

## **5.1 Conclusions**

From this research, many things are noticed and concluded. The following are the most important ones:

1. The type of network messages that are relevant to the file system deal with port number equal 139 and 445. Both of these ports refers to SMB working over NetBIOS, and then NetBIOS over TCP/IP.
2. To gain high-speed detection, the network packets should be filtered. Filtering the network packets reduce the amount of incoming data to speeds the analysis stage time.
3. Disk's resources should be indexed to be capable apply structural searching technique (like, search sorted binary tree) to achieve high-speed searching performance.
4. Some of the headers' fields of headers of TCP/IP protocols are really meaningful to monitoring process. These fields are:
  - a. Source IP, Destination IP, and Protocol fields of IP protocol header.
  - b. Source Port number and Destination Port number of TCP protocol header.
  - c. Command and Flags fields of SMB protocol header.

In addition, there are unused fields for monitoring process, which are:

- a. Checksum field of IP and TCP protocol header.
- b. TTL field of IP protocol header.
- c. Window field of TCP protocol header.
- d. EXTRA field of SMB protocol header.

5. The power of the network intrusion detection system depends on which network segment is sniffed and, thereby, on what is the extracted information from this segment.
6. During system implementation, several problems have been raised, the main ones are:
  - a. Finding a proper mechanism to aggregate the local shared resources to use them in Initialization Module of the Client subsystem. To solve this problem, a Binary Tree structure was implemented as dynamic link list of records. Each record contains one shared resource and two links to related resources records.
  - b. How to distinguish between clients' users when they access a shared resource, in addition to PC's IP and PC user account name, if there is another user on the same PC and may use the same PC account. To solve this problem, the users were classified into three - classes and each class is classified into two or three - levels.
  - c. The type of the response of the administrator against suspicious users. There are several actions that could be taken against suspicious users, such as warning the user, blocking user's IP, ending user's network session, etc. the simplest one that is adopted in this thesis is using warning message.

## **5.2 Suggestions for future work**

In the following, some recommendations for future work in NNIDSs are listed:

1. Utilization of users profiles in detection engine.
2. Applying Misuse analyze strategy with FMS.
3. Extend this work to detect others file system actions (such as, list file attribute, browsing files and folders, print a file, and so on).
4. Use another response type instead of Warning Message, such as prevention the attacks by ending user network session, or blocking user's IP.
5. Running FMS on a Wide Area Network (WAN) or Metropolitan Area Network (MAN). Taken into considerations all the necessary scalable factors to establish such a system, examples: distributed databases, local and global Administrator subsystems and their administrators' privileges.

## 4.1 FMS Interfaces

FMS system interface consists of two interfaces: Administrator interface which is installed in the Administrator computer, and Client interface which is installed in the remote computers.

The Client subsystem is executed by FMS\_Client.exe file that must run in the remote computers to be used by any user aims to access the network shared resources. At the Administrator side, the FMS\_Administrator.exe is executed in the Administrator computer to start monitoring the entire network for suspicious actions that performed on files and folders.

### 4.1.1 FMS Client Subsystem Interfaces

FMS Client subsystem interface is designed to make its use is easy. The main functions of it are: user authentication to access the network shared resources, system initialization, and start monitoring specific LAN.

When FMS\_Client.exe is executed, *FMS Main* frame will appear as shown in Figure (4.1).



**Figure (4.1)** FMS Main frame

Main frame consists of:

1. **Setting** option: this option will be used by system manager to initialize the system's information.

2. **User Name** textbox: to allow user to enter his/her own username for login.
3. **Password** textbox: to allow user to enter his/her password.
4. **Login** button: an already registered user can login after entering his/her username and password.
5. **Register** button: a user who aims to register with new username can use the Register button.

### A. Initialization of FMS Client subsystem

When there is need to initialize the system, system manager should click on **Setting** option. System manager decides if there is need to initialize the FMS.

To set the system information, system manager is asked to enter a specific password to perform setting process. FMS will check the entered password, if it is valid then **FMS Setting** frame will appear to him/her as shown in Figure (4.2). Otherwise, FMS will display an error message to him/her to try again.

