any degree, diploma, associateship, fellowship, titles in this or any other University or other institution of higher learning.

I further declare that the material obtained from other sources has been duly acknowledged in the thesis. I shall be solely responsible for any plagiarism or other irregularities, if noticed in the thesis.

Signature of the Research Scholar : …… Date:……

Name of Research Scholar: Shalvi Darshan Dave

Place: Ahmedabad

CERTIFICATE

I certify that the work incorporated in the thesis Network Access and admission restriction using traffic monitoring and vulnerability detection submitted by Smt. Shalvi Darshan Dave was carried out by the candidate under my supervision/guidance. To the best of my knowledge: (i) the candidate has not submitted the same research work to any other institution for any degree/diploma, Associateship, Fellowship or other similar titles (ii) the thesis submitted is a record of original research work done by the Research Scholar during the period of study under my supervision, and (iii) the thesis represents independent research work on the part of the Research Scholar.

Signature of Supervisor: …… Date: ……

Name of Supervisor: Dr. Bhushan Trivedi

Place: Ahmedabad

Originality Report Certificate

It is certified that PhD Thesis titled Network Access and admission restriction using traffic monitoring and vulnerability detection by Shalvi Darshan Dave has been examined by us. We undertake the following:

a. Thesis has significant new work / knowledge as compared already published or are under consideration to be published elsewhere. No sentence, equation, diagram, table, paragraph or section has been copied verbatim from previous work unless it is placed under quotation marks and duly referenced.

b. The work presented is original and own work of the author (i.e. there is no plagiarism). No ideas, processes, results or words of others have been presented as Author own work.

c. There is no fabrication of data or results which have been compiled / analysed.

d. There is no falsification by manipulating research materials, equipment or processes, or changing or omitting data or results such that the research is not accurately represented in the research record.

e. The thesis has been checked using **Turnitin** (copy of originality report attached) and found within limits as per GTU Plagiarism Policy and instructions issued from time to time (i.e. permitted similarity index <=25%).

Signature of the Research Scholar : …… Date: ……

Name of Research Scholar: Shalvi Darshan Dave

Place : Ahmedabad

Signature of Supervisor: …… Date: ……

Name of Supervisor: Dr. Bhushan Trivedi

Place: Ahmedabad

PhD THESIS Non-Exclusive License to GUJARAT TECHNOLOGICAL UNIVERSITY

In consideration of being a PhD Research Scholar at GTU and in the interests of the facilitation of research at GTU and elsewhere, I, Shalvi Darshan Dave having (Enrollment No.) 119997493014 hereby grant a non-exclusive, royalty free and perpetual license to GTU on the following terms:

a) GTU is permitted to archive, reproduce and distribute my thesis, in whole or in part, and/or my abstract, in whole or in part (referred to collectively as the "Work") anywhere in the world, for non-commercial purposes, in all forms of media;

b) GTU is permitted to authorize, sub-lease, sub-contract or procure any of the acts mentioned in paragraph (a);

c) GTU is authorized to submit the Work at any National / International Library, under the authority of their "Thesis Non-Exclusive License";

d) The Universal Copyright Notice (©) shall appear on all copies made under the authority of this license;

e) I undertake to submit my thesis, through my University, to any Library and Archives. Any abstract submitted with the thesis will be considered to form part of the thesis.

f) I represent that my thesis is my original work, does not infringe any rights of others, including privacy rights, and that I have the right to make the grant conferred by this non-exclusive license.

g) If third party copyrighted material was included in my thesis for which, under the terms of the Copyright Act, written permission from the copyright owners is required, I have

obtained such permission from the copyright owners to do the acts mentioned in paragraph (a) above for the full term of copyright protection.

h) I retain copyright ownership and moral rights in my thesis, and may deal with the copyright in my thesis, in any way consistent with rights granted by me to my University in this non-exclusive license.

i) I further promise to inform any person to whom I may hereafter assign or license my copyright in my thesis of the rights granted by me to my University in this non-exclusive license.

j) I am aware of and agree to accept the conditions and regulations of PhD including all policy matters related to authorship and plagiarism.

Signature of the Research Scholar: _____

Name of Research Scholar: Shalvi Darshan Dave

Date: _____ Place: Ahmedabad

Signature of Supervisor: _____

Name of Supervisor: Dr. Bhushan Trivedi

Date: _____ Place: Ahmedabad

Seal:

Thesis Approval Form

The viva-voce of the PhD Thesis submitted by Smt. Shalvi Darshan Dave (Enrollment No. 119997493014) entitled Network Access and admission restriction using traffic monitoring and vulnerability detection was conducted on …….…… (day and date) at Gujarat Technological University.

(Please tick any one of the following option)

☐ The performance of the candidate was satisfactory. We recommend that he/she be awarded the PhD degree.

☐ Any further modifications in research work recommended by the panel after 3 months from the date of first viva-voce upon request of the Supervisor or request of Independent Research Scholar after which viva-voce can be re-conducted by the same panel again.

(briefly specify the modifications suggested by the panel)

☐ The performance of the candidate was unsatisfactory. We recommend that he/she should not be awarded the PhD degree.

(The panel must give justifications for rejecting the research work)

---------- ----------

Name and Signature of Supervisor with Seal

1) (External Examiner 1) Name and Signature

---------- ----------

2) (External Examiner 2) Name and Signature

3) (External Examiner 3) Name and Signature

ABSTRACT

Network is a mix of healthy, potentially vulnerable and already infected hosts. One would always want to keep the already infected as well as potentially vulnerable hosts separate from healthy hosts. The objective behind this is to stop transitive spread of vulnerability from one infected host to another and also prevent potential attack on other shared resources. Network admission solutions try to solve the problem of keeping such hosts in a separate functioning section from clean hosts. Network administrators normally prevent the vulnerable and infected hosts from accessing resources outside the network periphery, in lieu of the fact that they may launch an attack on outside resources which may backfire on the network performance itself.

Along with applying these rectifying methods, one would also want to prevent such incidents from happening by protecting potentially vulnerable but not infected applications either by restricting access. Most of the solutions implement this security by denying access of admission of such potentially vulnerable or infected hosts into network and by restricting overall network as well as Internet access for them. This implementation may effectively work if we considered every potentially vulnerable host as infected hosts. We categorize potentially vulnerable host and an infected one as: A windows machine running with lower service update version is considered as potentially vulnerable but a machine having a virus is an infected host. Tight security restrictions in organization may lead to productivity loss if absence of security update prevents a user to access official mails.

Our research is a concentrated effort to maximize alert capturing with context information, simplify alert aggregation using a novice approach, without compromising with network performance. We propose a model that can be as secure as conventional solutions without compromising network productivity. The proposed model advocates the fact that not the machine but the application running on that host is either potentially vulnerable or infected so the access control policy focuses on the process and not at a gross level of host. The model is designed to keep the compromised applications separate from clean applications. It also builds up the run time data information for such applications running in the network to maintain the global black list of such applications with highest confidence level. Our research also generates precise set of advices to network administrator for attack prevention in real time.

Acknowledgement and / or Dedication

Endless thanks go to Lord Almighty for all the blessings he has showered onto me, which has enabled me to write this last note in my research work. During the period of my research, as in the rest of my life, I have been blessed by Almighty with some extraordinary people who have spun a web of support around me. Words can never be enough in expressing how grateful I am to those incredible people in my life who made this thesis possible.

As they say, any journey cannot be accomplished without the able support of people who directly or indirectly lend their time or resources, in order to reach your goal. First and foremost, I would like to convey my heartfelt gratitude to my parents, my husband, my son and my entire family for their continuous motivation and adjustments during this crucial tenure.

I would like to thank from the bottom of my heart to my supervisor Dr. Bhushan Trivedi, for believing in me, even when i gave up from time to time. His blessings, motivation and constant support in worst and best times, has finally paid off in terms of our research. It would not have been possible for me to constantly strive for better performance without his extraordinary vision. As a teacher and a guide, he never leaves the hands of his students. He is with you like a shadow during good and bad. He has always believed in me more than anyone else. I would also like to sincerely thank Dr. Pramode Verma, Dr. Haresh Bhatt and Dr. D.B. Choksi for their frank reviews and suggestions which made our journey a lot easier.

I would like to address special thanks to the unknown reviewers of my thesis, for accepting to read and review this thesis. I wish to thank the authors, developers and maintainers of the open source used in this work. I would like to appreciate all the researchers whose works I have used, initially in understanding my field of research and later for updates. I would like to thank the many people who have taught me starting with my school teachers, my undergraduate teachers, and my post graduate teachers.

Last but not the least; I would like to thank Cyberoam Technologies Pvt. Ltd. for providing

us with industry support for developing our research prototype model and also for funding our patent filing procedure. A special thanks to Mr. Jimit Mahadevia, for his able inputs and technical guidance, which made our research industry compliant.

Table of Content

List of Abbreviation

NAC: Network access and admission control.

IDS: Intrusion detection systems

IDPS: Intrusion detection and prevention system.

HIDPS: Host based Intrusion Detection and Prevention Systems

NIDPS: Network based Intrusion Detection and Prevention Systems

VLAN: Virtual local area network

HIDS: Host-based intrusion detection system

NIDS: Network-based intrusion detection system

NI: Network interceptor

EC: Event Collector

TP: True Positive

TN: True Negative

FP: False Positive

FN: False Negative

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances

APHIDS: A Programmable Hybrid Intrusion Detection System

UTM: Unified Threat Management

PCF: Partial Completion Filters

AIA: Application Intercepting Agent

NI: Network Interceptor

PMU: Pattern Matching Unit

ACM: Aggregation and Correlation Module

AQM: Application Quarantine Module

SSH: Secure Shell

SSL: Secure Socket Layer

DMZ: Demilitarized Zone

DOS: Denial Of Service

DDOS: Distributed Denial Of Service

NAT: Network Address Translation

VPN: Virtual Private Network

List of Abbreviation

WTLS: Wireless Transport Layer Security

DIDS: Distributed Intrusion Detection Systems

NSM: Network Security Monitor

IE: Internet Explorer

List of Figures

List of Tables

CHAPTER – I

Introduction

1.1 Background:

The past decade has witnessed a phenomenal growth of Internet. The term Internet was coined from Internetwork, which conventionally means bringing multitude of networks on a common platform. This is being possible irrespective of the networks with different size, scale and operational mechanisms. Traditionally, we define network as a set of heterogeneous resources operating in a shared environment. These resources can be anything from a host machine, printer, and server to a router. Different applications running on host machine access these shared resources periodically. Network with maximum throughput allows multiple applications accessing same resource at the same time instance. However, resource sharing invites problems such as confidential data outflow, virus or other attacks, surfeit bandwidth utilization etc.

1.1.1 Network access and admission control:

Network Access and admission control (NAC) is a set of methodologies, which are defined and implemented with the purpose of optimizing resource sharing without compromising security of the network [1]. The prima facie of NAC is to allow access / restrict end systems from accessing critical resources based on their "health". End systems can be traditional host machines, printers, IP phones, IP security cameras, etc. "Healthy" host / end system is the one which is assessed by NAC system and is granted access to shared resources.

Assessment of host determines the overall health and the type of privileges it ought to receive. State of a host includes current version of operating system, anti-virus signature, status of firewall, installed software or patches. The NAC can operate in two modes; the first utilizes basic facilities provided by the operating system of the host to report back or

special processes known as agents to shoulder the responsibility of reporting. This information is relayed to a centralized controller using Agent-based or Agent-less NAC technique depending on the mode chosen by the administrator. Agent-based NAC depend on specially designed operating system independent agents to retrieve host information. Agent-less NAC utilizes operating system's management interface to query operating system and receive snapshot for that checkpoint. From this information and information received at past checkpoints, the controller categorizes a host as healthy or unhealthy. The unhealthy hosts need restriction. VLAN steering switches are instructed to divert traffic to and from such unhealthy hosts or processes residing on such hosts in other direction for better control in case of unhealthy hosts. It is possible to manage these switches remotely and usually from a central place it is possible to monitor and control the entire network. The remote controlling of this switch is possible to be done by a few methods. One is to use SNMP commands when SNMP client is running on the host under consideration. The other option is to use remote login methods like SSH or Telnet to send commands to the target host.

NAC Components:

FIGURE 1.1 shows the components that define basic NAC methodology:

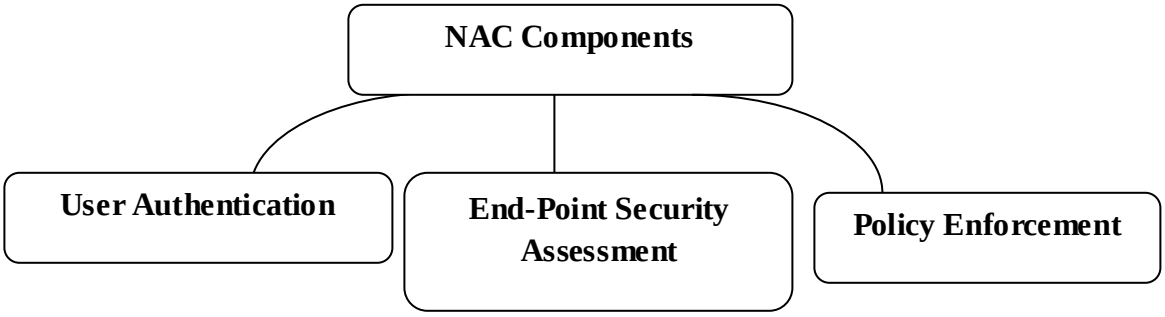


FIGURE 1.1 NAC Components

Functional Specifications of NAC:

Authentication: Authentication is the first step any end system needs to comply with, before it is granted entry into a network. Authentication is the process of verifying user credentials. It can be a simple username/password authentication scheme, iris scan,

fingerprint scan or any other complex procedure. A successful authentication is followed by authorization. Authorization process is in essence implementation of set of pre-defined policies, which includes level of access allowed, time of the day when access is allowed, set of resources allowed for access etc. Citing an example for the same, a clerk is authorized to utilize printer only during job hours whereas a manager is authorized for the same resource without any time restriction. Once authentication and authorization process is complete, NAC allows the entry of end system to the network [1] [2].

Assessment and Validation: An end system, authenticated and authorized, continues to access resources of the network. However, this does not guarantee that the end system cannot misbehave. Post admission to the network, the health of the end system can be compromised. This can happen due to a misbehaving application running on the end system itself, or through an attack from a resource outside the network. Because of this, there is a tendency to spread this attack transitively to other systems. Assessment is the functionality to keep a tab on activities on each end system in the network. Periodic monitoring of end systems includes check of current version of operating system, anti-virus signature, status of firewall, installed software or patches etc. NAC assessment commonly falls into two different arenas: Agent-based and Agent-less [3]. Agent-based NAC rely on software / program known as an agent. This agent needs to be installed on every end system for assessment. Agents are either persistent or dissolvable in nature. Persistent agent, name per se, remain installed on the host and stays in running mode. Dissolvable agent, are basically JAVA or ActiveX components, and disappear after they are utilized. Agent-less NAC perform assessment using remote scanning methods, such as running a vulnerability scan, or by using RPC (remote procedure call) or WMI (Windows Management Instrumentation) to query an end system [3]. Then again, passive scanning, using intrusion detection looks for malicious intent based on actual traffic monitoring.

Based on the assessment conducted by NAC, an end system is validated using the pointers provided. If any threat or vulnerability is detected, the process of validation is activated [3] [4]. Validation ensures that the end system with vulnerabilities is removed from normal network access and shifted to a quarantine area. The shifting of unhealthy system to quarantine area is facilitated by technique known as VLAN steering. VLAN steering technique instructs switches to divert traffic to and from such unhealthy end systems or processes residing on hosts in other direction for better control in case of vulnerability. These switches can be managed remotely and usually from a central location. One of the

methods uses SNMP commands when SNMP client is running on the host under consideration [6]. The other option is to use remote login methods like SSH or Telnet to send commands to the target system. Figure 1.2 shows the technique of VLAN steering to quarantine an unhealthy host:

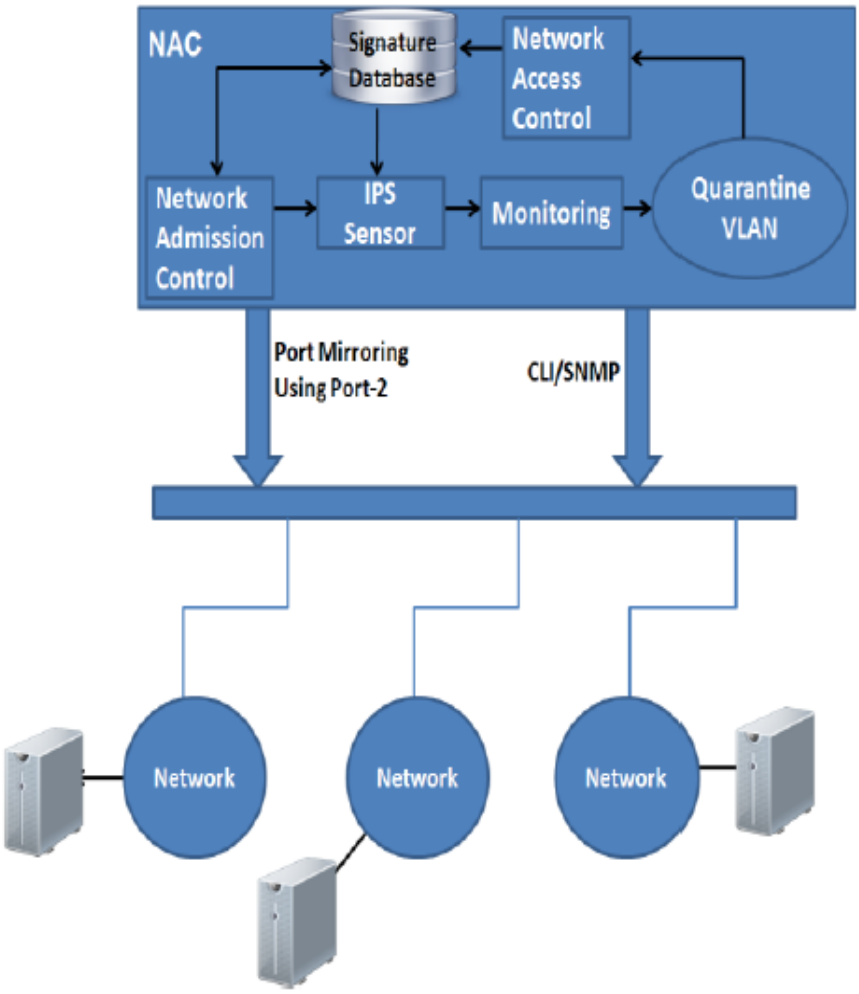


FIGURE 1.2 VLAN Steering for Host Quarantine

Policy Enforcement:

Policy enforcement process ideally involves very granular (i.e. flow-based) network communication policies and not just simple VLAN assignment [6] [7]. After all, putting all of the "unhealthy" end systems into the same quarantine VLAN means they would cross-infect each other with new vulnerabilities. Policies designed by network administrators describe how incoming traffic on switch ports should be handled related to filtering, prioritizing and cataloguing. Remediation is the process of rectifying or removing a

problem / threat in order to become amenable with these pre-defined policies. A remediation process as part of a NAC solution allows users placed in a network quarantine state to get back into healthy state.

There are many policy enforcement methods defined for NAC. The following table, Table 1.1 describes the advantages and disadvantages of each of them:

TABLE 1.1 Comparison of NAC policy enforcement techniques

Methods	PROS	CONS
802.1x	• Highest security standards and authentication based Multiprotocol. • In 802.1x Authentication and authorization occurs before a host even accesses the network. Multiple authentication schemes are supported, and 802.1X is designed to be extensible as new technology arrives. • Centralized administration, Good scalability, Real time detection, High level of security this are characteristics of 802.1x • Very familiar model to end-users ,broadest platform support, handles • Guest user best. • Hosts that do not support 802.1x can be granted access to the network using manually configured exception by MAC address. • Implementing RADIUS and EAP protocol	• For compliance check must use an agent software • Some times its difficult or complex to deploy • All elements on the network must be configured to use 802.1x • Difficult manageability. Not all of networking element can support 802.1x and Not all of the elements residing on the network are 802.1x capable (AS-400, Printers etc.) • It is possible to spoof the MAC address of an exception element is order to receive the same access that element has to enterprise network. • The cost for implementing a solution which is based on 802.1x is currently HIGH (time, resources, infrastructure, upgrade, etc.)
DHCP	• DHCP is easy to install and configure. Because DHCP is well supported, it will work with any host that uses DHCP to request an IP address. • Easy to intercept requests for IP assignment. • Fewer devices are required because it only needs to be in-line with the DHCP server. • It is Easy and fast to deploy.	• Control is easily bypassed by devices assigning a static IP. • Detected elements are only those using DHCP. • Won't catch devices behind a NAT device (i.e. routing WAP). • Detection of elements is done at Layer 3 only. • An element can connect to the network without being detected. Access to at least the local subnet will not be restricted
ARP	ARP is used in any IP host and will always work without any configuration changes to hosts.	Cannot solve the problem of network traffic.
Inline Block	• An in-line device checks traffic shaping and catch attack. • Elements detection is performed at layer 3.	• Deployment is time consuming. Deployment must involve a network re-architecture. • Some elements may only generate layer 2 traffic. Elements can infect and entering on their local subnet and cannot be stopped. Elements detection is checked only at layer 3.
Walled Garden	Like DHCP and ARP management, this method works with any host that uses DNS. Often, the user ends up at a Web page, where he has to authenticate or accept an agreement. Relies on the fact that users will eventually use a Web browser and thus be redirected to a Web page.	Not widely used.

Problem Areas:

We have identified and summarized the following problem areas of NAC [3] [6] [7]:

1) Security assessment of host periodically checks for currently installed programs / software patches applicable, operating system and kernel routines. This assessment is very time and resource consuming. These assessments also have a tendency to generate many false alarms.
2) Access control enforcement usually quarantines the potentially vulnerable host / infected host after assessment and validation fails. This affects the overall productivity of network and also hampers network performance.
3) Periodic scan for host assessment cannot run at higher frequency due to efficiency reasons, hence there are high chances that any host starts violation of policy amid two scans. That means after the host is recognized as healthy, it might change its status to unhealthy and the monitor missed that. It might be again able to change its state back to healthy to avoid detection.

1.1.2 Intrusion Detection and Prevention systems (IDPS): A means to achieve NAC capabilities:

NAC functionalities are achieved by firewalls, intrusion detection systems or intrusion detection and prevention systems. The choice of using any of the above relies on the complexity of security policy defined and considered necessary for the network. Firewalls operate on a pre-defined policy of allowing / blocking traffic to and fro the network. They are installed on network boundary and require human intervention periodically to instruct for allowing or denying the traffic. They are incapable of inspecting the contents of traffic which flow through the network.

IDPS complement firewalls and other security devices employed for a network. IDS keep track of hostile actions or attacks in the network. After an attack is detected, the event is logged by the IDS in a database or a file. An alarm is generated signalling the presence of an attack.

Brief Background of IDPS:

Intrusion Detection term was first and foremost coined by Anderson [4]. His work describes the core functionalities performed by an Intrusion Detection System (IDS). He defines an intrusion as a potential threat or a deliberate attempt by an intruder, in order to secure access to critical network resources, harm or render system functionalities slowdown or to retrieve and manipulate vital information. Anderson [1] summarizes intrusion as:

- Penetration where an attempt to retrieve or gain access to crucial system files is successful. Also, such access is unauthorized and is termed as successful attack.
- Risk is unintentional, unpredictable or accidental leak of information or data, caused due to hardware malfunctioning or wrong software design.
- Vulnerability as opposed to risk is known design or hardware fault that results in misuse of system data.
- Attack is defined as a thoughtfully sketched plan and successful execution of the same.

Followed by this was Denning's work [5] which presents basic principle behind almost all existing IDS today. Since then, multitude of IDS and Intrusion Detection and Prevention Systems (IDPS) have been designed and implemented. All of them majorly work towards securing network from security threats and violations.

Even after an attack on network, an IDPS analyzes system audit data to ascertain the extent to which the attack has caused damage on network. IDPS can thus prevent such attacks in future, making them valuable as static alert analyzers or real-time preventers. A summarized report [18] on recent attack trends says:

- Data and Resource misuse and malfunctioning has increased by almost 250% in last ten years.
- Major corporate companies around the world suffer from at least one malicious incident.
- Loss of approximately $25 billion is reported by major software and telecom industries in United States alone.

The above study proves that even though many popular and successful IDPS exist today, still major work needs to be accomplished to achieve security benchmark without compromising performance. Accuracy and performance is a big trade-off faced by almost all existing IDPS.

Need for IDPS:

Ideally, we define intrusions as attacks against network resources such as crucial services, data-centric attacks on client / server applications, host-based attacks like privilege acceleration, illicit logins and access to system sensitive files, or malware akin to viruses, worms and Trojans. This leads to compromise in the availability, breach of confidentiality or malfunctioning of a network resource. Such intrusions leads to critical denial of services, response failure of system or important database compromised. Intrusion detection and prevention systems are means of detecting unauthenticated or unauthorized use of a network resource or attacks imposed on the same.

Intrusion Detection and prevention systems are implemented as a combination of software and hardware to detect and report such malicious actions. An Intrusion Detection and Prevention System (IDPS) ideally operates behind the firewall as shown in Figure 1.3, looking for evidences in patterns in network packet payload that may lead to an attack on network resources. Thus, IDPS are used as the second and ultimate level of defence in any corporate or enterprise network against attacks to protect crucial network resources.

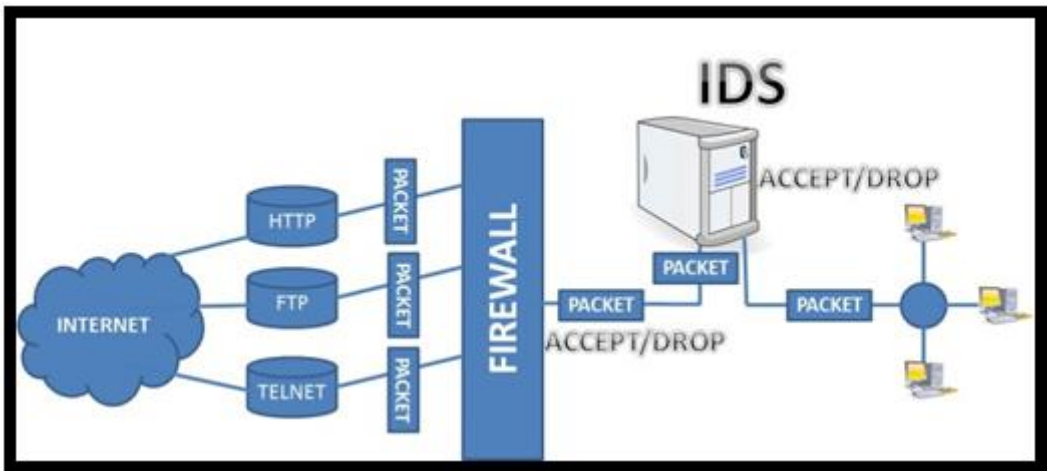


FIGURE 1.3 An archetypical deployment of firewall and IDPS

The fundamental question that needs to be answered is: Why do we deploy an IDPS, in presence of a firewall? In order to answer this, we need to first understand the capabilities and limitations of a firewall, and how an IDPS complements it. By understanding this basic functional difference, we realize why we require both firewall and IDPS to provide concrete security. Firewalls are incapable of detecting or handling typical network and application layer attacks such as Denial of Service and Distributed Denial of Service attacks, scan attacks, or Trojans. Due to exponential growth of the Internet, the high probability of threat scenario prevalent over the Internet has been the basic reason for the network security experts to take into consideration IDPS as fundamental aspect for network security.

One more reason for IDPS popularity is attacks launched by internal authorized users along-with attacks from external entities. Internal users basically are fraudulent employees misusing their privileges for launching attacks. These internal frauds cannot be prevented by a firewall since firewalls are designed to block unwanted network traffic into or out of any network. Packet filtering firewalls normally scan a packet payload for layer 3 and layer 4 protocol information. Their defence mechanism is not dynamic. As a result, IDPSs, originating in the year 1980, was introduced by Anderson [4]. It was much later established formally in 1987 by Denning [5]. Since then, much advanced research has been conducted on IDPS in the recent years.

Classical Approaches of IDPS:

Traditional Intrusion detection systems follow two diversified approaches for attack detection: signature-based or anomaly-based [8] [9] [10]. A signature-based IDPS relies profoundly on pre-defined signatures stored in a signature database [11]. These signatures are configured keeping in mind various types of attacks identified for detection. If an attack occurs, then an entry is made into the database. The signatures are reactive in nature because until the threat is known and a remedy exists for the threat, a signature cannot be created. Figure 1.4 describes a generalized model of signature-based IDPS:

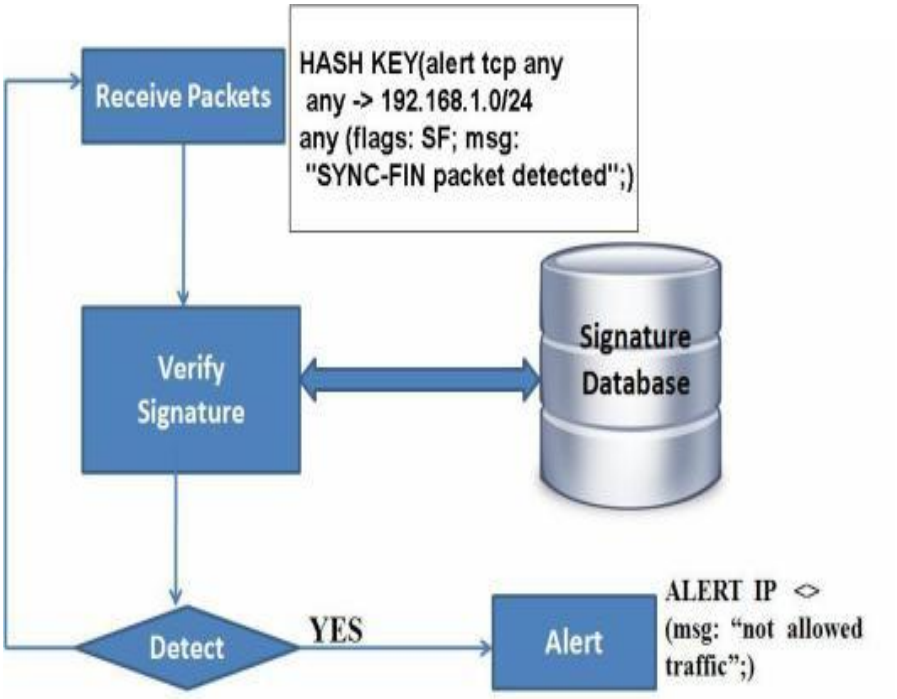


FIGURE 1.4 Signature based IDPS

Anomaly-based detection systems have a baseline defined for behaviour of systems [12]. Activities that comply with this baseline are termed as "normal" and which do not adhere to the baseline is termed as "abnormal" behaviour. Baseline or normal behaviour for a system, for instance, kernel information, system log, network packet payload, application-running information; operating system information etc is stored into the database. Monitoring traffic for abnormal behaviour increases the chances of detecting previously unknown attacks [12]. Figure 1.5 shows a typical anomaly based IDPS:

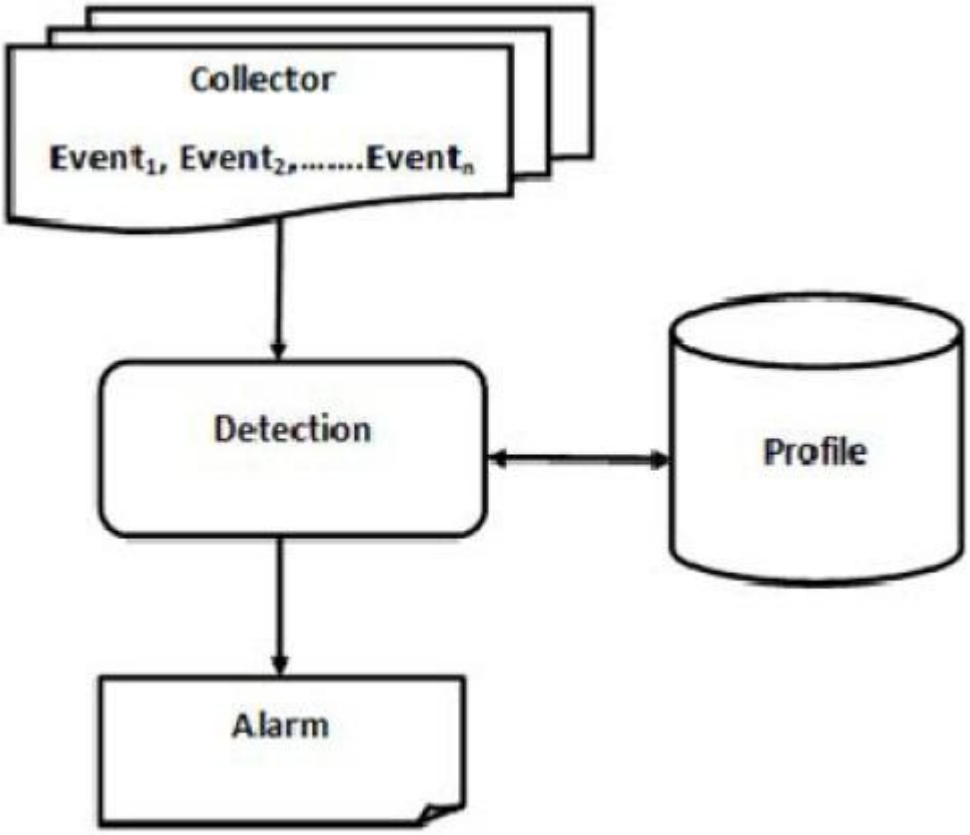


FIGURE 1.5 Anomaly based IDPS

Existing Scenario, Challenges and drawbacks of IDPS:

Intrusion Detection and Prevention Systems (IDPS) are nowadays considered benchmark in network security arena. Highly advanced open-source and commercially available IDPS strive to protect critical network resources against malicious attempts. Two fundamental categories of IPDS are either misuse-based or anomaly-based [13]. Misuse-based IDPS has low false positive rate but they are unable to detect novel or unknown attacks [13] [14]. Anomaly based detection is capable of detecting unknown attacks but is prone to high false positive rate [15] [16]. Many hybrid IDPS have been introduced recently [17] [18] [19]. However, our literature survey section describes in detail how these hybrid IDPS still face challenges in detecting complex attack scenarios. These IDPS face major challenges in detecting related attack scenarios, high false positive rate, and correlating raw alert log with precision. They are unable to distinguish between actually malicious or benign attacks. They simply raise alarms on ad-hoc basis for each possible attack detected.

Network administrator is usually burdened with the task of mining this raw alert log, which is generally a large mass of unrelated, single attack instances. Also, the entire process of detection, alert generation and prevention hampers overall network throughput and productivity. Finally, the analyzed alert log presented to the network administrator is not precise enough for taking adequate preventive actions [19].

One of the major challenges faced by an IDPS in current scenario is to segregate various logs such as that produced by router, system, firewall and host level IDPS alerts, with alerts from a network based IDPS. This either requires a skilled IDPS analyst or an automated implementation. In both the cases, one needs to be constantly aware about latest attack trends, Trojans, viruses, latest operating system advancements and accordingly keep the signatures accurate and in sync with latest intrusions.

1.2 Motivation and Objectives:

Intrusion detection and prevention systems are designed and deployed to secure networks against intruders. These systems are normally the very first barricade any network traffic has to overcome. Smart hackers are crafting advanced techniques to override the detection capabilities of existing IDPS.

This thesis first considers the existing problems which need to be addressed in network security scenario and which are the baseline for motivation and objectives of this research.

Problem 1: Inability of existing IDPS to correctly identify attack direction: We identify two types of intruders: external intruders and internal intruders. External intruders reside outside the periphery of the network and craft attacks on network resources to extract crucial information or data. Internal intruders again can be classified as: vulnerable intruders and malicious intruders. Vulnerable intruders are victims of attacks launched on them from other sources. They themselves do not carry any malicious intent but because they are victimized, they spread the vulnerability to other resources on the network. Malicious intruders reside within the network, having access to critical data and harm the network internally by launching attacks from within the network. Existing IDPS solutions fail to identify and segregate intruders as external or internal.

Problem 2: Real-time attack identification with precision: Yet another functionality of IDPS is traffic monitoring and assessment for vulnerabilities. For traffic monitoring, host-based IDPS perform techniques such as port scan or system scan. This hampers host performance and is very time consuming. Continuous context switching increases processor load also. As against this, Network-based IDPS inspect network packets for possible intrusions. Inspecting each and every packet again is cumbersome process. Furthermore, since packets are inspected as and when they arrive, in case of distributed attacks, Network-based IDPS fail to detect them.

Even after an attack is detected, IDPS solutions fail to identify the actual reason behind the attack and the cause of attack. An attack source details are normally IP address and port no with timestamp. If the attacker is an external entity, and is not a vital resource for network, a simple remedy such as blocking the host machine will suffice. On the other hand, if an attacker is an internal entity or in case of external entity, an important resource for the network, simply blocking the host will hamper network performance. The need of

the hour is to extract as much information about the attack as possible for precise identification of the application which caused the attack along with version, timestamp, criticality etc.

Problem 3: Aggregation and Correlation of raw alert mass and false positive rate: Any suspicious activity in the network detected by IDPS is stored in a database. This database consists of raw alerts generated and stored by IDPS under suspicion. Not all of the alerts are attacks. Each and every alert needs to be analyzed for attack identification. For precise identification of an attack, two scenarios should be considered:

1. A single raw alert itself might be an attack.
2. In case of distributed attacks or attacks which stretch across multiple alerts, proper aggregation of related alerts is necessary.

Proper aggregation of alerts reduces the no of false positives present in raw alert mass. This also helps the network administrator in taking precise preventive actions since he only needs to analyze actual alerts.

Problem 4: To maintain optimal network productivity while security enforcement

The main focus for an IDPS is to provide maximum possible defense against attacks. However, during peak network traffic, there is complete possibility of network productivity being compromised. A typical IDPS action involves blocking malicious hosts in case of attacks. In a typical case of network with n hosts, consider a case where IDPS blocks m hosts (m=50%n). If out of the blocked m hosts, p number of hosts is running critical services, the network productivity goes down by almost 50%, which is not desirable. Thus, security should be provided keeping network productivity at benchmark level.

