IEEE CSIDC 2004
National University of Sciences and Technology Pakistan

Omni Secure

Securing Networks, Protecting Against Cyber Terrorism

Team Mentor Dr Saeed Murtaza
shouzib@yahoo.com

Team Members
Muhammad Bilal Anwer
Mbanwer82@yahoo.com
Muhammad Shoaib Alam
Mail_Shoaib@yahoo.com
Qasim Javed
Qasim_nust@yahoo.com
Muhammad Raza ur Rehman
netwizio@yahoo.com

1. Abstract

Computer Networks have become very important in our lives considering their application ranging from an office/home network to the Internet. The Internet alone provides us with many different services, such as email, voice and video conferencing, exchange of files, news etc. However due to unreliable nature of computer networks, this expansion has also exposed us to 'global evils'. Cyber terrorism is one of the most threatening among them. Critical Infrastructures like Dams and Airports have been hacked in the world.

These days one of the most effective ways to bring down a complete network is a denial of service attack. Imagine if DoS attacks were launched on the 911 services. Such attacks target the accessibility of these services thus rendering them non-operational. It is obvious that the breakdown of any critical service for even a few minutes could result in an extreme situation causing severe damage. For example, the terrorists might plan a bomb explosion at a certain place as well as coordinate a Denial of Service attack on the critical infrastructures including hospitals, fire brigades, 911 etc. In such a situation these services would fail to respond to the incident, unable to dispatch the rescue teams to the incident site and provide first aid. In case of a big terrorist activity, like the attacks on the World Trade Center, the timely response of these services, or even worse the service break down would result into a high death toll.

Omni Secure helps maintain the accessibility of the services by detecting Denial of Service attacks. This would ensure service availability in case of an emergency. Moreover, timely detection of such attacks is important, as it would enable the service authorities to be aware of the attack thus enabling them to adopt alternate measures. This would also reduce the response time involved in responding to the incident hence saving human lives. Omni Secure also provides the data mining capability on network traffic hence providing the ability to crack the terrorist communication on Internet.

1.1 Performance Requirements

- Timely and accurate detection of Denial of Service attacks
- Real-time alert reportage
- Attack Mitigation
- Fault Tolerance
- Ability to work at high speed

2. System Overview

2.1 System Description

Omni Secure has many Functional (F) and Non Functional/Quality (Q) Requirements.

F1: The administrator of the service computer network should be able to know the current statistics of the network including abnormal traffic or break-in attempts (known attacks) by hackers.

F2: The administrator should be able to manage the system remotely to a certain extent, so that even if he/she is not present at the terminal, they could take actions against any kind of attack.

F3: The system must be flexible enough to generate a concise report/log of the on going activity for auditing purposes.

F4: The system should provide a platform to detect novel attacks, so that the future attempts of such attacks can be stopped.
F5: The system should immediately alert the concerned authorities in case of an attack and if possible trace back the attacker.
F6: All the modules of the system should have a centralized management interface
F7: The system should also provide an infrastructure so that data mining could be done on network traffic.
Q1: The system should be able to accommodate high-speed traffic
Q2: The system should be able to perform efficiently.
Q3: The system should be scalable and extensible. Scalable means that the system should be upgradeable to accommodate higher speeds. Extensible means that the system should be able to accommodate different type of network traffic analyzers.
Q4: The system should be able to detect attacks (novel and known) with high level of accuracy.
Q5: The system management should have an easy to use graphical user interface.

2.2 Omni Secure General View

The main requirements of the Omni secure are efficiency, scalability and extensibility. Although hardware implementations of network security are present but they don't fulfill all of these requirements. Furthermore, high-end machines (like network processors) were not available to us. So we decided to build a distributed layered architecture for network traffic analysis. This architecture is shown in the figure below.

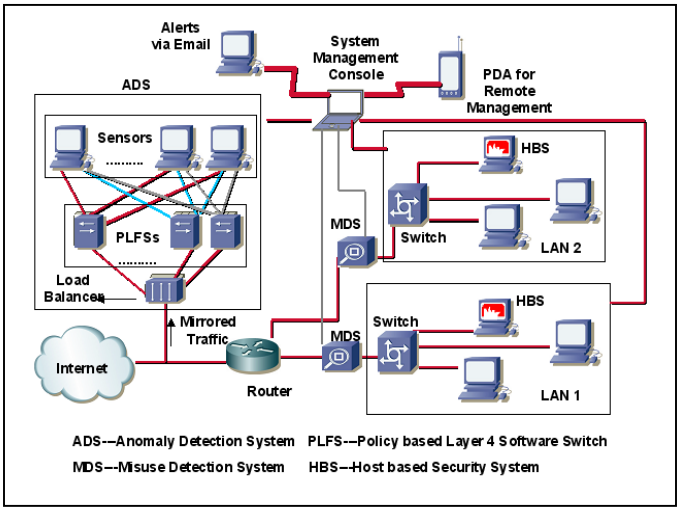


Figure1 Omni Secure

The main modules of Omni Secure are: System Management Console, Anomaly Detection System (ADS) (Novel attacks detection, QoS choking attacks detection), Misuse Detection System (MDS) (detecting known attacks), and Host base Security System (HBS). (Strengthening computer security), Reportage Modules. (Reporting Intrusions via email and PDA) Response Modules. (Taking Response against Intrusions)

3 Omni Secure Design Methodology

Software Engineering practices are the most important things in Software Engineering. We decided to use RUP as the tool to guide us in Software Engineering. We have used incremental and evolutionary model for software development as specified by the Rational Unified Process. We have used this model because we have to improve and modify many modules for many iterations and this sort of engineering is well handled by Iterative Incremental Techniques. In Inception Phase the Requirement Analysis sort activities are done. Elaboration elaborates

the requirements conceived in Inception Phase. Construction Phase in our case consists of 9 different activities. All of these activities are implemented as different spirals (Spiral Model of Software Engineering Life Cycle). 8 of these 9 activities have been implemented and tested or are at the end of implementation phase. The network Traffic Data Mining Module is yet to be implemented.