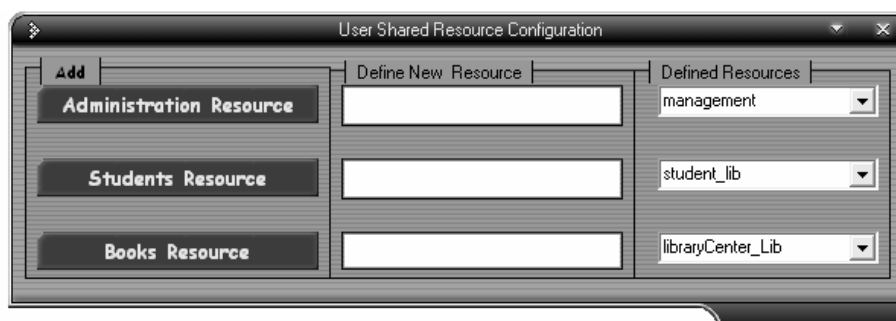


**Figure (4.2)** FMS Setting frame

FMS Setting frame consists of the following buttons:

1. **Detect Administrator** button: this button is used to detect the Administrator PC using broadcasting technique.
2. **Define User resources** button: this button is used to define the user's shared resources on the current host. Shared resources types depend on the applied environment. FMS is applied in a university environment. Therefore, the type of shared resources may be files and folders of students, employees, books, thesis, etc. Figure (4.3) illustrates the frame that will be displayed when Define User Resources button is clicked on.

This frame defines three types of shared resources, which are: *administration*, *students*, and *books*. Each type has its textbox to enter a new file/folder name with this type, button to add the new file/folder, and ComboBox that show which resource(s) had been defined.



**Figure (4.3)** User Shared Resources Configuration frame

3. **Configure ACL** button: when a click is made on this button, **Configure ACL** frame will appear, as shown in Figure (4.4).



**Figure (4.4)** Configure ACL frame

The above frame contains two buttons, *Send Shared List to Administrator*, and *Receive ACL from Administrator*. When **Send Shared List to Administrator** button is clicked on, the shared resources of the current host are gathered into a list and send to Administrator subsystem to define their permissions.

While clicking on **Receive ACL from Administrator** button, FMS will receive list of pre-defined permissions of the sent list of shared resources.

**4. Set Time Threshold Value** button: this button is used to set time threshold value for counting a number of user actions repetitions, which affects the Alarm level as discussed in Chapter 3 at section 3.3.1.1. When this button is clicked on, FMS will ask the system manager to enter number of minutes to set the time threshold. By default, FMS Client subsystem set  $t$  value to 15 minutes. Value of  $t$  range is chosen to be from one minute to 2 hours due to logical reasons, if entered value less than of 1 minute or greater than 120 minutes, an error message will appear to the manager.

**5. Receive Users' info list** button: this button is used to receive a list of information for all registered users from Administrator subsystem. The information are: PC's IP, PC name, PC account name, username, user type, and PC description. This list will be utilized to define each user to the system by his/her information.

## **B. Registering into FMS**

The following steps show how a new user can use FMS Client subsystem for registration:

**Step1:** click on a Register button in Main frame, then the *Register* frame will appear as shown in Figure (4.5). Then, the user must enter the following information:

1. **User Name**: represent client's user nickname that will be used for login, its length should be between 4 and 12.
2. **Password**: password to be used with the above username for login, its length should be between 6 and 14.
3. **Confirm Password**: retyping the above password to confirm it.
4. **Person Type**: select type of user from FMS list of user types. The types of users in FMS are: Visitor, Student, and Employee. Visitor is the user who isn't defined in the Registration database. Student is the user who is defined in Registration database by his/her name, study stage, and level. When user is Student type, **Full name** textbox will be enabled and user should enter student name as stored in Registration database. Employee is the user who is defined in Registration database by his/her name, level, and identification card number. When user is Employee type, **Full name** textbox should be filling, and **College ID** textbox will be enabled to enter his/her identification card number.