The main aim of this thesis is to provide exhaustively analyzed raw alerts, correlated and aggregated, with minimum false positives and maintaining optimal network productivity. The secondary goal is to generate set of correct and precise advices for network administrator for taking preventive actions. In particular, our research will focus on the objectives described below:

1. Propose and incubate a hybrid model which incorporates efficient features of host-based and network-based IDPS. Traffic monitoring will be done locally and in case of an alert, information for the alert will be stored centrally in a database. Assessment of raw alerts and analysis will be done centrally.
2. Real-time attack identification.
3. Bifurcating raw alerts and related alerts. Correlating related alert into a single attack scenario.
4. Reducing false positives by more than 95% with optimal network productivity.
5. Final alert log ably supported by confidence level for attacks.
6. Generate precise advice for prevention of attacks.

1.3 Keywords:

Traffic Monitoring, Network access control, Network admission control, Authentication, Authorization, Policy Enforcement, quarantine, assessment, vulnerability detection, Intrusion detection systems, firewalls, Intrusion detection and prevention systems, false positives, raw alerts, alert correlation, alert aggregation, deployment, architecture, alert correlator, alert masking.

1.4 Contributions of the study:

This thesis provides major contributions in the field of intrusion detection and prevention methodology as discussed in the objectives above. We summarize and group the contributions as follows:

1. In-depth study of taxonomy of IDPS and their open research areas in today's scenario of complex networks and operating systems. The study found a few open research areas out of which we have addressed one which we are presenting as a Ph D thesis.

2. We have designed, implemented, tested and proposed a hybrid architecture, wherein traffic monitoring is performed locally and analysis of raw alerts, aggregation and correlation is performed centrally. The proposed architecture helps us to identify attacks in real time.

3. Our proposed model also addresses a problem of analyzing mass of raw alerts which is a huge bottleneck in identifying actual attacks. Our mechanism proposes innovative way of aggregating and correlating alerts. In this way, we successfully reduce false positive rate by more than 95%.
4. Implementation of our model also generated a final alert log which describes confidence level of each aggregated attack scenario. Our results proved that it optimizes network productivity.
5. Our model uses the final aggregated and correlated alert log to generate precise advice to network administrator for prevention of future attacks.

1.5 Research Methodology utilized for research work:

1.5.1 Hypothesis:

Research Hypothesis:

(i) A hybrid approach to log attack information and aggregation produce an optimized solution for judging the attack information and quarantine the processes involved and not the entire host unless necessary. It is possible to design a model to achieve attack detection in shorter period of time, analyze, aggregate and correlate raw alerts, mask false positives, and generate preventive advice.

1.5.2 Research Methodology:

This section describes the research methodology used for the rationale of carrying out this research. Our research follows a hybrid qualitative + quantitative approach. A quantitative research is data collection and sampling in quantitative or finite manner which our model does. Additionally, it helps the administrator to make decision based on qualitative measurements of network analysis and aggregated raw alerts. The aggregation and correlation part is purely quantitative. This data is subjected to a rigorous analysis and experimental methods. We have studied and surveyed almost all existing IDPS systems in terms their qualitative parameters, with their experimental procedures and data used for the same. This study first formed the base of our problem identification process. After

finding research gaps, we conclude that for providing solution to existing problems which we have identified, we need to use data sets available for testing and proving our proposed model. This proves our research is quantitative.

Classifying our approach further, we have followed experimental as well as simulation kind of research methodology. The model which we have proposed and implemented is experimental. Our model can work on any third party IDS and their results. Each of the host machines of our target network contains a data interceptor module. This module intercepts socket data, and simultaneously gathers alert information from the IDS sensor. Next module logs alert information into an alert database. Third module in sequence reads and analysis the alert database contents, whom we call an analyzer module. The complete set up is tested with existing as well as real network data.

At times, when we need to test in certain cases when data set does not contain attack information we wanted, we have generated test scenarios ourselves. That is the simulation part of our model. , The simulation module is developed in similar lines with SysLog protocol to make sure it can work with any other protocol which is comfortable with syslog. We have used this mass of raw alters to test our model for analysis.

Thus, a qualitative + quantitative research, with experimental and simulation approach forms the basis of our research work.

1.5 Organization of the remainder of thesis:

In Chapter 2, we describe our literature survey in the area of IDPS. We have segregated our survey based on parameters such as IDPS approach, deployment, architecture, responsiveness of existing IDPS. We have also analyzed in detail patents of commercially popular IDPS such as CISCO, Juniper and Fortinet.

Chapter 3 details our proposed architecture for achieving our objectives already described. We also discuss two different deployment modes for our proposed prototype. System flowchart is also shown.

Chapter 4 discusses experimental setup for solution validation, algorithm, datasets for experimentation and proof of concept.

Chapter 5 describes experiment results, parameterized comparison with open source and commercial IDPS.

Chapter 6 concludes our research with objectives achieved with justification, conclusions of our work, and scope of future enhancements possible in our research.

CHAPTER II

Literature Survey

2.1. Intrusion Detection and Prevention Systems (IDPS) - Taxonomy:

IDS complement firewalls and other security devices employed for a network. IDS keep track of hostile actions or attacks in the network. After an attack is detected, the event is logged by the IDS in a database or a file. An alarm is generated signalling the presence of an attack. Figure.1 shows taxonomy of IDS systems:

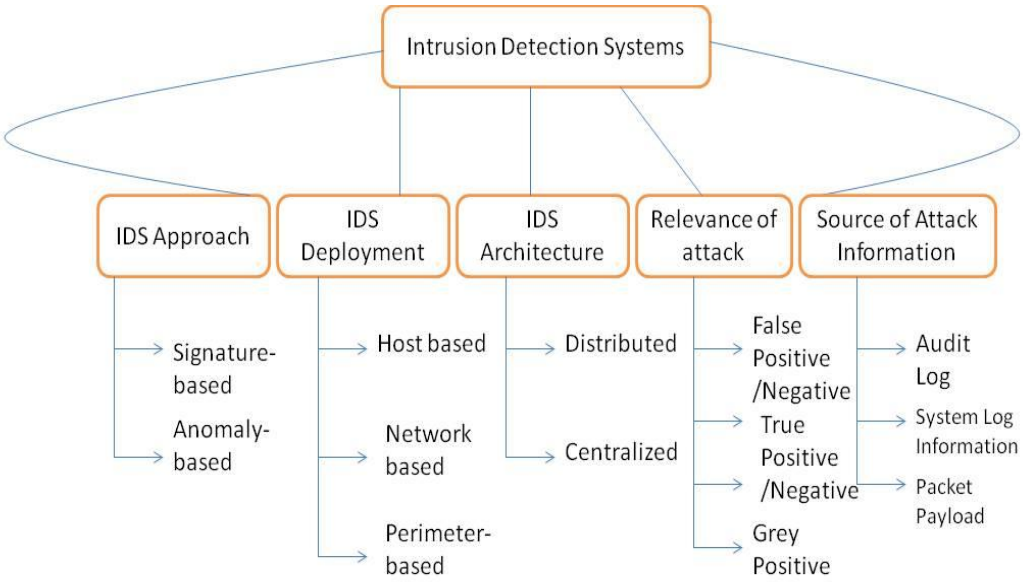


FIGURE 2.1 Intrusion Detection Systems Taxonomy

Growth of the Internet and increase in complexity of operating systems and networks lead to further enhancements in detection scenario. Researchers were now more bothered about the reactive measures of an IDS. There was a huge demand for IDS systems to not only detect attacks and raise alarms, but also provide measures to prevent these attacks. Intrusion Detection systems were assigned a new task of prevention of detected attacks. Almost all IDS are now Intrusion Detection and Prevention Systems (IDPS). The following diagrams, Figure 2.2 and Figure 2.3, show the difference between IDS and IDPS functioning:

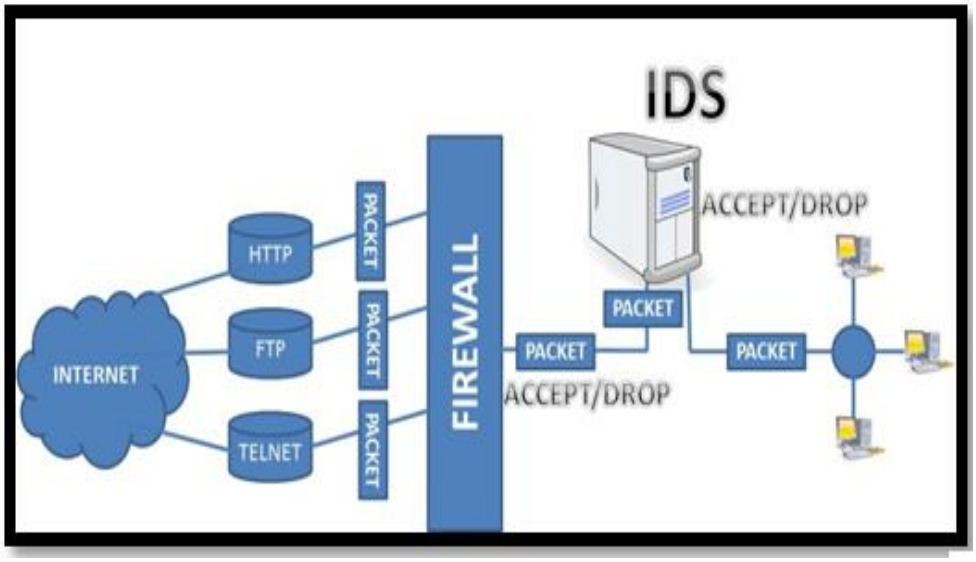


FIGURE 2.2 An Intrusion Detection System in Network

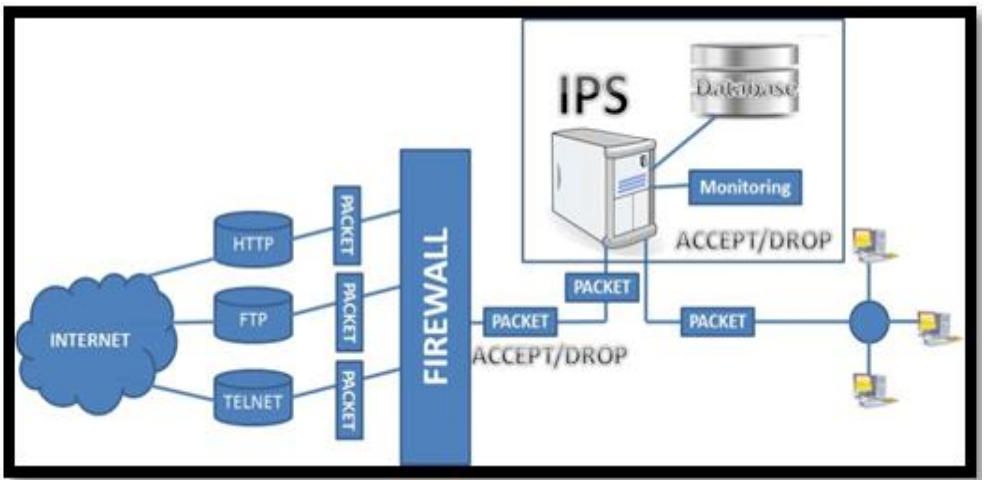


FIGURE 2.3 An Intrusion Prevention System in Network

Intrusion detection and prevention systems are more reactive in nature. They work on the analysis and policy enforcement functionalities defined by NAC. For analysis, they rely on alert log stored in a database or files. This alert log is assessed for determining the cause of attack, the victim of the attack and criticality of attack. After analysing the alerts, appropriate actions are taken by IDPS for preventing those attacks from network.

Before describing survey on IDPS, we first discuss existing scenario in NAC, the advantages and disadvantages of the same.

A European patent publication no. EP2164228 A1 claims "Hierarchical application of security services with a computer network" whereby, techniques are described for

hierarchical application of security services with a network device. In particular, the network device receives security classification information that maps a security class to one or more computing devices. The security class identifies security capabilities of the computing devices. The network device also receives network traffic associated with the computing device and applies a set of patterns defined by a policy associated with the security class to the network traffic to detect a set of network attacks. Based on the application of the set of patterns, the network device forwards the network traffic. As a result of receiving security classification information, the network device may become aware of the security capabilities of the computing device and only apply those patterns required to augment these detected security capabilities, thereby preventing application of overlapping security services through application of these services in a hierarchical manner. But, it scans the network and collects the network security snapshot which comprises association of host with installed patches, application, etc. When that particular host create sends or receives a network data, tailor made pattern matching is applied for that host. Like if it is known that only certain applications are installed on that host then attack related to those applications are scanned and matched with. Thus, it requires scanning of entire network. Moreover, it applies pattern matching for whole bunch of applications installed on that host instead of being specific to identified vulnerable application. Thus, the present invention is more time consuming and costly.

US patent publication no. US6816973B1 claims a method and system for adaptive network security using intelligent packet analysis are provided. The method comprises monitoring network data traffic. The network data traffic is analyzed to assess network information. A plurality of analysis tasks are prioritized based upon the network information. The analysis tasks are to be performed on the monitored network data traffic in order to identify attacks upon the network. But this invention creates a network map which composes information regarding different devices, oses, services installed in the network and then use that information to analyze the network packet. Thus, this invention is fully dependant on prior network map making it stringent.

DISADVANTAGES OF NAC:

All of the available systems for network access control suffer from at least one of the below mentioned problems:

- Most of them do not run periodic scan for host assessment at higher frequency due to efficiency reasons, hence there are high chances that any host starts violation of policy in between two scans which means after the host is recognized as healthy, it might change its status to unhealthy and the monitor misses that. It might be again able to change its state back to healthy to avoid detection.
- Most of them have tendency of removing the entire host from the network once found unhealthy. Thus, due to just one vulnerable process, all network traffic of host will be blocked, which can have practical impact on overall productivity. Rather, this behaviour can be used to have denial of service attack on that host.
- Many of them are dependent on prior network map which makes them stringent.
- Many of them apply pattern matching for whole bunch of applications installed on that host instead of being specific to identified vulnerable application which makes them more time consuming and costly.

2.1.1 IDS Approach:

Traditional Intrusion detection systems follow two diversified approaches for attack detection: signature-based or anomaly-based. A signature-based IDPS relies profoundly on pre-defined signatures stored in a signature database. These signatures are configured keeping in mind various types of attacks identified for detection. If an attack occurs and the signature for that attack does not match with any of the signatures in the database, a new entry is made into the database. These signatures are reactive in nature because until the threat is known and a remedy exists for the threat, a signature cannot be created. Anomaly-based detection systems have a baseline defined for behaviour of systems. Activities that comply with this baseline are termed as “normal” and which do not adhere to the baseline is termed as “abnormal” behaviour. Baseline or normal behaviour for a system for instance information of kernel routines, audit log, network packet payload, list of running applications; operating system information etc is stored into a data store. Monitoring traffic for abnormal behaviour increases the chances of detecting previously unknown attacks.

Elaborating typical anomaly-based IDPS, as the name indicates, detects anomalous system behaviour using the following general approach. It first classifies the system behaviour under observation into normal and abnormal system behaviour. Based on this classification, the anomaly detector identifies deviation from the normal system behaviour. Finally, it takes the needed action based on the system analysis. For instance, Lee et al [20] explore an anomaly detector for the sendmail program. They use the execution sequence of system calls as the system behaviour to be analyzed. So a database of normal sequence of system calls is built using a set of training sendmail execution traces. Then the evaluated sendmail execution traces are compared with the database thus built. If in a sequence of calls, a system call is not present in the database, then it is labelled as abnormal. However, there may rarely be system calls which are invoked and are also part of normal sendmail execution. So in order to filter out such outliers, they examine a window of system calls. If in case in the window more than a threshold number of calls are abnormal, then it is an anomaly. Thus in this manner, Lee et al [20] classify sendmail execution as either benign or malign. System calls need not be the only system behaviour to detect an anomaly.

An anomaly detector can also use other control-flow information [23] [24] [25]. The main advantage of an anomaly detector is its potential to adapt to system dynamics. For instance, an anomaly detector can detect zero-day attacks. These are attacks that are hitherto unknown. So it is important to thwart such attacks due to the ease of spread of these attacks. A heuristic leveraging the observed traffic anomaly on the zero-day [21] is an interesting defence mechanism to zero-day attacks.

However, there are issues with effective anomaly detection. Anomaly detection requires a wide variety of training data in order to accurately predict the system behaviour. For example, Lee et al. [20] observe that when the sendmail anomaly detector heuristic is applied to network traffic, the strong temporal variations in network traffic result in a very high error rate. So the anomaly detector needs to keep pace with the system input and the system response. Sommer et al. [22] discuss in depth the various issues to effective anomaly detection. In this work we concentrate on misuse detection IDPS. But it is important to stress that anomaly detection is important and very relevant to network security.

In contrast to an anomaly detector, a misuse detection IDPS functions by using a database of prior attacks. So a misuse detection IDPS compares the packet bytes with the attack

database. In case the packet bytes match the database, then the IDPS flags it as an intrusion attempt. Attack descriptions or signatures are contained in the misuse detection systems. They are matched against audit data stream finding proof of known attacks [26]. Keeping focus on analysis of system data such as audit logs and generating few FP (false positives) is the biggest benefit of misuse detection systems. Detecting only known attacks having predefined signature is the biggest drawback of misuse detection systems. New attacks, whenever found, should be added to signature database. Additionally, signature-based IDPSs are more susceptible to attacks intended at triggering a large amount of detection alerts. This is done by generating traffic designed on purpose in order to counterpart the signatures used in the analysis process. An attack like this can be used to weaken the resources on the IDPS computing platform and to put out of sight attacks within the huge number of alerts formed. As compared to firewalls, all packets at layer 3 and 4 along with application level protocols will be scanned by misuse based IDPS looking for back door Trojans, worms, Denial of Service attacks, buffer overflow attacks, detecting scans against the network, etc. Higher visibility to detect signs of compromised hosts and attacks is provided by an IDPS. Firewall is still needed to block the traffic before it penetrates the network. Additionally IDPS should also be used to ensure monitoring of traffic that gets past the firewall.

2.1.2 IDPS Deployment:

Deployment of IDPS plays a vital role in the eventual performance of the same. Deployment determines what IDPS monitors and scans and the kind of attacks it is capable of detecting. Host-based IDPS are deployed at host-level in a network. They monitor individual host information such as system calls, process usage statistics, and file access log. To achieve this, they usually perform scanning using port-scan or vulnerability scan [50].

Network-based IDPS deploy an IDPS sensor at network ingress point, a central location in a network from where all the traffic passes through [50]. They monitor network inflow and outflow traffic and inspect packet payload information for malicious content.

Perimeter-based IDPS, also known as network-node IDPS monitor network packets destined for a particular end system [50] [26]. They are distributed in nature. Multiple sensors are placed at different locations along the boundary of network. These sensors

perform load balancing for traffic monitoring. They are delegated the task of monitoring packet information only for a set of end systems in the network.

Primarily IDPS is deployed at host or network level [50].

The basic characteristics of an IDPS are determined by the deployment. An IDPS deployment categorizes it as network based intrusion detection and prevention system (NIDPS) or host based intrusion detection and prevention system (HIDPS). An IPS sensor is generally put at network ingress point [50] in an NIDPS. What the IPS sensor does is monitoring network traffic and it also inspects network packets for any suspicious activity. A network-based system can keep track of different hosts by identifying the monitoring component properly (for e.g. at a network ingress point).

HIDPS are deployed on individual hosts and have access of system-level information, such as patterns of system calls, file access or process usage. They detect vulnerabilities by system log monitoring or by scanning techniques. It is by using techniques such as port scan that they detect potential vulnerability. Figure 2.4 shows HIDPS deployment scenario.

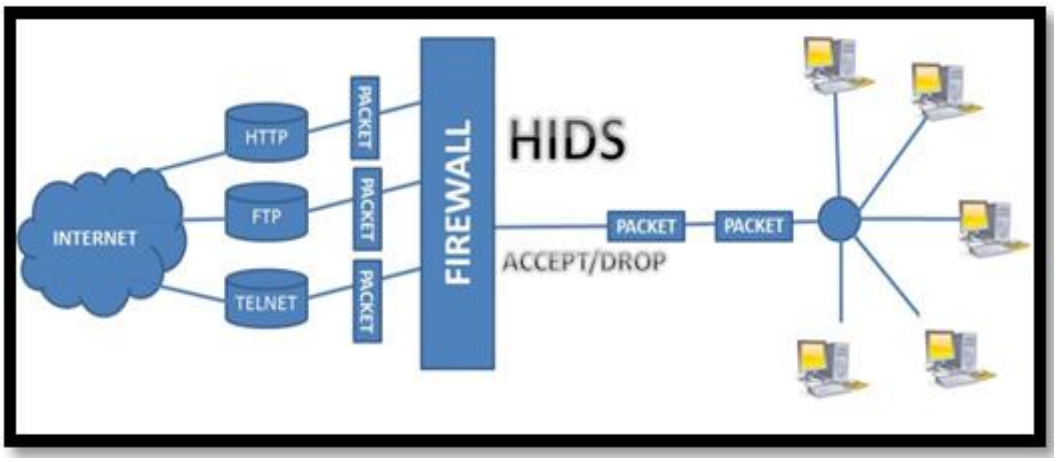


FIGURE 2.4 Deployment Scenario for HIDPS

Nowadays, the host based approach has gained greater prominence than it had a decade ago. First, it is due to the explosive growth of the Internet that modern operating systems have grown more complex. Its outcome is that extensive monitoring has become increasingly complex to achieve. Secondly, administrators are inclined to keep their focus on the impact of an HIDPS on host performance. A popular HIDPS (usually known as a "web application firewall") is ModSecurity [34]. It is a module (i.e. a pluggable software

component) for the Apache web server. It intercepts incoming and outgoing requests and analyzes them. If a malicious request is detected; ModSecurity can drop it and thus stop it from being processed by the Apache server. Thus the NIDPS approach makes it possible to monitor data and events without having any effect on host performance and that is the main advantage of the NIDPS approach. That it is not host-based is its main disadvantage (this specially applies to systems that analyze network packet payload). Also, NIDPS does not perform precisely for protocols that use data encryption techniques such as SSH and SSL, especially if the encryption key is not given. Using a host-based component to access data after decryption is its possible solution, but it levies load on the host that is being monitored.

The above mentioned problem aggravates with IPv6 slowly replacing IPv4. This is because As IPv4 is getting replaced with IPv6, the problem will grow in importance as authentication and confidentiality of data (through cryptography) is one of the main design goals of IPv6. Figure 2.5 shows deployment scenario for NIDPS.

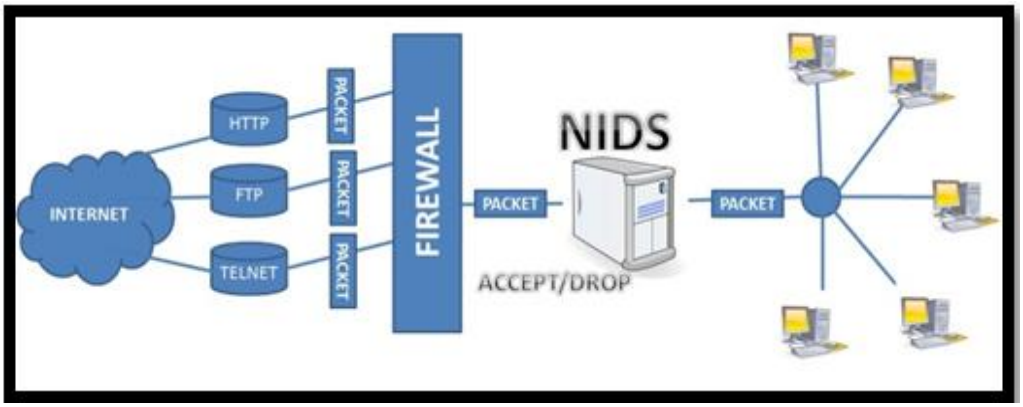


FIGURE 2.5 Deployment Scenario for NIDPS

Reconstruction of network traffic is another common problem for NIDPS. Data streams are divided into TCP segments and IP datagram. The traffic should be reassembled into the original form to analyse the content. Modern networks operate at high speed (up to 10Gbs). For an arbitrarily powerful system, the traffic reconstruction would be theoretically possible. Whereas NIDPS faces performance and implementation constraints. First, depending on system time-outs and data throughput (i.e. resource consuming, NIDPS should save a significant amount of data for and elongated period. Secondly, the methods used by operating systems in order to execute diverse network stacks and handle packet reconstruction are different. Hence it becomes necessary that the NIDPS engine

implements some functionality which is also context-aware. The drawbacks discussed above were given a new dimension in the form of evasion and inclusion attacks, first defined by Ptacek and Newsham [35]. Interactions are crafted by attackers for fooling NIDPS. E.g. some data previously sent is overwritten in NIDPS memory or NIDPS is forced to drop unanalysed data.

Advantages offered by HIDPS and NIDPS were merged by EMERALD system into (virtually) single IDPS [28]. It still has open issues like problem of data accumulation from varied sources, alert aggregation & correlation with appropriate metric definition. Because of these issues, the development of enhancements was closed after the initial POC of EMERALD.

NIDPS is bifurcated into two more categories. Sniffing every incoming and outgoing packet at network boundary in order to analyze the behaviour is done by a standard technique, promiscuous-mode network intrusion. Exactly one sensor is positioned on each network segment in Promiscuous-mode Intrusion detection & prevention systems. The packets destined for a specific destination machine are sniffed by network node IDPS. Network-node IDPS are intended to function in a distributed environment [38]. Spatially and temporally distributed data should be captured in order to detect as well as prevent as many attacks on network as possible. E.g., an attack for which an alert has been generated from different detection locations is deemed as distributed attack. In addition to this, alert generated for same attack but at different time intervals is an indication of a co-ordinated, repeated attack. A hybrid IDPS incorporating centralized detection feature of NIDPS and localized, scattered feature of HIDPS is required to capture data spatially and temporally.

NIDPS AND HIDPS: A Comparison

The deployment of IDPS requires certain points to be considered and we describe them in this section. Having reviewed few host-based and network-based deployments, we discuss details of each with pros and cons. .Comparison with network-based intrusion detection systems made it clear that host-based IDPS have certain advantages. Accessing semantically rich information about the operations performed on a host is one of the advantages of HIDPS, while NIDPSs analyzing network traffic need to reassemble, parse, and interpret the application-level traffic to identify application-level actions [35]. .Encryption of application –level traffic makes it more conspicuous. In this case, it becomes necessary to equip a network-based monitor with the key material required to

decrypt the traffic; otherwise, it will not be possible to access the application-level information. Besides, HIDPSs have to process a limited amount of information as the OS and applications generate events at a lower rate compared to the rate of network packets over busy links. Since it is more difficult to desynchronize the view that the intrusion detection system has of the status of a monitored application with respect to the application itself, HIDPS are less prone to evasion attacks. Ultimately it is more possible for a HIDPS to perform a focused response because of easy identification and termination of a process performing an attack. One of the popular HIDPS, Hawkeye solution [27] is discussed here. The architecture of Hawkeye solution includes components such as a centralized server for management, agents, database, monitoring console and demilitarized zone (DMZ). This proposed architecture is better than other HIDPS since it provides advantages such as packet capturing via multithreaded TCP / UDP functions, passive network monitoring, log and view packet in hex-format, and abnormal behaviour detection with its cause. It is possible to trace the source IP address in an abnormal case. However, it is a packet dependent detection mechanism. It is very difficult to detect an attack if it is distributed across multiple packets. Thus, the process of detection needs to be done on information or stream and not packets.

One more problem of this architecture is described in [28][29]. According to it, each and every packet should be inspected by monitoring the network flow. They propose adaptive sampling algorithm in order to address this issue. Future behaviour can be predicted by this algorithm based on experiential samples. Next sampling interval can be predicted by using the weighted least squares method. Erroneous predictions by the weighted least squares predictor are indication of an alteration in the network traffic behaviour and require a change in the rate of sampling. Alternatively, what this algorithm looks for is patterns in network interchange and makes it possible to detect few attacks for e.g. DOS attacks. When a detection system is deployed at host level, an IDPS that monitors network scenario can only measure trends in network traffic and thus detect attacks such as DOS. Nowadays co-existence of IPv4 and IPv6 is a prominent trend.

IPv4 address space is almost depleted. A massive investment having been made in IPv6 networks, designers are confident that IPv6 networks should be progressively ideal and organized. The switch from IPv4 to IPv6 is going to be a very time-consuming process [30]. The CIDP [30] is a proposed architecture with hierarchical, distributed, three-dimensional IDPS. It consists of UTM (Unified Threat Management), NIDPS at the

network boundary, Subnet NIDP deployed in each subnet, HIDPS in IPSec tunnel endpoint, and a public domain server. The function of subnet NIDP incorporated in one of the three layers is to detect back door attacks. Subnet NIDP is capable of protecting other subnets from a host launching an attack, except the subnet itself. Host-based IDPS are required for this protection. It is only on certain endpoints that this architecture deploys on. It is better that the deployment is done on each host as it protects the network from attacks from server and client.

The IPv4 address space would have been used up long back without NAT. As applications behind the NAT don't have any other way to know the real address/port used by the hosts [38, 39], different other applications are affected by the translation of address/port by NAT. In case of an attack which is launched from inside the network or a host is vulnerable in the network, IP addresses of the invader and victim is essential. IDPS can perform correctly with NAT if it is re-examined. On deploying IDPS on perimeter router behind NAT device, actual identity of invader and victim is unknown. The presence of a NAT device that changes the packet headers should be known to the IDPS deployed in the network. [31] A possible solution for it is to deploy two IDPS: one each above and below the NAT. By doing this, the two IDPS will refer invader and victim with separate identities even if they relate to the same threat scenario. Hence these alerts will be treated as separate alarms increasing the number of alerts overwhelming the security operator. To verify real hosts' identities mixed up in the security concern based on NAT table, output of analysis module is analyzed by the identification module.

It is difficult to integrate IDPS to NAT box. It may not be possible for every NAT device to provide such information outside the box. Attacker and victim information can be assured to be correct if IDPS functionality is deployed on a host. Because of certain characteristics of wireless network, building an IDPS in wireless environment is inconvenient compared to wired environment. E.g. while using wireless devices; company's trusted workers may need "inside" kind of connectivity. On the other hand, "outside" kinds of connectivity might be needed by visitors on connecting to company's wired network through an access point inside the corporate firewall. Placing a firewall between "inside" and "outside" is very tough [31]. Secondly, it is important to deploy the IDPS engine from where the entire user traffic flows through. A wireless network can be attacked from all directions targeted at any machine. This makes it difficult to identify a single path to deploy an IDPS engine so that all traffic can pass through it. Thus, compared

to wired networks, it is not easy to build an IPS engine in wireless environment. WTLS-Based IPS model is proposed [32] to find a solution for this issue. According to this model, logical sole path has to be built between every wireless terminal and its destination. This makes it easy for an IPS engine to detect and prevent the traffics of user. Figure 2.6 shows WTLS IPS setup in wireless environment:

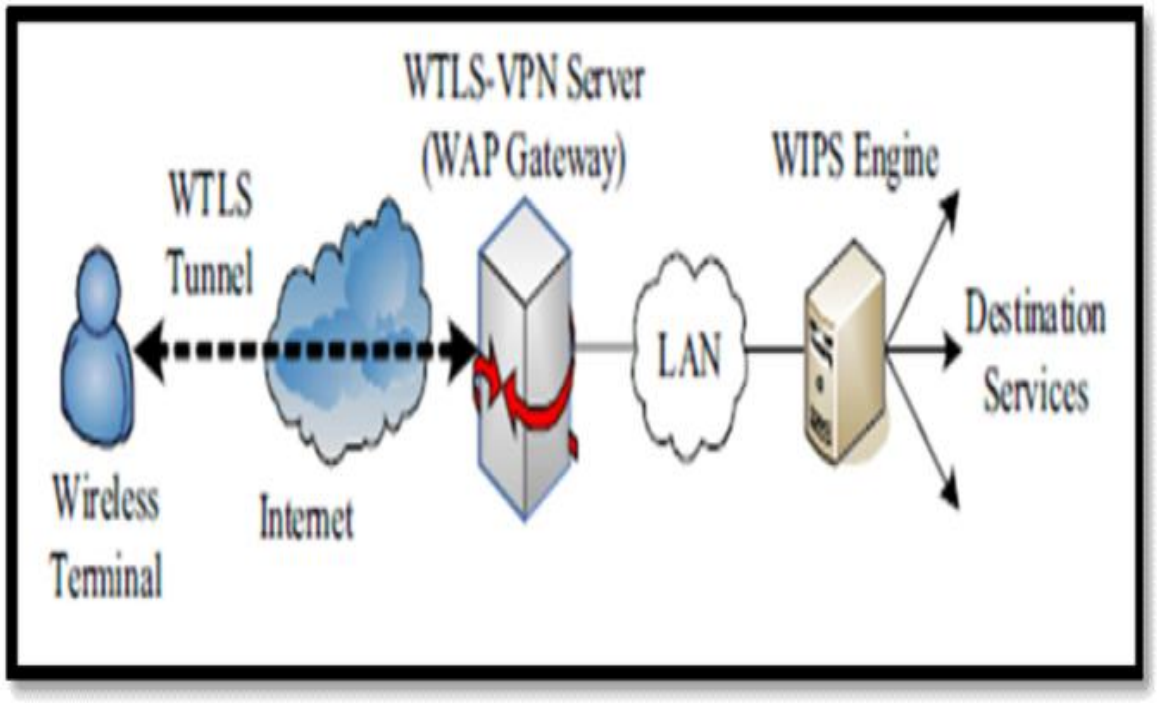


FIGURE 2.6 Wireless Setup [32]

VPN tunnel should be created between wireless device and gateway in order to redirect through. If HIDPS is deployed on wireless device, tunnelling will not be needed. It is necessary to capture data, distributed spatially and temporally so that maximum number of attacks may be detected and prevented. For e.g. an attack that is detected at different monitoring locations is possibly a distributed attack. In addition, if an alert for the same attack is generated at different time intervals, it is a possibility of co-ordinated, repeated attack. A hybrid system that combines centralized traffic monitoring feature of NIDPS as well as distributed local feature of HIDPS is required to capture data spatially and temporally.

2.1.3 IDPS Architecture:

If deployment of an IDPS answers to the "what" part of an alert verification, architecture of an IDPS answers to the "how" part of the same. Architecture of IDPS can be

categorized as distributed or centralized. This determines how IDPS performs its two vital operations: Traffic Monitoring and Assessment for vulnerabilities. A distributed IDPS delegates the traffic monitoring and assessment functionalities to multiple entities. Each entity who is delegated the above-mentioned tasks is capable of alert detection and verification. They also communicate and share these detection results with one another. Figure 2.7 describes distributed IDPS:

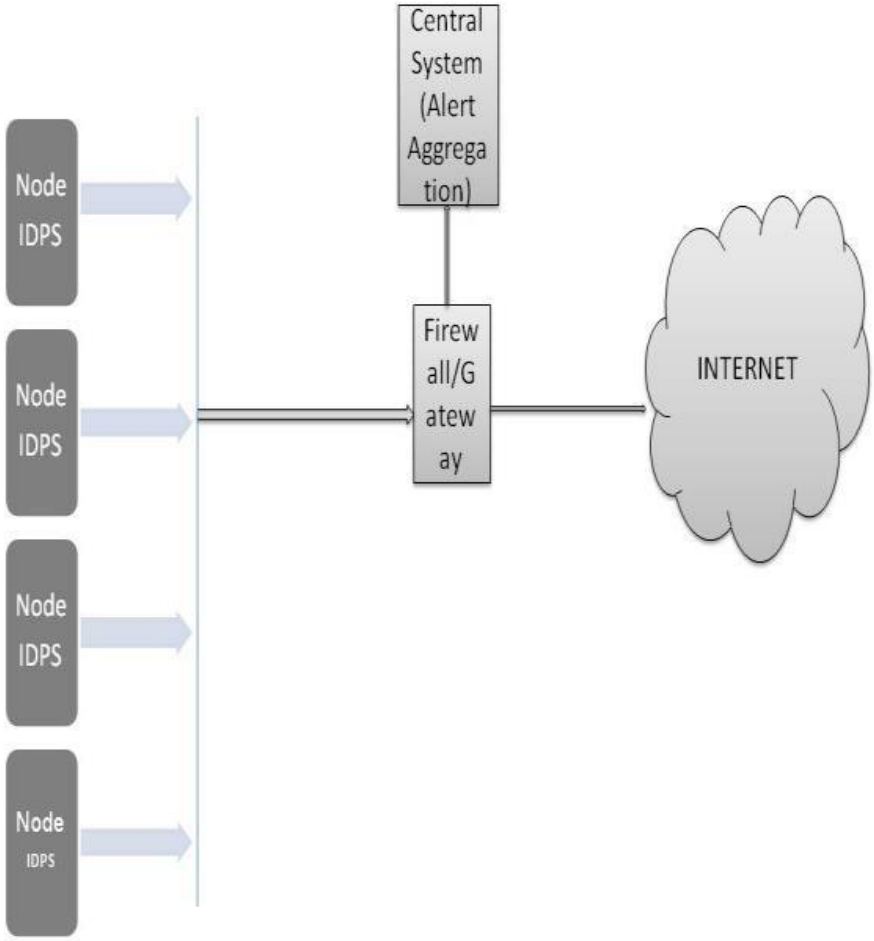


FIGURE 2.7 Distributed IDPS

In a centralized IDPS, multiple sensors are placed in the network for detection of attacks. These sensors monitor traffic flow, collect and store attack information in a database. The attack log is then relayed to a centralized component responsible for vulnerability assessment. Only one event analyzer centrally analyzes the entire network information provided by multiple sensors. Figure 2.8 shows a centralized IDPS architecture:

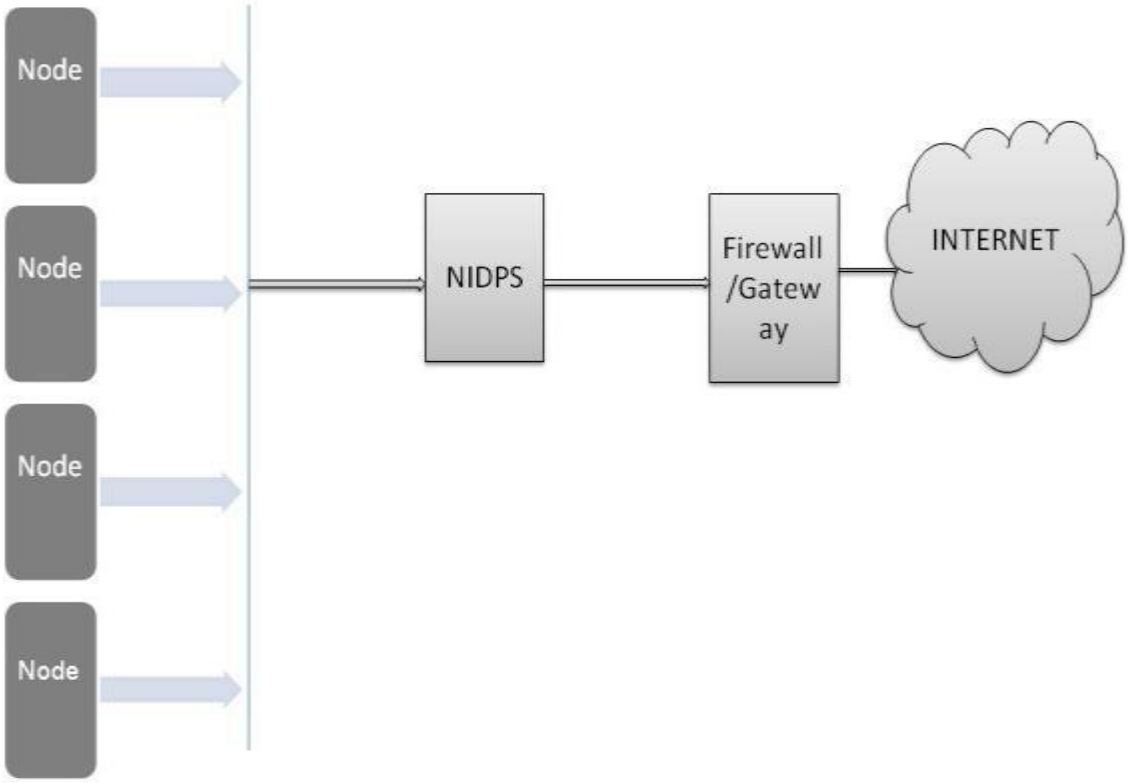


FIGURE 2.8 Centralized IDPS Architecture

The type of information that an IDPS analyzes in order to raise an alarm or an alert is largely dependent on the architecture of the same. Two primary functions performed by NIDS or HIDS are: traffic monitoring and analysis. IDPS implemented in distributed manner performs both of these functions in a distributed manner. An earliest example of this kind of architecture is Distributed IDS [31], capable of performing analysis locally as well as centrally. It uses network security monitor (NSM) components and uses signature-based and behavioural detection locally. Analysis is performed centrally using rule-based expert system. Hence, DIDS is a combination of HIDS and NIDS capable of sharing results with each other. Thus results are analyzed and shared at different levels.