Whole Life Cycle of OmniSecure is shown as follows

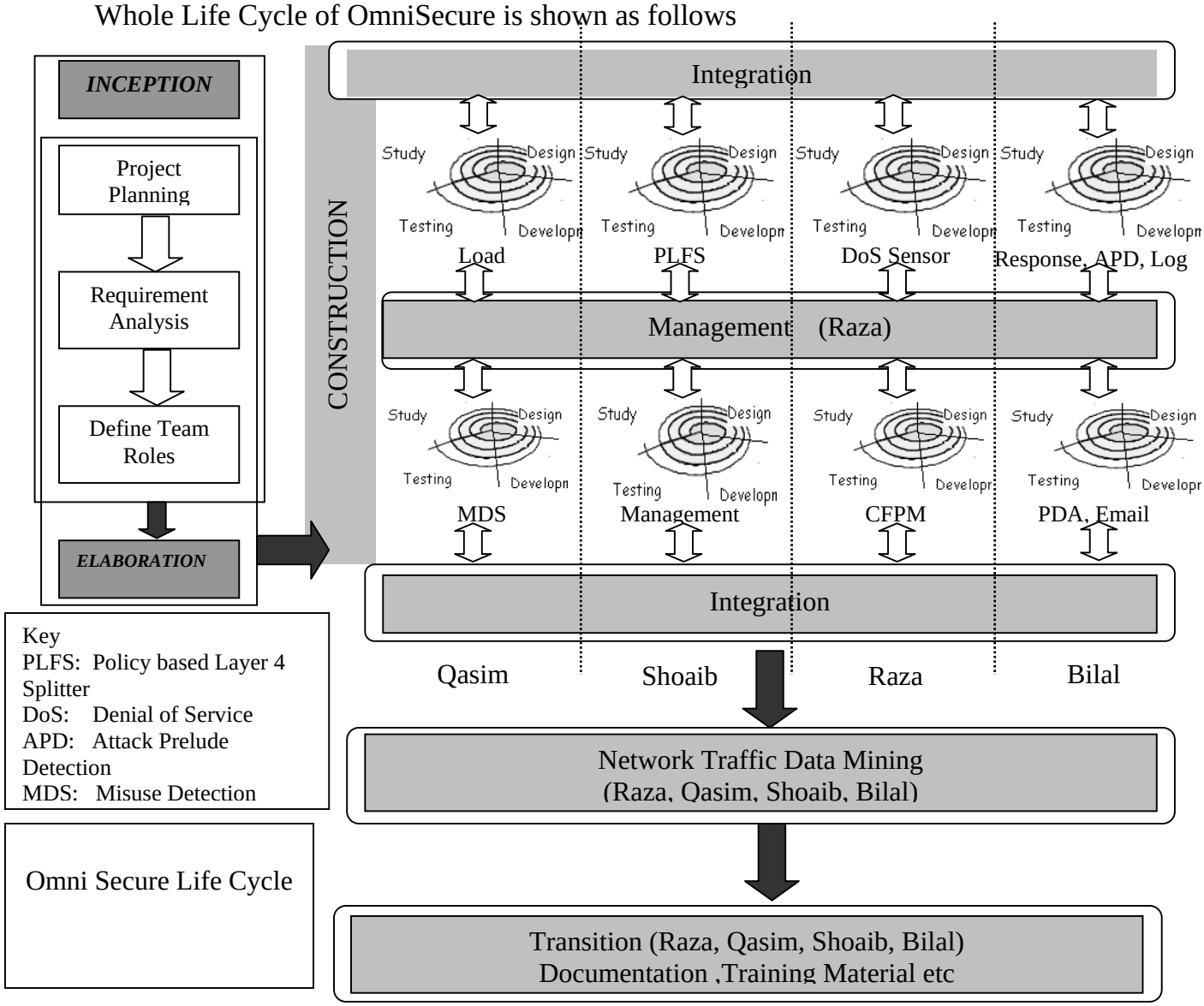


Figure 2 Life Cycle of Omni Secure

The main tools and methodologies used in Omni Secure development are

Artifact	Model	Tool
Requirements	Use Case Diagram	Rational Rose 98
Static Modeling	Class Diagram	Rational Rose 98
Dynamic Modeling	State Chart Diagram, Sequence Diagram	Rational Rose 98
Implementation	C	GCC, GDB, Valgrind
	C++	GCC, GDB, Valgrind

	Java	Personal Java, JCCKIT
Project Management		Microsoft Project
Version Control		Concurrent Versioning System
Automated Testing		Network Simulator 2, Iptraf, NetDude TCPDump, TCPReplay

4 Benefits and Innovations

The real benefit of the Omni Secure is that it is designed to secure the networks of critical infrastructures like 911, Fire Brigade etc. The Cyber Attack on these critical infrastructures has not been given much consideration before

Innovation of Idea: Securing critical services

OmniSecure is unique in this aspect that it concentrates on providing security to the computer network of services that are critical for human safety. These include the 911 service, fire brigade, hospitals etc.

Innovation of Application: Remote management using a PDA

Omni Secure enables the administrator to view system statistics and manage the system remotely using a PDA. This meets the 24/7 operating requirements of critical services such as 911, hospitals, fire brigades etc.

Further more the implementation of Omni Secure is also unique in the sense that it applies Novel and Efficient techniques for the purpose of Network Analysis and Cyber Attacks Detection.

The major benefits of the Omni Secure for the society are as follows

Real-time detection of attacks

The system is able to detect attacks in real-time and alert the authorities as soon as possible. Real-time attack detection will ensure that the important servers and the applications controlled by them are intact.

Detection of novel attacks

Omni Secure provides a platform to detect novel attacks (the attacks having no specific pattern), thus strengthening the security of critical services' computer networks. This would also ensure that accessibility to and integrity of Important Servers is intact.

Reduced Incident response time

Real time attack detection would ensure that accessibility to the emergency services is maintained. Hence facilitating quick response in the case of any emergency

Network Traffic Data Mining

Another important benefit of the Omni Secure is that the sensors performing data mining on the network traffic can be easily integrated into the system. This would help in tracking the activities of the terrorists online.

5 Implementation and Engineering Considerations

Use case Model

The main use case diagram modeling the interaction between the System Administrator and the system is as shown in the figure 3.