The image shows a screenshot of a software window titled "FMS Register Window". The window contains a registration form with the following fields and controls:

- User Name**: A text input field containing the text "nickname".
- Full name**: A disabled text input field.
- Password**: A text input field with masked characters "\*\*\*\*\*".
- Confirm Password**: A text input field with masked characters "\*\*\*\*\*".
- College ID**: A disabled text input field.
- Person Type**: A dropdown menu currently showing "Visitor".
- Submit**: A button located at the bottom of the form.

**Figure (4.5)** FMS Register frame

**Step2:** after entering the above information, the user should click on **Submit** button. If the entered information are accepted by Administrator



subsystem as new user, then success message will appear. Otherwise, an error message corresponding to the information error will appear and another try is made.

### C. Login into FMS

When **FMS Main** frame is appeared to the user as shown in Figure (4.1); the user should enter his/her username and password, and then click on Login button to let him/her view the shared resources. FMS Client subsystem will check the validity of these information, by sending them to FMS Administrator subsystem. If username and password are valid, he/she will be permitted to enter the system. Otherwise, an error message will appear to him/her to make another try.

### D. Controlling FMS

When user login succeeded, the user gain control the system and the *Main Control* frame will appear see Figure (4.6).



**Figure (4.6)** Main Control frame

Main Control frame contains three menus: *Tools*, *Monitoring*, and *Help*.

1. *Tools* menu: this menu contains the following options as shown in Figure (4.7).



**Figure (4.7)** Tools menu of Main Control frame

(a) **Reconfigure** option: when system manager clicks on Reconfigure option; FMS Client subsystem will ask him/her to enter a password. If the password is accepted, then the FMS Setting frame, as shown in Figure (4.2), will appear to him/her. The system manager will re-enter the new setting of the system information to be restarted again. If he/she chose to restart later, then **Monitoring** menu will be disabled to prevent FMS from using the invalid system information, as shown in Figure (4.8).



**Figure (4.8)** Main Control frame with disabled Monitoring menu

(b) **StartUp Program** option: this option is used to add/remove FMS Client subsystem to/from Windows OS start up program list, as shown in Figure (4.9).



**Figure (4.9)** Tools menu after add FMS to start up menu

(c) **Logout** option: this option is used to logout the user from the system and the Main frame is appeared again. When clicking on Logout option, FMS will ask user to re-enter his/her password. If the entered password incorrect corresponding to login username, an error message will appear.

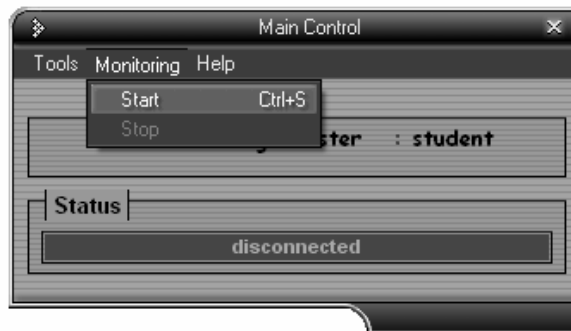
(d) **Hide** option: this option is used to hide the Main Control frame from the screen, and put it as small icon in the Start bar beside Windows clock, as shown in Figure (4.10). If user clicks on the above icon, the Main Control frame will maximized and again appears on the screen. The target of this job is to make the monitoring process in the background, and without confusing the PC's user.



**Figure (4.10)** FMS Client Subsystem icon

(e) **Exit** option: this option is used to end FMS execution. A click on Exit option will make FMS ask user to confirm end of program. Then FMS will ask user to enter his/her password as Logout from the system. If user enters his/her password, FMS will prompt user to wait until save all setting.

2. **Monitoring** menu: this menu is used to control the monitoring of current FMS Client subsystem on current host against suspicious actions. This menu has two options, **Start** and **Stop** options, as shown in Figure (4.11).