However, almost all centralized IDPS rely on few limited data sensors, with one analyzer to perform monitoring and analysis of alerts. This leads to missed attacks and false negatives. In addition, scalability remains a major issue [31].

EMERALD [41] and APHIDS [42] are earliest implementations of DIDS. EMERALD is NIDS while APHIDS is HIDS, both with distributed architecture. Source of information and analysis techniques are different in both. However, both of them share a common

hierarchical architecture. Local modules at host level process and pass on their result to modules at higher level, till it reaches root level modules. Analysis is finally performed by a centralized component. Thus we can say they are not truly distributed.

The distributed layer of APHIDS uses distributed agent engines to perform major functionalities. These agents perform analysis on event notifications which are triggered and raise an alarm. Delay in the analysis process is reduced considerably because of this architecture. However, the alerts generated are not logged at a common data store. Attacks such as DDOS are not possible to be detected since they require analysis of previously stored alerts over a longer time span.

In case of EMERALD, different information sources such as system audit log, network packet payload, application logs and alerts generated by IDS. This information is filtered, parsed as well as formatted and relayed for further processing. A profiler engine applies analysis process on this relayed information in order to detect vulnerabilities. Certain events, which require analysis of complete network streams, are missed since analysis is done at host level [41].

Network packets do not provide all the necessary information for alert aggregation and analysis. We also require and need to capture operating system related data also. Socket information such as IP address and port of source and destination, application name, version and information about data usage i.e. how much data is uploaded and downloaded etc. This kind of detail forms a good basis for precise correlation and analysis.

2.1.4 Source of Alert Information:

There are different sources from which IDPS retrieves information for an attack. Host based IDPS monitor user activities on a system with the help of system audit logs [40] [41] [42]. These audit logs themselves are likely to be tampered with in case of a successful attack. This fact imposes a constraint on HIDPS to detect and raise an alarm before the audit log information is manipulated. We identify the following source of alert information for an HIDPS:

Process Statistics: Modern operating systems contain variety of commands which provides a snapshot of processes currently running on a system. These commands provide micro-level information about the processes since they directly access kernel memory. Examples of such commands for UNIX environment are ps, pstat, vmstat etc. A major

bottleneck is gathering and structuring this data for continuous audit. There is no mechanism to structure this information in order to extract alert details.

Resource Usage / Accounting: One of the sources for system behaviour is Accounting or Resource usage. It provides detailed usage statistics of shared resource consumption by users of that system. These shared resources can be disc or network usage, memory, running processes etc.

Syslog: Syslog utility is essentially an audit service, for those applications running on a given system. It receives name of the application in form of a string input parameter. It appends timestamp and system name on which the application is running to the name of the application. This information is stored either on a remote site or locally. Host based IDPS use this information to detect application anomalies in the system.

Network-based IDPS monitor overall traffic inflow and outflow. The sensor sits inline for traffic monitoring. Network packets are regularly inspected for any suspicious activity. Certain categories of attack such as denial-of-service attacks can only be detected by network traffic monitoring. Also, since network packets are inspected before they access any resource of the network, detection of an attack is more precise and time-bound.

2.1.5 Relevance of Attack

We define true positive (TP) as the scenario where an alert is generated for an attack successfully. It denotes that an attack is detected in true sense. A false positive (FP) describes the opposite scenario to TP. It signifies that an alarm is signalled for an attack which did not take place in actuality. It has been noted that more than TP, FPs prove to be a bigger headache to the network administrators during alert analysis [88][89][90][91]. Another term false negative (FN) denotes a situation when IDPS does not generate an alarm and an attack is actually launched.

Attack relevance is determined by rate of TP and FP. We assign RTP as rate of true positives and RFP as rate of true negatives. The RTP parameter determines how effective an IDPS is during attack detection process.

$$\text{RTP} = \left[\frac{TP}{TP+FN}\right] * 100\%$$

As the value of FN goes towards 0, RTP value goes towards 100% and it denotes effectiveness of IDPS solution.

In the same manner, we define RFP as how effectively IDPS responds in the FP scenario:

$$\text{RFP}=\left[\frac{TP}{TP+FP}\right]*100\%$$

As the value of FP goes towards 0, RFP value goes towards 100%. It means IDPS is effective against FP.

Figure 2.9 shows a comparative analysis of RTP and corresponding effectiveness of different popular IDPS such as strataguard [89], Snort, stonegate IDPS, intrupro [90] and packetalarm [91] NIDPS:

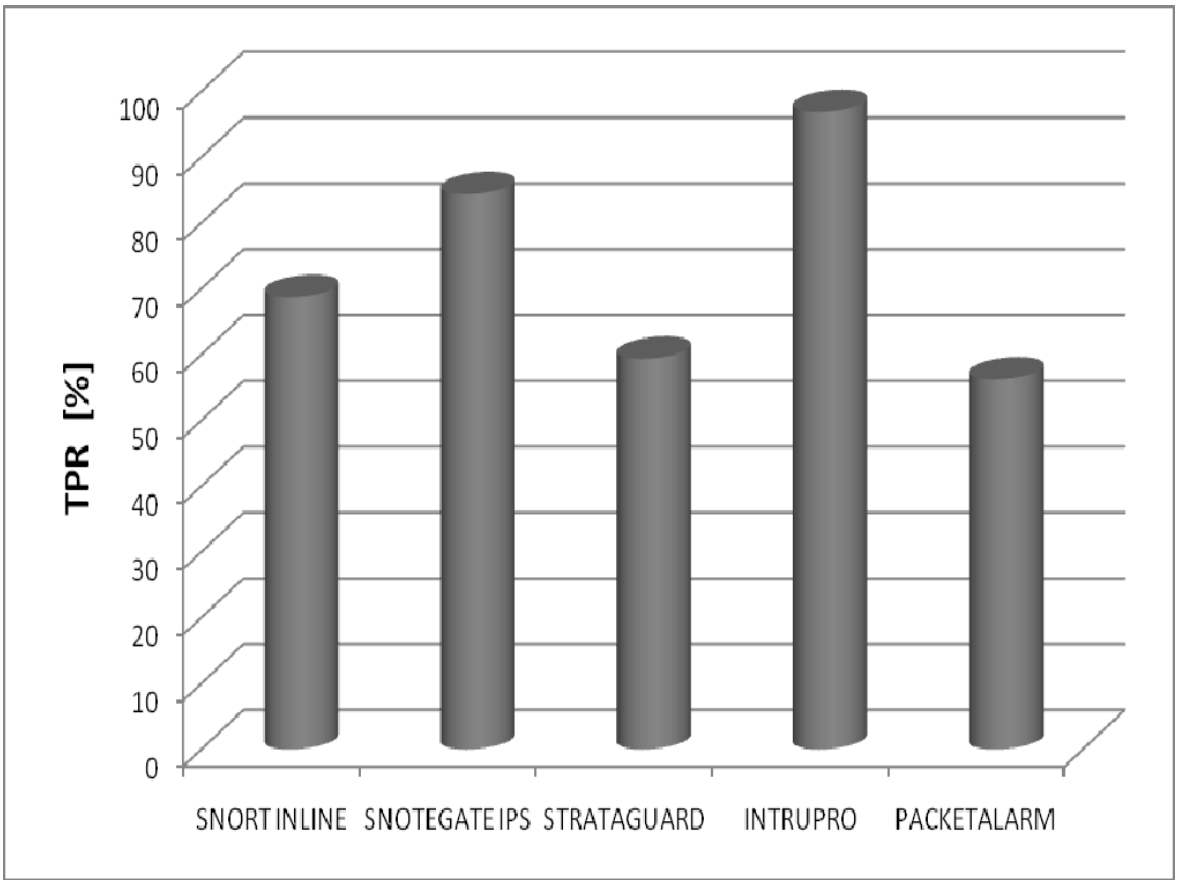


FIGURE 2.9 RTP for Various NIDPS

The chart shows that effectiveness of intrupro [90] is maximum.

Figure 2.10 shows RFP for the same IDPS discussed above. It shows the behaviour of various IDPS against FPs and effectiveness of the same solutions:

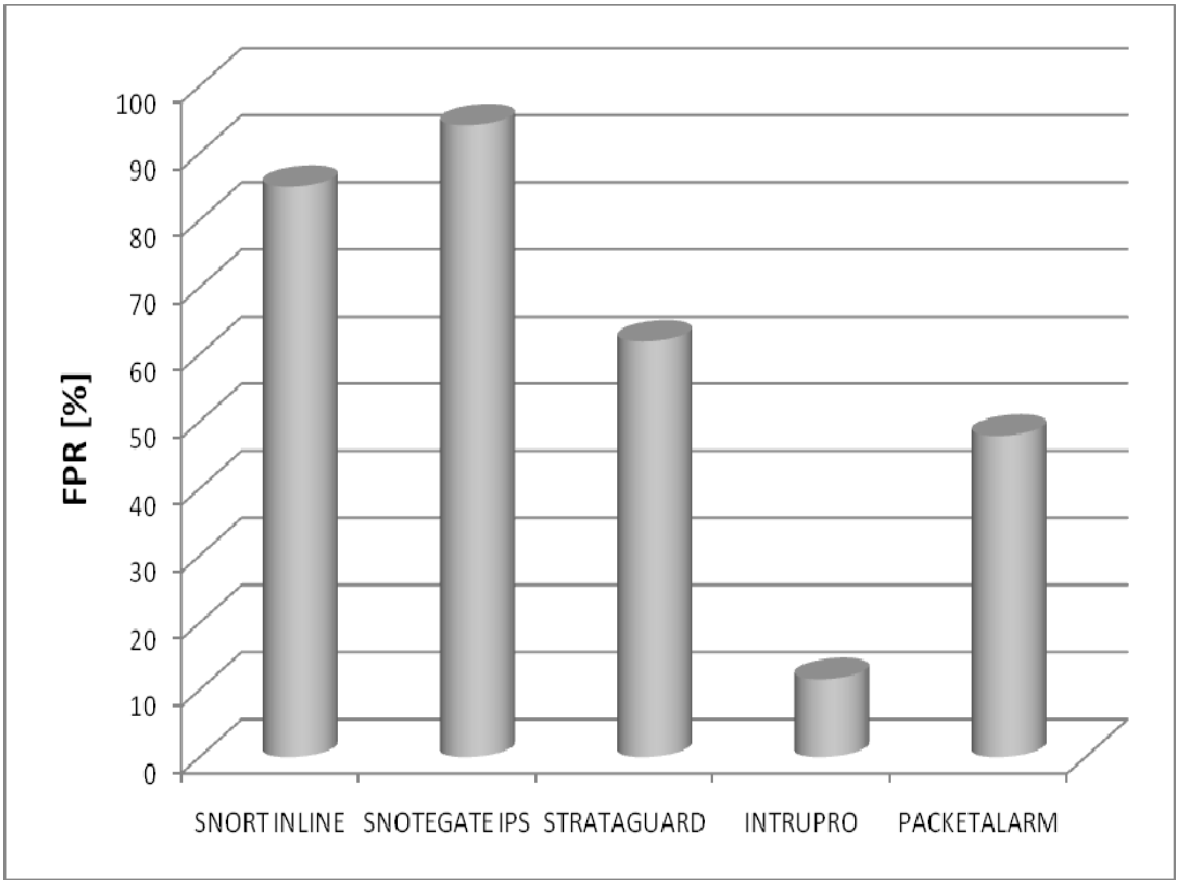


FIGURE 2.10 RFP for Various NIDPS

Raw alerts log is a combination of TP and FP. Almost all IDPS face the problem of determining how relevant an alert generated for an attack is. This is done via analysis of raw alerts logged by an IDPS. This analysis is called alert aggregation and correlation.

In order to take maximum possible advantage of hybrid deployment, monitoring, aggregation and correlation of alerts should be performed in a distributed manner. The correlated alerts should then be analyzed locally by the administrator. One such architecture described in [40], as shown in figure 2.11 consists of placing different IDPS sensors at host-level, subnet-level and at edge-router level. Each of these sensors throws alerts according to their configuration and signatures specified. Aggregation and correlation is done centrally. In this case, multiple sensors throw same kind of alerts repeatedly. During aggregation and correlation, it becomes difficult to filter same alerts. Also, correlation is not précis since basic attributes for correlation are missing.

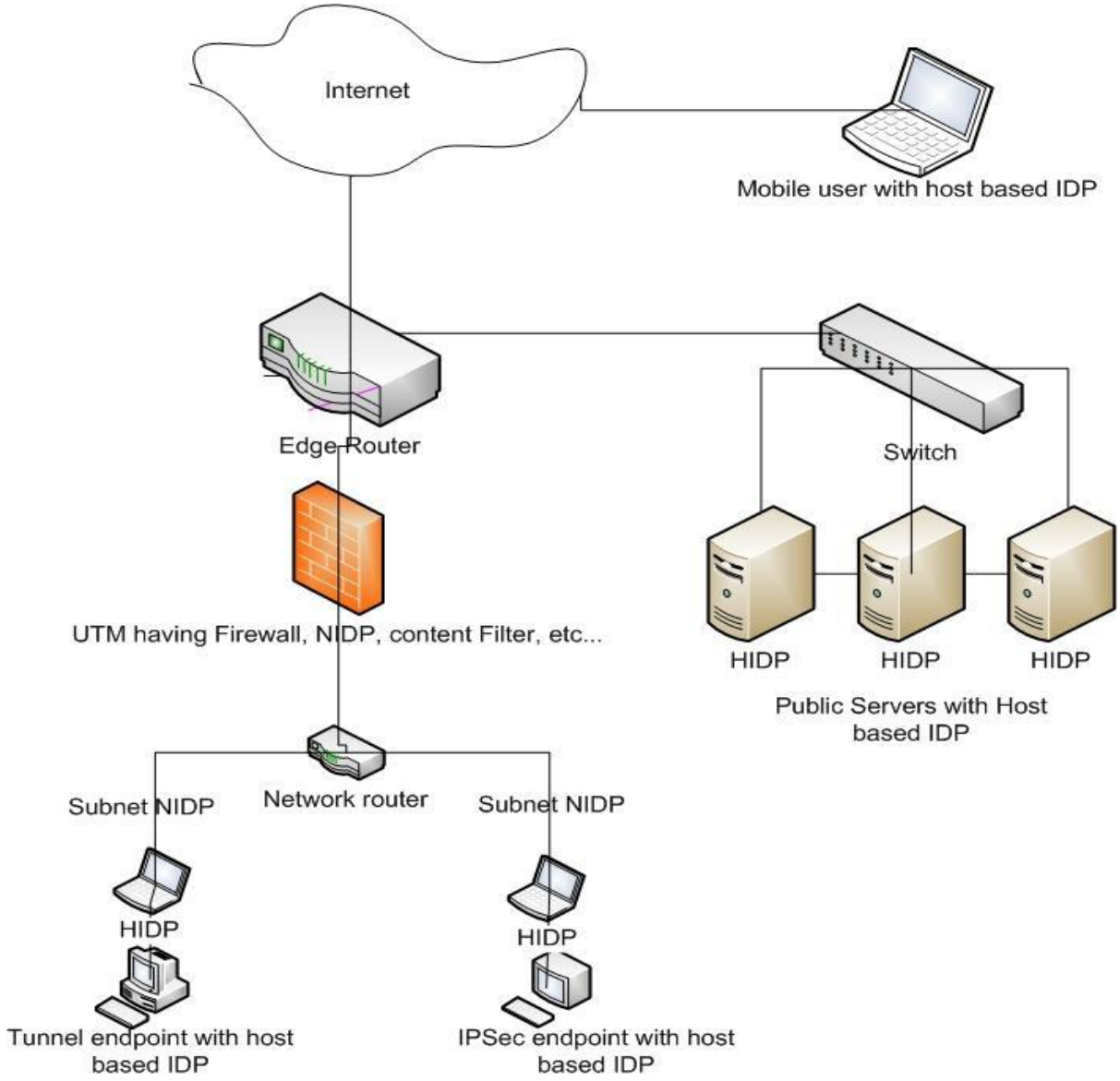


FIGURE 2.11CIDP Architecture and Components [40]

Citing an example for the same, we consider a scenario when alerts are generated when an application request to visit a website. This website is already vulnerable for a particular version of Internet Explorer, but not any other browsers such as safari or chrome. During correlation, if this information is not present in alerts, it leads to wrong correlation of alerts. Consequently, the response behaviour of IDPS will also be misled. In the given scenario, applying proper patch or fix for a given version of Internet Explorer will minimize the response cost. Also, precise problem will be attended to. Also, alerts should be generated only when request to visit this website comes from Internet explorer and not any other browser. Even the version of Internet explorer should be known and specific.

The authors in [43] suggest adding a verification layer at a centralized location. Correlation is performed after this layer functions. Verification layer performs four different types of checks named: check for target host health, verification for vulnerability, reliability of IDPS sensor and reach ability check for attack. They add this layer to reduce FP and FN. However, there are two possible drawbacks here:

- Adding verification layer centrally adds additional load on the server. All the alerts are thrown on the server for verification, because of which server ultimately suffers performance degradation. Some of the checks such as reach ability check for attacks should be done at sensors itself. If the target application in this case is not vulnerable, an alarm should not be raised for the same, leading to considerable reduction in false positive rate.
- Verification for reach ability of attack requires information about database and network topology. This kind of verification for each and every possible attack hampers the network performance. Also, if single server implements multiple protocols, FP rate tends to go high. This kind of alert verification should be performed using application information and not IP address and ports.

In order to reduce FP rate, one needs to check whether an attack is actually going to reach the destination. This check if performed at sensor will considerably reduce FP rather than done centrally on the server. Even alert correlation is affected in presence of large no of false positives. If one has information about all the running applications when alert was generated, correlating precise true alerts becomes a lot easier.

Statistical analysis in [44] describe total percentage ratio of false alerts in terms of FP and FN in live scenario. It claims that out of total alerts generated by any IDPS, approximately 93% are FPs and only 7% turn out to be FN. So reducing false positive rate is vital for performance and accuracy of any IDPS.

It is also mentioned in [44] that of the total FPs, almost 90% are generated due to wrong policy configuration and not due to security loophole. It is also observed that all such FPs majorly occur due to traffic similarities between protocols.
Examples of such events are as follows.

- Alert for "SQL Injection comment attempt" is generated because Bit Torrent clients bind to port 80. The traffic monitored in this scenario seems similar to SQL injection attempt.
- When BitTorrent clients bind to port 10000, a port monitored by signature specified, "VERITAS Backup Agent DoS attempt" alert is generated. Here, alert is generated because traffic seems similar to DoS attempt.

IDPS generates alerts in the above two cases, because sensor primarily matches IP and port only. It is worthy to note here that in both the cases, no malicious content or traffic is sent. Still, IDPS considers it as malicious attempt. If along-with IP and port, IDPS also verified the application involved, alerts will not be generated for the same. Nowadays, multitude applications run on same or randomly defined ports.

In the paper [45], authors suggest that when any application runs with root privileges, there is a possibility of vulnerability or attack. Their rationale behind this is an application with root privileges harms more. This is generally not true when application itself is vulnerable or in case of malware attacks. An attacker with user privileges may possibly hack userid and password from a browser. There is even higher possibility for NTLM (NT LAN Manager) vulnerability attack with user privileges than an application with root privileges.

We suggest that user / root privileges should be linked to application using these privileges. This will lead to correct and precise analysis of all malicious traffic for correlation. Privileges provide initial possibility of threat or attack, but it cannot be the only parameter for detection.

The authors of [46] concentrate on two types of attacks: PCF (Partial Completion Filters) and Scan attacks. PCF attacks are similar to DoS attacks. The proposal is divided into three steps:

- PCF is implementation of proprietary data structure to hold value of counters. First step finds out the operating range of the same
- Finding traffic outside the threshold of PCF
- Analysis of FPs and FNs

Nowadays, setting the initial range of threshold can be difficult. In the proposed algorithm by the authors, counters are calculated on the basis of source IP address and port with difference calculated between SYN and FIN segment. To explain the difficulty in maintaining counter value, we consider the case of web protocol. There can be two diverse behaviours for the same protocol. One scenario, as shown in figure 2.12, where host is using proxy server and same connection is retained for multiple HTTP requests / responses. In second case, a browser considers future connection requests using connect-ahead function. Also, authors suggest edge-router level implementation. This may also lead to faulty operative frequency calculation, subsequently leading to higher FP rate.

In our proposal, we tap actual host in proxy server scenario. For this, we capture the value of 'X-Forwarded-For' tag present in http request header. Thus, even if request is sent by proxy server on behalf of any host, we find original host. Thus, in case of concurrent connection, we can set counters precisely.

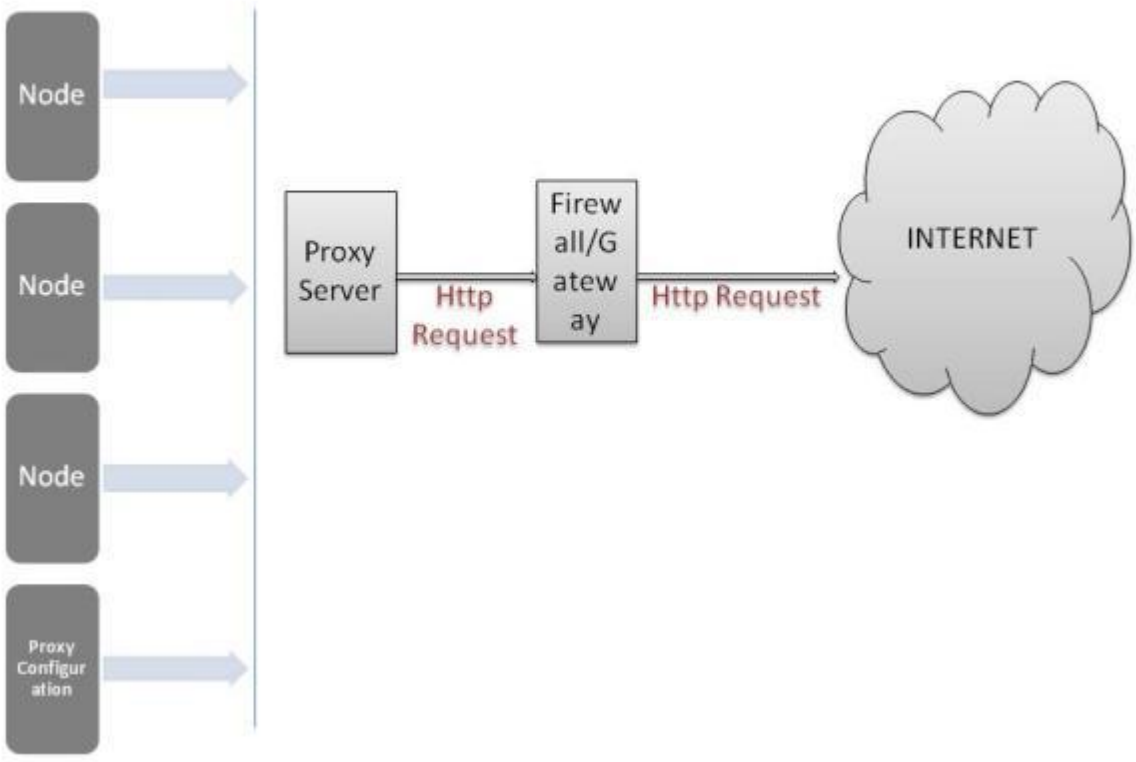


FIGURE 2.12 Typical Deployment of Proxy Server in Local Network

The concept of implementation of Locality Buffering is suggested in [47]. This technique adapts packet payload in order to accelerate processing of sensor. This is done by enhancing locality of memory access and reducing rate of cache misses. Packets are retained and assigned memory using locality buffer allotment process. Buffer allotment is done using either known source port, known destination port or both. However, nowadays, applications tend to use unknown and random ports. Thus, if buffer allotment uses application information with ports, the problem will be solved to a great extent. Even signature grouping done here should consider overall application level scenario, rather than just port information.

Generalisation of Snort rules is discussed in [48]. The technique they are using matches packet payload against snort signatures. Any successful match generates an alert with low severity, assuming that it is a variation of some known threat. This assumption is based on the fact that applications use random ports during attack. We suggest that generalization of rules must be combined with application itself rather than port.

This paper [48] also mentions generalization of content or data with generalization of rules. The paper also says that using this technique IDPS accelerates by almost ten times, which is in operating limit. But this cannot be true for Ethernet with gigabit speed. Generalization of rules also imposes load on the sensor. Citing an example for the same, if one match is ignored for a rule, consequently all the packets will match against the same signature. Rules should be independent of ports and not compromise network performance also.

Two varied approaches; anomaly detection and rule generation are discussed in [49]. They detect anomalous behaviour using the technique of data mining anomalous traffic scenarios from network connections. From the detected anomalies, they then integrate anomaly detection scheme with Snort using a weighted signature generation scheme. Figure 2.13 describes the same. The extracted signature with weights assigned is as follows:

alert tcp$EXTERNAL NET any <> $HOME NET any (msg :00 possible pod attack00; itype : 8; dsize : 1500 <> 1530; threshold : type both; track by dst; count 15 seconds 1; sid : 900; 001; rev : 0;)

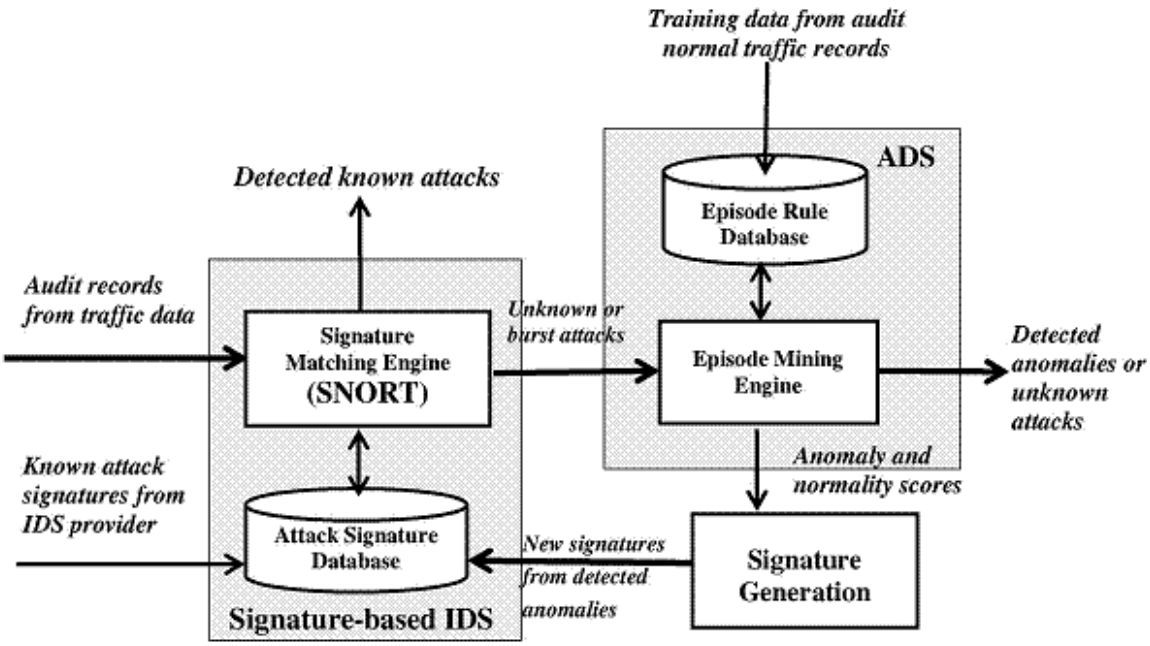


FIGURE 2.13 Anomalous Data Detection Technique [49]

The proposed scheme searches destination for payload data size in the range of 1500 to 1530 for TCP request packet. A total count of 15 such occurrences generates an alarm. The rule is generated using analysis performed by anomaly detection system. Blocking action will block the host, in this case source and destination, and not the application responsible for anomaly. If during weighted signature generation if precise relation is established between application misbehaviour responsible for this threat, effective security is achieved.

2.2 Core IDPS Functionalities:

Any IDPS performs three main functions: inspect incoming and outgoing traffic, which involves network packet inspection, payload verification, operating system log information, specific target based monitoring. The second function is looking for specific signatures in the form of patterns for possible attack instances. The third function depends on the nature of IDPS. A reactive IDPS would raise an alarm and log the alert information into a file or a database. We discuss each function in the following sections.

2.2.1 Traffic Monitoring Sensor:

Traffic Monitoring is the very first step for detection of attacks. Any IDPS monitors network traffic for detecting a possible intrusion. Usually a sensor is placed at network, host or at the boundary-level of network for performing traffic monitoring [50]. During peak hours, since the inflow and outflow of traffic is high, normally sensors are overloaded. This results in packets being dropped or intrusion going undetected [23][26]. Network level sensors are the best bet for Denial of service attacks. However, network level sensors cannot function properly for encrypted traffic i.e. in presence of SSH or SSL used for data encryption, unless key is provided or shared. This complexity is growing with migration of IPv4 to IPv6. The main goal of IPv6 networks is to provide confidentiality and authentication of data [24]. Another major bottleneck faced by network level sensors is reconstruction of network packets. These sensors need to accumulate large amount of data, which is dependent on system timeouts. This is highly resource consuming and compromises the overall network throughput.

Host level sensors provide security against internal intruders [50]. They work best to gather system level information. But, modern day operating systems are growing complex day by day. Hence, it is difficult to perform in-depth monitoring of system data. Also, host level sensors increase load on the system and hampers system performance.

2.2.2 Pattern Matching & Verification Component:

Matching and verifying network payload against pre-defined signatures or baseline behaviour is not enough to detect attacks accurately. Pattern matching is a crucial functionality of IDPS. It largely affects the overall performance of IDPS and its effect on network productivity and throughput. Pattern matching algorithms broadly belongs to two categories: Program-based or Component-based. Major program-based pattern and

verification algorithms are Boyer-Moore (BM) [75], Aho-Corasick (AC) [76], KnuthMorris-Pratt (KM-P) [74]. KM-P algorithm always matches using previously matched information. It never re-matches an already matched symbol in a pattern. It also never matches for unknown patterns and uses pre-stored information. BM algorithm matches patterns from the right side of string rather than the left. This is done to reduce total no of comparisons required. However, the same algorithm increases the complexity linearly as no of patterns grow, thereby also decreasing network throughput. AC algorithm uses finite state machine and matches multiple patterns simultaneously. It defines different output states, which indicate that a pattern match is successful. However, the main drawback of this algorithm is explosion of states in exponential fashion.

Component based verification broadly uses either network processor based pattern matching or bloom filters [77]. Bloom filters compute hash and matches incoming string using the computed hash value. It is dependent on packet length. Each packet requires a separate bloom filter, which is a drawback for worms with long definitions. Network processor based algorithm is shift based algorithm with memory related hashing technique used. It defines patterns with fixed size 4, an assumption which is no longer feasible.

2.2.3 Attack Response Module:

Attack Response module is the component responsible for providing response in case of alerts. IDPS can be active, reactive or passive [70], depending on the way it generates response to alert situations. A passive IDPS logs the alert details into a file, database or as an audit record. Passive IDPS, apart from logging alert details, also raises an alarm for the same.

Reactive IDPS also takes preventive measures for the corresponding attack launched [73]. Reactive IDPS reconfigures system policy in order to block the attacker, uses TCP connection reset to stop any new connection requests, terminating a user login session or increasing traffic monitoring. The motive behind this is to restrict the intruder from accessing resources and curbing the attack consequences.

Proactive IDPS takes on-time intervention tactics to prevent an attack in real time [72] [73]. The major difference between reactive and proactive IDPS is that the latter prevents an attack from taking place, whereas reactive IDPS acts after an attack has occurred. An

example of proactive response would be that it stops a network packet from reaching the destination by dropping the connection itself.

2.3 Attacker, Intruder & Vulnerabilities:

Attacker and vulnerabilities are bound tightly with each other. An attack is launched after vulnerability in the system is exploited or exposed. Vulnerability is exploited by a threat or unauthorized access to a weakness in internal controls, network or system security components, hardware or software modules etc. [74]. Vulnerability does not necessarily lead to attacks. It is an indication of possible threat to the system. Any attacker always relies on such exposed vulnerabilities to launch an attack and gain access to critical network resources.

2.4 Raw Alert log processing: Aggregation and Correlation Techniques

Host based intrusion detection and prevention systems (IDPS) works on different events generated by operating systems. Operating system generates such events when it expects something malicious for example, when CPU utilization is high, some application tries to write in system file area or certain service is not responding etc. An alert generated by HIDPS contains information such as hostname, service name, event timestamp, and signature id and signature name [50] because these system works only at application level and have visibility of only applications. Network based IDPS, on the contrary, works on network packet data. In an NIDPS, an IPS sensor is placed at network ingress point, or some other place where one can monitor the packets of entire network. The IDPS sensor monitors network traffic and inspects packet transmissions for malicious behaviour. It generates alert when it observes such malicious behaviour. The alerts generated by NIDPS or perimeter based IDPS contains information such as source IP, destination IP, event timestamp, signature id, signature name and protocol details [50] because these systems work at network layer and have visibility of only network protocols and network details. Such IPS sensors can be placed at various locations along the boundary or periphery of the network. It is then known as Perimeter based IDPS [51]. Perimeter based IDPS is typically deployed at the entry points of different subnets. The purpose of such deployment is to

protect every subnet from generating threats within the network and across the network but within the organization.

This information contained in an alert is crucial for the network administrator to identify the attacker's identity and to take correct preventive decisions for that attack, which includes applying operating system fixes, applying application fixes, blocking of unused network services and so on. However, a single alert may not indicate complex attack scenarios such as DDOS, application targeted attacks followed by port scan, botnets etc. [50] In addition, the number of network packets and log entries that are inspected periodically has a direct impact on number of false alerts generated by an intrusion detection system (IDS) [54]. Statistical analysis shown in this paper [59] states that in live scenario 92.85% of false alerts are false positives and 7.15% are false negatives. Scalability issue in IDPS occurs when network grows and it leads to aggravation of above-mentioned problems.

In order to address these issues, alert correlation is used. Effectiveness of Alert correlation technique depends on how it deals with above-mentioned problems. The problem with the current IDPS systems is that when any host is found involved in malicious activities, it blocks the entire set of processes of the same host irrespective of the fact that most of such processes are innocuous. When one or few processes are malfunctioning and the host is blocked, that will affect a lot of productive time.

2.4.1 Differentiating need of raw alerts and Correlated Alerts:

- **Alert as Raw data and Its Consequences**:

There are certain alerts, which are sufficient to provide attack information. An example of such alert is when any user inside the network is aware about certain system vulnerability such as buffer overflow. Such user will launch an attack on this vulnerable system and an alert is generated. This alert describes the attack in entirety. However, for the same vulnerability, an outside user, unknown to this vulnerability, will first perform port scan followed by application probing. Once he gets system information, he would launch the same attack as inside user and similar alert will be generated. The major difference here is, even though both alerts are similar, the first case describes the entire attack. However, in the second case, attack instances for port scan, application probing and buffer overflow has to be correlated to describe the entire attack.

- **Addressing IDPS issues by Correlating Raw alert log**:

In order to address complex attack scenarios, certain issues such as alert flooding, alert context, false alerts and scalability should be addressed. Alert flooding is caused by an IDPS at the lowest level of abstraction. The fact that a human security expert then studies the mass of alerts intensifies the probability of wrong or inappropriate decision taken. Alert context means relating similar attack instances. The existing detection systems are usually capable of detecting various types of attack, but in between the occurrences of attacks, lay various instances of similar attacks, which the detection systems are not able to differentiate from one another [59]

Correlation of attack alerts tries to address IDPS issues mentioned above such as alert flooding, alert context, false positives and Scalability. Alert flooding is caused by multiple alerts generated for the same attack instance. For example, when a user launches a large ICMP echo attack, it would be normally continuous attack on a destined system. Alert will be generated for every large ICMP echo packet. We can group such alerts on the basis of source IP, destination IP, and attack and time interval. Such temporal correlation of alerts will reduce the log size considerably.