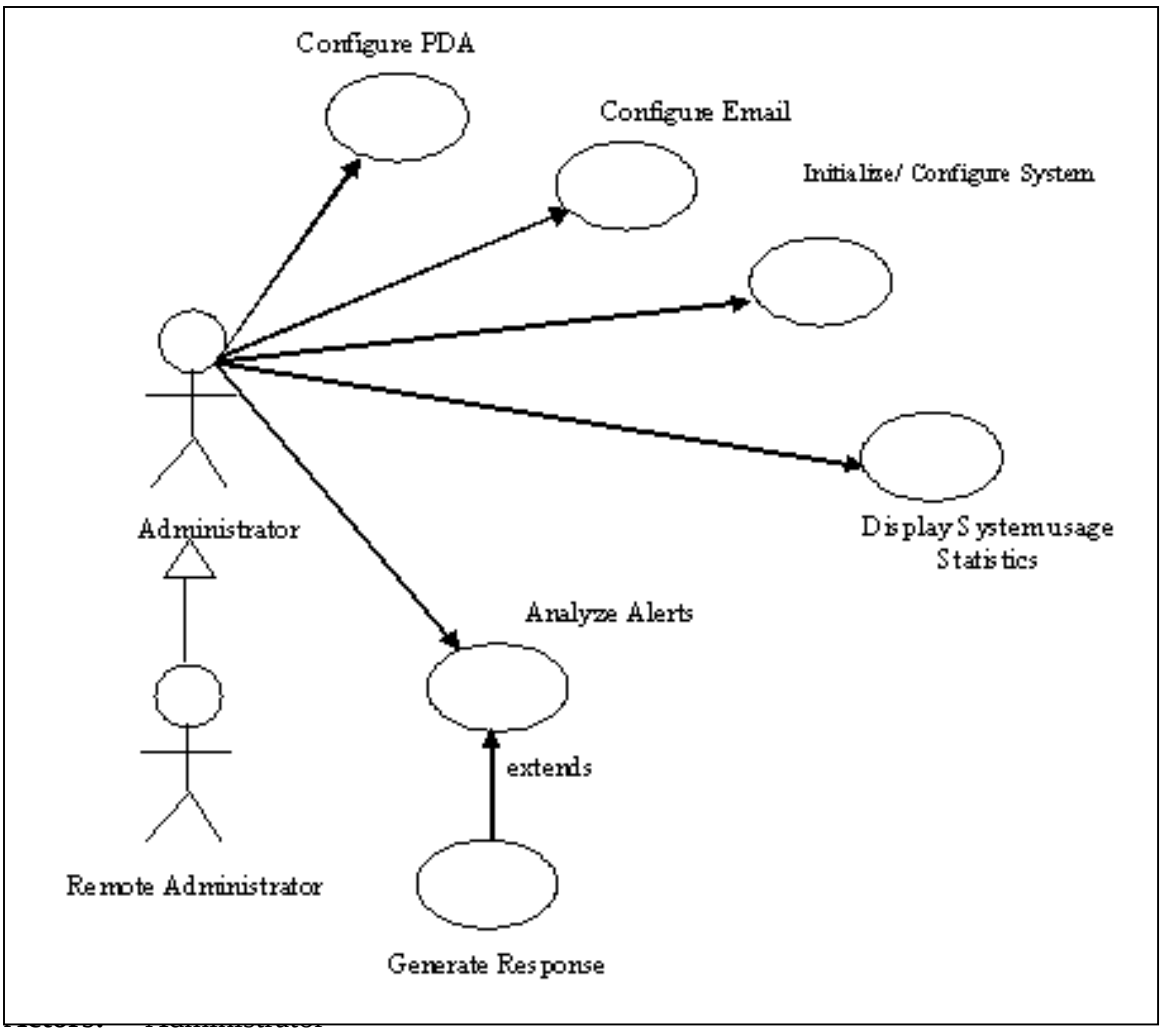
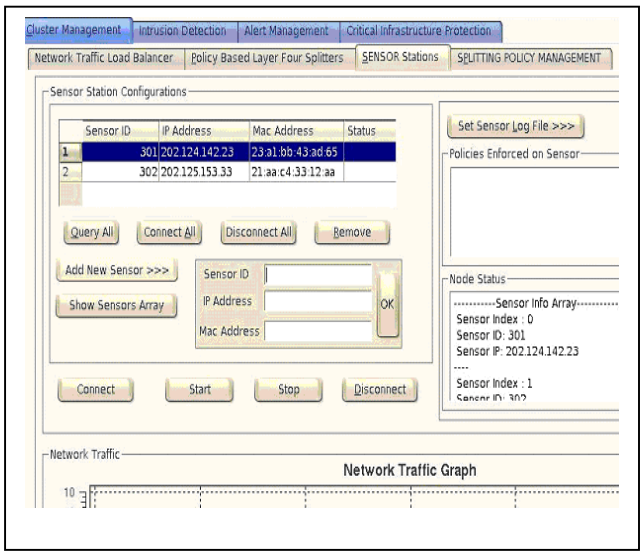


Figure 3 Omni Secure: Use Case Diagram

Use Case: Initiate/Configure System
Actors: Administrator, Remote Administrator
Purpose: To start up the whole system for data capturing, analyzing and generating Alerts. And loading policies for various issues.

Overview:

In this use case the administrator can initiate the whole system using the management console. The system is initialized with the default policies already developed by the OMNI SECURE team. Moreover if the administrator wants to load new policies he can load them after defining them. This use case has further more specialized use cases like configuring Load Balancers, PLFS, and DoS Sensors etc.



Actor Action	System Response
Initialize	Start the load balancer, PLFS Switch and sensor with default policies.
Configure System	Specify the new configuration policies for different modules
Load Configuration Files	System works according to the new configuration files loaded.

Use Case: Analyze Alert
Actors: Administrator, Remote Administrator
Purpose: To maintain a specific flow of alert to the administrator.
Overview: In this use case the administrator can define on the management console that how the alerts should be handled by the OMNI SECURE system. It can be configured to send the email or alert to the PDA.

Actor Action	System Response
Analyze Alerts	The system administrator can analyze alerts being displayed on the system GUI. These alerts are communicated to the system management console from different intrusion detection modules like Misuse Detection System, Critical Server Security Systems

Use Case: Display System Usage Statistics
Actors: Administrator
Purpose: To help the administrator about the future needs of the Omni Secure like adding another senor or adding another splitter.

Overview: This use case helps the system administrator to decide whether any new sensor is needed or not. The administrator can choose to see the various system statistics of Load Balancer; Policy based Layer Four Splitters and different Sensors.

Actor Action	System Response

Click on the respective graph tabs	The graph of sensors and alerts are shown

Use Case: Configure Email
Actors: Administrator
Purpose: To change the email configuration, according to which the email is sent to the email address of the Administrator.

Overview: The administrator can configure the system to send an email to him or some specific mail address in case of an alert is generated from any of the sensors. Administrator specifies the address as well as the SMTP server through which the email is to be sent. The

Actor Action	**System Response**
Add Email options	
Save email configuration	The configuration file is saved.

Use Case: Configure PDA
Actors: Administrator
Purpose: To modify the policies that defines the PDA interaction with OMNI SECURE.
Overview: The administrator can interact with the management console to change the policy that defines different parameters of the PDA. The PDA runs the server while the client is on the Management Console. Management console runs the client and connects with the PDA, which facilitates policy updating at the PDA. This also allows administrator that alert should be sent on the PDA or not.