Alert context means grouping similar attack instances. As explained in the previous section for buffer overflow attack, we can group similar alerts on the basis of source IP, destination IP and time interval. False positive means alert generated by an IDPS to protect or report system vulnerability which does not exist. Considering the same example, if external user had tried different attacks for which system is not vulnerable, and IDPS has generated alerts for the same, and then correlation will help to mask them as false alarms.

Scalability issue occurs for an IDPS when large number of alerts is generated as network grows. As correlation deals with alert flooding and false positives, final number of alerts that needs to be inspected reduces considerably.

The network event and alert logs or TCP/IP connections are analyzed by almost all existing IDPS. The purpose of analysis is to detect non-compliant activity within or outside the network, which would compromise network performance. The alert log comprises of information regarding source and victim of the attack and might also include attack type [54] (e.g., SQL injection, buffer overflow, or denial of service). Over a period of time, same attack instance might span over many network connections or stored as log file

entries, leading to a mass of attack alerts. During analysis of this alert mass, it is difficult to correlate these multiple alert entries to the same attack instance. To address these issues and accurately correlate this kind of mass alerts, data mining techniques such as building classification models from identified attacks or using rules to associate similar attack instances have been widely used in the past[67] [68] [69] [70] [71]. In this section, we discuss at length existing solutions available to analyze and correlate heap of attack alerts, problems or issues with these solutions and our proposed solution for improvement.

2.4.2 Time based aggregation:

In the approach mentioned in [55] authors use attack thread reconstruction approach. This approach depends on rigorous sorting of alerts within a temporal window of fixed length, sorted according to the source, destination, and attack type (classification). Another approach clusters similar attack occurrences within a fixed time window and matching their source and target [56]. Here the raw alerts are logged in a relational database. This approach fails when alert rate generation is high. Combination of alerts and time window used to group those alerts leads to more number of false positives. Routine administrative activities like valid port scan can be categorised as a critical alert and can affect the overall alert grouping.

2.4.3 Similarity based aggregation:

The proposal for aggregating raw alerts described in [57] closely relates to our proposed solution. Link-based clustering technique repeatedly unifies alerts into more generalized ones. It tries to decipher the rationale behind the majority of alerts, and if the alert is not critical or if it is a false alarm, it is removed subsequently. This approach fails in situations when attack instances occur only within a limited time window. This is because attacks which are unified into generalize alerts are taken from different time window to find out generalized patterns. Furthermore, this algorithm only finds root causes for mass alerts. Attacks with less number of alerts are ignored completely in this approach. In our proposed model, we group attack instances only when they are correlated.

2.4.4 Partial Completion Filter based Aggregation:

A weighted signature generation scheme is used for integration of ADS (Anomaly-based detection system) with SNORT in [58]. This hybrid system tries to combine advantage of signature based system like Snort and anomaly based detection. ADS first try to detect

anomalies from already mined anomalous traffic episodes from Internet. Once anomaly is detected, HIDS (Host-based intrusion detection system) extracts signatures from it and adds them into the SNORT signature database. As this anomalous traffic is not correlated, pattern generated from this traffic is more likely to have false alarms. Anomaly detection is local to network and networks differ in behavioural tendencies. So anomaly for one network may be genuine traffic for another. Without doing any localized correlation of anomalous traffic in network, this scheme will generate more false alarms.

One of the proposals uses partial complete filters (PCF) to co-relate and aggregate attacks [62]. But these filters are crafted so as to aggregate only scanning and bandwidth attacks. Also, since these filters allow counters used for packet hashing to decrease, false negative rates tend to be higher. It generates hash values using source and destination address and maintains SYN & FIN tracks to find out PCF counter. Higher number of PCF counters points towards attack. But this approach only works for TCP and not for any other protocol. It correlates SYN and FIN packets to find out attack but it doesn't check the relevance of the attack. It can generate many false alarms when some running services are temporarily closed or when administrator runs some diagnostic tools in network to find out legitimate open ports.

2.4.5 Situation based aggregation:

Aggregation and Correlation (AC) algorithm [63] establishes Aggregation relationship and Correlation relationship between alerts. Aggregation relationship establishes situational relation between alerts and Correlation relationship tries to find out duplicate alerts from log messages. Also, the algorithm tries to find out common characteristics between alerts and aggregate them into situations.

Following illustration in Figure 2.14 explains how Correlation relationship is being established by finding out duplicate alerts from log messages. We can see that different sources are trying to attack a Web server running on Host with an IP Address 10.10.10.61. They are trying to launch phf cgi script attack to find out details of groups and user passwords by trying to access /etc/groups and /etc/passwd files.

```
WebIDS: 934190025 pattern(cgi) phf 10.10.10.62
        "GET /cgi-bin/phf?/etc/group HTTP/1.0" 200 442
WebIDS: 934190025 pattern(UrlSuccess) ^2 10.10.10.62
        "GET /cgi-bin/phf?/etc/group HTTP/1.0" 200 442
WebIDS: 934190025 decision(followup) warnings 10.10.10.62
        "GET /cgi-bin/phf?/etc/group HTTP/1.0" 200 442
WebIDS: 934190027 pattern(suspiciousCgi) passwd 10.10.10.62
        "GET /cgi-bin/phf?/etc/passwd HTTP/1.0" 200 444
WebIDS: 934190027 pattern(cgi) phf 10.10.10.62
        "GET /cgi-bin/phf?/etc/passwd HTTP/1.0" 200 444
WebIDS: 934190027 pattern(UrlSuccess) ^2 10.10.10.62
        "GET /cgi-bin/phf?/etc/passwd HTTP/1.0" 200 444
WebIDS: 934190027 decision(followup) warnings 10.10.10.62
        "GET /cgi-bin/phf?/etc/passwd HTTP/1.0" 200 444
RealSecure: 934190025 1 HTTP\_PHF 10.10.10.62:9285 10.10.10.61:80
            URL="/cgi-bin/phf?/etc/group"
RealSecure: 934190027 1 HTTP\_PHF 10.10.10.62:9304 10.10.10.61:80
            URL="/cgi-bin/phf?/etc/passwd"
RealSecure: 934190027 1 HTTP\_Unix\_Passwords 10.10.10.62:9304
            10.10.10.61:80 URL="/cgi-bin/phf?/etc/passwd"
```

FIGURE 2.14 Alert Scenarios for AC Algorithm [63]

WebIDS has generated alert messages as a warning because phf cgi script has been requested and web server returned 200 OK success codes so apparently it is found. WebIDS generated two warning messages because it has established Correlation by finding out unique attack (URL as an attack data, here two different resources has been requested in URL query parameter). Real Secure has generated three warning alert messages because it has established correlation between Same Source, same destination & same URL (URL as an attack data or attack class).

AC algorithm tries to find out common characteristics between alerts and aggregate them into situations. We studied a commercial product called FortiAnalyzer [73], which generates similar reports which AC algorithm refers as a situation. We covered both in following explanation in Table 2.1:

TABLE 2.1 Comparison of Alert Aggregation in AC Algorithm and FortiAnalyzer

Alert Characteristics	**AC Algorithm Situation**	**FortiAnalyzer Report**	**Attack Scenario**
Alerts with the same source, the same target and belonging to the same alert	Situation 1	Not Available	This allows one to detect, for example, an attacker who is launching a series of

Alert Characteristics	**AC Algorithm Situation**	**FortiAnalyzer Report**	**Attack Scenario**
class			Web server attacks against a single Web server.
Alerts with the same source and destination	Situation 2-1	Top Sources for Most common destinations	This detects an attacker who runs a multiple attacks against the one or more services available on the target
Alerts with the same target and belonging to the same alert class	Situation 2-2	Top Destinations for Most common attacks	This situation can be used to detect a distributed attack against a single target
Alerts with the same source and belonging to the same alert class	Situation 2-3	Top Sources for Most common attacks	This situation allows one, for example, to find an attacker who is running a series of name server attacks against a set of name servers.
Alerts with the same source	Situation 3-1	Top attack sources	The goal is to detect a single attacker who runs various attacks against different targets
Alerts with the same target	Situation 3-2	Top attack destinations	This allows one to detect distributed attacks

Alert Characteristics	AC Algorithm Situation	FortiAnalyzer Report	Attack Scenario
Alerts belonging to the same attack class	Situation 3-3	Top attacks per category	This situation is triggered if, for example, a large number of people are trying out a new attack that was recently posted to a hacker mailing list

It is a good example of characterising different scenarios into situations. But it should have a consideration of whether those source and destinations are part of perimeter or outside of perimeter. When we consider alert attack as a characterizing parameter, we should also consider the severity of attack along with applicability of the attack. What if Web server is running on the destination is not vulnerable to this attack? What if this attack is being generated from inside the network then outside the network? Many such questions remain unanswered here. In our proposed solution, we try to answer these questions.

One more technique based on fuzzy-logic is proposed [64] for aggregation. It assigns scores to prioritize alerts and further reduce number of alerts by alert rescoring technique. Core to their architecture is alert scoring metrics and inference component. They compute several metrics on alerts received and evaluate alerts using these metrics as indicators. From these computed metric values, a Fuzzy logic inference engine calculates the overall alert score. On reception of a new alert, the associated scoring metrics such as Applicability metric, Importance of Victim metric, Sensor Status Metric, Service Vulnerability Metric etc. are computed based on the available security parameters, environmental parameters and vulnerability knowledge base using tools such as Nmap and Nessus. These tools are capable of scanning the network and list down running or installed applications and services which can be mapped with the generated alert. But it might not be accurate enough in case similar application or services are running on same host. So it is advisable to have application or service name when alert is being generated.

One more algorithm called Generic algorithm (GA) [65] works more on training data set to eliminate false alarms and to generate appropriate alerts. This pre classified data set is generated by capturing network data using sniffer for a period of time. Longer the time, more enhanced the pre classified dataset will be. Data set includes information like source network information (IP & Port), destination network information (IP & Port) & protocol used (TCP, UDP). Once this data set is prepared, Administrator needs to mark every entry (connection) as a intrusion or clean. So this pre-classified data set must include both type of traffic entries.

This pre-classified data set is then used to create rules. This rules are more like conventional network firewall rules where 5 tuple (Source network information, destination network information and protocol) information will be used as a matching criteria and action will be Allow (if pre-classified set has marked it as a Clean) or Deny (If pre-classified data set as marked it as a Intrusion).

An example of a rule can be:

if {the source network information of connection contains: (IP=10.1.1.1 & port = Any); destination network information of connection contains: (IP=1.2.3.4 & Port=80); & protocol used=TCP} then {Intrusion}

This rule will detect an intrusion if the rule is matched by IDS and any service requested from this address is rejected. However, in practical scenarios, only specific service should be rejected rather than any service from this address and it may not block all intrusions which have not occurred during building of data set.

Another way of Alert aggregation is to define a similarity operator between two alerts and generate output between 0 and 1[66]. If two alerts are more similar then output will be more near to 1. Similarity can be compared on the basis of Time interval (if to alerts are more near on time span then consider them similar), IP Address (if they belong to same network segment), User data (if it belongs to same user) and Process & Service data (If it is being generated by same service group or process group). But it doesn't consider version information of process or service. It also doesn't talk about adding the process and service information in alerts.

2.5 IDPS Performance Evaluation Criteria:

The two most commonly used factors for evaluating or measuring an IDPS performance are false positive (FP) rate and attack detection rate. The trade-off between the two can be visualized using Receiver Operating characteristic curve (ROC) [78][79][80][81]. However, these two parameters are not sufficient to measure overall impact of IDPS and we require more information for the same. Many techniques such as Bayesian rate for detection [82], cost expected [78], data sensitivity [79] and capability of attack detection [80] have been suggested.

Our survey [63][77][78][79][80][82] says that we require accuracy score, alert precision score and confidence level for overall performance evaluation of IDPS. We also need network productivity and throughput values as optimal performance indicators.

We define the following terms:

Ta = Total raw alerts generated and logged.

Tp = Total true positives

Tn= Total true negatives

Fp= Total false positives

Fn= Total false negatives

We define accuracy score as:

AS (Accuracy Score) = $Tp + Tn / Ta$

Let AP (Alert Precision Score) be defined as number of attacks actually detected and belong to a class of attack.

AP = $Tp / Tp + Fp$

Confidence Level (CL) measures exact correctness of class of attack detected.

CL = $Tp / Tp + Fn$

There is a trade-off between the two scores: Alert precision score and confidence level. As the number of detections increase by lowering of the threshold, the confidence level will increase, while precision score is expected to decrease. We can plot a graph showing the alert precision vs. Confidence level for a given IDPS to evaluate the performance of IDPS.

2.6 Survey Conclusions:

Our literature survey leads us to the following conclusions in different functionalities of IDPS:

- Traffic Monitoring:

a) Network level sensors are incapable of processing encrypted data
b) Crucial information is missed by network level packet inspection
c) Host level sensors are more time and resource consuming
d) Host level sensors cannot cope with modern day operating system complexities

- Access and Admission Restriction

a) Single policy for internal and external users/hosts aggravates problems
b) Restrict admission of malfunctioning host in case of validation leads to productivity problems

- Vulnerability Detection

a) Architecture: IDPS architecture should be able to capture benefits of local and centralized location in network.
b) Deployment of IDPS should be hybrid, a combination of host based and network based deployment, for effective network performance
c) Information Source: IDPS Rule set / signatures incapable of detecting known attacks with new strategy / unknown attack source.
d) Relevance of alert log: Improper correlation of raw alerts leads to wrong preventive actions.
e) IDPS should always maintain an optimal balance between accuracy and network productivity with highest possible confidence level for detection.

CHAPTER – III

Proposed Alert Aggregation Model

3.1Problem Statement:

To generate highly optimized aggregated and correlated raw alerts by application profiling, mask as many false positives as possible, using traffic monitoring & vulnerability detection, generate set of precise advices for prevention in real time, without compromising network productivity.

3.2Scope of our research

Our proposed prototype model is flexible enough to be used with any existing open source of commercial IDPS. The model aims to provide enhancements in overall detection and prevention approach, and concentrate on identified problem areas of existing IDPS. The scope of our research is defined as:

- Simulation & utilization of raw alert dataset with standard parameters and our own configured ones
- Bifurcation of self-determining alerts and related alert scenario from raw alert log.
- Highly optimized aggregation and correlation of raw alerts into meaningful alert context
- False positives masked with high accuracy and confidence level
- Grey positives identified and correlated with correct alert context
- Quarantine / block applications to achieve optimal network productivity
- Generate preventive advice for network administrator for preventive decisions

3.3Objectives of our research

The overall objectives of our research are summarized as:

- Use a hybrid deployment approach for combining advantages of host and network based IDPS
- Identify attacks in real-time
- To use alert aggregation for eliminating false positives.
- To use correlation to group multiple alert instances to form a single threat scenario with highest confidence level and accuracy.
- Quarantine applications instead of hosts for better network productivity and throughput.

3.4Original Contribution by the thesis

Although multitude researchers have worked towards enhancing existing Intrusion detection and prevention systems (IDPS), major breakthroughs in terms of prevention of attacks on network resources is not yet accomplished. Critical open research areas hammering the network security experts are: Increasing rate of false positives and negatives, smart crafting of known attacks in unknown forms, and identifying and analyzing unknown attacks [68][69][70]. Attack detection and prevention is not the only remedy. It is equally important to analyze the strategy behind these attacks, relate similar attack instances and understand the motive behind these attacks. Almost all IDPS generate alerts after attack detection and do not concentrate on the analysis of those attack scenarios, which is the need of the hour.

We concentrate on both false positives and grey positives. Grey positive indicates possibility of an attack, and does not directly indicate an attack. It can either result into a true positive or false positive, depending on the context of an alert. For example, we consider the case of an alert generated during nmap SYN scan. If it is generated for Trojan infection, it is a false positive. But, if an alert is generated for an attempt to scan VNC protocol, then it is categorized as a grey positive. In our proposed solution, we perform

aggregation and correlation using application related information. Therefore, we can identify alert context for both false and grey positives. Our aggregation and analysis results in proper preventive decisions and assists network administrators to react in response to attacks in real time.

We segregate an alert considering direction of attack. The alert is identified as Inbound / Outbound, client-side / server-side. This is crucial during quarantine process. If an inside client/server is found vulnerable, administrator can quarantine the associated application and apply hot-fixes, if possible. However, if an outside client / server are identified as an intruder, it will be blocked.

All of the already mentioned solutions have their own pros and cons. So we propose a combined approach by bringing best of all together. Summarizing the research gaps, we aim to achieve the following enhancements with respect to existing IDPS approaches:

- To restrict the admission & access of Host in the network like NAC but only when it is required.
- Isolate and restrict the access of applications running on the host when found potentially vulnerable or infected.
- Make our proposed model more practically implementable by working at network traffic level but at the same time provide a solution to classic NIDPS problems like False alarms.

As already discussed at length in survey chapter, both host based and network based conventional approaches have their own advantages and issues. We have therefore used a hybrid deployment and architectural approach for our proposed model, which extracts maximum possible information both at host level and network level. The task of intercepting live application data is performed locally at host level, and complete analysis of global attack scenario is done centrally. Thus, our hybrid model takes maximum advantage of both the approaches.

As our solution is based on IDPS technology, our solution also inherits the problems of IDPS technology. We propose a more efficient approach to reduce raw alert log size with maximum accuracy and high confidence level. To achieve this, we add semantic information to alert generated by an IDPS such as application name, version, severity of

alert and timestamp. This information leads us to actual source of attack with application name and version with attack severity and timestamp. Our results shown at the end proves our claim.

For testing, we have simulated raw alert dataset and we show how this semantic information helps our aggregation and correlation methodology. It reduces and classifies the raw alert data to provide precise preventive advice to administrator for prevention of attacks at run-time. The result also indicates that we take a much smaller granularity by addressing processes on the hosts and not the hosts themselves. Throughout the experimentation, our solution maintains optimal network productivity.

Thus our proposed solution achieves run-time attack prevention without compromising network performance. It is more in close proximity to IDPS technology but we are enhancing the same, which can be applicable to solve the problems of network admission and access restriction.

3.5Proposed System Architecture:

We have designed and presented following subsystems, which constitute our proposed architecture:

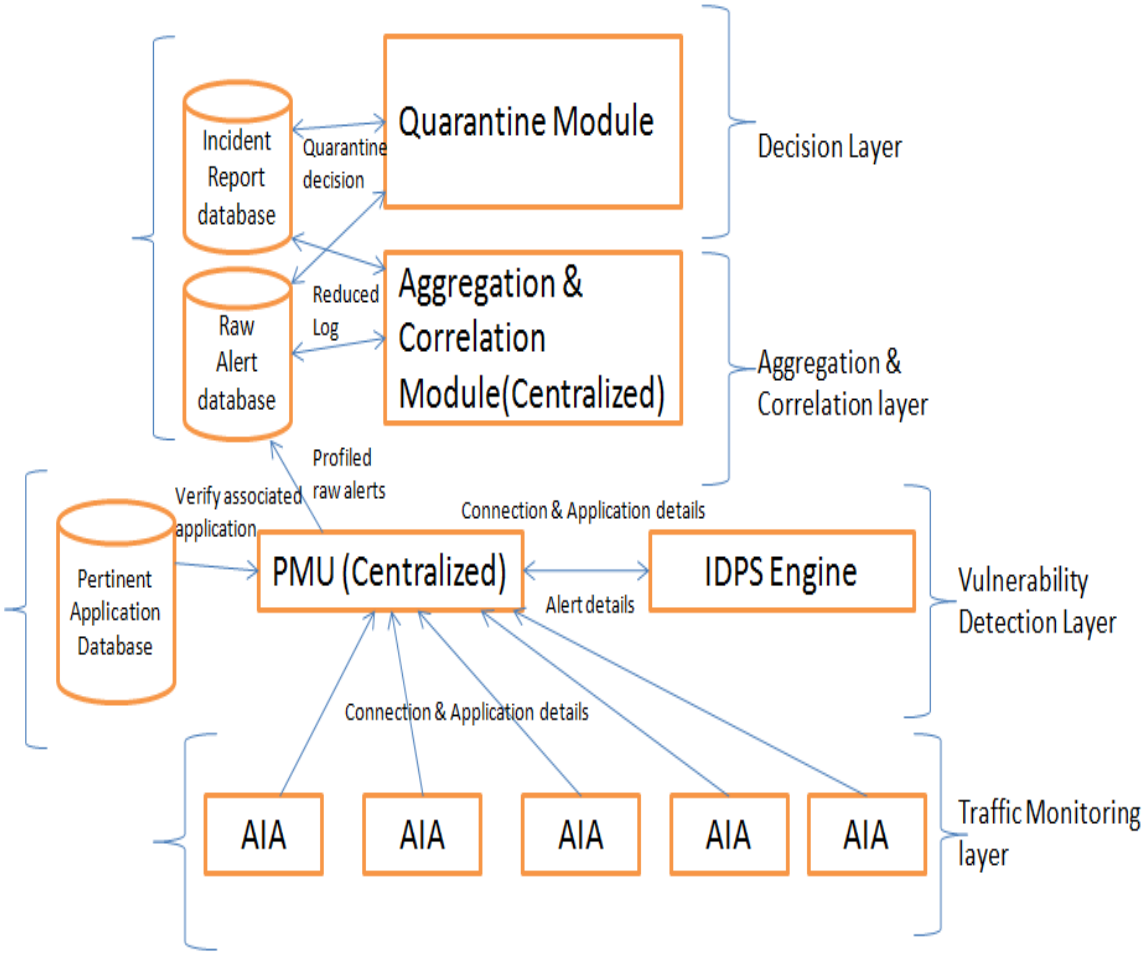


FIGURE 3.1 System Architecture

If deployment of an IDPS answers to the "what" part of an alert verification, architecture of an IDPS answers to the "how" part of the same. Architecture of IDPS can be categorized as distributed or centralized. This determines how IDPS performs its two vital operations: Traffic Monitoring and Assessment for vulnerabilities. A distributed IDPS delegates the traffic monitoring and assessment functionalities to multiple entities.

The type of information that an IDPS analyzes in order to raise an alarm or an alert is

largely dependent on the architecture of the same. Two primary functions performed by NIDS or HIDS are: traffic monitoring and analysis. IDPS implemented in distributed manner performs both of these functions in a distributed manner. An earliest example of this kind of architecture is Distributed IDS [31], capable of performing analysis locally as well as centrally. It uses network security monitor (NSM) components and uses signature-based and behavioural detection locally. Analysis is performed centrally using rule-based expert system. Hence, DIDS is a combination of HIDS and NIDS capable of sharing results with each other. Thus results are analyzed and shared at different levels.
However, almost all centralized IDPS rely on few limited data sensors, with one analyzer to perform monitoring and analysis of alerts. This leads to missed attacks and false negatives. In addition, scalability remains a major issue [31].

3.5.1 System modules:

Our entire architecture is segregated into main four layers: traffic monitoring layer, vulnerability detection layer, aggregation and correlation layer and decision layer. Each layer is responsible for performing functions and then relay information to the next layer. We now explain the basic functionality of each layer.

- **Traffic monitoring layer**: As mentioned earlier, traffic monitoring is the very first step in attack detection scenario. This layer monitors inflow and outflow of network traffic. For each connection initiated, Application Intercepting Agent (AIA) [61] [72] reads initiator application details from by hooks in socket calls. It forwards these details with network packet payload to Packet Matching Unit (PMU).

- **Vulnerability detection layer**: Two components PMU and IDPS engine (any standard IDPS such as Snort or Suricata) work in sync to detect vulnerabilities in packet sent by AIA. An alert sent by IDPS contains “sid” of the signature which matched the vulnerability. “sid” is a unique identifier of signatures of IDPS. PMU performs lookup for this “sid” in Pertinent Application data store (data stores are described in next section). If a match is found, it stores profiled alerts (raw alerts with application details) in raw alert database.

- **Aggregation & Correlation layer**: This layer is responsible for false positive reduction. Aggregation and correlation module at this layer reads raw alerts and

masks false alerts, filters alerts for routine network maintenance activities, and group alerts which are part of single threat scenario with reason for grouping.

- **Decision layer**: Application Quarantine Module (AQM) at this layer reads the correlated log, and quarantines the applications. It also marks them as "quarantined" in the alert database. Preventive advices are stored in incident report database.

3.5.2 Data Stores

Global Application Store:

This data store provides the overall network level snapshot. Global application store is implemented as a data structure to store list of running hosts in the network. For each host, it also stores complete list of application and connection details dynamically. The first level stores a two-tuple hash of MAC address and hostname of each active host. Whenever any application requests for a connection to send or receive data, a five-tuple hash {Source IP, Destination IP, Source Port, Destination Port, Protocol} with application name and version is inserted into the global application store. As shown in figure 4.2, it is implemented as a two-level linked list.

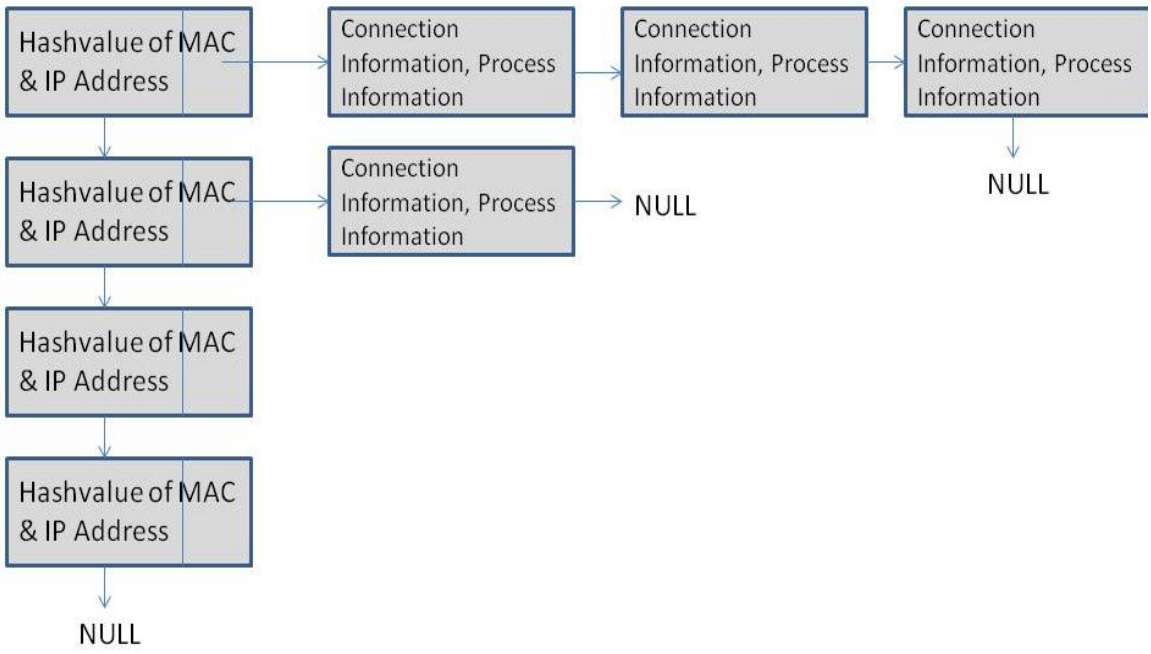


FIGURE 3.2 Global Application Store (Generic)

The number of hosts live in the network always remains dynamic. From the time when the network comes up, the number of hosts actually up and running in the network differs periodically. Also, on each host, varied number of applications runs at different time instance. In order to maintain a list of live hosts at a given time instance and list of applications running on each host at that same time instance, we implement Global application store as a linked list. Nodes are added and deleted as and when a host comes live or goes down. Also, nodes are added or removed according to the running status of applications on each host. The first level contains information of hosts in network. Each node of first level list point to another linked list. Each node of second level list represents application and connection information of each application running on a host. Figure 3.3 shows a snapshot of working hosts and applications running on them.

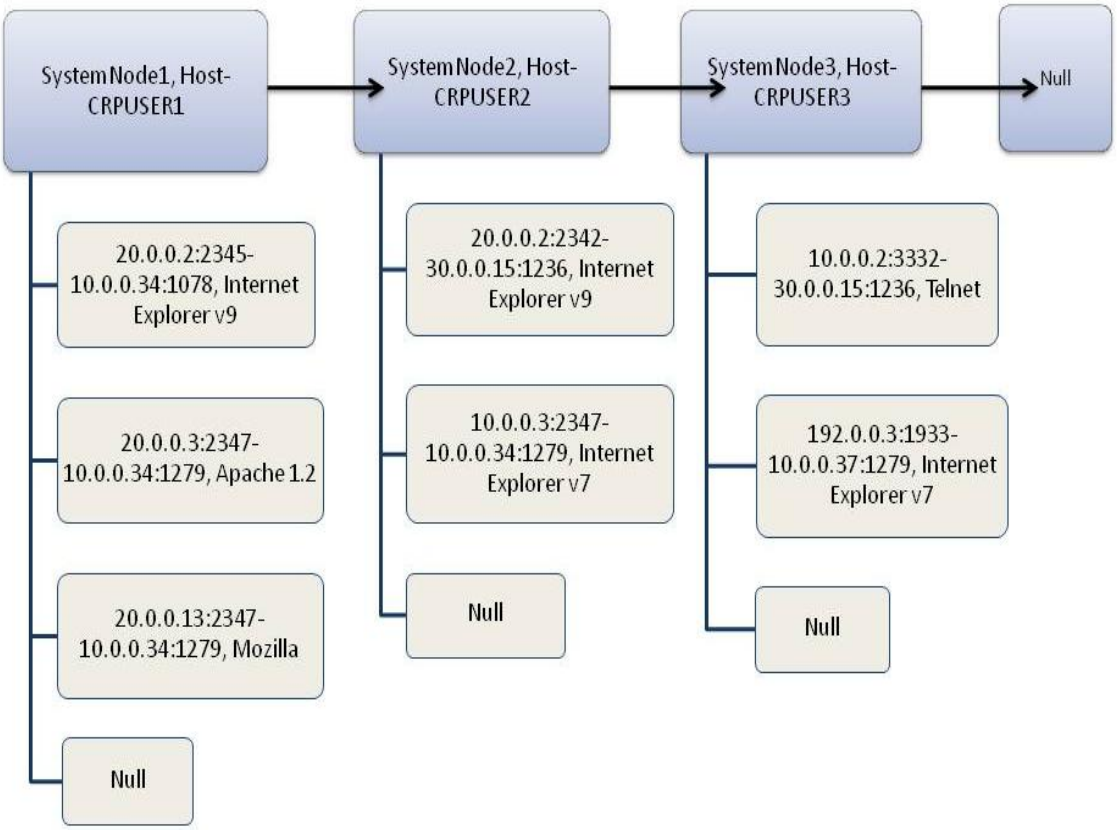


FIGURE 3.3 Global Application Store

As shown in the figure 3.3, the first level linked list contains three nodes for hosts 'SystemNode1', 'SystemNode2' and'SystemNode3'. The second level list for each node shows the total number and details of connection made by applications running on each host. There are three applications running on 'SystemNode1', two on 'SystemNode2' and 'SystemNode3' each.

Alert Store Database (Raw Alert Log generated by pattern matching unit):

The pattern matching unit (PMU) stores raw alert incidents into an alert database. It is configured centrally at server level. Raw alert log contains network level information including source IP, destination IP, name of attack from signature and attack severity. It also contains system related information such as name of the application and version. Information stored in alert store database is used for aggregation and correlation. Therefore it is very crucial to store as much granular information as possible. Network level information provides initial idea about the source of attacker and corresponding victim. Attack severity is used to calculate criticality of attack as minor, major, critical or fatal. System level information such as name of application and version helps in identifying root cause of attack or vulnerabilities in network resources which need to be attended to. Direction of attack helps us in identifying whether attack is launched from within or outside the network. It also helps us identify whether a client or server resource is victimized.

Incident Report Database:

This database is a result of final root cause analysis performed by our aggregation and correlation module. This database contains precise set of preventive advices for network administrator. It contains cumulative alert score and individual incidents or alerts correlated as a single alert scenario. The cumulative alert score is stored host-wise and application-wise. It decides the confidence level of our final aggregated and correlated log. This score is used for taking preventive actions by the network administrator.

Alert Score (in %): 0-10: Confidence Level: Low
10-30 Confidence Level: Medium
30-50 Confidence Level: High
50-70 Confidence Level: Critical

>70 Confidence Level: Fatal

Cumulative Score for each correlated entry helps admin take preventive measures as:

1) Host-wise alert score shows the following statistics:
    a. Total alert incidents
    b. Total applications participating for alerts

When the alert score for a host is high, critical or fatal, admin can block the host itself.

2) Application-wise alert score shows the following statistics:
    a. Total alert incidents for a given application
    b. Total hosts participating

When the alert score is high, critical or fatal, admin can quarantine the application and block further connection requests from the same.

Pertinent Application Information store:

We have implemented an interface where administrator defines relevance of attack signatures to different applications. In order to identify application relevance to an attack, we have implemented a parser, which reads signature file of IDPS and displays a table of signatures to the administrator. The complete algorithm for this parser, known as rule set tokenization algorithm is described in detail in the next section. The administrator specifies the corresponding application for each signature. The signature and its application are stored into Pertinent Application Information store. Alert correlator uses this information to find out false positives and grey positives during aggregation and correlation process generated by intrusion detection engine and marks them. Table 3.1 shows few actual entries from our signature database. Against each signature parsed and read from signature file, administrator specifies the pertinent application name and version.

TABLE 3.1 Pertinent Application Information Store

Signature ID	Signature Name	Pertinent Application Name	Pertinent Application Version
2017478	IE Memory Corruption	Internet Explorer	7 to 9

Signature ID	Signature Name	Pertinent Application Name	Pertinent Application Version
	Vulnerability		
100000447	Mozilla Firefox DOMNodeRemoved attack attempt	Mozilla Firefox	Any
2101809	Apache Chunked-Encoding worm attempt	Apache	1.3.x
2002993	Rapid POP3S Connections - Possible Brute Force Attack	NA	NA
7393	Smtp_auth_failure	Telnet	Any

3.5.3 Rule set tokenization algorithm

Each entry in pertinent application information store is a result of our Rule set tokenization algorithm, which we discuss in this section. As mentioned earlier, we have implemented a parser, which reads signature file of IDPS and displays a table of signatures to the administrator. The administrator specifies the pertinent application for each signature parsed.

A unique sid is allocated to each signature in the IDPS signature file. One needs to upgrade IDPS signatures whenever unknown threats are detected. For proof of concept, we have taken signature file from Emerging Threats (ET), used by Snort and Suricata, popular open source IDPS. Our parser scans and tokenizes the signature file. After parsing, it generates a new file, which contains three fields: sid, signature name and direction of attack, i.e. to / from client / server.

ALGORITHM: To tokenize and Parse signature file of IDPS:

Input: Signature file from emerging threats.
Output: Signature classification file, in which each record contains fields ‘sid’, ‘signature name’ and ‘flowdirection.’

Step 1: Open the signature file and create a new file for storing result.

Repeat step 2 to 7 while end of file is false
Step 2: Read a Line from the signature file.
Step 3: Tokenize this line using semicolon as token separator.
Step 4: Compare each token with substring ‘flow’. If match found, store the value in variable ‘flow_direction’ else repeat step 4.
Step 5: Compare each token with substring ‘msg’. If match found, store the value in variable ‘signature_name’ else repeat step 5.
Step 6: Compare each token with substring ‘sid’. If match found, store the value in variable ‘sid’ else repeat step 6.
Step 7: Save the variables ‘sid’, ‘signature_name’ and ‘flow_direction’ as a record in a file.
Step 8: Close the files.
After the entire file is parsed, we insert the resultant file’s records into database.

3.5.4 Attack Direction Classification

In our proposed solution, we identify classification of attack direction as a crucial parameter. It determines the preciseness of preventive action taken by our Application Quarantine Module (AQM), which is described in detail in next chapter.
Our first classification decides whether an alert is generated from with the network or outside the network. Second classification is determined by verifying an attempt is made to read or write, from the system call intercepted by our Application Intercepting Agent (AIA). The detail of how AIA intercepts system calls during connection request is also described in next chapter.

As explained in the previous section, each signature contains a token “flow”, with values: ‘to_client’, ‘from_client’, ‘to_server’ and ‘from_server’. This value and the system call (read / write) used, together identifies the attack direction as:

Client-side Inbound / Outbound or Server-side Inbound / Outbound.

CLIENT SIDE ATTACK

In this scenario, client is having a known vulnerability. This client sends a request to a remote server for some service. There is a strong possibility that the remote server launches an attack on this vulnerable client. The connection direction determines whether this attack can be classified as inbound or outbound.

A. Inbound Attack (Attack on inside client):

A client accesses a remote server website using IE. If the website is already vulnerable, it may pose a threat to IE application. Falling in the category of such threats is ‘heap buffer overflow’ attack. In this scenario, IE is vulnerable client, also termed as a victim. The corresponding signature from Emerging Threats (ET) is shown below:

#alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"ET WEB_CLIENT Possible Apple Quicktime Invalid SMIL URI Buffer Overflow Attempt"; flow:established,to_client; content:"|3C|smil"; nocase; content:"|3C|img src="; nocase; distance:0; content:!"http"; nocase; within:20; content:"|3A|//"; within:20; isdataat:700,relative; content:!"|3C 2F|smil|3E|"; nocase; within:700; content:!"|0A|"; within:700;classtype:attempted-user;
reference:url,securitytracker.com/alerts/2010/Aug/1024336.html; reference:bugtraq,41962; reference:cve,2010-1799; sid: 2103192; rev:2;)

Our algorithm stores three fields from the above rule: ‘sid’,’msg’ and ’flow’. ‘sid’ and ‘flow’ contains values ‘to_client’ and 2103192 respectively. As soon as IDPS matches the signature for this attack, our pattern matching unit (PMU: described in next chapter) compares the ‘sid’ returned by IDPS with the sid in the database. If match is found, PMU stores the attack direction as ‘to_client’ and ‘msg’ to the database for the corresponding sid.

For further granular processing, PMU uses application connection information sent by AIA for this attack. If AIA intercepted read() system call, then the attack is inbound. If AIA intercepts write() system call, then classification of the attack is outbound.

Figure 3.4 describes the scenario shown above for client-side inbound attack. Here the system call hooked by AIA is read().