Actor Action	**System Response**
Change PDA Configuration	Client-Server interaction results in policies being updated for PDA.
Restore defaults	Revert to default policies

Construction of Omni Secure

Anomaly Detection System

Anomaly Detection System is composed of the following three parts

1. Load Balancer
2. Policy based Layer4 Splitters
3. Sensor Cluster

Load Balancer

Introduction

One of the key requirements of Omni Secure is the ability to work on high speed In order to make OMNI SECURE work at high speed it was decided to make OMNI SECURE

ADS a distributed clustered architecture. The load Balancer was the interface of the ADS to the mirrored network traffic. The main requirements of the Load Balancer are

1. Capture Network Traffic
2. Forward Network Traffic
3. Forward Network Traffic as fast as possible

Design and Implementation

The load balancer has to forward the network traffic as fast as possible. Traditionally most of the network management and security applications used the Raw Sockets (Layer 3) for the network traffic Sniffing. Recently libpcap and the packet sockets (Layer 2) have also gained ground.

Layer 3 Sniffing vs. Layer 2 Sniffing

The usage of raw sockets was rejected in the load balancer development because the raw sockets use layer 3 sniffing and each packet is routed through the kernel to the user process while layer 2 sniffing techniques directly pass the packet from the network interface or the driver to the user process. This is especially very effective in the case of packet capturing and forwarding both.

Libpcap vs. PF_Packet Sockets

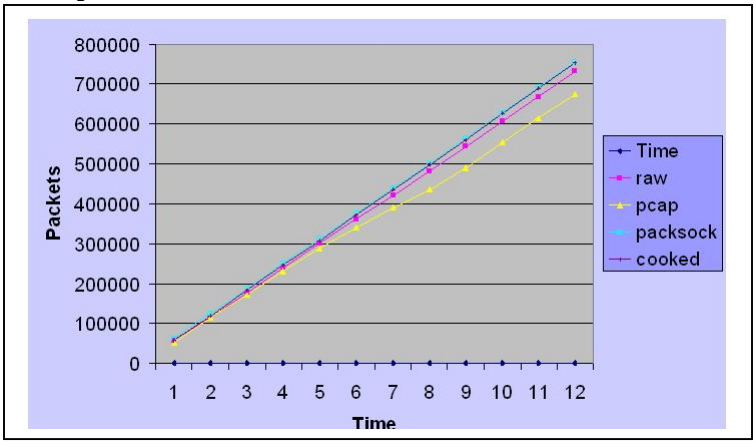
Libpcap and PF_Packet sockets both use layer 2 sniffing techniques but Libpcap is a generalized packet-capturing library. And the generalization results in some loss of efficiency. So Libpcap was rejected and PF_Packet sockets based packet capturer and forwarder was implemented from scratch.

Packet forwarding Policy

The load balancer can load balance the traffic in many ways. For example it can forward each packet to every different PLFS (Policy Based Layer 4 Splitters) in a round robin base or it can forward the packets based on time. We are opting for the second option cause it is more efficient to timely load balance among the PLFS because in the first case the switching for every new packet can also cause great burden on the Load Balancer. For a predetermined time slot packets are sent to PLFS 1, for the next time slot the packets are sent to the second PLFS and this process continues in a round robin basis.

Testing

For testing the network traffic was dumped using the TCPDump tool. A dump of one Giga Byte was made and replayed using TCP Replay tool.



Policy based Layer 4 Splitters (PLFS)

The key requirements of the OMNI SECURE include scalability, flexibility and extensibility. OMNI SECURE must be able to accommodate different types of sensors and a number of sensors foe network traffic data mining and Intrusion Detection. To insure that different types of sensors and more number of sensors can be added to the OMNI SECURE each of the sensors in the ADS subsystem of OMNI SECURE must get only the traffic for which it is configured. This is the role of Policy based Layer 4 Splitters. The basic purpose of the PLFS is to distribute the traffic amongst the sensors according to the configuration on the sensors.

Design

Flexibility of PLFS

They key requirement while designing the splitters was that they should be fully configurable. But to make the splitter fully configurable requires special designing in the PLFS. The design of the PLFS marks the following requirements:

1. A number of policies should be definable on each of the PLFS.
2. There should be no limit on the policies being general or specific.
3. Policies should be definable according to a no of parameters like the source IP, destination IP, source port, destination port or any combination of these parameters.
4. Packets belonging to the same attack should be forwarded to the same sensor otherwise the attack cannot be detected.
5. While splitting traffic according to the user-defined policies, the packet duplication should be kept at minimum.
6. The load balancing should be done so that the disparity of load among sensors is minimum.

PLFS has been designed with these options in mind.

Implementation

Policy Definition

In our system the splitter loaded with policies for each sensor does the traffic splitting. The user can define any number of policies according to the requirement. These policy definitions enable the splitter to send the respective traffic flows to the corresponding sensor. There can be multiple policies loaded for a single sensor. All the sensors have a common rule-set. The policies define rules against which the packet header is matched and the packet is then forwarded to the sensor for which that policy is defined. In this way policies are used as a pre-processing mechanism to reduce the amount of traffic to be analyzed by the sensor. The user can define a

specific policy or a very general one. Generally a more general policy will load the system more than the specific one.

As all the user-defined policies may not be mutually exclusive so two or more policies might result in the forwarding of same packets to more than one sensors thus degrading system performance. This is termed as the problem of packet duplication. In order to deal with this problem, a similarity measure is determined for each possible pairing of policies by monitoring the number of similar packets forwarded by each policy in the pair. The similarity measure is indicative of the amount of overlap in two policies.

The policies are reloaded for multiple sensors to a single sensor when the similarity measure for the pair exceeds a certain threshold defined by the user.

Load Balancing among Sensors

Balancing the load among different sensors is one of the major requirements of good traffic splitter. Different people have investigated efficient load balancing for intrusion detection system. Traffic should be distributed among sensors so that their performance is maximized, by keeping the load disparity at minimum. Assuming the cluster of N identical sensors, the ideal situation is to distribute 1/N of the total load to a single sensor. In our approach when a disparity in sensors' load crosses a threshold and a need arises for shifting some traffic load from more loaded sensor to less loaded one, a reloading of policies is done. This mechanism of policy reloading to shift traffic load works well to evenly balance the load among sensors. While balancing the load among different intrusion detection sensors, the issue of flow perseverance arises. When packets of a single attack are distributed among different sensors, the attack would go undetected. This problem is handled as our approach of reloading policies for shifting the traffic load ensures that packets of a single attack are not distributed among different sensors. Hence the attack detection rate is not lowered due to shifting of flow.

We can divide the problem of load balancing into three steps; load evaluation, policy selection, and policy reloading. Our system keeps track of system-load parameters such as memory occupied CPU usage etc. for all sensors. These parameters are sent to system manager from each sensor for evaluating the need for balancing the load. We also approximate the load the due to a policy on a sensor. This is called the Policy Load Factor (PLF) of the policy. It is approximated by determining the factor of traffic forwarded by that policy to the sensor. So when the disparity is observed among sensors, the suitable policy to be reloaded is selected by comparing PLF of each policy.