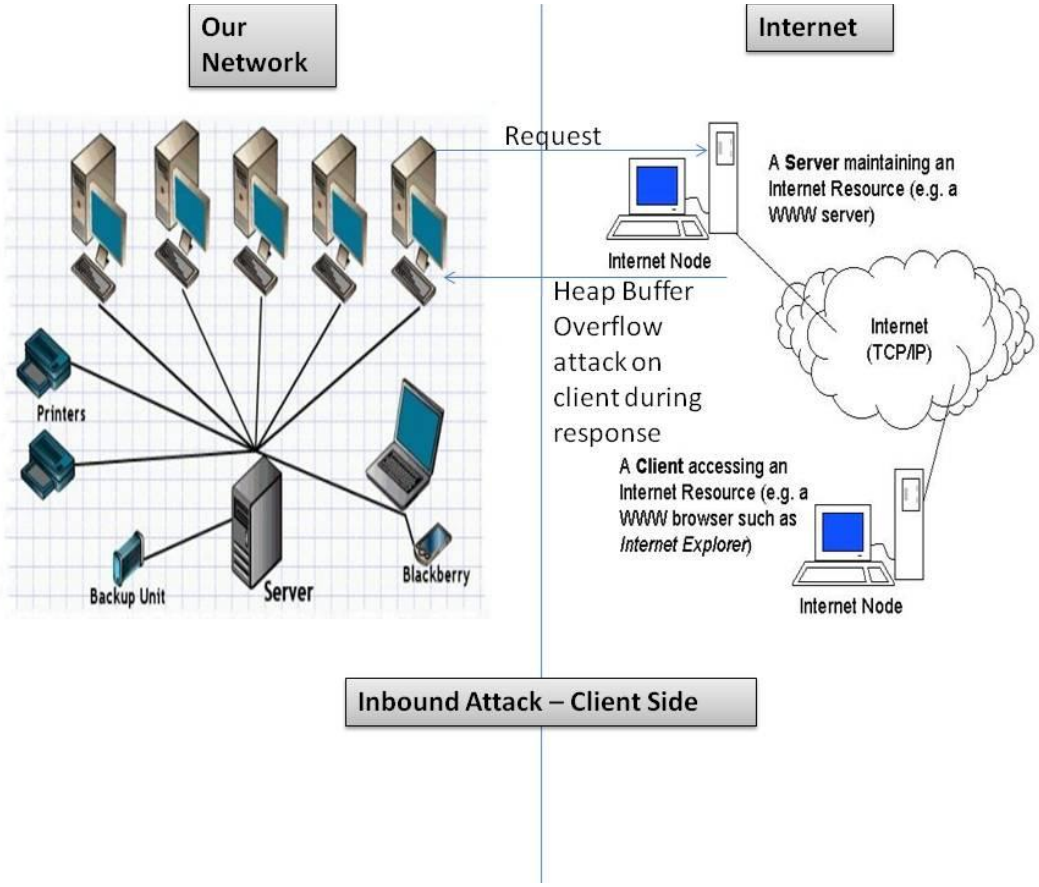


FIGURE 3.4 Inbound Attack on Client

B. Outbound Attack (Inside server attacking on outside client):

IIS server is hosted in our network. Any machine from outside the network uses IE and sends a request to our IIS server. Vulnerability on our server launches an attack on remote machine's IE. Some client from the internet tries to access the site using Internet Explorer. If our server is infected or compromised, then it can launch an attack on the remote client's IE. The signature in this case can be as given below:

alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET WEB_CLIENT Iframe in Purported Image Download (jpeg) - Likely SQL Injection Attacks Related"; flow:established,from_server; content:"|0d 0a|content-type|3a| "; nocase; content:" image/jpeg"; nocase; distance:0; within:30; content:"<iframe"; nocase; distance:0; pcre:"/content-type\x3a\s+image\/jpeg/i"; pcre:"/<iframe.*?src.*?>.*?<\/iframe>/im"; classtype:web-application-attack;
reference:url,doc.emergingthreats.net/bin/view/Main/2008313; sid:2008313; rev:7;)

Here, the direction of attack as given in the signature is 'from_server' and and AIA intercepts read() system call. Hence we term this attack as outbound attack on client.

SERVER SIDE ATTACK

When anyone requests any service from the server, he can also land an attack along with the request. If running application server has any known vulnerability, services can suffer due to attack. Again, depending on direction of connection, attack can be inbound as well as outbound.

A. Outbound Attack:

Our client machine launches an attack named "SQL Injection Attempt" on a remote server. The remote server is victimized in this scenario. The signature for this attack is given below:

alert http $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"ET WEB_SERVER SQL Injection Attempt (Agent CZ32ts)"; flow:to_server,established; content:"|0d 0a|User-Agent|3a| CZ32ts|0d 0a|"; nocase; classtype:web-application-attack; reference:url,doc.emergingthreats.net/2009029;
reference:url,www.Whitehatsecurityresponse.blogspot.com; sid:2010621; rev:4;)

As shown in the signature, attack flow is 'to_server' and AIA intercepts write() system call and sends it to PMU. Therefore, PMU stores information about this attack outbound attack on server.

B. Inbound Attack:

A remote client launches an attack on IIS server, which is part of our network . AIA

intercepts read() system call and flow defined in signature is 'from_client'. PMU records this type of attack as inbound attack on server. Figure 3.5 describes the same:

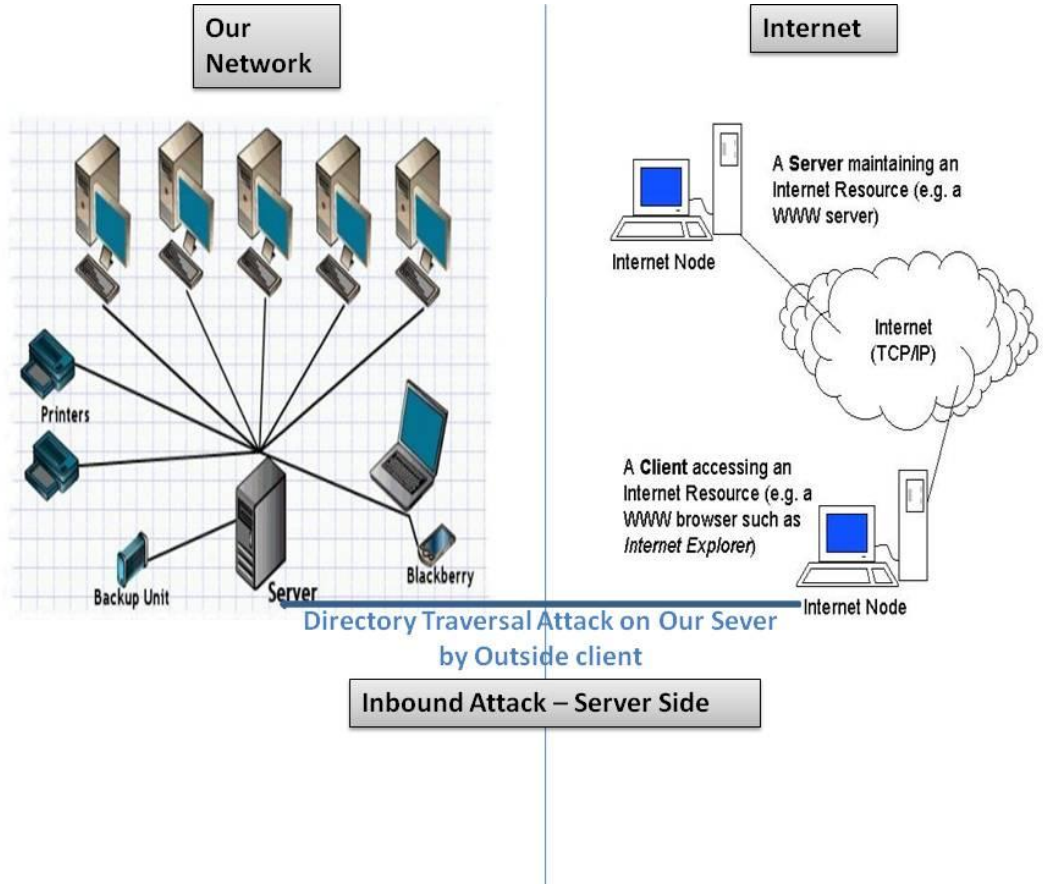


FIGURE 3.5 Inbound Attack on Server

Deployment of IDPS plays a vital role in the eventual performance of the same. Deployment determines what IDPS monitors and scans and the kind of attacks it is capable of detecting. Host-based IDPS are deployed at host-level in a network. They monitor individual host information such as system calls, process usage statistics, and file access log. To achieve this, they usually perform scanning using port-scan or vulnerability scan [50].

Network-based IDPS deploy an IDPS sensor at network ingress point, a central location in a network from where all the traffic passes through [50]. They monitor network inflow and outflow traffic and inspect packet payload information for malicious content.

Perimeter-based IDPS, also known as network-node IDPS monitor network packets destined for a particular end system [50] [26]. They are distributed in nature. Multiple sensors are placed at different locations along the boundary of network. These sensors perform load balancing for traffic monitoring. They are delegated the task of monitoring packet information only for a set of end systems in the network. Primarily IDPS is deployed at host or network level [50].

3.6Deployment Model

Our system architecture has five components as already described in figure 3.1:

1) Application intercepting agent (AIA)

2) Pattern matching unit (PMU)

3) Intrusion detection engine

4) Alert aggregation & Correlation module (ACM)

5) Application quarantine module (AQM)

Our system can be deployed in two modes as shown in Figure 3.6 and Figure 3.7. Application intercepting agent sends raw alerts to Pattern matching unit in out of band deployment as depicted in figure 3.6 which is not in the direct path of network traffic. Deployment shown in figure 3.7 describes inline mode where the pattern matching unit sits in direct line so as to intercept the entire network traffic. In inline mode the application intercepting agent only sends associated application information to pattern matching unit. It works as an agent running on that machine and so we call it application intercepting agent. Inline mode is useful when changes in existing network are possible and PMU can be deployed just before edge router or switch. Out of band is appropriate when one doesn't want to change any network topology.

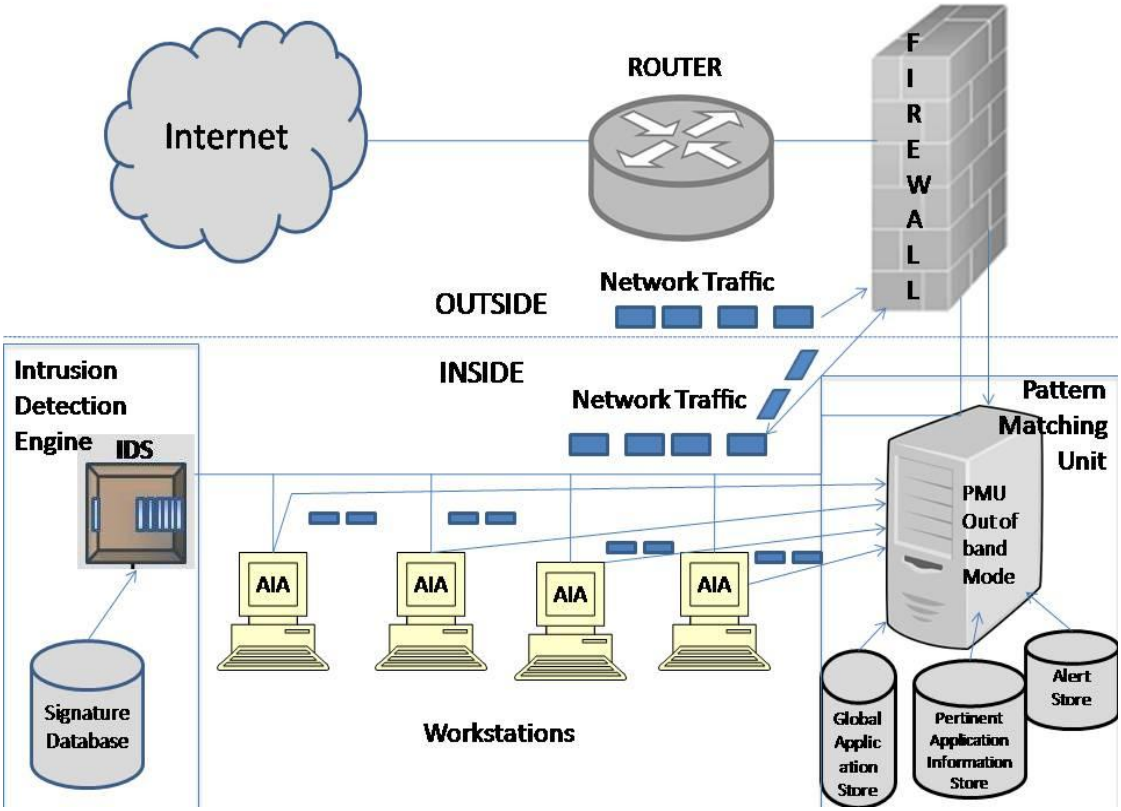


FIGURE 3.6 Out of band Deployment mode

The deployment in Figure 3.6 improvises network performance and minimizes the efforts of a network administrator. It describes a network comprising of workstations and server which is protected by a firewall and an IPS engine. The local network interacts with the external world via a router. It is assumed that the router sends and receives network packets from local network to the internet.

Router is a device recognized in the networking arena, that which is capable of making connections between multiple networks at the transport layer of OSI model. It examines protocol information present in the packet after which it makes the decision of forwarding the same. The router described in Figure 3.6 forwards the network packets from inside network to internet and vice versa after inspecting them.

The IDS engine inspects network packet header and payload information for detection of known vulnerabilities. Vulnerabilities in network security is an amalgamation of three situations: a system or resource susceptible to flaws, an attacker gaining access to

that flaw and the capability of an attacker to exploit that flaw by launching an attack on the system or resource. For identifying such type of potential threats to network resources, an IDS engine uses "signatures" stored in a signature database as shown in Figure 3.6. Signatures define set of rules to detect attacks or intrusive activities on network. The signatures in the signature database contain possible alert information. The packet information is matched against the signature, and a positive match is termed as a threat or an attack.

As shown in Figure 3.6, our deployment is based on the fact that separate preventive measures need to be taken for vulnerabilities within or outside the network. PMU is deployed in out of band mode. Out of band mode means traffic may bypass PMU and hence AIA needs to send all the details to PMU. Also, all the data stores are implemented centrally. AIA is deployed at each workstation.

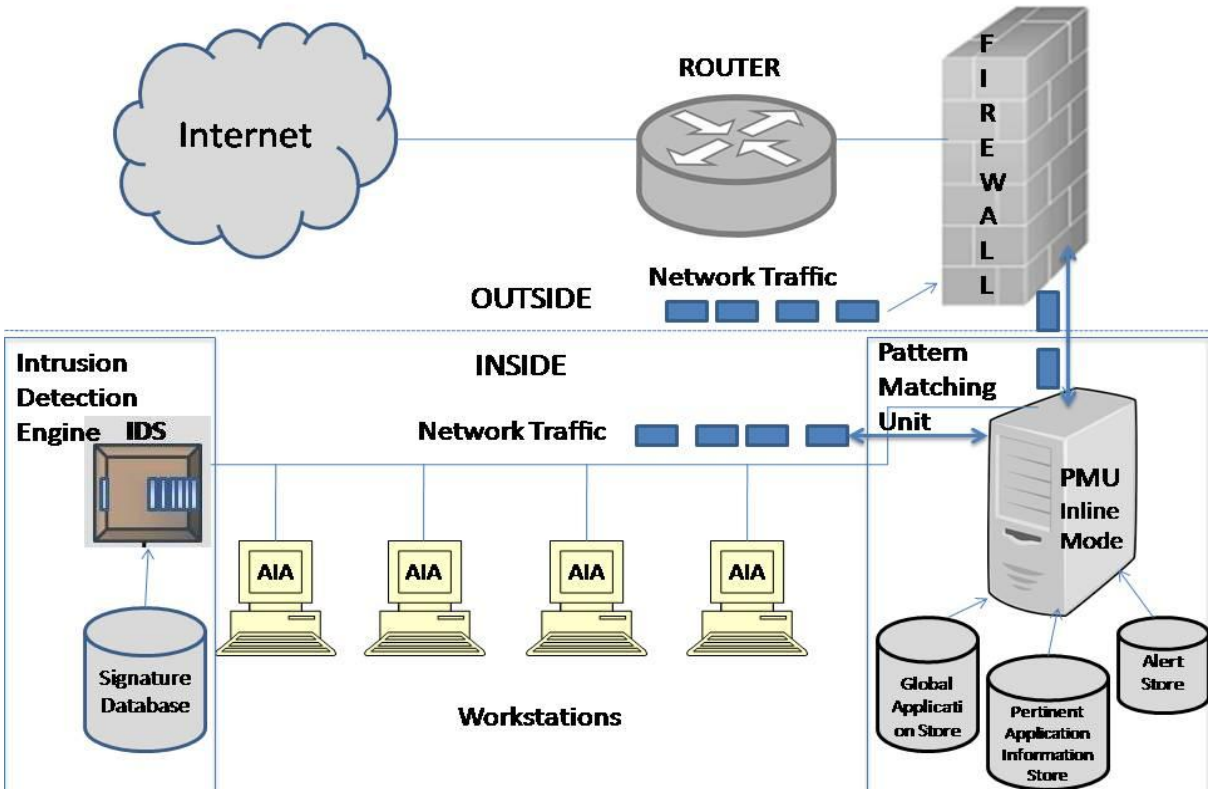


FIGURE 3.7 Inline mode Deployment

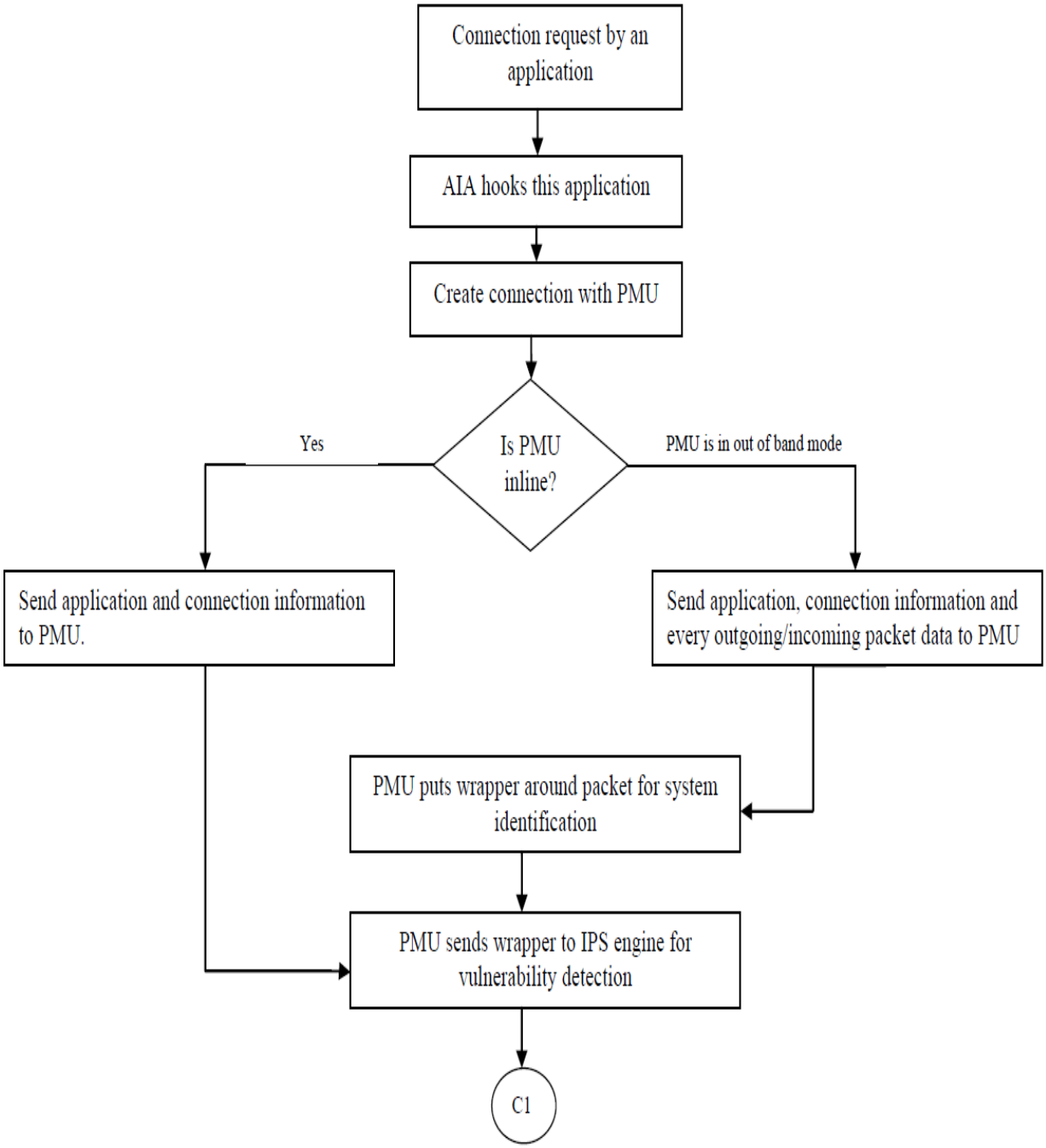
The proposed model advocates a hybrid system, where information about malicious attempt is derived locally and preventive decision making is done centrally. The hybrid

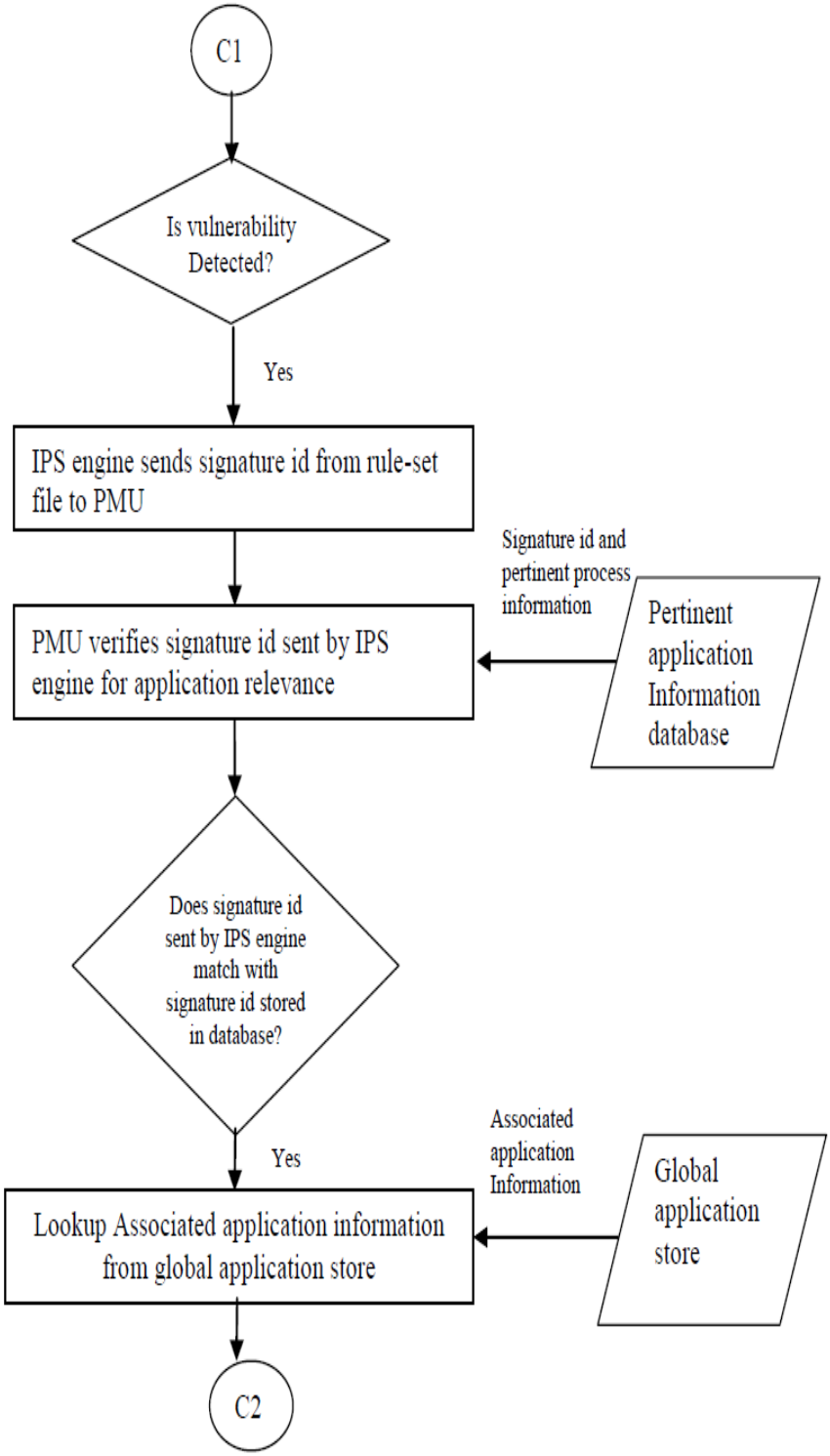
nature of model overcomes two major flaws: Implementing major assessment functionalities at host-level is time consuming as well as resource consuming and it also aggravates network productivity issues. Overall host performance suffers. Network level sensors miss crucial information about an alert due to which possibility of attack increases. The proposed model takes advantages of both the approaches and this leads to maximum productivity in the network. An agent (AIA) to intercept application related information is deployed on each workstation in the local network. This agent relays the application information to a centralized engine (PMU). The engine tries to find out vulnerability, if present with the process. If vulnerability is detected, the application is blocked and the information for the vulnerable application is stored in an alert store database. Network administrator uses this database for preventive actions.

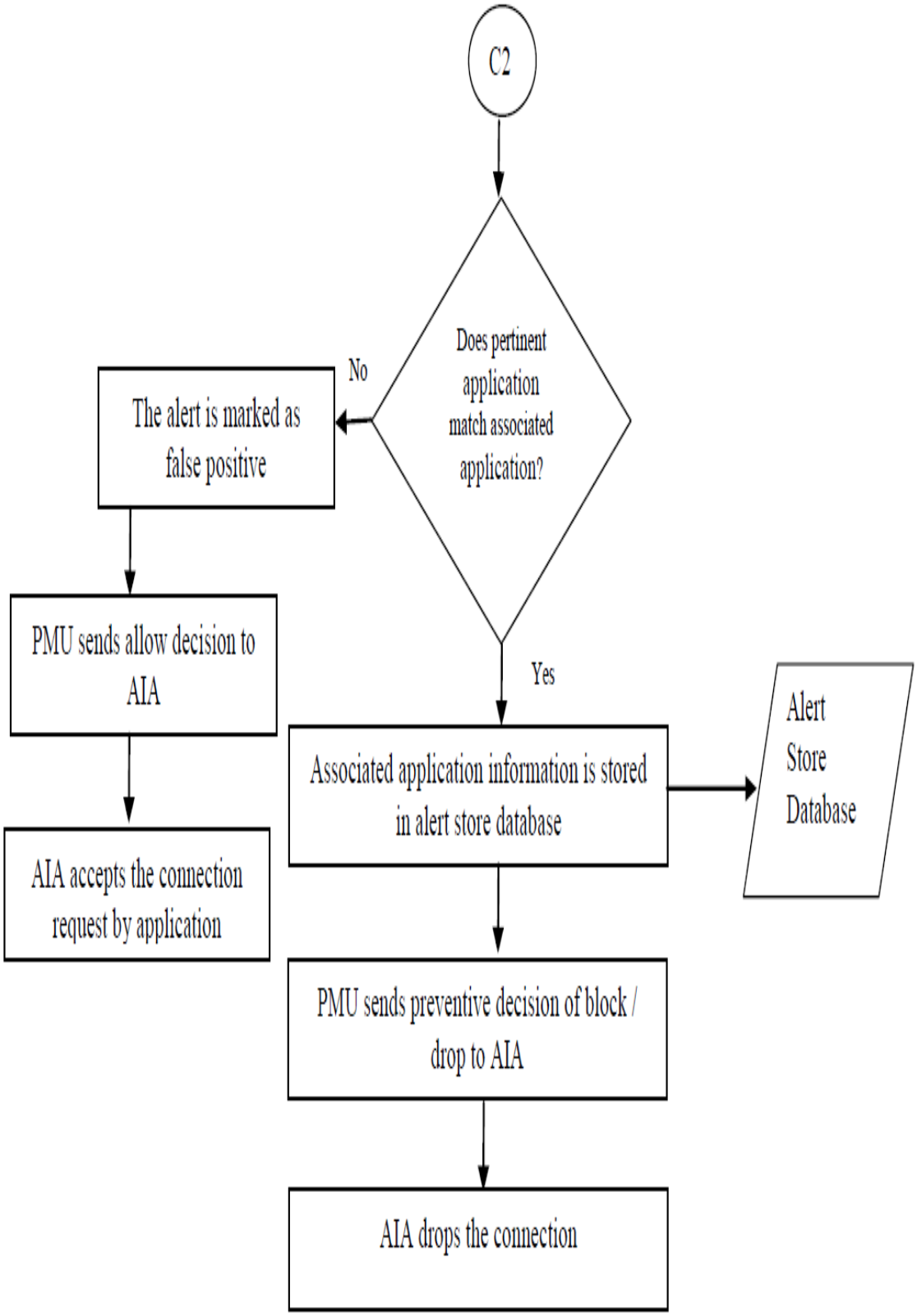
A connection is described as an attempt by an application running on a host machine to communicate with another application on another host. The intent of connection can either be healthy communication or malicious. Our model tries to tap the malicious intent of an application.

3.7 System Flowchart:

The following diagram shows the system flow of our proposed prototype:




C1
Is vulnerability Detected?
Yes
IPS engine sends signature id from rule-set file to PMU
Signature id and pertinent process information
Pertinent application Information database
PMU verifies signature id sent by IPS engine for application relevance
Does signature id sent by IPS engine match with signature id stored in database?
Yes
Associated application Information
Global application store
Lookup Associated application information from global application store
C2


C2
Does pertinent application match associated application?
No
The alert is marked as false positive
PMU sends allow decision to AIA
AIA accepts the connection request by application
Yes
Associated application information is stored in alert store database
Alert Store Database
PMU sends preventive decision of block / drop to AIA
AIA drops the connection

CHAPTER – IV

Workflow of the Model

4.1 Application Intercepting Agent (AIA):

Application Intercepting agent (AIA) is deployed on each host in the network. It consists of two main modules: Network Interceptor (NI) [61] and Event Logger [72]. The following diagram, Figure 4.1 shows the conceptual framework for NI and event logger:

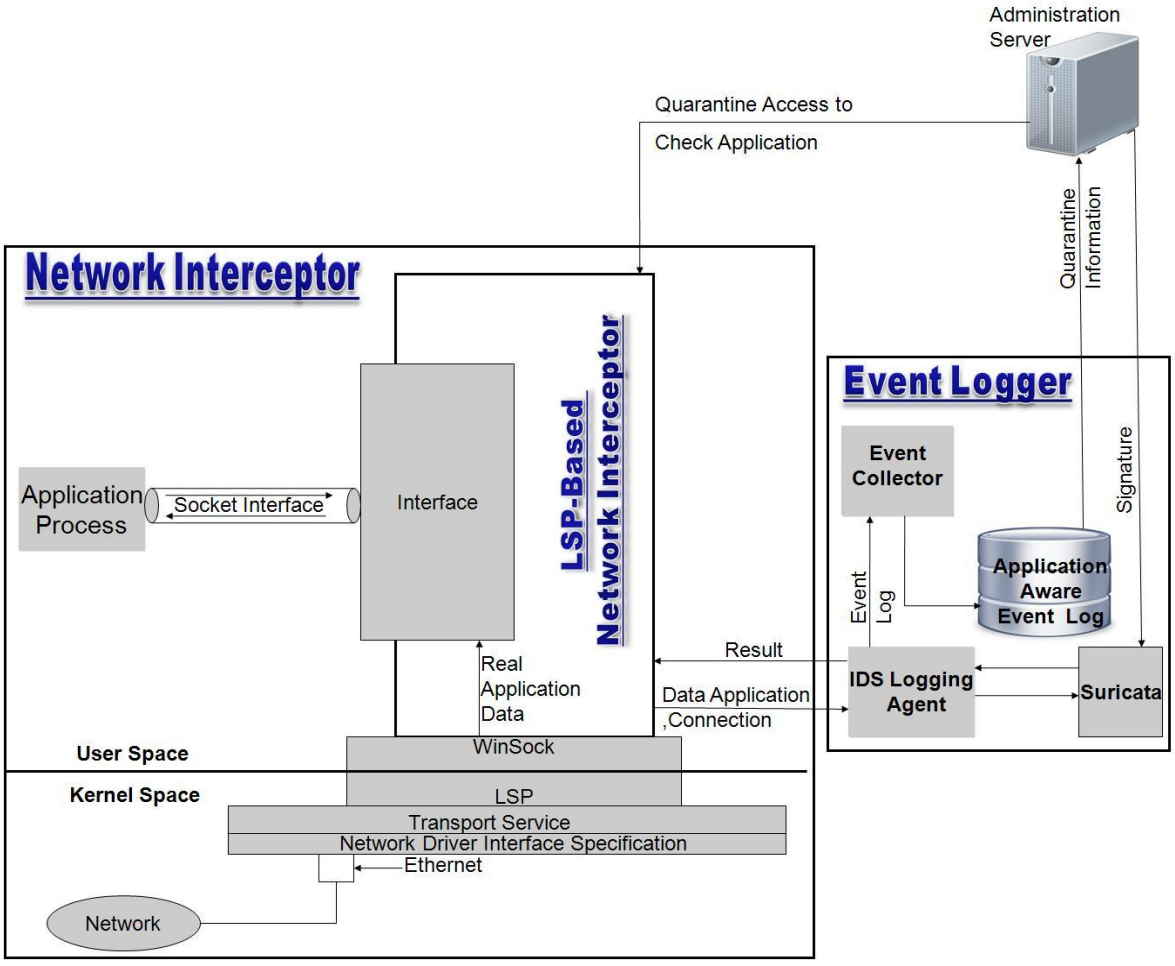


FIGURE 4.1 Conceptual framework for NI, Event Logger [61]

4.1.1 Network Interceptor (NI):

Network Interceptor [61] performs hooks on system functions, and because of this it captures real-time traffic. Due to this, it detects an attack in real-time.

Network Interceptor [61] has two options on Windows operating system: NDIS Framework, capable of intercepting raw network data and operates at Kernel Level. Another approach works at an intermediate level using WINSOCK Hooking, to intercept socket function calls and socket data. We choose WINSOCK hooking approach, since it provides information of application which has initiated connection, with interception of socket call. Various techniques such as replacing WINSOCK DLL, Hijacking DLL, writing Layered service provider (LSP) driver etc. are available to implement WINSOCK Hooking. Since other techniques leads to DLL version compatibility issues, Microsoft recommends LSP driver implementation [83]. Therefore, we are using WINSOCK 2 SPI Framework to implement Transport Layered Protocol.

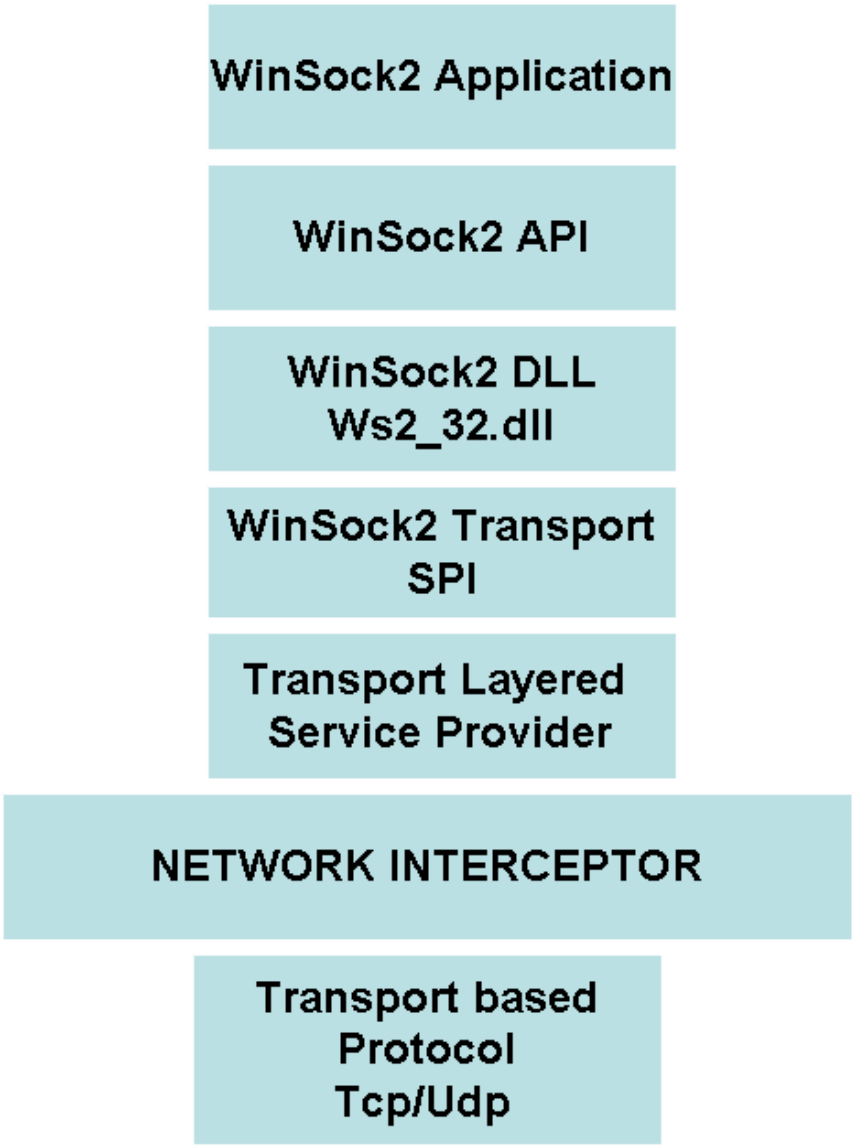


FIGURE 4.2 Layering for NI

We implement NI as a standard DLL. NI is a functional extension to an already existing transport service provider (TSP). We implement NI as a protocol stack of services, in order to provide functions for connection set up, data transfer, to apply flow control, error control, etc. NI is registered with WSCInstallProvider [83], which is an API of SPI framework. In this way, NI can use LSP SPI Architecture of Windows. Once we register this NI, all transport SPI functions implemented by NI are made accessible to ws2_32.dllby callback function mechanism via the LSP's dispatch table. Therefore, NI overrides and intercepts Winsock 2 functions before they are processed by ws2_32.dll. We are mainly interested in functions used by any application to send or receive data. Such

functions are normally send, sendto ,recv, recvfrom, etc. These functions help in intercepting real time data. When any application initiates a connection for the first time, our NI DLL also gets initialized along with it. During the time of initialization, pointers of our callback functions get registered into LSP dispatch table. Therefore, when the application calls any socket functions to initiate a connection or to send/receive data, it first calls our function. Our implemented function obtains the access of socket handle and data which is transmitted. It sends this data to IDPS for inspection and waits for its result. The result is communicated to pattern matching unit via event logger.

4.1.2 Event Logger:

The Event Logger [72] stores the alert log in alert store database. On a high-performance network, the inflow and outflow of events is very high. The speed at which it stores the alert log generally does not match with the flow of incoming and outgoing network packets. In order to store each alert log and not miss any of them, we have implemented an asynchronous architecture. We use the concept of classical producer-consumer scenario. Event Logger is divided into the three functional sub-systems: Event Receiver, Queue and Database Plug-in [72]. Figure 4.3 shows the working of Event Logger:

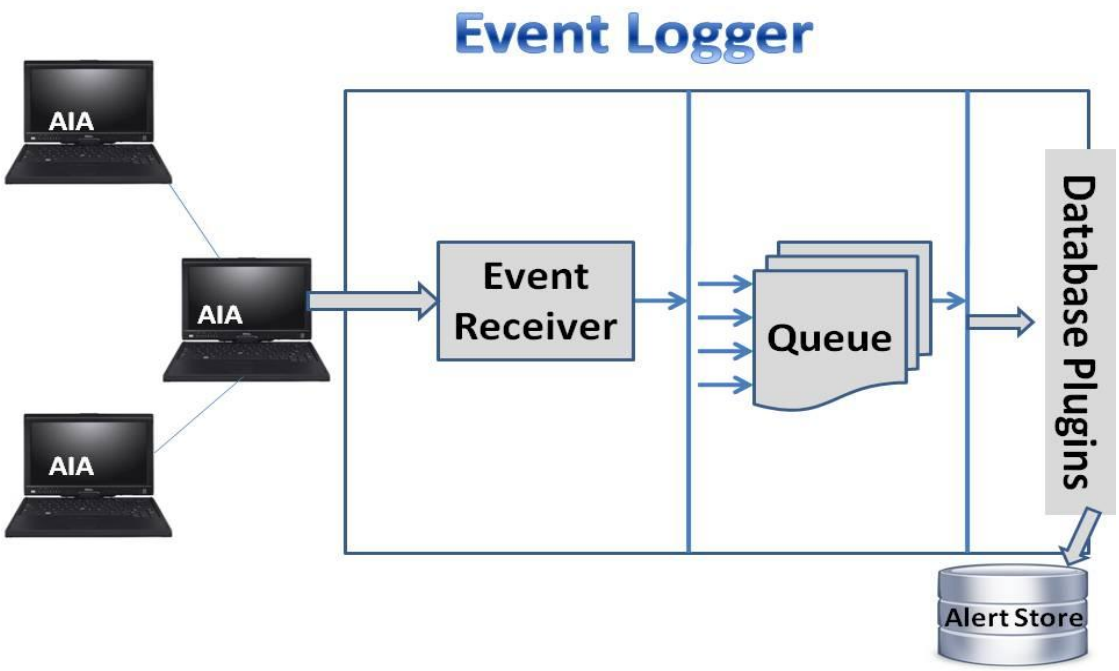


FIGURE 4.3 Event Logger

AIA uses UDP protocol to send event log to the Event Logger. Event Receiver receives this log and inserts it in the queue. The database plug-in fetches this log from the queue and stores it in the database.

Figure 4.4 describes the communication flow between PMU and AIA. As shown, a workstation has Process-A running on it. On the same workstation, AIA is deployed. Process-A tries to send a request to a website hosted on the Internet. AIA has a dedicated TCP connection with PMU. Before the packet reaches IDS engine deployed in our network, AIA intercepts the system call at socket level for this connection request. From the socket call intercepted, it obtains all the information about application as well as connection. AIA now sends this information to PMU. PMU verifies this against pertinent application data store and sends the decision of allow/block and AIA allows or drops the connection.

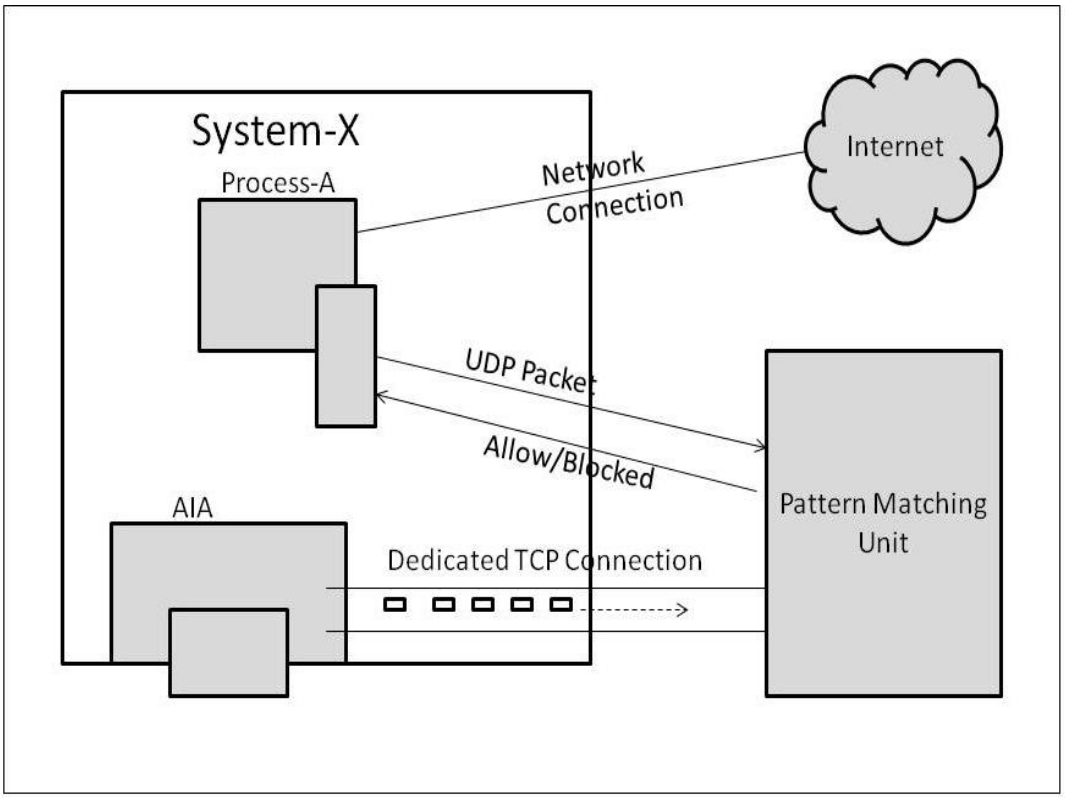


FIGURE 4.4 Communication flow for AIA

4.2Pattern Matching Unit (PMU):

As shown in figure 4.5 and 4.6, PMU has two different deployment modes: inline and out of band. A UDP packet listener will listen on a dedicated UDP port, to gather application and connection information sent by AIA instances. Once a UDP packet is received, listener will store application information and five tuple hash {Source IP, Destination IP, Source Port, Destination Port, Protocol} in a global memory store. Global application store is implemented as a two-dimensional linked list, where first dimension represents a unique system and each system node points to another linked list representing application and connection information for that system.

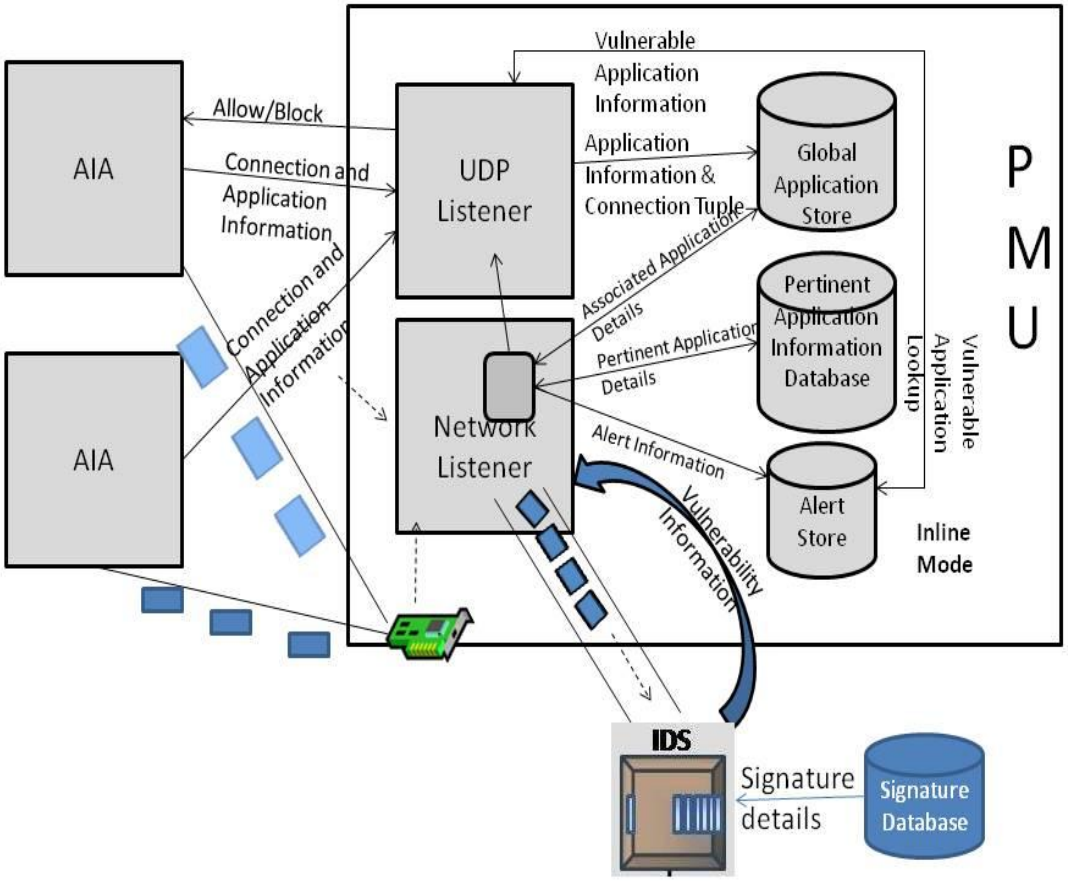


FIGURE 4.5 PMU working in inline mode

As stated earlier, when PMU is deployed in out of band mode, it starts TCP server listening on a dedicated port. AIA running on each system initiates a dedicated TCP connection

from each one of them. All the packets transferred on that system is sent to PMU using this dedicated TCP connection. So, TCP listener will have multiple incoming packet streams. Each packet stream is uniquely identified and differentiated by system identifier. PMU puts a wrapper on every packet to associate it with respective system. It also accumulates all these packets in a common packet queue.

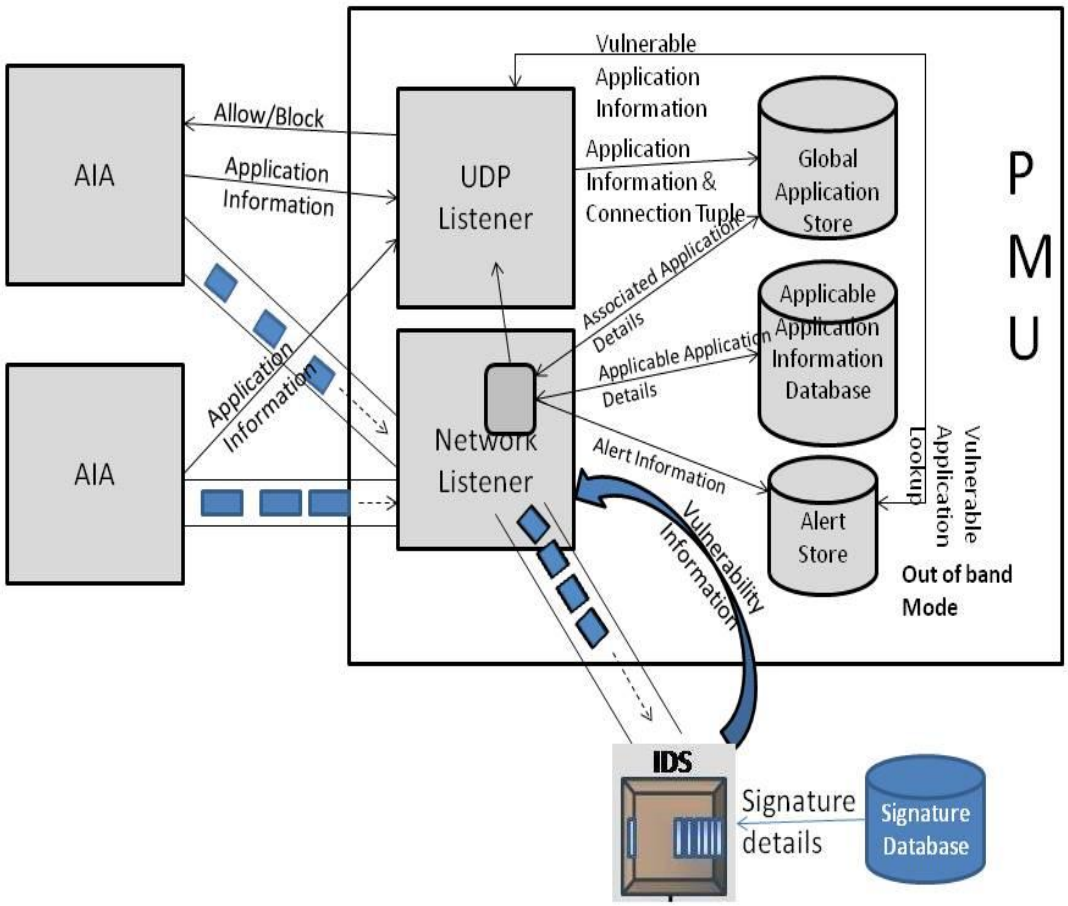


FIGURE 4.6 PMU working in out of band mode

Another process running under TCP listener keeps a watch on this queue. As soon as packet arrives, it sends that packet to IDS engine using standard available interface and API provided by IDS engine. IDS engine inspects the packet for known vulnerabilities using standard signature rule files. If any vulnerability is found, it responds back to the calling application with signature rule id information.

PMU has another data store which provides pertinence of signature to different application or processes. PMU looks up the triggered signature rule id in this data store to fetch the information about pertinent application.

From a packet wrapper (Hashvalue of combination of MAC ID & Source IP Address), it retrieves system identifier to lookup second level linked list representing that particular system's application and associated connection information.

From a packet, it fetches five-tuple information to lookup into retrieved second-level linked list. This lookup provides associated application information. PME now matches associated application information and pertinent application information. If associated application information belongs to this list, then it concludes that application is either compromised or attacked. In that case, PMU stores this information into an Alert store database.

If it doesn't match, it is considered as a false alarm. Administrator takes a decision on every entry in alert database; mark it as a quarantine decision.

4.3Aggregation and Correlation Module (ACM):

Host based intrusion detection and prevention systems (IDPS) reacts to different events generated by operating systems running on machines which host these systems. Operating system generates logs of seemingly malicious events for example, when CPU utilization goes high, an application tries to write in system file area or a service is not responding etc. HIDPS generates an alert when such event is reported. An alert generated by HIDPS contains application level information such as hostname, service name, event timestamp, and signature id and signature name [20]. It is because these systems work only at application level and have visibility of only application headers. Network based IDPS, on the contrary, works on network packet headers. In an NIDPS, an IPS sensor is placed at network ingress point, or some other place where one can monitor the packets flowing through entire network. The IDPS sensor monitors network traffic and inspects packet transmissions for malicious behaviour. It generates alert when it observes such malicious behaviour. The alerts generated by NIDPS or perimeter based IDPS contains network level information such as source IP, destination IP, event timestamp, signature id, signature name and protocol details [20]. It is because these systems work at network layer and have visibility of only network protocols and network header details. Such IPS sensors can be placed at various locations along the boundary or periphery of the network. It is then known as Perimeter based IDPS [21]. Perimeter based IDPS is typically deployed at the

entry points of different subnets. The purpose of such deployment is to protect every subnet from generating threats within the network and across the network but within the organization.

This information contained in an alert is crucial for the network administrator to identify the attacker and to take correct preventive decisions for that attack, which includes applying operating system fixes, applying application fixes, blocking of unused network services and so on. However, a single alert may not indicate complex attack scenarios such as DDOS, application targeted attacks followed by port scan, botnets etc.[20] In addition, the number of network packets and log entries that are inspected periodically has a direct impact on number of false alerts generated by an IDS [1].

An alert is an indication given by an IDPS for possibility of an attack. This alert may singularly describe an attack in entirety. However, certain attacks may be distributed across multiple alerts. In our approach, we have considered both the cases. Thus, we aggregate alerts only when it is required.

There are certain alerts, which alone are adequate to provide attack information. An example of such alert is when an insider is aware about certain system vulnerability such as buffer overflow and tries to attack. Our proposed system generates an alert in this case in its entirety. However, for the same vulnerability, it is also possible that an outsider performs port scan followed by application probing to learn about this vulnerability and similar alert is generated. The major difference here is the first case describes the entire attack while second one does not. In the second case, attack instances for port scan, application probing and buffer overflow need to be correlated for complete attack description. The strength of our approach is that it can correlate such events to exactly pinpoint the attack.

Addressing IDPS issues by Correlating Raw alert log: Alerts generated by IDPS provide very basic information of attacks, from which it is very difficult to retrieve bigger picture. In order to address complex attack scenarios and generate aggregated alerts related to them, specific issues such as alert flooding, alert context, false alerts and scalability should be addressed. These issues are described as:

Alert Flooding: Alert flooding is caused by multiple alerts generated by IDPS for the same attack instance. For example, when an attacker launches a large ICMP echo attack, it

would be normally continuous attack on a destined system. Alert will be generated for every large ICMP echo packet. The fact that human security expert then studies these alerts intensifies the probability of wrong or inappropriate decision taken. We handle it by collapsing multiple similar alerts into one.

Alert context: Context means alerts which are manifested as part of a larger attack sequence**.** It means relating similar attack instances. The existing detection systems are usually capable of detecting various types of attack, but in between the occurrences of attacks, lay various instances of similar attacks, which the detection systems are not able to differentiate from one another [62]. Our approach handles that as well.

False alerts: False positive means alert generated by an IDPS to protect or report system vulnerability which does not exist. Network administrator performs routine maintenance activities by port scan or similar probes. IDPS misclassifies this activity as an alert. Such alerts should be filtered from alert log.

Scalability: This issue bothers an IDPS when large number of alerts is generated as network grows and consequently number of probes increases. With proper aggregation, this problem is also handled to an extent by our approach.

Our approach tries to address these issues. We focus primarily on generating only one alert per attack, even if the attack manifested itself into multiple alerts. Such temporal correlation of alerts will reduce the log size considerably. For example, in case of buffer overflow attack, we can group similar alerts on the basis of source IP, destination IP and time interval. For the same example, if external user had tried different attacks for which system is not vulnerable, and IDPS has generated alerts for the same, and then correlation will help to mask them as false alarms. The task of alert information retrieval is handled locally and is sent to a centralized module for further processing. Thus, we successfully manage the trade-off between information gathering and network performance to a large extent.

ACM module covers three aspects of any attack:

1) Attack information – Attack information is basic raw alert log data being sent by any standard IDS. It includes information about attack, severity of attack, network details and signature rule ID which has detected the attack.

2) Application participation in attack –we identify details of attacking and victim applications details like application name, application version, etc.

3) Pertinent application to specific attack – An attack is generally targeted towards some particular application vulnerability. E.g. Attacks for different application servers like Tomcat and Apache will be different even though application protocols are HTTP. Our solution uses signature rule set database with indication of application pertinent to attack.

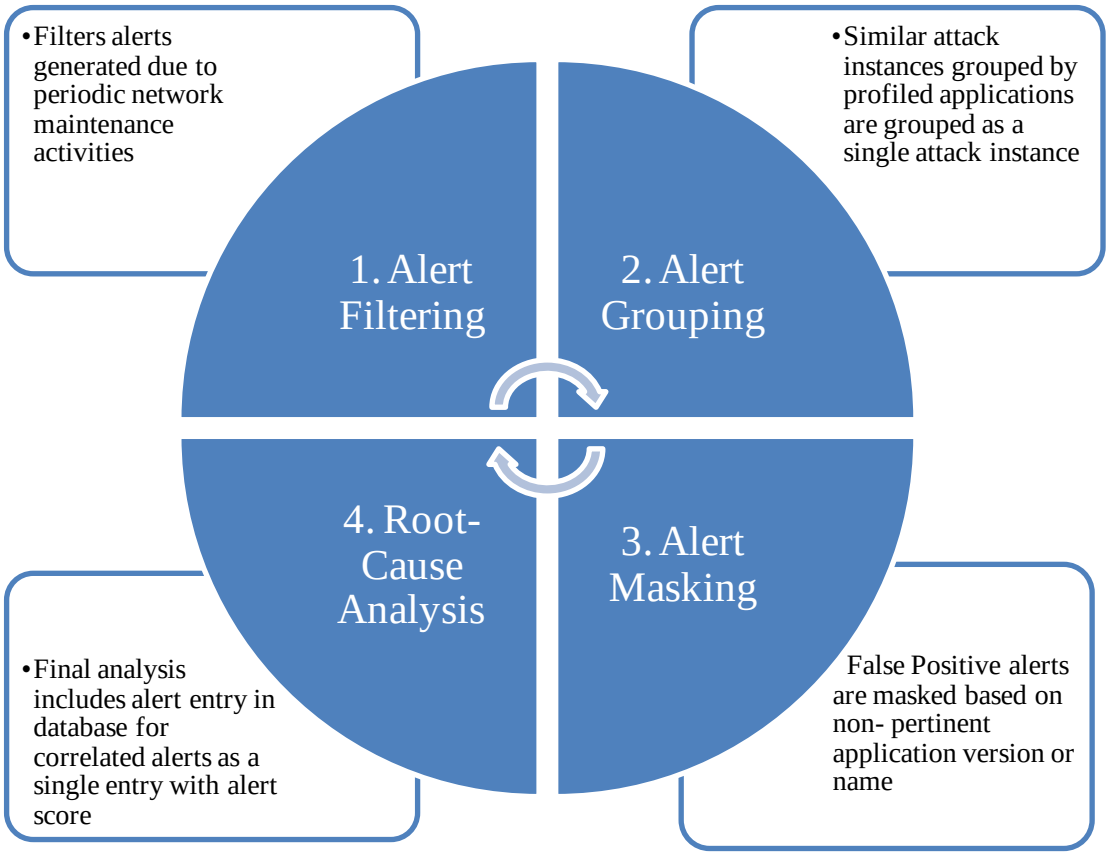


FIGURE 4.7 Aggregation and Correlation Module

Figure 4.7 describes our alert aggregation and correlation module (ACM) which is a four stage process: Alert Filtering, Alert Aggregation, Alert masking and root-cause analysis. This module works on Alert Store generated by PMU. This store contains raw alerts with pertinent application information with alert details. This application information is primarily used by our ACM module to correlate alerts into a single alert scenario with precise information about the application name and version responsible for the attack.

Alert Filtering:

There are times when system goes into maintenance mode and requires debugging and trouble shooting. When alert is generated due to activities performed for such trouble shooting applications, it should be filtered because they are irrelevant for attack identification. Administrator has a predefined list of such applications in our system by providing application name and application versions. Any alert generated by this application will be marked for filtering by our system.

Alert Aggregation:

Alert aggregation is merging of events, which are part of common attack, or duplicates in nature. When sequence of attack is generated by same application running on one host or different hosts, it should be aggregated as they are forming a common attack. E.g. If one botnet application is spread across the network and multiple hosts are trying to generate DOS attack to one particular server on internet or multiple servers on internet as a whole, we will aggregate them by application name, application version, variety of destination IPs and attack and this reconstruction will give us attack information of Distributed DOS.

Alert Masking:

When an attacker launches an attack on the nonexistent system or vulnerability, it normally generates false alarms. E.g. If attacker probes that port 80 is open and launches an attack for IIS which is not running on that system. It will generate a false alert by IDPS. When any alert is being generated by our IDPS, it contains information of application which participated in that attack along with applicability of that attack. We compare both the information and mark the false positives being generated.

Root-cause analysis:

This module decides confidence level of each log entry in alert store database to perform correlation of individual alert incidents into meaningful alert scenario. It scans each database entry, which has an initial confidence level set to 1. If an entry has been marked as false positive, this module decreases the confidence level for that entry to 0. Any subsequent entry for same application name, version or same host will result into increase in its confidence level. All such entries will be correlated to form a single threat scenario,

with threat level defined. The final outcome of this step is Incident Report Database with security threat level defined as low, medium, high, critical & fatal.

4.4Application Quarantine Module (AQM):

Application Quarantine Module uses incident report database for preventive advices. For example, root cause analysis concludes that Mozilla running on machine 20.0.1.15 is being compromised. Administrator marks this application on that host as a quarantine application. Thus, when Application intercepting agent (AIA) asks for permission of network admission or access for mozilla, application quarantine module will look up the alert store to check if mozilla running on 20.0.1.15 is marked as quarantine or not. If match is found then it will reply with BLOCK status and AIA will block the network access for mozilla. So it doesn't block the access of host machine but only blocks vulnerable application for network access. This improves the network security without compromising network productivity.

4.5Complete working of all modules:

We now discuss stepwise working of all the modules. We also discuss how they interact with each other in order to achieve our objectives.

Application intercepting agent (AIA) intercepts network traffic being sent or received by any application on the network under consideration. When any application tries to communicate, it opens a socket to send or receive the data. AIA intercepts this traffic and collect details of that application. This application information is passed to pattern matching unit (PMU) along with packet payload of connection.

PMU sends the received network data to intrusion detection engine to scan and detect the attack. Intrusion detection engine can be any general IDPS such as Snort or Suricata [84] [85]. It receives the network information from PMU and scans that traffic for attack using its rule set database. If any attack is found that it generates the alert log and sends it back to PMU. This alert log contains id of signature which matched against the network packet data. Once PMU receives the alert information from intrusion detection engine, it performs lookup on Pertinent Application Information Store database for the signature id of alert. If

a match is found, it verifies the pertinent application for the matched signature id with application associated with the alert.

The application which initiated this malicious connection is called an associated application. The global application store plays an important role of verifying the pertinent application as well as associated application for the matched signature. This is implemented as a two level linked list, snapshot of which is shown in the next section. We have chosen Linked list as a data structure because no of hosts and no of applications running on them can grow or shrink dynamically. In the first level linked list, each node stores a hash value to identify unique host IP Address and host name. Each node of the first linked list points to a set of nodes of second level linked list. Each node of the second level linked list stores connection as well as application information about each connection initiated from that host. PMU performs a lookup in this store. If a match is found, it inserts alert details along with pertinent application information indicated by rule set database in another database called alert store.

Alert aggregation and Correlation module (ACM) works on this alert store database which contains application information and whether the application is pertinent to the alert as described. We have described this module in detail in following sections. ACM generates the final aggregated log which is useful to administrator to take preventive actions and also useful for application quarantine. This unit reduces raw alerts log size by more than 95% and also removes false positives.

Application quarantine module (AQM) works on aggregated alert log. When AQM receives the permission request from an application for network admission or access, it checks for quarantined application information in final aggregated incident report database. If any application is already marked for quarantine then AQM blocks that application's connection process.

4.6Workflow Algorithm:

We have discussed each module in detail in previous section. We have discussed how each module communicates with each other at different stages. We now present a complete algorithm of our workflow:

Step 1: AIA intercepts connection request by an application

Step 2: AIA sends connection and application details to PMU along with packet data transferred over that connection.

Step 3: PMU sends connection packet data to IDS engine.

Step 4: If IDS engine scans the data and if vulnerability or attack is found then sends an alert back to PMU. PMU compares associated application of that connection with pertinent application stored in signature database.

Step 5: If successful match, alert details are stored in alert store with confidence level 1.

Step 6: ACM parses each entry in raw alert store and performs the following actions:

a) Filter alerts generated for network maintenance and trouble-shooting

b) Mask false positives. In our approach, alerts generated for non-pertinent applications are false positives.

c) Group alert incidents for profiled applications as a single alert scenario.

d) Root-cause analysis: This step decides confidence level of each log entry to perform correlation of individual alert incidents into meaningful scenario.

d.1: If entry is marked as false positive, decrease confidence level to 0.

d.2: If otherwise, we mark it for further tracking.

d.3: Whenever next incidence occurs for same application or same host, we increase confidence level and mark it as security threat with threat level.

d.4: All subsequent log entries from that host for the particular application will be correlated to this incident to form a single threat scenario.

d.5: The final outcome of this step is Incident Report Database, used by AQM with security threat level defined as low, medium, high, critical & fatal.

CHAPTER – V

Experimental Setup & Results

5.1Snort vs. Suricata

For proof of concept, we are using Suricata IDPS engine [86]. Our PMU, which is deployed centrally, sends application and connection details to Suricata for detection. We now discuss why we are using Suricata over Snort.

Snort is single-threaded. Suricata is multi-threaded. In case of Snort, whenever CPU is overloaded during peak traffic hours, it will start dropping the packets. In order to inspect more traffic using Snort with a single CPU, one needs to run multiple Snort instances. In addition to this, when Snort runs with multiple instances, each instance needs to normalize the traffic. However, Suricata, designed to work with multiple CPU's, will use all of the CPU's on the sensor. It performs load-balancing of the traffic across all of the CPU's, so there is little tuning needed in this regard [86]. Suricata improves performance by normalizing the traffic once (depending on configuration), then pushing the normalized traffic to worker threads which perform the payload inspection.

For performance evaluation and comparison, we have used Snort and Fortinet IDPS. The purpose of choosing Snort is because it is the most popular and widely used open source IDPS available in market today. Fortinet is again a very popular IDPS in the commercial category. For comparison with CISCO and Juniper, we have studied their patents and logged documents available.

5.2Network and Operating system specifications

We have configured a network with 4 client workstations and two server nodes. Application Intercepting Agent (AIA) is installed on each client node. Pattern matching unit (PMU) is installed on one of the servers and Suricata is installed on a separate server.

We perform tests with PMU in inline as well as out of band mode. We have used Postgre SQL database for data stores. The hardware configuration for each is given below:

Internet speed: 40 Mbps

Network (Ethernet) speed: 1 Gbps (Giga bits per second)

Client Node configuration

Make	Dell
Processor	Intel core i3 (4^{th} generation)
Processor frequency	3 GHz
Hard disk	1 TB
Reading Speed	5400 rpm (Revolutions per minute)
Storage Interface	SATA (**Serial AT Attachment)**
RAM	4 GB DDR3 SDRAM (**double data rate type three synchronous dynamic random-access memory)**
Operating System	Windows 7
Cache memory	3 MB

Server configuration

Make	Dell
Processor	Intel® Xeon® E5-2403
Processor frequency	2.20 GHz
Hard disk	1 TB
Reading Speed	7200 rpm (Revolutions per minute)
Storage Technology	RAID (redundant array of independent disks**)**
Storage Interface	SATA
RAM	4 GB UDIMM (Unregistered dual in-line memory module)
Operating System	Windows Server 2012
Cache memory	15 MB

Dataset generation

We have considered port scan and mail username/password vulnerabilities for test data. The following database log is both for port scan and mail username/password vulnerability. It contains raw alerts generated and stored in out alert log database with profiled application information.

We have tested our proposed architecture on simulated attack scenarios, which forms the basis of our experimentation in terms of raw alert log. This raw alert log, stored in alert

store database, is used in a live network, with our units deployed locally as well as centrally. Application intercepting agent is deployed locally on each host in the network and pattern matching unit is deployed centrally on a server.

We have simulated archetypal network user's day to day activity. We have simulated connection events of normal Internet browsing by emulating different browser applications and mail sending events by emulating mail application, routine system administrator's activity to generate Port scans for checking network health and different attacks typically generated on or by these applications. For proof of concept, we have simulated approx. 50000 raw alerts using syslog generation utility. Syslog generation utility simulates the syslog protocol and generates alerts similar to that of Snort and Suricata. This utility sends this alert log to Pattern matching unit to store these alerts with pertinent application information.

5.3Snapshot of data stores

5.4.1 Pertinent Application Information Database

For storing data, we are using Postgre SQL database. We have implemented an interface where administrator defines relevance of attack signatures to different applications. In order to identify application relevance to an attack, we have implemented a parser, which reads signature file of IDPS and displays a table of signatures to the administrator. The administrator specifies the corresponding application for each signature. The signature and its application are stored into Pertinent Application Information store. ACM module uses this information to find out false positives generated by intrusion detection engine and marks them. Table 5.1 shows few actual entries from our signature database. Against each signature parsed and read from signature file, administrator specifies the pertinent application name and version.

TABLE 5.1 Pertinent Application Information Store

Signature ID	Signature Name	Pertinent Application Name	Pertinent Application Version
2017478	IE Memory Corruption Vulnerability	Internet Explorer	7 to 9
100000447	Mozilla Firefox DOMNodeRemoved attack attempt	Mozilla Firefox	Any
2101809	Apache Chunked-Encoding worm attempt	Apache	1.3.x
2002993	Rapid POP3S Connections - Possible Brute Force Attack	NA	NA
7393	Smtp_auth_failure	Telnet	Any

5.4.2 Alert Store

The pattern matching unit stores the alert log in an alert database. It is implemented as a centralized system. Alert log contain information such as source and destination IP, name of attack, severity, application name and version. This alert information logged is of vital importance. The Source and Destination IP provides to the administrator the network level information of the attacker and the victim. Name and severity of the attack gives information on criticality of the attack. We define levels of severity as Minor, Major, Critical and Fatal. Application name and version gives application related information, which is used to identify root cause of attack in the network or actual vulnerable application, which requires hot fixes. Attack type can be client-side attack or server-side attack. Direction of connection can be inbound or outbound.

Table 5.2 shows table of raw alerts & contain few of the actual logs for explanation purpose:

TABLE 5.2 Raw Alert Store

Alert ID	Alert Type	Alert Name	Timestamp	Application Name	Application Version	Network Information	Direction
1	Connection	Connection	1/12/2015 10:40:45	Internet Explorer	7.x	10.1.1.1:50000 – 20.1.1.1:80	I_C
2	Intrusion	IE Memory Corruption Vulnerability	1/12/2015 10:40:46	Internet Explorer	7.x	10.1.1.1:50000 – 20.1.1.1:80	I_C
3	Connection	Connection	1/12/2015 10:40:46	Internet Explorer	7.x	10.1.1.1:50001 – 20.1.1.1:12345	I_C
4	Intrusion	Port Scan	1/12/2015 10:40:47	Internet Explorer	7.x	10.1.1.1:50002 – 255.255.255.255:25	I_C
5	Connection	Connection	1/12/2015 10:40:48	Internet Explorer	7.x	10.1.1.1:50003 – 10.1.1.2:25	I_C
6	Intrusion	Port Scan	1/12/2015 10:40:49	Nmap	2.x	10.1.1.3:10000 – 255.255.255.255:25	O_S
7	Connection	Connection	1/12/2015 10:40:49	Google Chrome	5.x	10.1.1.4:20000 – 20.1.1.1:80	O_C
8	Intrusion	IE Memory Corruption Vulnerability	1/12/2015 10:40:52	Google Chrome	5.x	10.1.1.4:20000 – 20.1.1.1:80	I_C
9	Connection	Connection	1/12/2015 10:41:05	Outlook	7.x	10.1.1.1:50004 – 10.1.1.2:25	O_C
10	Connection		1/12/2015 1:40:15	Telnet	3	10.202.4.1:5050-195.130.217.97:25	I_C
11	Intrusion	SMTP_AUTH_FAILURE01	1/12/2015 1:40:30	Telnet	3	10.202.4.1:5050-195.130.217.97:25	I_C
12	Connection	SMTP_AUTH_FAILURE01	1/12/2015 1:41:45	Telnet	3	10.202.4.1:5052-195.130.217.97:25	I_C
13	Intrusion	SMTP_AUTH_FAILURE01	1/12/2015 1:42:15	Telnet	3	10.202.4.1:5052-195.130.217.97:25	I_C
14	Connection		1/12/2015 1:44:24	Microsoft Outlook	8	10.202.4.25:4000 - 195.130.217.97:25	I_C
15	Intrusion	SMTP_AUTH_FAILURE01	1/12/2015 1:44:45	Microsoft Outlook	8	10.202.4.25:4000 - 195.130.217.97:25	I_C
16	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:02	Apache	1.2.1	10.202.3.25:4000 - 195.130.217.97:2000	I_S

Alert ID	Alert Type	Alert Name	Timestamp	Application Name	Application Version	Network Information	Direction
17	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:03	Apache	1.2.1	10.201.5.25:3330 - 195.130.217.97:80	I_S
18	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:04	Apache	1.2.1	10.201.4.25:2200 - 195.130.217.97:80	I_S
19	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:04	Apache	1.2.1	10.201.4.25:2100 - 195.130.217.98:4500	I_S
20	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:05	Apache	1.2.1	10.201.4.26:4300 - 195.130.217.98:1233	I_S
21	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:06	Apache	1.2.1	10.201.4.25:4200 - 195.130.217.99:1333	I_S
22	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:07	Apache	1.2.1	10.201.4.23:3322 - 195.130.217.97:25	I_S
23	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:08	Apache	1.2.1	10.201.4.22:2340 - 195.130.217.98:80	I_S
24	Intrusion	SMTP_AUTH_FAILURE01	1/12/2015 2:04:09	Microsoft Outlook	8	10.201.4.24:5400 - 195.130.217.99:25	I_C
25	Connection	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:09	Apache	1.2.1	10.201.4.25:2300 - 195.130.217.97:2334	I_S
26	Connection	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:09	Apache	1.2.1	10.201.4.22:1200 - 195.130.217.98:5455	I_S
27	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:10	Apache	1.4.1	10.201.4.23:2300 - 195.130.217.98:25	I_S
28	Intrusion	SMTP_AUTH_FAILURE01	1/12/2015 2:04:10	Microsoft Outlook	8	10.201.4.25:4133 - 195.130.217.97:1233	I_C
29	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:10	Apache	1.4.2	10.201.4.22:2103 - 195.130.217.97:80	I_S
30	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:11	Apache	1.4.2	10.201.4.25:4330 - 195.130.217.99:25	I_S

Correlating raw alert log by Aggregation & Correlation Module (ACM):

As discussed in the previous chapter, ACM is a four stage process: Alert Filtering, Alert Grouping, Alert masking and root-cause analysis. This section describes these four stages and their outcome. Each stage works on alert store database shown in fig. 6. This database stores raw alert log and application profile for detected alerts.