Testing

The Table 2 depicts the results when the tests were carried without using the mechanism of dynamic reloading of policies. We define and enforce the policies for sensors statically. For judging the performance of the system, we recorded values for system load, memory occupied and attack detection rate (ADR) at each sensor. These values were updated after every 90 seconds once the system was invoked. After this we ran the system with dynamic policy reloading mechanism and recorded values for system load, memory occupied and attack detection rate at each sensor updated every 90 seconds. Table 3 shows the result of it.

	Sensor 1	Sensor 2	Sensor 3
System Load	23%	68%	37%
Mem Occupied	52%	87%	63%
ADR	71%	75%	83%

Table 2: Results without dynamic reloading of policies

	Sensor 1	Sensor 2	Sensor 3
System Load	24%	62%	45%
Mem Occupied	58%	83%	65%
ADR	68%	77%	79%

Table 3: Results with dynamic reloading of policies

	Sensor 1	Sensor 2	Sensor 3
System Load	31%	54%	48%
Mem Occupied	61%	81%	70%
ADR	71%	79%	77%

Table 4: Results getting leveled, with dynamic reloading of policies

	Sensor 1	Sensor 2	Sensor 3
System Load	41%	51%	64%
Mem Occupied	67%	74%	74%
ADR	73%	81%	58%

Table 5: Low detection rate on Sensor3

The readings in Table 5 were recorded 4 minutes after the readings in Table 4. Here sensor 3 is the most heavily loaded. The attack detection rate for sensor 3 is very low because the policies reloaded for this sensor is forward denial of service traffic.

DoS Detection Sensors

Design

The main requirements of the DoS detection sensor are as follows:

- Detect DoS attacks accurately
- Detect DoS attacks in real time
- The sensors must be optimally configurable
- The sensors should use efficient coding techniques and paradigms.

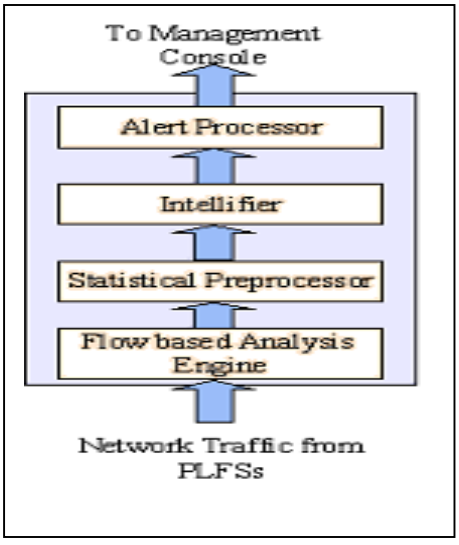


Fig : Conceptual Model of DoS detection Sensors

The DoS Detection Sensors were designed according to the Pipe and Filters design pattern. Each Layer in the DoS detection sensors performs some sort of data computation and passes on the results to the next layer. The Conceptual Model of the DoS detection sensor is as shown in last figure:

Flow based Analysis Engine

The basic purpose of the flow based analysis engine is to separate the network traffic coming to each sensor according to the policy on the sensor. The sensors are fully configurable. The FBAE is meant to make flow records for each type of flow and then pass it on to the Statistical Pre-processor. Generalized Flow based analysis mechanism enables us to perform Anomaly Detection on a number of parameters in different situations.

Statistical Pre processor

The statistical pre processor reads the FBAE records and then compares it to the normal profile. Recent research in Statistics has shown that probability density function based (PDF) methods (methods keeping into account the whole profile of the event) are more robust as compared to the mean and variance based methods. Kolmogorov Smirnov Test is a goodness of fit test often used to detect Deviations between the two data sets. We decided to use the KS test and its variants for detecting the deviations between the normal and the incoming traffic.

Kolmogorov Smirnov Test

Kolmogorov Smirnov Test is based on the empirical cumulative distribution function (ECDF). Given N ordered data points y_1, y_2, y_3... y_N the ECDF can be defined as $\mathbf{E_m = n_m/N}$ Where n_m is the number of points less than y_m where m=1, 2…N. The two-sample KS test is elaborated by the following equation $\mathbf{D=max_{-\infty<x<\infty}|T(x)-S(x)|}$

Where the T (x) and S (x) are the two ECDFs to be analyzed by the KS test.

The KS test has many advantages like that it is not dependent on the underlying distribution and further it is in exact test (does not depend on the number of data points). But it has certain limitations also like that it is more sensitive near the center than at the tails.

Anderson Darling Test (AD)

The Anderson Darling test was an attempt to improve the faults in the Kolmogorov Smirnov Test i.e. the KS test is more sensitive at the center than the tails. In order to come across this problem Anderson, Darling proposed following variation in KS test. The AD test is given below

$$\mathbf{A=max_{-\infty<x<\infty(}|T(x)-S(x)|)*(T(x)(1-T(x))^{-1/2}}$$

Manikopoulos Test

Manikopoulos [6] modified the KS test to especially cater for denial of service attacks. Denial of Service attacks detection should be robust against the flash crowds and subtle variations in network traffic. So to cater for this Manikopoulos modified the KS and added the difference of the area between the two curves to the original KS D factor so the Manikopoulos test is given by

$$M=\max_{-\infty<x<\infty}\{|T(x)-S(x)| + f(N)*(\Sigma_{i=1\ldots k}|T(x)-S(X)|)\}$$

Where f (N) is the function catering for the total no of observations in the following analysis time window.

Intellifier- The Intelligent Classifier

Neural Networks are criticized for heavy computations.

Our distributed architecture enables us to use neural networks with considerable ease. The Intellifier is designed to be a library containing implementation of various forms of intelligent techniques meant for classification. In future we are going to extend the Intellifier to other techniques also and the implementation of these techniques for the anomaly detection.

We have tested the following three neural networks in OMNI SECURE

- Back Propagation Neural Network
- Perceptron Back Propagation Hybrid
- Fully Connected Perceptron Back Propagation Hybrid

Back Propagation Neural Network

The Multi layer Perceptron with back propagation or the feed forward back propagation neural network is most widely used Neural Network Architecture. It utilizes the generalized Delta Rule for training.

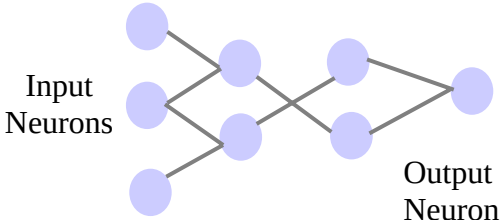


Figure Back Propagation NN

Perceptron Back Propagation Hybrid (PBH)

Perceptron Back propagation Hybrid can be regarded as a derivative of back propagation neural network architecture. It uses the same training rule but the architecture is a bit modified so that some extra links are added from the neurons of the input layer to the neurons of the output layer.