Alert Filtering:

In the raw alert log shown in Table 5.3, one of the alerts is triggered from host 10.1.1.3. This alert is generated because of port scan activity performed in network. However, this activity is performed by application called NMap with version 2.x. This particular application is in a pre-defined list by specifying it in black list of Applicable Application of administrator hence, alert 6 is filtered.

TABLE 5.3 Filtered Alerts

Alert ID	Alert Type	Alert Name	Timestamp	Application Name	Application Version	Network Information	Direction
6	Intrusion	Port Scan	1/12/2013 10:40:49	Nmap	2.x	10.1.1.3:10000 – 255.255.255.255:25	O_S

Alert Grouping:

In our raw log, Host 10.1.1.1 has visited a vulnerable site running on 20.1.1.1. Exploit residing on this site exploits internet explorer running with version 6 to 9. It creates a backdoor entry using which it sends certain commands. In this particular case, it ask compromised Internet explorer to perform port scan to get a list of mail servers running in network. After that, it asks compromised IE to connect one mail server and send an email. From alert log, we can see that IE running on source 10.1.1.1 has performed all these activities in specific time duration. At the same time, user has also opened a connection with a mail server to send a specific email. But this last activity is not a part of threat scenario. It should be treated out of threat context. When we aggregate these alerts using source IP, destination IP, timestamp, application name and version, we aggregate first four alerts and correlate it to threat scenario. But, the last activity is performed by different application using same source IP, destination IP and timestamp. So it will not participate in

above aggregation and is treated as a normal activity. In the table 5.4, alerts with ID 1, 2,3,4,5 are aggregated as a single attack scenario. Similarly alerts with ID 10,11,12,13,24 are also aggregated.

Similarly, for vulnerability "Apache Chunked-Encoding worm attempt", our pertinent application database has pertinent application specified as Apache with version <=1.3.x. Using this information from the pertinent application database, alerts with ID 16 to 23 and also 25,26 will be merged as a single alert scenario, even though they are generated for different source and destination. Also, they are generated in different time intervals. Thus we precisely correlate separate alerts with common context.

TABLE 5.4 Grouped Alerts

	Alert ID	Alert Type	Alert Name	Timestamp	Application Name	Application Version	Network Information	Direction
Grouped Alerts with same context	1	Connection	Connection	1/12/2013 10:40:45	Internet Explorer	7.x	10.1.1.1:50000 – 20.1.1.1:80	I_C
	2	Intrusion	IE Memory Corruption Vulnerability	1/12/2013 10:40:46	Internet Explorer	7.x	10.1.1.1:50000 – 20.1.1.1:80	I_C
	3	Connection	Connection	1/12/2013 10:40:46	Internet Explorer	7.x	10.1.1.1:50001 – 20.1.1.1:12345	I_C
	4	Intrusion	Port Scan	1/12/2013 10:40:47	Internet Explorer	7.x	10.1.1.1:50002 – 255.255.255.255:25	I_C
	5	Connection	Connection	1/12/2013 10:40:48	Internet Explorer	7.x	10.1.1.1:50003 – 10.1.1.2:25	I_C
Grouped Alerts with same context	10	Connection		1/12/2013 1:40:15	Telnet	3	10.202.4.1:5050- 195.130.217.97:25	I_C
	11	Intrusion	SMTP_AUTH_FAILURE01	1/12/2013 1:40:30	Telnet	3	10.202.4.1:5050- 195.130.217.97:25	I_C
	12	Connection	SMTP_AUTH_FAILURE01	1/12/2013 1:41:45	Telnet	3	10.202.4.1:5052- 195.130.217.97:25	I_C
	13	Intrusion	SMTP_AUTH_FAILURE01	1/12/2013 1:42:15	Telnet	3	10.202.4.1:5052- 195.130.217.97:25	I_C
	24	Intrusion	SMTP_AUTH_FAILURE01	1/12/2015 2:04:09	Microsoft Outlook	8	10.201.4.24:5400 - 195.130.217.99:25	I_C

Grouped Alerts with same context

Alert ID	Alert Type	Alert Name	Timestamp	Application Name	Application Version	Network Information	Direction
16	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:02	Apache	1.2.1	10.202.3.25:4000 - 195.130.217.97:2000	I_S
17	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:03	Apache	1.2.1	10.201.5.25:3330 - 195.130.217.97:80	I_S
18	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:04	Apache	1.2.1	10.201.4.25:2200 - 195.130.217.97:80	I_S
19	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:04	Apache	1.2.1	10.201.4.25:2100 - 195.130.217.98:4500	I_S
20	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:05	Apache	1.2.1	10.201.4.26:4300 - 195.130.217.98:1233	I_S
21	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:06	Apache	1.2.1	10.201.4.25:4200 - 195.130.217.99:1333	I_S
22	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:07	Apache	1.2.1	10.201.4.23:3322 - 195.130.217.97:25	I_S
23	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:08	Apache	1.2.1	10.201.4.22:2340 - 195.130.217.98:80	I_S
25	Connection	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:09	Apache	1.2.1	10.201.4.25:2300 - 195.130.217.97:2334	I_S
26	Connection	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:09	Apache	1.2.1	10.201.4.22:1200 - 195.130.217.98:80	I_S

Alert Masking:

Host 10.1.1.4 has visited the same vulnerable site running on 20.1.1.1. However, application used to visit this site is Google chrome with version 5.x. same alert is generated by IDS system. This vulnerability is applicable to IE with version 6 to 9. Therefore, applicability of signature is not harmonized. This alert is masked as false positive alarm. User working on Host 10.202.4.25 has accidently put wrong username and password combination in his Outlook mail client. Even though it is identical SMTP_AUTH_FAILURE01 alert, pertinent application is not identical. Consequently it is considered as a false positive alert and has been masked. So in entirety, 7,8,9,14 & 15

event IDs are masked. Similarly alert ID 29,30 also have same alert “Apache Chunked-Encoding worm attempt”. However the pertinent application version is <=1.3.x. Whereas the associated application here has version 1.4.x. Hence, these two entries will also be masked. Table 5.5 shows masked alerts.

TABLE 5.5 Masked Alerts

Alert ID	Alert Type	Alert Name	Timestamp	Application Name	Application Version	Network Information	Direction
7	Connection	Connection	1/12/2013 10:40:49	Google Chrome	5.x	10.1.1.4:20000 – 20.1.1.1:80	O_C
8	Intrusion	IE Memory Corruption Vulnerability	1/12/2013 10:40:52	Google Chrome	5.x	10.1.1.4:20000 – 20.1.1.1:80	I_C
9	Connection	Connection	1/12/2013 10:41:05	Outlook	7.x	10.1.1.1:50004 – 10.1.1.2:25	O_C
14	Connection		1/12/2013 1:44:24	Microsoft Outlook	8	10.202.4.25:4000- 195.130.217.97:25	I_C
15	Intrusion	SMTP_AUTH_FAILURE01	1/12/2013 1:44:45	Microsoft Outlook	8	10.202.4.25:4000- 195.130.217.97:25	I_C
29	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:10	Apache	1.4.2	10.201.4.22:2103- 195.130.217.97:80	I_S
30	Intrusion	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:11	Apache	1.4.2	10.201.4.25:4330- 195.130.217.99:25	I_S

Root-cause analysis:

Our final correlated log contains precise and granular information about an attack. Correlation action field describes whether single raw alert itself was sufficient for attack detection or the attack was a coordinated sequence of events. The field contains values ‘single’ or ‘merged’ as per the analysis performed. Alert name is the ‘msg’ field or name of the attack stored in the pertinent application store database. When PMU matches “sid” from this database for an alert, it stores name of alert value for the matched ‘sid’ in alert store database. We successfully match multiple alerts with different ‘sid’ to form a single alert scenario. From the table, we can see that ‘IE Memory Corruption Vulnerability’ and ‘Port Scan’ are actually an attempt by vulnerable IE to form a correlated attack sequence. For a merged entry, firstoccurrence_lastoccurrence field is computed from the timestamp

field in alert store. It tells the administrator, the first and last time instance in case of correlated attack scenario. Pertinent application is helpful for taking quarantine actions. If almost 80% of applications running on a host is malfunctioning, in that case, the host itself is quarantined. Source field provides the host IP responsible for such attacks. ContactedDestinations field provides the targeted host's IP addresses. Correlated ID and CorrelatedBy field describes the alert ID of merged alerts and which of the four stages were responsible for merging the alerts.

As already discussed, we categorize direction of attack as inbound or outbound client-side or server-side. We store these values as I_C, O_C, I_S, O_S. This information is critical for administrator to determine whether the attack was from within the periphery of the network or from outside the network boundary. It also tells us whether a client or server was found to be vulnerable.

After Alert filtering, grouping and masking, in the above raw alert log new Coorelated alert database looks like as follows. Alert IDs 1,2,3,4 & 5 are being merged in to new aggregated alert and alert 10, 11, 12 & 13 are also merged. Alert ID 6 is deleted by alert filtering and alert ID 7, 8, 9, 14, 15, 16-23, 25, 26 is deleted by Alert masking as discussed in respective sections. Table 5.6 shows our final log after aggregation and correlation.

TABLE 5.6 Final Aggregated & Correlated Log

Correlation Action	Alert Names	First Occurance - Last Occurance	Aggregation Time interval (sec)	Pertinent Application	Source	Contacted Destinations	Correlated Alert ID	Correlated By	Direction
Merged	IE Memory Corruption Vulnerability, Port Scan	1/12/2015 10:40:47 - 1/12/2013 10:40:48	300	Internet Explorer	10.1.1.1	20.1.1.1:80, 20.1.1.1:80, 20.1.1.1:12345,255.255.255.255:25,10.1.1.2:25	1,2,3,4,5	Alert Grouping	I_C
Merged	SMTP_AUTH_FAILURE 01	1/12/2015 1:40:15 - 1/12/2015 2:04:10	300	Telnet	10.202.4.1	195.130.217.97:25	10,11,12,13	Alert Grouping	I_C
Merged	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:02 - 1/12/2015 2:04:09	215	Apache	10.201.4.22	10.201.4.22, 10.201.4.23, 10.201.4.25	16,17,18,19,20,21,22,23,25,26	Alert Grouping	I_S

From above aggregated logs, administrator can now concentrate only on relevant security incidents. Administrator can better investigate applications and patches applied on machine 10.1.1.1. IE running on that machine seems compromised by Memory corruption vulnerability. Another security incident is more about user than a machine. User working on 10.202.4.1 is doing suspicious work and trying to guess username or password of co-workers, which can be even a CEO of the company. Hence, it should be treated with exigency.

5.4.3 Incident Report database

Root-cause analysis module also creates two tables during the process. These two tables define alert score metrics host-wise and application –wise. Application Quarantine Module

(AQM) uses these tables and the final correlated log for taking preventive decisions. Alert score measure is defined as follows:

Alert Score: 0-10 Confidence Level: Low
10-30Confidence Level: Medium
30-50 Confidence Level: High
50-70 Confidence Level: Critical
>70 Confidence Level: Fatal

Cumulative Score for each correlated entry helps admin take preventive measures as:

3) Host-wise alert score shows the following statistics:
   a. Total alert incidents
   b. Total applications participating for alerts

When the alert score for a host is high, critical or fatal, admin can block the host itself.

4) Application-wise alert score shows the following statistics:
   a. Total alert incidents for a given application
   b. Total hosts participating

When the alert score is high, critical or fatal, admin can quarantine the application and block further connection requests from the same. Table 5.7 & 5.8 shows host-wise and application-wise alert score and confidence level.

TABLE 5.7 Host-wise Alert Score

Host	Incidents Reported	Applications Participated	Confidence Level
192.168.3.1	30	5	Medium
192.168.3.2	20	10	Medium
192.168.3.15	1000	1	Fatal
10.0.1.32	2	1	Low
10.0.1.27	450	15	Fatal
20.0.2.33	50	3	Critical
10.0.1.35	80	2	Fatal
192.168.1.65	65	10	Critical
192.168.3.45	2	2	Low
20.0.1.4	55	3	Critical

TABLE 5.8 Application-wise Alert Score

Application Name	Incidents Reported	Hosts Participated	Confidence Level
I.E 7.x	160	1	Fatal
Chrome 8.x	10	1	Low
Mozilla 4.x	15	3	Medium
Apache 1.2.1	45	1	High
Telnet	75	1	Critical

5.5 Experiment Results

Application quarantine module (AQM):

This module is used to block applications and hosts which are found vulnerable by ACM module. For blocking hosts, we use VLAN Steering method.

All existing IDPS including Snort, use static policy enforcement methods such as ARP management, 802.1x etc. All these methods for policy enforcement prevent intrusions by blocking further flow of traffic from one to another network (i.e. LAN to WAN or LAN to DMZ).

The traffic can still flow within the network and infect other machines in the same LAN or WAN. Such vulnerable or infectious hosts gain unwanted access to network resources.

VLAN Steering

VLAN Steering deals with enforcement of policy to implement dynamic access control by which such hosts can be isolated from LAN and cannot affect the service and availability of resources. Therefore, we have chosen VLAN Steering in our system.

IPS sensor in our system performs VLAN Steering for policy enforcement to implement dynamic access control by which we isolate vulnerable or infected hosts from LAN. IPS sensor assigns a quarantine VLAN or production VLAN to access switch ports. Managed Access switches have their profiles pre-defined in the IPS Sensor. They are also called distribution switches. The profiles enable the Sensor to use protocols such as SNMP,

Telnet, or SSH to send commands to the switches for blocking or stopping a port, switching VLANs etc. Figure 5.1 describes how port mirroring is used to perform VLAN steering:

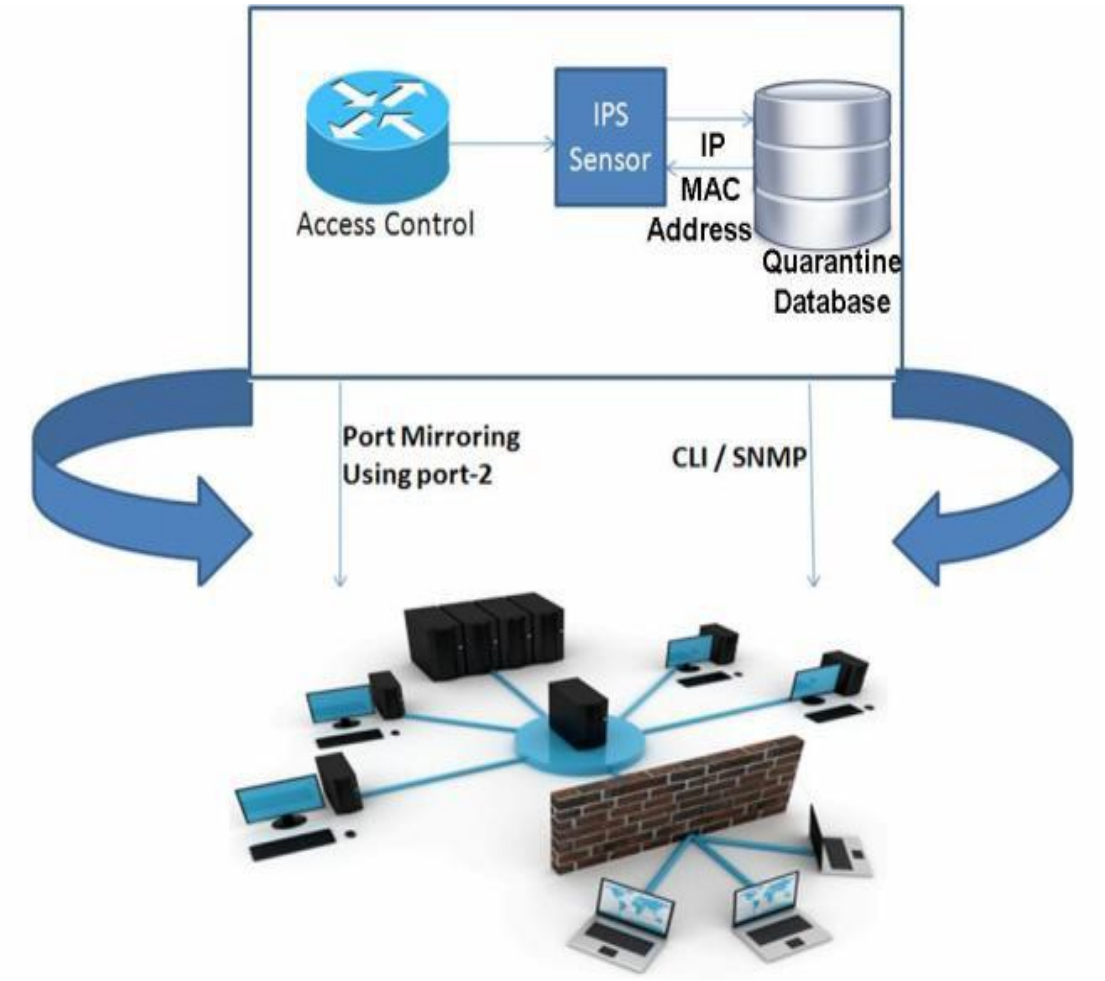


FIGURE 5.1 VLAN Steering using Port Mirroring

As we have mentioned in root cause analysis, Internet explorer running on machine 10.1.1.1 is being compromised. Administrator marks this application on that host as a quarantine application. Thus, when Application intercepting agent asks for permission of network admission or access for Internet explorer, application quarantine module (AQM) will look up the alert store to check if IE running on 10.1.1.1 is marked as quarantine or not. If match is found then it will reply with BLOCK status and Application intercepting unit will block the network access for IE. So it doesn't block the access of host machine but only blocks vulnerable application for network access. This improves the network security without compromising network productivity.

A list of quarantine applications is also stored in a separate table. For the above given scenario, after all the four steps of aggregation and correlation, the quarantine table, shown in Table 5.9 will contain the following details:

TABLE 5.9 Quarantine Details

Host IP Address	Application Name	Application Version
10.1.1.1	Internet Explorer	<= 7.0
10.202.4.1	Telnet	<=3
10.201.4.22	Apache	<=1.3.x

Then above table contains three fields: host IP address, application name and version. This table is read by AQM, and it can take precise preventive decision based on the analysis provided. It says that on two hosts with given IP addresses, IE and telnet are vulnerable with version specified. AQM quarantines the application and if necessary, applies necessary patches or hot fixes. The applications are monitored in the quarantine area. When the application vulnerabilities are attended to, they are again given access to network resources. In this way, the hosts on which applications were infected continue to function normally. Thus, network productivity is maintained at all times.

5.6 Performance Evaluation:

For evaluating the performance of our proposed model and comparing it with other IDPS, the very first requirement is simulation of attack scenarios. These attack scenarios should be crafted so as to target known vulnerabilities. We first discuss how we have simulated attack scenarios taking an example. We have simulated real attack scenario where internal user is trying to guess a mail username and/or password of another internal user.

Usually mail server uses authentication methods and responds with a typical error code when authentication fails. We have created IDPS Signature to track SMTP AUTH ERROR

reply code. We applied this signature in Fortigate to generate alert when ever such incident occurs. The following is a snapshot of this scenario simulation:

```
AHDLAP03762:~ test$ telnet 195.130.217.97 25
Trying 195.130.217.97...
Connected to service152.mimecast.com.
Escape character is '^]'.
220 service152.mimecast.com ESMTP ; Fri, 26 Dec 2014 08:27:32 +0000
EHLO mydpc.com
250-service152.mimecast.com Hello [103.250.31.254 (103.250.31.254)]
250-AUTH LOGIN
250-AUTHmLOGIN
250-STARTTLS
250 HELP
AUTH LOGIN
334 VXNlcm5hbWU6
test123
334 UGFzc3dvcmQ6
test123
535 Incorrect authentication data - http://kb.mimecast.com/Mimecast_Knowledge_B
se/Administration_Console/Monitoring/Mimecast_SMTP_Error_Codes#535
Connection closed by foreign host.
AHDLAP03762:~ test$
```

For this simulated attack, Fortigate gives the following response:

{date=2014-12-25 time=23:57:39 logid=0419016384 type=utm subtype=ips eventtype=signature level=alert vd="root" severity=critical srcip=192.168.99.21 dstip=195.130.217.97 sessionid=35028 action=detected proto=6 service=SMTP attack="SMTP_AUTH_FAILURE01" srcport=55185 dstport=25 direction=0 attackid=7393 profile="test" ref="http://www.fortinet.com/ids/VID7393" incidentserialno=1250547066 msg="custom: SMTP_AUTH_FAILURE01," crscore=50 crlevel=critical}

Analyzing this situation, we now consider how effective this response is. From the given snapshot, there can be two possibilities:

- It is a legitimate user who has genuinely forgotten his username/password
- Someone is actually trying to craft an attack.

Comparing this with our solution, our alert log will contain two more fields: Application name and version. In this case it will contain: Application_name="telnet".

In our case, we consider this type of scenario as grey positive, since as per the two possibilities considered above; it might eventually be a false positive or true positive. For concluding a grey positive to TP or FP, we require context behind the alert. Alert context is derived from granular information and not from standard parameters such as IP and port only. Rationale behind this is that we need to cover both the possibilities here. There are high chances that a user has actually forgotten his username / password. For this first case, in our raw alert log, we check whether standard email clients such as outlook or thunderbird are used. We also check the pertinent version for these applications. If the application name and version match, we mask this alert as FP. However, if a user is using non-standard application with a non-pertinent version, we mark it for quarantine. In case of Fortinet, it will directly raise an alarm and block the host.

As already mentioned, we have simulated total 50000 network attack scenarios. This simulated alert log is stored in alert store database. Our modules work on this log to aggregate, filter, mask and correlate these raw alerts. The chart shown in Figure 5.2 shows how our proposed approach successfully reduces false positives from raw alerts and makes the log precise for attack prevention. The total log is reduced by more than 90% of the original size.

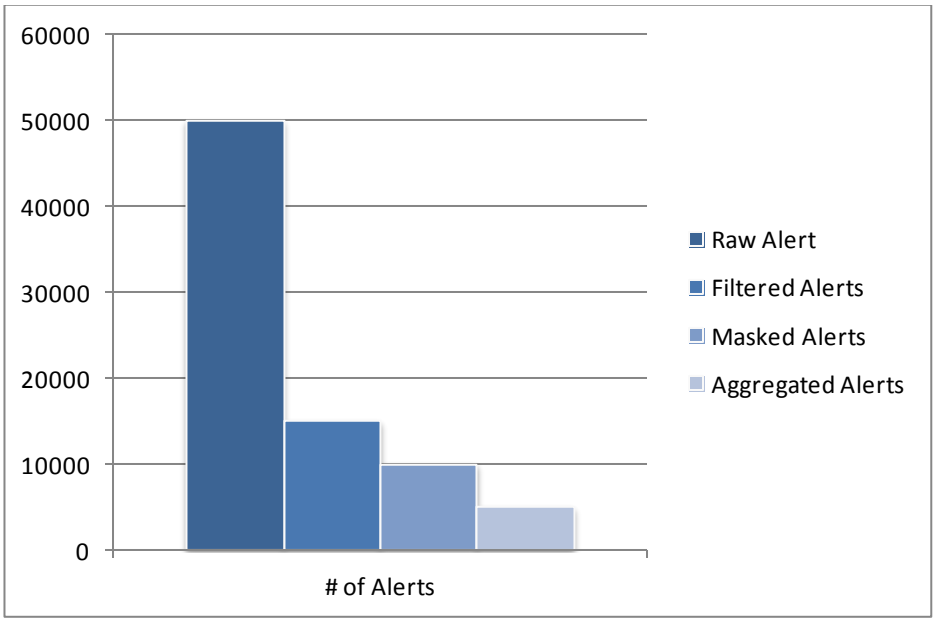


FIGURE 5.2 Alert Aggregation

As shown in our final correlated log, administrator marks the application responsible for the attack as quarantine. Our application quarantine module uses this log to quarantine this application. However, the host on which this application was running can continue to access network resources.

Network Productivity:

Snapshots in Figure 5.3 & 5.4 show network productivity along with network utilization. High network utilization is because of large number of attacks being generated in the network. Host based quarantine and Application based quarantine both achieves same level of network utilization but network productivity is largely affected in case of Host quarantine approach. It is due to the fact that host quarantine approach quarantines Host on which vulnerable application is running. Other applications running on that host also participate in network productivity as they are not vulnerable applications and they are sending or receiving productive information. So when a Host is quarantined, it lowers down the overall network productivity. In our approach we quarantine only the vulnerable application and not the Host. So non-vulnerable applications can still send and receive productive network traffic. Consequently, it doesn't affect overall network productivity. It proves that our approach of application quarantine either maintains or improves network resource productivity.

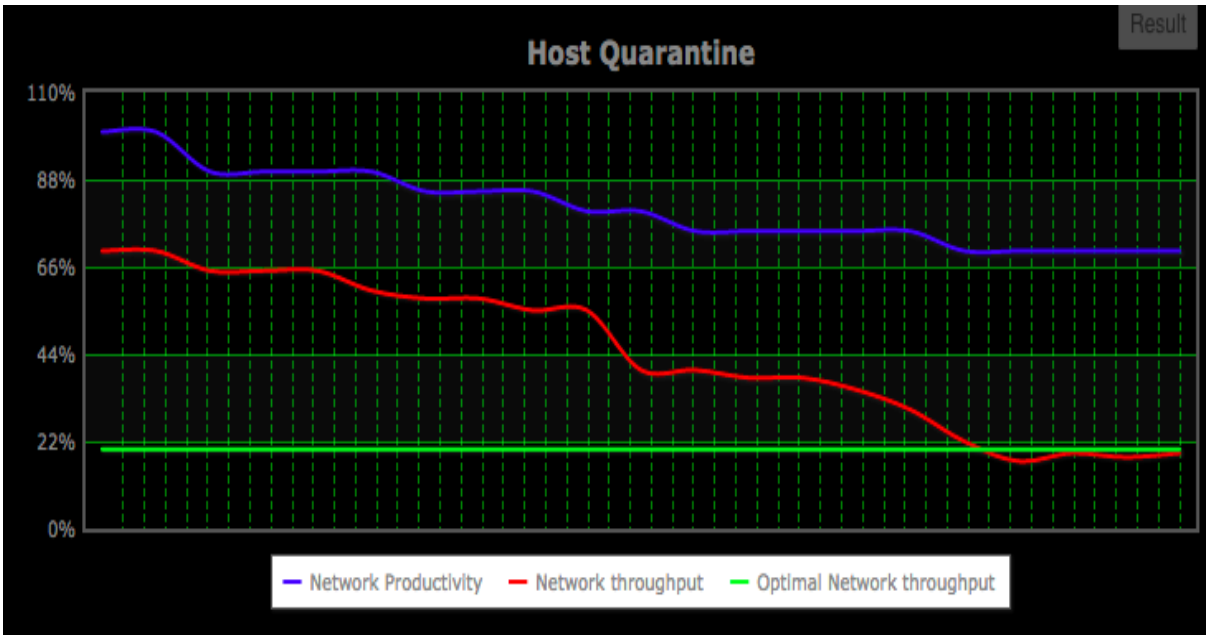


FIGURE 5.3 Network Productivity: Host Quarantine

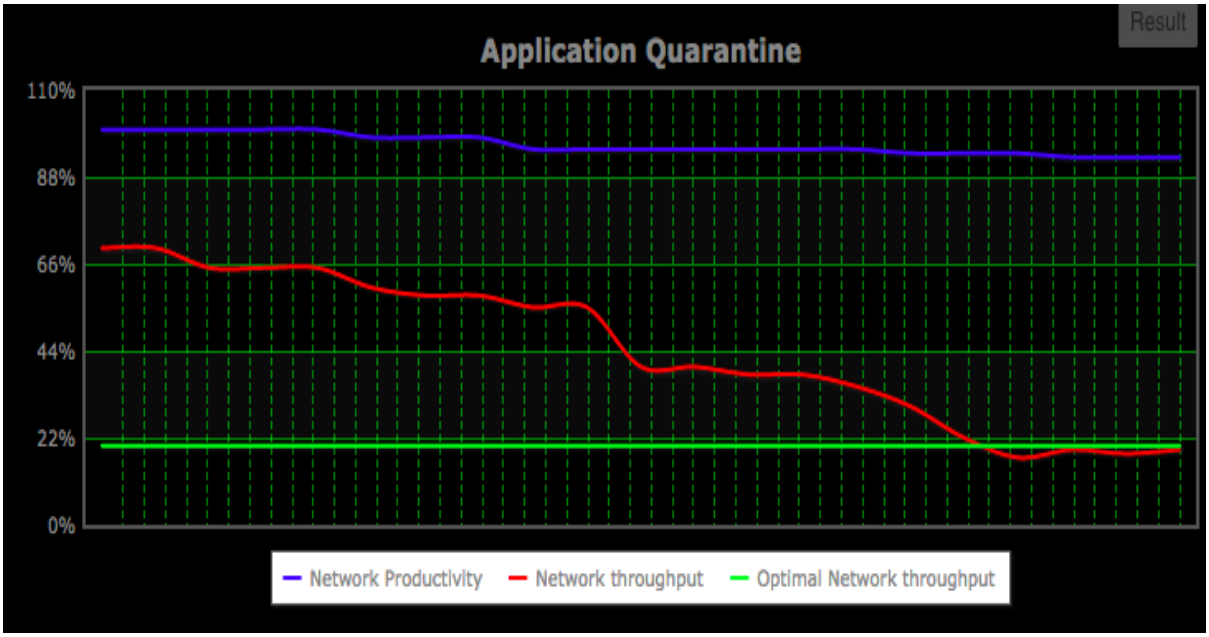


FIGURE 5.4 Network Productivity: Application Quarantine

Accuracy:

We run our solution on each of these logs and monitor the performance on our performance monitor. The snapshot in figure 5.5 shows that because we provide scenario tracking information in our raw alerts itself, aggregation of individual incidents and correlating them to create attack scenarios is more effective than standard IPS:

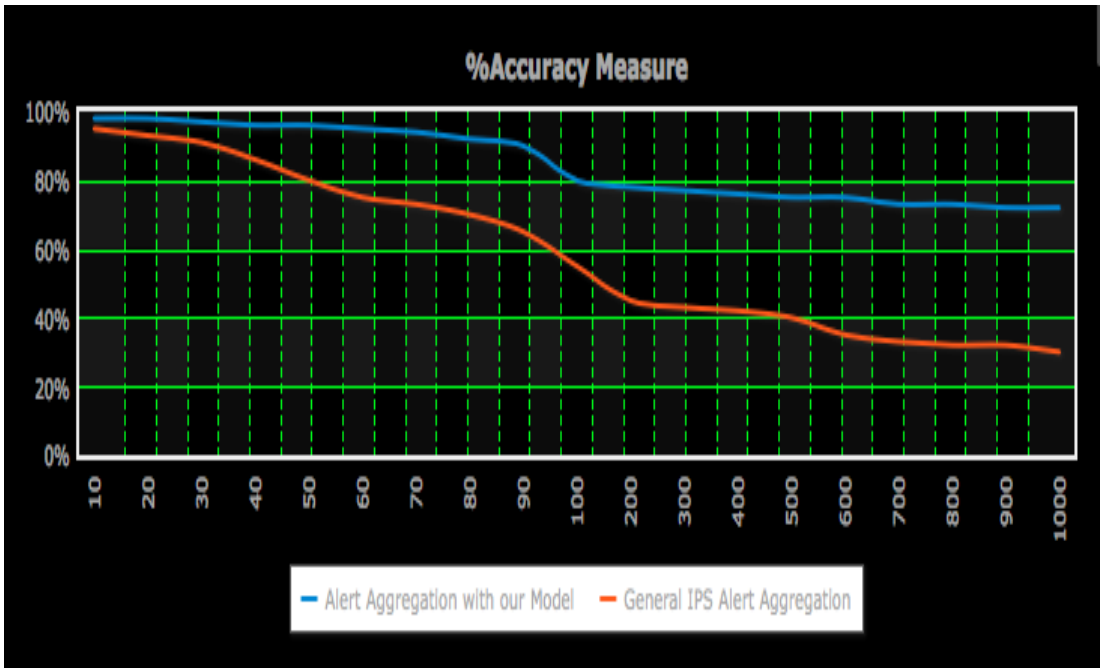


FIGURE 5.5 Accuracy comparison with Other IDPS

5.7 Comparison with Open source and commercial IDPS:

For comparing our proposed system with other popular IDPS available, we observed the outcomes of our approach. Then we observed the performance of Snort, on our dataset (since it is open-source, it was readily available and configurable) in our network. We also configured Forti-Analyzer (Fortinet IDPS), in our network. For comparison with other products CISCO [87], we studied their patent documents (since they were costly for us to procure for live experiments).

CISCO [87] Patents:

US6816973 B1 (Also published asUS6499107) - Method and system for adaptive network security using intelligent packet analysis

This patent creates a network map which comprises information regarding different devices, Oses, Services installed in the network and then uses that information to analyze the network packet. Our invention generates and uses the information in real time. No dependency on the any prior Network Map.

US6954775 B1 - Parallel intrusion detection sensors with load balancing for high speed networks

Load balancing across different nodes has been used to enhance detection power and hence scalability. In our case we are working on Hybrid deployment, so processing is already distributed.

Table 5.10 describes parameterized comparison of our proposed model with existing open source and commercial IDPS. We have identified different parameters such as deployment approach, architecture, aggregation approach, performance measures etc. As we have already discussed in our survey chapter, deployment and architecture play a major part in overall efficiency and effectiveness of an IDPS solution. We have also discussed at length how conventional deployment approach of host based or network based has their own drawbacks in terms of high false positive rate, inability to identify root cause of attack etc. Therefore, our hybrid approach takes advantage of both host based and network based approach and offers effective, novice and a simple solution to provide better security with balanced performance. We describe the comparison in Table 5.10:

TABLE 5.10 Comparison of proposed model with other IDPS

Comparison Parameters	Snort	CISCO	Juniper	Fortinet	Our approach
Deployment (host-based, perimeter-based, network-based, hybrid)	Network-based	Hybrid	Network-based	Hybrid	Hybrid
Architecture (local, centralized, hybrid)	Centralized	Hybrid	Centralized	Hybrid	Hybrid
Real-time attack identification (true/false)	False	False	False	False	True
Attack identity (victim/attacker)	False	False	False	False	True
Attack boundary (from/to inside/outside the network)	False	True	False	False	True
Attack Relevance (is aggregation required?)	False	True	False	True	True
Aggregation approach	None	Does not work for applications using dynamic ports.	Does not correlate alerts	Aggregation is done on IP & Port. Cannot detect attacks which are distributed or correlated.	Aggregation is performed for alerts which are part of single distributed attack. Correlation is done to form a single threat scenario.
Quarantine approach	No database to store alerts generated. So need to be configured externally	Host blocking	None	Host blocking	Application is quarantined, host functions normally

Comparison Parameters	Snort	CISCO	Juniper	Fortinet	Our approach
Network throughput & Productivity	Considerable decrease in throughput because of centralized approach.	Because of host blocking, it can be affected whenever rate of alerts is high.	Does not affect much since no quarantine approach is applied	Network productivity down by 30% and throughput down by 40%.	Almost 80% achieved against optimal benchmark

CHAPTER – VI

Conclusions and Future Enhancements

6.1 Objectives Achieved

The model focuses on run-time alert information retrieval and remediation. The model advocates hybrid deployment and assessment, wherein details of application are retrieved locally from each workstation and sent to a central engine (PMU) for assessment. This distributed nature enhances load balancing and leads to better productivity in the network. Network performance is not compromised. The model defines an alert as an application malfunctioning rather than host malfunctioning. It thus quarantines a vulnerable application rather than a host. This feature provides several other advantages. The model does not rely on any pre-defined pattern matching policy, nor does it define a static list of applications installed on host. Rather, whenever an application initiates a connection from a host, at run-time we extract the application information. This dynamic nature of information retrieval provides precise details of an attack and attacker.

The alerts stored in alert store database provide information about vulnerability of a application with its name and version. This information is crucial for a network administrator for taking preventive measures and makes prevention more effective.

The model also reduces considerable number of false positives from the set of alerts sent by IPS engine to PMU as shown in the below table:

Correlation Action	Alert Names	First Occurance - Last Occurance	Aggregation Time interval (sec)	Pertinent Application	Source	Contacted Destinations	Correlated Alert ID	Correlated By	Direction
Merged	IE Memory Corruption Vulnerability, Port Scan	1/12/2015 10:40:47 - 1/12/2013 10:40:48	300	Internet Explorer	10.1.1.1	20.1.1.1:80, 20.1.1.1:80, 20.1.1.1:12345,255.255.255.255:25,10.1.1.2:25	1,2,3,4,5	Alert Grouping	I_C
Merged	SMTP_AUTH_FAILURE01	1/12/2015 1:40:15 - 1/12/2015 2:04:10	300	Telnet	10.202.4.1	195.130.217.97:25	10,11,12,13	Alert Grouping	I_C
Merged	Apache Chunked-Encoding worm attempt	1/12/2015 2:04:02 - 1/12/2015 2:04:09	215	Apache	10.201.4.22	10.201.4.22, 10.201.4.23, 10.201.4.25	16,17,18,19,20,21,22,23,25,26	Alert Grouping	I_S

Our final correlated log contains precise and granular information about an attack. Each field in the log is helpful to the administrator not only for preventive actions, but also to verify the effectiveness of signatures against known and unknown vulnerabilities. Correlation action field describes whether single raw alert itself was sufficient for attack detection or the attack was a coordinated sequence of events. The field contains values 'single' or 'merged' as per the analysis performed. Alert name is the 'msg' field or name of the attack stored in the pertinent application store database. When PMU matches "sid" from this database for an alert, it stores name of alert value for the matched 'sid' in alert store database. We successfully match multiple alerts with different 'sid' to form a single alert scenario. From the table, we can see that 'IE Memory Corruption Vulnerability' and 'Port Scan' are actually an attempt by vulnerable IE to form a correlated attack sequence. For a merged entry, firstoccurrence_lastoccurrence field is computed from the timestamp field in alert store. It tells the administrator, the first and last time instance in case of correlated attack scenario. Pertinent application is helpful for taking quarantine actions. This application will be quarantined and appropriate patches or hot-fixes will be applied to the same. The application is then monitored and after it is termed 'healthy', it is again given access to network resources. If almost 80% of applications running on a host is malfunctioning, in that case, the host itself is quarantined. Source field provides the host IP responsible for such attacks. ContactedDestinations field provides the targeted host's IP addresses. Correlated ID and CorrelatedBy field describes the alert ID of merged alerts and which of the four stages were responsible for merging the alerts.