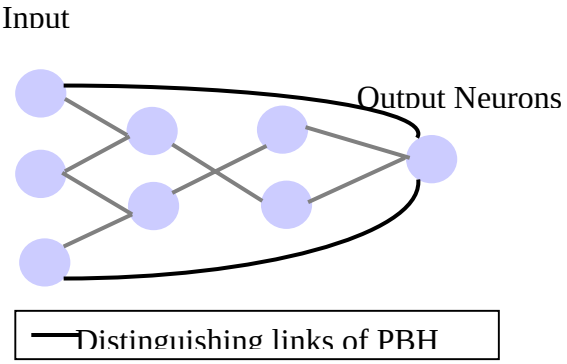
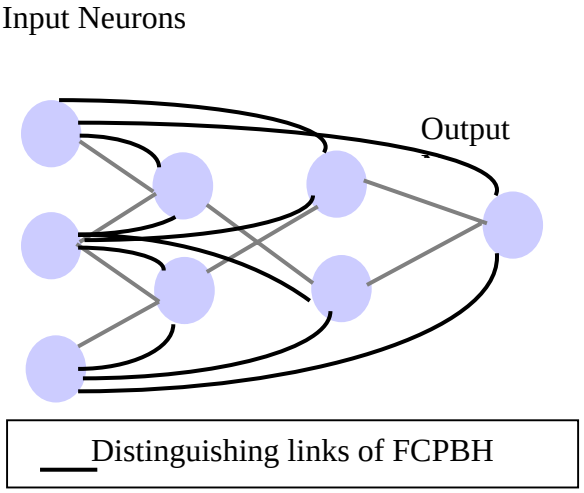


Fig Perceptron Back Propagation Hybrid

Fully Connected Perceptron Back Propagation Hybrid (FCPBH)

On the same line of thinking as Perceptron Back Propagation Hybrid we conceived a new Neural Network Architecture, which we named as Fully Connected Perceptron Back Propagation Hybrid (FCPBH). In FCPBH we introduced the direct



effect of the input to each neuron. This neural network architecture is shown below

Alert Processor

Alert Processor processes the alerts and handles the updating of normal profile.

Alert updating is performed as follows. Let T_{old} be the reference model before updating, T_{new} be the reference model after updating, and T_{obs} be the observed user activity within a time window. The formula to update the reference model is

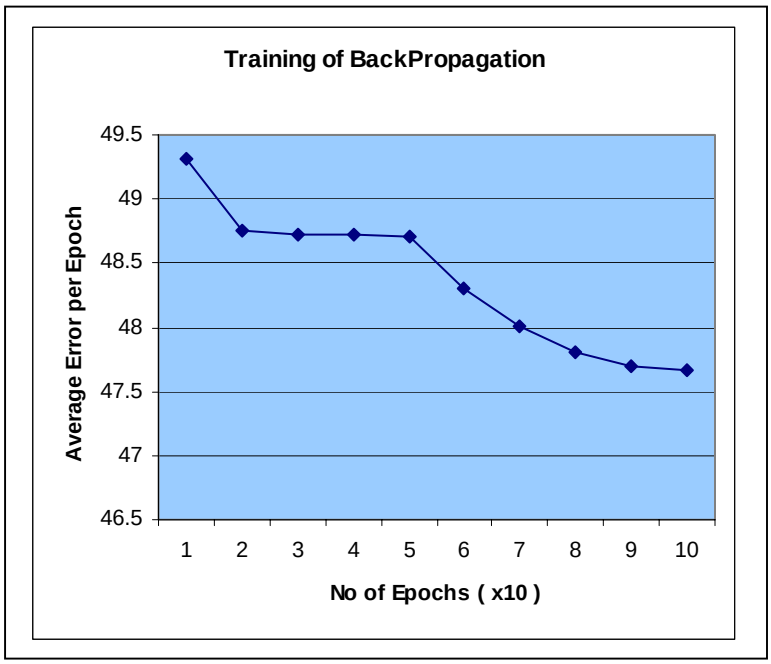
$T_{new}=T_{old}(1-s*a)+T_{obs}*s*a$

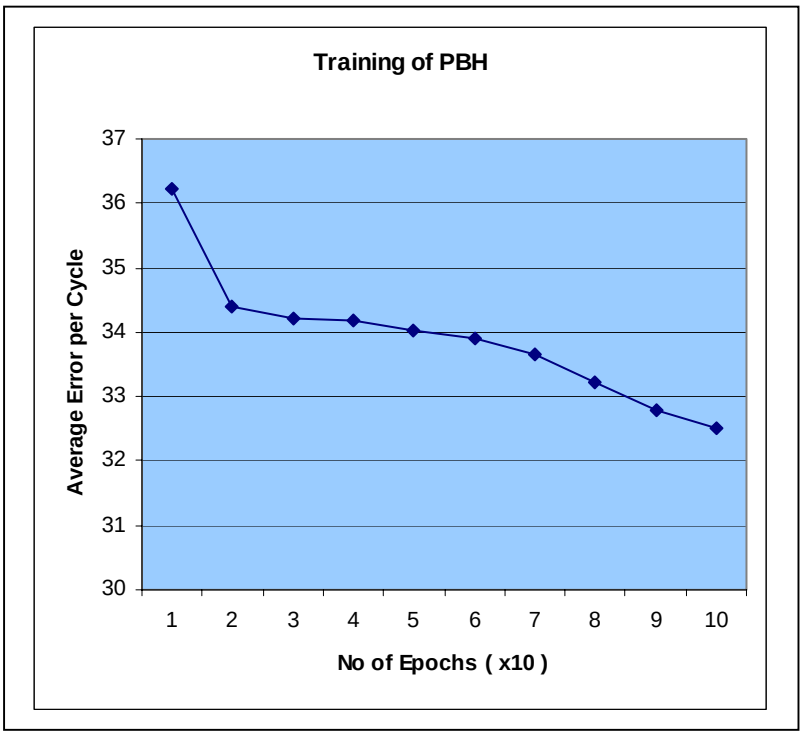
Where **a=predefined learning rate, s=0 (if attack) (s=output of the Neural Network), s=out otherwise**

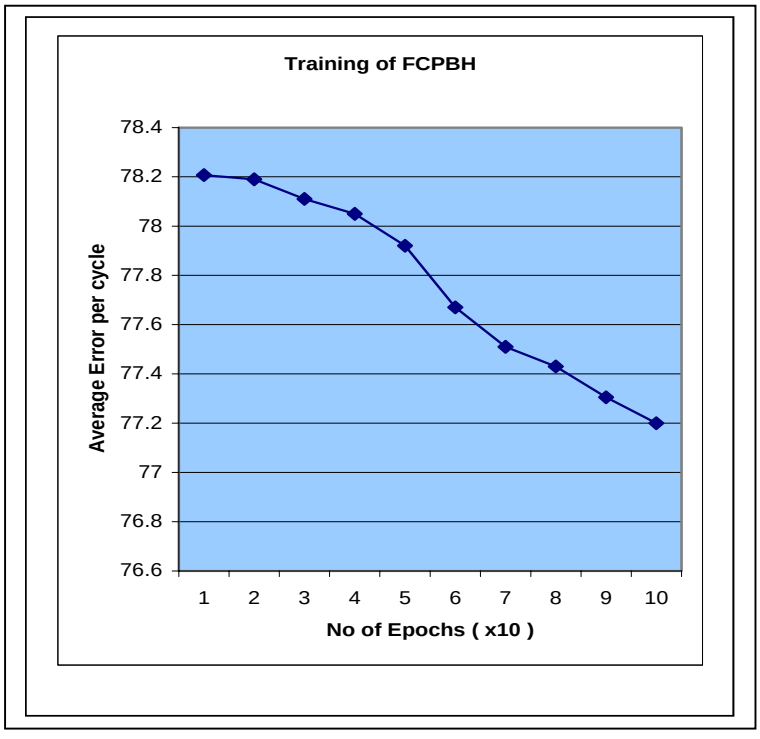
Testing

Neural Network	Back Propagation	PBH	FCPBH
Learning Rate	0.01	0.01	0.01
Average Error per Epoch	47.679	32.89	77.9
Error Last Epoch	47.47	32.5022	78.01
Error Last Cycle per Pattern	0.21	0.14	0.3
Total Cycles	100	100	100
Total Patterns	5000000	5000000	5000000
No of Layers	3	3	3
Neurons in Layer1:Layer2:Layer3	2:3:1	2:3:1	2:3:1

Table 5.7: NN Training with 5000000 patterns







The above tests how that the results of training of PBH are more superior to rest of the neural networks. FCPBH performance was really bad. The results of the neural networks classification on the test data after the above training are as follows

Neural Network	Back Propagation	PBH	FCPBH
Trained at Learning Rate	0.01	0.01	0.01
Correct Classification Rate	75%	81%	66.098%
False Positives rate	21.04%	10.9%	13.23%
False Negatives Rate	4%	8.1%	20.87%

Table 5.8: NN Training test results

Response, Attack Prelude Detection, Alert Logging

Due to the problems in counter measures against the intrusions we decided to implement an IP blocker program as a response module that can block specific IP on firewalls and routers. The design of the IP blocking mechanisms is marked by the following requirements

- Any IP or port should be block able on the firewall or router
- This Blockage should take place in minimum time.
- The IP Block list already present, on Firewall should not be changed.

Implementation

Due to the unavailability of the router we decided to convert a Linux machine into a router. Linux provides the blocking of port and IPs through its IPTable /IPChains Interface. This has the added advantage that many commercial routers and firewall use the IP Tables for blocking and unblocking of the traffic.

Attack Prelude Detection

Introduction

Almost every cyber attack today starts with some port scanning.

There are many port scan techniques like TCP connect, TCP half connect, Stealth scanning, Xmass tree scanning and NULL scanning all these techniques require TCP packets and these techniques need the RST packet in one way or the other to detect the port status as shown in the following table.

Technique	Flags Used	Reply (if listening port)	Reply (if port is closed)
TCP connect	SYN	SYN/ACK	RST/ACK
TCP half connect	SYN	SYN/ACK	RST/ACK
Stealth	FIN		RST/ACK
Xmass tree Scan	URG/PSH/FIN		RST/ACK
NULL scan	No Flag		RST/ACK

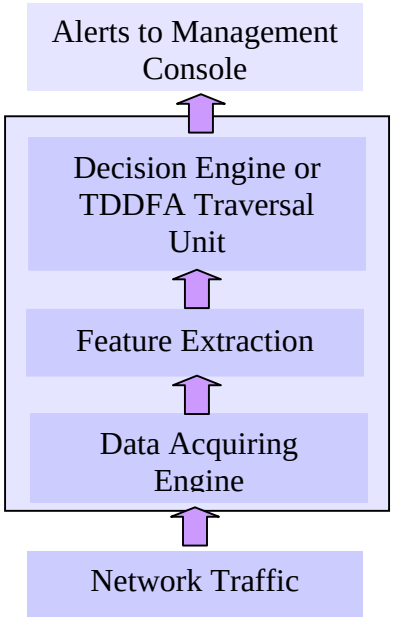
Table 5.10: Port Scan Techniques

Design

The port Scan attacks can't be detected by one single event. They are basically a series of events that leads to a final stage of decision whether port scanning is being done or not. Deterministic Finite Automata (DFA) also models the system as a series of states. So DFA can be used to model the port scanning attacks. But stealth port scans (port scans distributed over time) are difficult to detect with DFA based mechanisms because DFA does not incorporate time in its analysis. So Time Dependent Deterministic Finite Automata (TDDFA) were selected for Attack Prelude or Port Scan Detection in OMNI SECURE

Implementation

The APD was also developed as per Pipes and Filters Design Pattern. The conceptual model of APD is shown in the figure. Data Acquiring Engine sniffs the data from the network interface. From this network interface the network traffic extracted is given to the Feature extractor, which decodes the IP and TCP header and uses these network traffic parameters to extract the features, which are fed to a TDDFA. Each time the DFA is traversed the attack variable is incremented by one and when it reaches a specific value the alert is generated.



Testing

The system has gone under testing and has detected TCP connect, Half Connect, Stealth, Xmas and null scans quite successfully.

Alert Logging

The alerts are logged in MySQL database. In order to log alerts there MySQL's client server architecture was used. Although embedded server was also an option but client server architecture was used. There were two reasons to use client server architecture over embedded server architecture

- Client server architecture can be extended from one machine to 2 machines easily.
- Embedded server is needed for high speed data logging and in this case the speed was not important than the future extensibility that can be provided through client server architecture

Implementation

Currently alerts are logged in one table which has the following fields to log the alerts.

SrcIP: Source IP i.e. IP of the attacker
DstIP : Destination IP i.e IP of the victom is stored in this field
SrcPort : Attacker's port is stored in this field
DstPort :Attacked Port on which the attack is launched
CreationTime: This is the alert creation time
LastTime: This is the time till the alert is spanned
CurrentDateandTime: This is the date and time at which the alert is logged

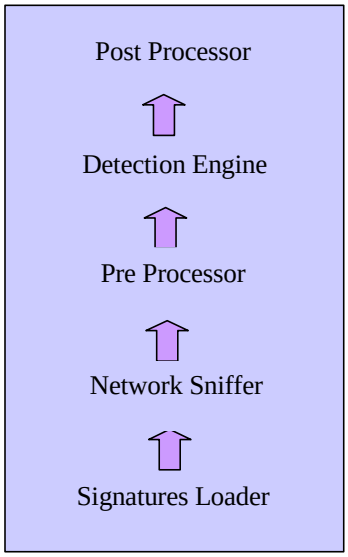
Testing

The alert logging module was tested to log the alerts and then the alerts were checked in the tables alert logging was going quite successful.