As already discussed, we categorize direction of attack as inbound or outbound client-side or server-side. We store these values as I_C, O_C, I_S, O_S. This information is critical for administrator to determine whether the attack was from within the periphery of the network or from outside the network boundary. It also tells us whether a client or server was found to be vulnerable.

The proposed model concentrates on macro-level vulnerabilities present in an unhealthy application and prevents the application from accessing network resources.

We have achieved the following objectives through our research:

- Hybrid Architecture ensures we utilize benefits of both host based and network based detection approach

- By intercepting associated application information at run time, we can prevent attacks at the time of their occurrence
- Precise aggregation and correlation of alerts of same context using application profiling with confidence level
- Attack direction classification helps in identifying victim / attacker from within / outside the network. This results in quarantine / block action.
- False positive reduction by almost 90%
- Application quarantine approach ensures optimal network productivity

An advantage of this model is that it targets the vulnerable application and prevents the application and not the host from illicit access to network resources. The model achieves this without compromising host performance since traffic scanning occurs centrally.

We now discuss how we have achieved effective performance in terms of accuracy for TP and reduction in FP against popular IDPS such as Snort and Fortinet.

Figure 6.1 show that from total log size of almost 50000 raw alerts, how our model generates maximum number of true positives (TP). As we have mentioned earlier, we also focus on grey positives, alerts which has a probability of a TP or FP according to the context in which it is analysed.

For example, we consider the case of an alert generated during nmap SYN scan. If it is generated for Trojan infection, it is a false positive. But, if an alert is generated for an attempt to scan VNC protocol, then it is categorized as a grey positive. In our proposed model, we perform aggregation and correlation using application related information. Therefore, we can identify alert context for both false and grey positives. Our aggregation and analysis results in proper preventive decisions and assists network administrators to react in response to attacks in real time. It also describes how our aggregation and correlation module effectively reduces false positives from original log.

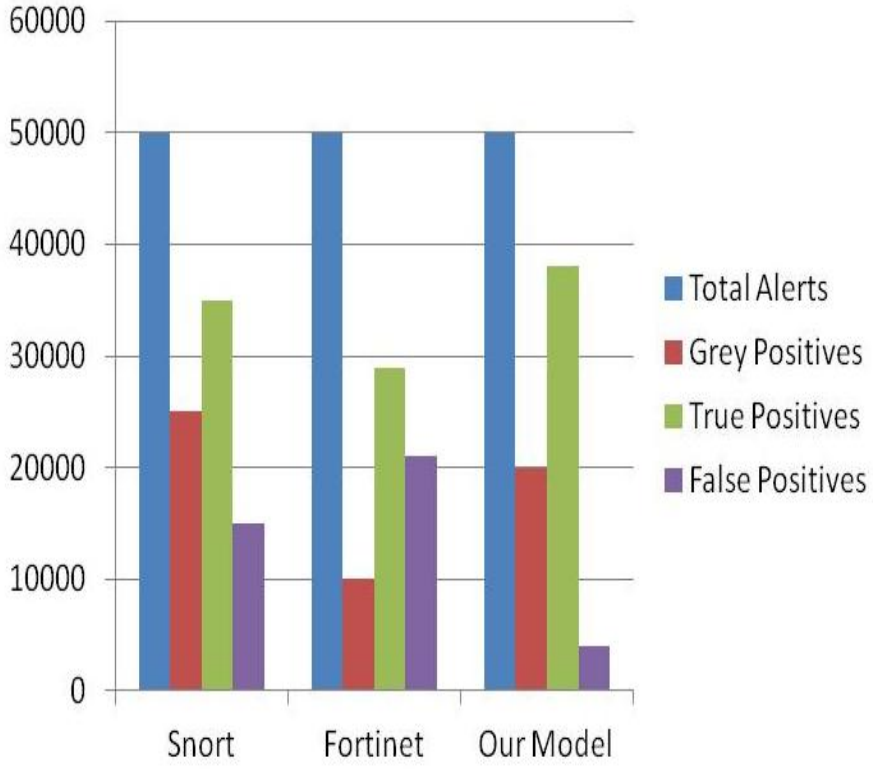


FIGURE 6.1 Comparison of Proposed model with other IDPS

Attack relevance is determined by rate of TP and FP. We assign RTP as rate of true positives and RFP as rate of true negatives. The RTP parameter determines how effective an IDPS is during attack detection process.

$$\text{RTP} = \left[\frac{TP}{TP+FN}\right] * 100\%$$

As the value of FN goes towards 0, RTP value goes towards 100% and it denotes effectiveness of IDPS solution.

Figure 6.2 describes how we achieve effective rate of true positive RTP as compared to other IDPS:

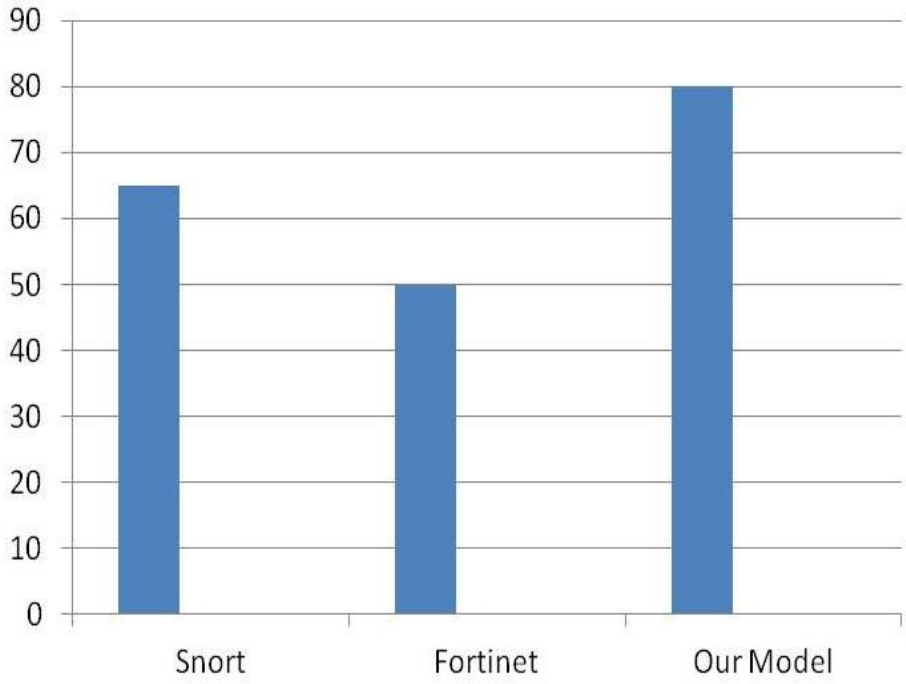


FIGURE 6.2 Comparison of RTP for Proposed Model with other IDPS

In the same manner, we define RFP as how effectively IDPS responds in the FP scenario:

$$RFP=\left[\frac{TP}{TP+FP}\right] * 100\%$$

As the value of FP goes towards 0, RFP value goes towards 100%. It means IDPS is effective against FP.

Figure 6.3 describes how we achieve effective rate of false positive RFP as compared to other IDPS:

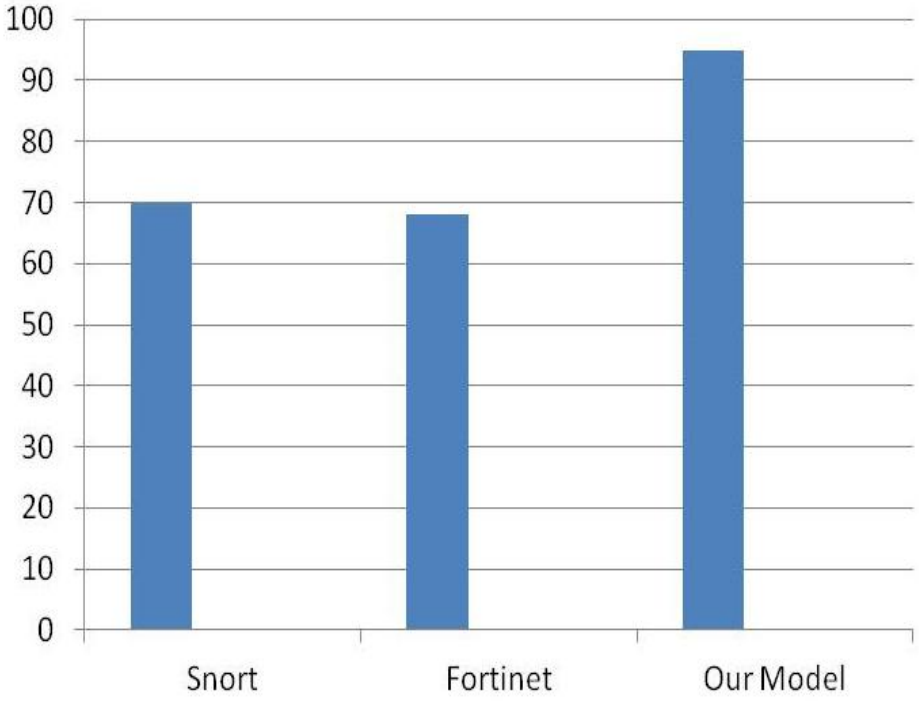


FIGURE 6.3 Comparison of RFP for Proposed Model with other IDPS

We now discuss how during reduction of false positives, our proposed model retains optimal network productivity and throughput. As discussed in the previous chapter, All existing IDPS including Snort, use static policy enforcement methods such as ARP management, 802.1x etc. All these methods for policy enforcement prevent intrusions by blocking further flow of traffic from one to another network (i.e. LAN to WAN or LAN to DMZ).

The traffic can still flow within the network and infect other machines in the same LAN or WAN. Such vulnerable or infectious hosts gain unwanted access to network resources. As shown in our final correlated log, administrator marks the application responsible for the attack as quarantine. Our application quarantine module uses this log to quarantine this application. However, the host on which this application was running can continue to access network resources.

Host based quarantine and Application based quarantine both achieves same level of network utilization but network productivity is largely affected in case of Host quarantine approach. It is due to the fact that host quarantine approach quarantines Host on which vulnerable application is running. Other applications running on that host also participate in network productivity as they are not vulnerable applications and they are sending or receiving productive information. So when a Host is quarantined, it lowers down the overall network productivity. In our approach we quarantine only the vulnerable application and not the Host. So non-vulnerable applications can still send and receive productive network traffic. Consequently, it doesn't affect overall network productivity. It proves that our approach of application quarantine either maintains or improves network resource productivity.

Figure 6.4 shows how we achieve optimal network productivity as compared to other IDPS. We quarantine hosts only when confidence level of alerts is critical or fatal. Other IDPS in case of a host identified as source of an attack, simply block the host without analyzing context of alert.

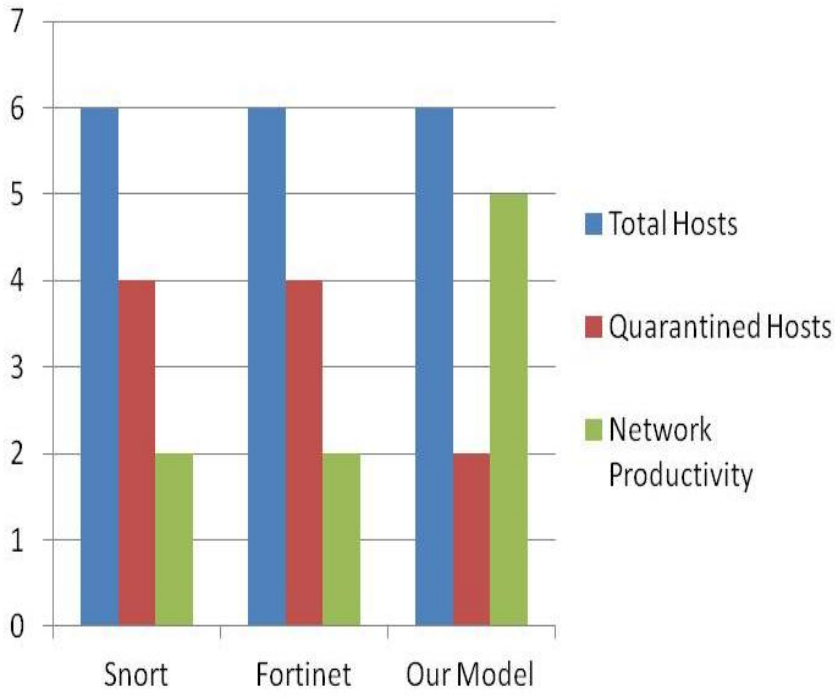


FIGURE 6.4 Network Productivity Comparison with other IDPS

6.2 Conclusion

There is still a long way to go, but our initial results clearly show that, we can successfully reduce the log size by more than 90%. For an attack with multiple alerts with same context, we aggregate into a single alert. False alarms and alerts generated for routine network maintenance are filtered. The reduced log provides precise attack information. The log semantics are used by network administrator to quarantine applications whose information is profiled in the log itself. We also prove how we achieve security without compromising network productivity.

6.3Possible Future Scope

Our final correlated log can be used by any IDPS capable of preventive features. Attack prevention is an area which requires highly optimized raw alerts. Our aggregated and

correlated log contains the root cause of a sequence of alerts. These correlated alerts as a single alert scenario provides a good source of input for attack prevention.

Our modules are flexible enough to be used with any existing IDPS. They complement the performance of IDPS and ensure that network throughput and productivity is not compromised for security.

Application quarantine is the highlight of our approach. The fact that we retrieve application details at the time of request, we prevent an attack at run time. We block the responsible application from gaining access to network resources. However, other applications running on the host continue to access normally. Incident reporting shows an exact report of host-wise applications participating in an attack and application-wise hosts responsible for an attack.

All these features provide an excellent platform for prevention of attacks at run time without compromising network performance.

List of References

[1] Porras, P and Neumann P 1997. EMERALD: Event monitoring enabling responses to anomalous live disturbances.

[2] Zamboni, D and Spafford, E 1999. New directions for the AAPHID architecture. In Recent Advances in Intrusion Detection.

[3] Peng Ning, Yun CUI, Douglas Reeves and Dingbang XU, 2004 ACM Transactions on Information and Security, Techniques and Tools for analyzing intrusion alerts.

[4] J.P. Anderson, 1980, "Computer Security Threat Monitoring and Surveillance", A Technical Report, Fort Washington, Pa.

[5] D.E. Denning, Feb 1987, "An Intrusion Detection Model," IEEE Trans. Software Eng., vol. 13, no. 2, pp. 222–232

[6] Lippmann R., Haines, J.W, Fried, 1999 DARPA off-line intrusion detection evaluation.

[7] Huff, D. 1954, How to lie with Statistics. W. W Norton & Co, New York.

[8] D. Wagner and P. Soto., *Mimicry Attacks on Host-Based Intrusion Detection Systems*. In Proceedings of the 9th ACM Conference on Computer and Communications Security, pages 255–264, Washington DC, USA, Nov2002.

[9] Polla D, McConnel,J, Johnson T Marconi, 1998. *A frame work for co-operative intrusion detection*, In Proceedings of the national conference on Computer security system.

[10] Dain, O. and Cunningham, R 2001.*Fusing a heterogeneous alert stream into scenarios*. In proceedings of the 2001 workshop on Data Mining for Security Applications. 1-13

[11] Kumar, S and Spafford, E.H. 1994. *A pattern matching model for misuse intrusion detection*. In proceedings of the 17'Th National Computer Security Conference 11-21.

[12] Heberlein, L.T. Dias, G.V, Mukherjee, B Wood, *a network security monitor*. In proceedings of the IEEE Synopsium on research in security and privacy

[13] D. Barbara, S Jajodia, L. Popyack and N.Wu., 2002, *Applications of Data mining in computer security*, Norwell, MA, Kluwer

[14] E. Eskin, A Arnold, M. Prerau, L Portnoy, S. Stolfo, 2002, *A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data*, Applications of data mining in computer security, Norwell, MA, Kluwer

[15] J. Zhang, M. Zulkerine, 2005, *Network intrusion detection using random forests,* in Proc. 3'rd Annual conference, Privacy, security trust (PST) , Canada, pp 53,61

[16] J. Zhang, M. Zulkerine, 2006, *A Hybrid Network intrusion detection using random forests,* in Proc. Intl. conference, Availability, Reliability, Security (ARES) , Vienna, Austria, IEEE CS press, pp 262-269

[17] J. Zhang, M. Zulkerine, Jun 2006, *Anomaly based network intrusion detection with unsupervised outlier detection,* in Proc. IEEE Intl. Conference Commu., (ICC), Istanbul Turkey, Vol. 5, pp 2388-2393.

[18] Mark Crosbie, Eugene Spafford, 1995, *Defending computer system using autonomous agents,* Technical Report CSD-TR-95-022, Department of Computer Science, Purdue University.

[19] E. Tombini , H Debar, L.Me., M Ducasse, Dec 2004, *A serial combination of anomaly and misuse based IDS applied to HTTP traffic,* in Proc. 20'th annual computer security appl. Conf., Tucson, AZ, pp 428-437.

[20] W. Lee and S. J. Stolfo. Data Mining Approaches for Intrusion Detection. In Proceedings of the 7th USENIX Security Symposium, 1998.

[21] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated Worm Fingerprinting. In In Proceedings of the 6th Conference on Symposium on Operating Systems Design and Implementation, 2004.

[22] R. Sommer and V. Paxson. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In Proceedings of the 31st IEEE Symposium on Security and Privacy, 2010.

[23] G. E. Suh, J. W. Lee, D. Zhang, and S. Devadas. Secure Program Execution via Dynamic Information Flow Tracking. In In Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems, 2004.

[24] T. Zhang, X. Zhuang, S. Pande, and W. Lee. Anomalous Path Detection with Hardware Support. In Proceedings of the 2005 International Conference on Compilers, Architectures and Synthesis for Embedded Systems, 2008.

[25] X. Zhuang, T. Zhang, and S. Pande. Using Branch Correlation to Identify Infeasible Paths for Anomaly Detection. In Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, 2006.

[26] Usman Asghar Sandhu, Sajjad Haider, Salman Naseer, and Obaid Ullah Ateeb, A Study of the Novel Approaches Used in Intrusion, International Journal of Information and Education Technology, Vol. 1, No. 5, December 2011 Detection and Prevention Systems

[27] Indraneel Mukhopadhyay, Mohuya Chakraborty, Satyajit Chakrabarti, A Comparative Study of Related Technologies of Intrusion Detection & Prevention Systems, Journal of Information Security, 2011, 2, 28-38

[28] S.Vasanthi, Dr. S.Chandrasekar , A study on network intrusion detection and prevention system current status and challenging issue, Proceedings. of International Conference on Advances in Recent Technologies in Communication and Computing 2011

[29] Khalid Alsubhi, Nizar Bouabdallah , Raouf Boutaba, Performance Analysis in Intrusion Detection and Prevention Systems, 12th IFIP/IEEE International Symposium on Integrated Network Management 2011

[30] Ke Yun, Zhu Jian Mei, Research of hybrid intrusion detection and prevention system for IPv6 network, 2011 International Conference on Internet Technology and Applications (iTAP), , vol., no., pp.1-3, 16-18 Aug. 2011

[31] Sourour, M.; Adel, B.; Tarek, A.;, Security Implications of Network Address Translation on Intrusion Detection and Prevention Systems,IEEE International Conference on Network and Service Security, 2009. N2S '09., pp.1-5, 24-26 June 2009

[32] Dong Lijun; Yu Shengsheng; Xia Tao; Liao Rongtao; , "WBIPS: A Lightweight WTLS-Based Intrusion Prevention Scheme," Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on , vol., no., pp.2298-2301, 21-25 Sept. 2007

[33] Usman Asghar Sandhu, Sajjad Haider, Salman Naseer, Obaid Ullah Ateeb, "A Survey of Intrusion Detection & Prevention Techniques", 2011 International Conference on Information Communication and Management,IPCSIT vol.16 (2011) IACSIT Press, Singapore

[34] Nwogu Emeka Joshua, "Network Intrusion Detection and Prevention Systems in Educational System", Bachelor Thesis of the Degree Programme in Business Information Technology Bachelor of Business Administration, 2012

[35] David Wagner, Paolo Soto, "Mimicry attacks on host-based intrusion detection systems", Proceedings of the 9th ACM Conference on Computer and Communications Security, 2002, pg 255-64

[36]Harley Kozushko. Intrusion Detection: Host-Based and Network-Based Intrusion Detection Systems, (2003).

[37] D. Senie, "Network Address Translator (NAT)-Friendly Application Design Guidelines", RFC 3235.

[38] T. Hain, "Architectural Implications of NAT", RFC 2993.

[39] R. Cohen. “On the establishment of an access VPN in broadband access networks”. Communications Magazine, IEEE, 41(2): 156-163. 2003

[40] Cheng-Yuan Ho; Yuan-Cheng Lai; I-Wei Chen; Fu-Yu Wang; Wei-Hsuan Tai; , "Statistical analysis of false positives and false negatives from real traffic with intrusion detection/prevention systems," Communications Magazine, IEEE , vol.50, no.3, pp.146-154, March 2012

[41] P.G. Neumann and P.A. Porras. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In NCSC ’97: Proc. 20th NIST National Information Systems Security Conference, pages 353–365, 1997.

[42] Thomas Heyman, Bart De Win, Christophe Huygens, and Wouter Joosen, “Improving Intrusion Detection through Alert Verification”, IEEE Transaction on Dependable and Secure Computing, 2004.

[43] Koller, R.; Rangaswami, R.; Marrero, J.; Hernandez, I.; Smith, G.; Barsilai, M.; Necula, S.; Sadjadi, S.M.; Tao Li; Merrill, K.; , "Anatomy of a Real-Time Intrusion Prevention System," International Conference on Autonomic Computing, 2008. ICAC '08. , vol., no., pp.151-160, 2-6 June 2008

[44] Ramana Rao Kompella, Sumeet Singh, and George Varghese, On Scalable Attack Detection in the Network, IEEE/ACM transactions on networking, vol. 15, no. 1, february 2007

[45] Konstantinos Xinidis, Ioannis Charitakis, Spiros Antonatos, Kostas G. Anagnostakis, and Evangelos P. Markatos, An Active Splitter Architecture for Intrusion Detection and Prevention, IEEE transactions on dependable and secure computing, vol. 3, no. 1, january-march 2006

[46] Uwe Aickelin, Jamie Twycross and Thomas Hesketh-Roberts, Rule Generalisation in Intrusion Detection Systems using SNORT, International Journal of Electronic Security and Digital Forensics (IJESDF), (1), pp 101-116, 2007

[47] Kai Hwang, Min Cai, Ying Chen, Min Qin, Hybrid Intrusion Detection with Weighted Signature Generation over Anomalous Internet Episodes, IEEE transactions on dependable and secure computing, vol. 4, no. 1, January-March 2007

[48] Sumit A. Khandelwal, Shoba. A. Ade, Amol A. Bhosle and Radha S. Shirbhate, A Simplified Approach to Identify Intrusion in Network with Anti Attacking Using .net Tool., International Journal of Computer and Electrical Engineering, Vol. 3, No. 3, June 2011

[49] Khalid Alsubhi, Nizar Bouabdallah, Raouf Boutaba, Performance analysis in Intrusion Detection and Prevention Systems, Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management, IM 2011, Dublin, Ireland, May 2011,pages 369-376

[50] Shalvi dave, Jimit Mahadevia, Bhushan Trivedi, "Efficacy Of Attack Detection Capability Ofidps Based On Its Deployment In Wired And Wireless Environment", International Journal of Network Security & Its Applications (IJNSA), Vol.5, No.2, March 2013

[51] Shalvi dave, Jimit Mahadevia, Bhushan Trivedi, Parameterized Analysis of Intrusion Detection and Prevention Systems and their Implications on Attack Alerts and Event Co-relation, International Journal of Computer Applications (0975 – 8887) Volume 65– No.9, March 2013

[52] Axelsson S., the Base-rate fallacy and its implications for the difficulty of intrusion detection, ACM Transactions on Information and System Security 2000; pp 186–205.

[53] Uwe Aickelin, Jamie Twycross, Thomas Hesketh-Roberts, Rule generalisation in intrusion detection systems using SNORT, Int. J. of Electronic Security and Digital Forensics,2007Vol.1,No.1,pp.101–116,DOI dx.doi.org/10.1504/IJESDF.2007.013596

[54] Alexander Hofmann, Bernhard Sick, "Online Intrusion Alert Aggregation with Generative Data Stream Modeling," IEEE Transactions on Dependable and Secure Computing, vol. 8, no. 2, pp. 282-294, March-April 2011, doi:10.1109/TDSC.2009.36

[55] F. Valeur, G. Vigna, C. Kru¨ gel, and R.A. Kemmerer, "A Comprehensive Approach to Intrusion Detection Alert Correlation," IEEE Trans. Dependable and Secure Computing, vol. 1, no. 3,pp. 146-169, July-Sept. 2004.

[56] F. Cuppens, "Managing Alerts in a Multi-Intrusion Detection Environment," Proc. 17th Ann. Computer Security Applications Conf. (ACSAC "01), pp. 22-31, 2001.

[57] K. Julisch, "Using Root Cause Analysis to Handle Intrusion Detection Alarms," PhD dissertation, Universitat Dortmund, 2003.

[58] Kai Hwang, Min Cai, Ying Chen, Min Qin, "Hybrid Intrusion Detection with Weighted Signature Generation over Anomalous Internet Episodes," IEEE Transactions on Dependable and Secure Computing, vol. 4, no. 1, pp. 41-55, Jan.-March 2007, doi:10.1109/TDSC.2007.9

[59] Cheng-Yuan Ho; Yuan-Cheng Lai; I-Wei Chen; Fu-Yu Wang; Wei-Hsuan Tai; , "Statistical analysis of false positives and false negatives from real traffic with intrusion detection/prevention systems," Communications Magazine, IEEE , vol.50, no.3, pp.146-154, March 2012

[60] Shalvi Dave, Bhushan Trivedi, Dashang Trivedi, "Simulation Of Security Agent Using Anomaly Based Detection and VLAN Steering", 2011 3rd International Conference on Computer Modeling and Simulation (ICCMS 2011)

[61] Shalvi dave, Jimit Mahadevia, Bhushan Trivedi, “Application Aware Event Logger”, International Journal of Computing, Vol-1,Issue-2,April 2011,pg-201-208

[62] Ramana Rao Kompella; Sumeet Singh; George Varghese;, "On Scalable Attack Detection in the Network," IEEE/ACM Transactions on Networking, vol.15, no.1, pp .14-25,Feb.2007 doi: 10.1109/TNET.2006.890115

[63] Herve Debar, Andreas Wespi, “Aggregation and correlation of intrusion detection alerts”, ACM dl, RAID '00 Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection, pg 85-103, springer-verlag,London

[64] Alsubhi, K.; Al-Shaer, E.; Boutaba, R., "Alert prioritization in Intrusion Detection Systems," Network Operations and Management Symposium, 2008. NOMS 2008. IEEE , vol., no., pp.33,40, 7-11 April 2008 doi: 10.1109/NOMS.2008.4575114

[65] N. Anitha, S.Anitha, B.Anitha, A Heuristic approach for alert aggregation in intrusion detection system, Journal of Computer Applications, Vol-5, Issue 3,2012

[66] Autrel, F. and Cuppens. (2005). Using an intrusion detection alert similarity operator to aggregate and fuse alerts. the 4th Conference on Security and Network Architecture Bat sur Mer, France.

[67] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, “A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection,” Proc. Third SIAM Conf. Data Mining, 2003, http://www.users.cs.umn.edu/~kumar/papers.

[68] W. Lee, S.J. Stolfo, and K. Mok, “Adaptive Intrusion Detection: A Data Mining Approach,” Artificial Intelligence Rev., vol. 14, no. 6,pp. 533-567, Kluwer Academic Publishers, Dec. 2000.

[69] W. Lee and S. Stolfo, “A Framework for Constructing Features and Models for Intrusion Detection Systems,” ACM Trans. Information and System Security (TISSec), 2000.

[70] M. Qin and K. Hwang, “Frequent Episode Rules for Internet Traffic Analysis and Anomaly Detection,” Proc. IEEE Network Computing and Applications (NAC '04), Sept. 2004.

[71] L. Ertoz, E. Eilertson, A. Lazarevic, P. Tan, J. Srivastava, V. Kumar, and P. Dokas, “The MINDS—Minnesota Intrusion Detection System,” Next Generation Data Mining, MIT Press, 2004.

[72] Shalvi dave, Jimit Mahadevia, Bhushan Trivedi, “Windows Based Application Aware Network Interceptor”, International Journal of Enterprise Computing and Business Systems, Vol 2, Issue 1, Jan 2012

[73]FortiAnalyzer Administration Guide found at http://docs.fortinet.com/uploaded/files/1174/fortianalyzer-admin-40-mr2.pdf

[74] D. E. Knuth, J.H. Morris, and V. R. Pratt. Fast Pattern Matching in Strings. SIAM Journal of Computing, 6(2):323–350, June 1977.

[75] R. S. Boyer and J. S. Moore. A Fast String Searching Algorithm. 20(10):762–772, October 1977

[76] A. V. Aho and M. J. Corasick. Efficient String Matching. Communications of the ACM, 18(6):333–340, June 1975.

[77] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood. Deep Packet Inspection Using Parallel Bloom Filters. Symposium on High Performance

Interconnects (HotI), Stanford, CA, USA, pages 44–51, August 2003.

[78] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In D. Barbara and S. Jajodia, editors, Data Mining for Security Applications. Kluwer, 2002.

[79] C. Kruegel, D. Mutz,W. Robertson, and F. Valeur. Bayesian event classification for intrusion detection. In Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC), pp. 1424, 2003.

[80] J. E. Gaffney and J.W. Ulvila. Evaluation of intrusion detectors:A decision theory approach. In Proceedings of the 2001 IEEE Symposium on Security and Privacy, pages 5061, Oakland, CA, USA, 2001.

[81] G. Gu, P. Fogla, D. Dagon, W. Lee, and B. Skoric. Measuring intrusion detection capability: An information-theoretic approach. In Proceedings of ACM Symposium on Information, Computer and Communications Security (ASIACCS 06), Taipei, Taiwan, March 2006.

[82] S. Axelsson, A preliminary attempt to apply detection and estimation theory to intrusion detection, Technical Report 00-4, Chalmers Univ. of Technology, Goteborg, Sweden, 2000.

[83]Layered Service Provider of Winsock2 available at http://www/microsoft.com/msj/0599/LayeredService.aspx

[84] Snort Intrusion detection & prevention system found at www.snort.org

[85] Suricata Intrusion detection systems found at www.suricata-ids.org

[86] Suricata advantages found at http://ossectools.blogspot.in/2011/04/network-intrusion-detection-systems.html

[87] Cisco Intrusion Prevention System [Online]. Available: http://www.cisco.com/en/US/products/ps5729/Products_Sub_Category_Home.html

[88] Edward Guillen, Daniel Padilla, Yudy Colorado, “Weaknesses and Strengths Analysis over Network based Intrusion Detection and Prevention Systems”, 2009. LATINCOM '09. IEEE Latin-American Conference on Communications, pp 1 - 5, DOI: 10.1109/LATINCOM.2009.5305047

[89] An NSS Group Report, Intoto Intrupro V3.0 Techical Evaluation. The NSS Group Security Testing Laboratories. June, 2005.

[90] PacketAlarm: Manual. Copyright © 2006. VarySys Technologies GmbH & Co. KG All rights reserved.

[91] Strata Guard™ v5.0, user guide. Copyright © 2002-2008 StillSecure®.All rights reserved.

[90] Dali, L.; Abouelmehdi, K.; Bentajer, A.; Elsayed, H.; Abdelmajid, E.; Abderahim, B.,”A survey of intrusion detection system Web Applications and Networking (WSWAN), 2015 2nd World Symposium on, 2015,Pages: 1 - 6, DOI: 10.1109/WSWAN.2015.7210351

List of Publications

Patents

1. Dave. S, Mahadevia J., Trivedi B., 2014, *A method and system for network access control based on traffic monitoring and vulnerability detection using process related information*, 4068/MUM/2014 A

Journal Publications

1. Shalvi dave, Jimit Mahadevia, Bhushan Trivedi, "Comparison of Static Policy Enforcement Techniques of NAC", IJETCAS, Issue 12, Volume 2, pp. 101-105, March-May, 2015

2. Shalvi dave, Jimit Mahadevia, Bhushan Trivedi, "Application Aware Event Logger", International Journal of Computing, Vol-1,Issue-2,April 2011,pg-201-208

3. Shalvi dave, Jimit Mahadevia, Bhushan Trivedi, "Windows Based Application Aware Network Interceptor", International Journal of Enterprise Computing and Business Systems, Vol 2, Issue 1, Jan 2012

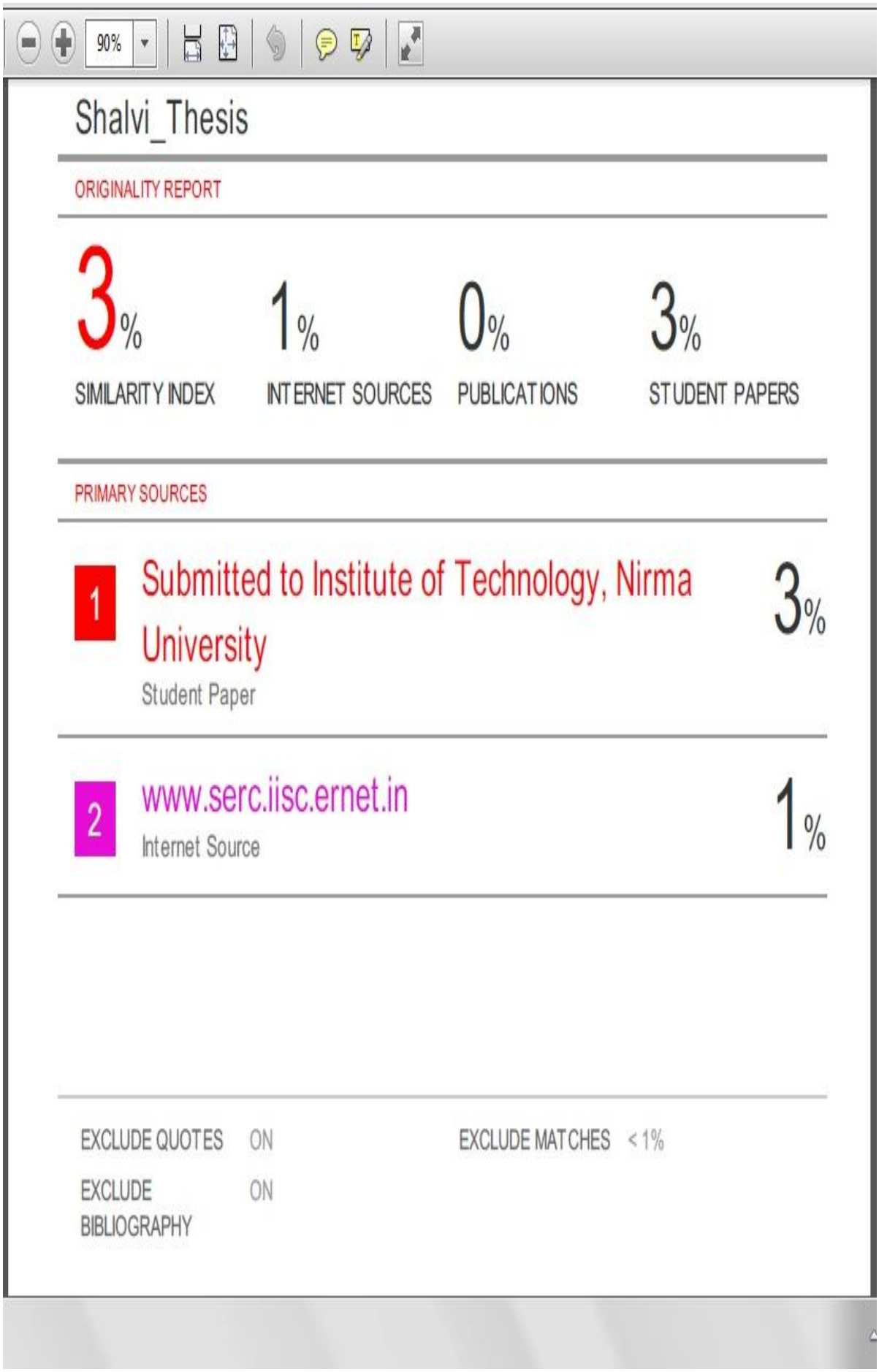
4. Shalvi dave, Jimit Mahadevia, Bhushan Trivedi, "Efficacy Of Attack Detection Capability Of IDPS Based On Its Deployment In Wired And Wireless Environment", International Journal of Network Security & Its Applications (IJNSA), Vol.5, No.2, March 2013

5. Shalvi dave, Jimit Mahadevia, Bhushan Trivedi,"Parameterized Analysis of Intrusion Detection and Prevention Systems and their Implications on Attack Alerts and Event Co-relation", International Journal of Computer Applications (0975 – 8887) Volume 65– No.9, March 2013

6. Shalvi Dave, Bhushan Trivedi, Dashang Trivedi, " Dynamic admission and access control using VLAN steering and anomaly based detection", International Journal of computer science and application, 0974-0767, Vol II Issue 2

Conference Publications

1. Shalvi Dave, Bhushan Trivedi, Dashang Trivedi, “Simulation Of Security Agent Using Anomaly Based Detection and VLAN Steering”, 2011 3rd International Conference on Computer Modeling and Simulation (ICCMS 2011)
2. Shalvi Dave, Bhushan Trivedi, Jimit Mahadevia, “Security policy implementation using connection and event log to achieve network access control”, International Conference on Advances in Computing and Artificial Intelligence ACAI, Rajpura/Punjab, India - July 21 - 22, 2011, pp 29-33


90%
Shalvi_Thesis
ORIGINALITY REPORT
3%
SIMILARITY INDEX
1%
INTERNET SOURCES
0%
PUBLICATIONS
3%
STUDENT PAPERS
PRIMARY SOURCES
1
Submitted to Institute of Technology, Nirma University
Student Paper
3%
2
www.serc.iisc.ernet.in
Internet Source
1%
EXCLUDE QUOTES ON
EXCLUDE BIBLIOGRAPHY ON
EXCLUDE MATCHES < 1%