Misuse Detection System

Each day unveils new vulnerabilities in computer software used around the world. This is due to insecure programming practices and the inherent nature of computer software, that, it cannot be bug free. To cater for these attacks against critical services Misuse Detection system was conceived.

Design



The Pipes and Filter Design Pattern was again used in the implementation of the Misuse Detection system of OMNI SECURE. Firstly the signatures should be loaded then the network traffic should be captured for analysis. Then the signatures are covered into suitable binary format using the Pre Processor. Detection Engine detects the pattern in the coming traffic. Then the Post Processor sends the alert to the manager.

Implementation

The basic conceptual model of OMNI SECURE's Misuse Detection System is as follows. First of all the system has to have a signature database, which in this case is a file containing all the signatures as specified by the administrator.

After loading the signatures into main memory, the network interface to be monitored is initialized. If the packet parameters match that of a service node, then the content node pointed to by this service node is consulted for the pattern to be searched in the packet payload.

If the string matches the regular expression, or the pattern specified in the signature base, an alert is echoed on the screen, this alert is also sent to the management console, so that it can be reported to the administrator. The alerts can be logged to a file as well, so that the administrator can view them later. The post processor does the job of sending the alert to the management console and logging alerts to a file.

Critical File Protection Mechanism

Introduction

To strengthen the network security of enterprise each and every layer of enterprise must be secured. For this very reason modules for the protection of the critical servers of the enterprise have also been developed in OMNI SECURE

Design and Implementation

There are a lot of security solutions available for the protection of computers but even then Computer Intrusions do take place. The guiding requirements marking the requirements of OMNI SECURE's host based security modules (CFPM, Critical File Protection Mechanism).

- The technique used should not overburden the computer.
- The technique should detect (or try to detect) most of the intrusions

The optimal intrusion detection in the case of Host based Systems corresponds to tracking and analysis of each and every system call. But this is computationally very heavy task. Further doing so would cause unpleasant delay for the users of the computer.

Keeping these requirements in mind we decided to use the File Integrity Checking as a method of detecting Intrusions. It is a simple yet intuitive way of detecting breaches in data Integrity because the hackers try to modify some important system files.

Testing

Different types of tests were conducted to test CFPM. Firstly the CFPM was tested to check the final changes in the files. Second test was to check CFPM against the changes by the malicious software (viruses, Trojans etc). Both of these tests were successful.

Omni Secure Management

The OMNI SECURE whole implementation has been modeled under the Model View Controller Design Pattern. The View and the main Controller functionality lie on the System Management Console (Wired or Wireless).

. The implementation on wired Management Console was done by using QT library. Multithreaded sockets were used for the implementation of both the Wired Console and for wireless management station. For wireless system management design there were many design considerations. The most important one was whether to use J2ME or Personal Java. Among these Personal Java was chosen.

For development, client server design pattern was selected. In this architecture PDA acts as the server and management console has got the client, which informs PDA of the alerts generated. The reason behind client server design pattern is the future extensibility, which can be provided in the PDA through management console. The future extension of OMNI SECURE would include complete management of OMNI SECURE through PDA.

The OMNI SECURE whole implementation has been modeled under the Model View Controller Design Pattern. The View and the main Controller functionality lie on the System Management Console.

As the Management Console performs the main controller functionality in OMNI SECURE, hence the issues involved in the design of the Management Console were enormous.

Some of other Major design issues were

- System Initiating Process Design
- Station Monitoring Protocol Design
- Alert Response Loop Design

System Initiating Process Design

System Initiating Process Design was designed to be highly configurable and easily manageable. The system administrator should be easily able to start and stop different processes. For this processes different daemon processes keep on listening to commands of start and stop from the Manager.

The start command corresponds to the following format
Start: destination: destination station ip source: system management console pay load: “start”
Stop: destination: destination station ip source: system management console ip payload: “stop”

Station Monitoring Protocol

Station monitoring protocol is designed to keep the management console updated about the current status of the cluster stations. The status query operation initiated by user from user interface follows this protocol. Moreover the continuous monitoring of station parameters (like cpu load, memory usage and network traffic load) is performed.

Request	destination: station ip	source: console ip	payload: "query"
Reply:	destination: console ip	source: station_ip	payload: "status"

Status can have the values of started and stopped.
If manager is not connected with station daemon server then the response shown to the user is "disconnected".

Alert Response Loop

The Alert Response loop is designed for managing the alert reportage and alert response mechanisms according to the configurations done in initiation process. These mechanisms are automated as they are initiated whenever sensors raise intrusion alarm.

5.6.3 Implementation

As mentioned before PDA Personal Java is being used on the PDA side. Whenever Management Console gets an alert it connects through its PDA Client to the PDA server and send the attack alert on the PDA. The alert is sent from management console to PDA in a specific format described below:

Alert Num.	Alert Level	Alert Type	Attacker IP	Attacked IP

Alert Num: Alert Number of the attack, which is stored in the Database at Management Console.
Alert Level: This is basically the attack alert level decided by Anomaly Detection System and Misuse Detection System.
Alert Type: This alert type basically tells whether alert is from Anomaly Detection System or Misuse Detection System.
Attacker IP: address of the attacker, sent from the detection systems.
Attacked IP” address of the victim’s IP on which the attack was launched.
These alerts are formatted in a GUI at PDA.

To display animated statistics on PDA JCCKIT library was used. The main challenge was to use graphing methods compatible to JDK1.1 or below

Network Traffic Data Mining

An important part of Omni Secure yet to be implemented is the Network Traffic Data Mining Module. In this module we intend to develop infer important statistics from

network traffic. One such implication can be the decoding of the emails. This is very important because the terrorist organizations are reported to have been using Internet as their communication tool. Recent Madrid bombings are reported to have been a result of communication via Internet. We are going to start this module by the start of MAY.

Cost

An important characteristic of out product is that all the things used in the system do not contribute to the cost criteria set by the CSIDC.

Summary

Omni Secure as the name indicates is a product to safe guard all the networks. This name is there due to the fact that it can be deployed to safe guard the network of any critical infra structure. The scalability and flexibility of the systems make the system work at higher speed. Further the system by safeguarding the networks of critical services and infrastructures can really become a tool to counter Cyber Terrorism.