**Figure (4.11)** Monitoring menu of Main Control frame

When **Start** option is clicked on, FMS will try to connect itself to the current LAN in order to monitor it. User can stop the monitoring by clicking on **Stop** option, as shown in Figure (4.12). If FMS cannot connect to LAN, an error message will appear to the user to retry again.

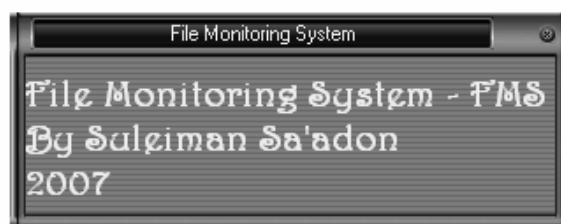


**Figure (4.12)** How to stop monitoring

**3. Help** menu: this menu has one option, which is **About**, as shown in Figure (4.13). Clicking on About option will display a message that demonstrate the current state of FMS, as shown in Figure (4.14).



**Figure (4.13)** Help menu of Main Control frame



### 4.1.2 FMS Administrator Subsystem Interfaces

FMS Administrator subsystem interface is designed to perform the following functions: preparing the FMS pre information, monitoring the entire network by FMS Client subsystems, respectively.

When FMS\_Administrator.exe is executed, *Administrator Access* frame will appear as shown in Figure (4.15).



**Figure (4.15)** Administrator Access frame

This frame consists of the following:

1. **Username** textbox: to allow Administrator to enter its own username for login.
2. **Password** textbox: to allow Administrator to enter the password corresponding to above username.
3. **Login** button: when Administrator enters its own username and password; Administrator should click on Login button to access the **Main Control** frame of FMS Administrator subsystem.

When Administrator enters username and password, and click on Login button, FMS Administrator subsystem will check if the entered username and password are both valid or not. If they are valid then the **Main Control** frame will appear as shown in Figure (4.16). Otherwise, an error message will appear for three times, then it terminates the execution.



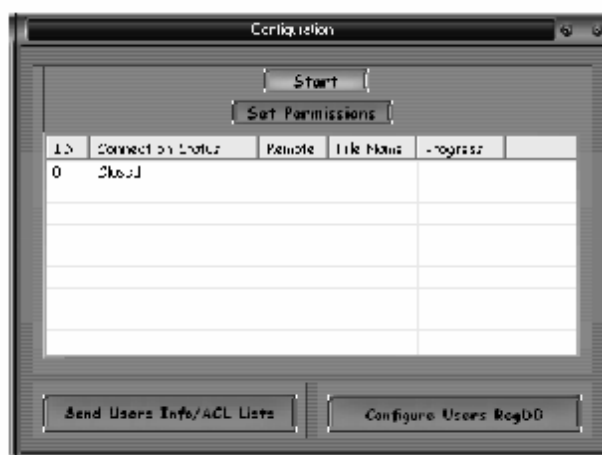
**Figure (4.16)** Administrator Main Control frame

### A. Configuration FMS Administrator subsystem

The following steps show how to configure the FMS Administrator subsystem:

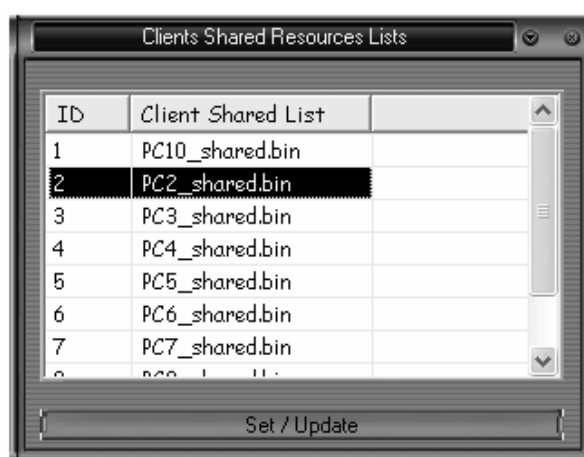
**Step1:** When the frame shown in Figure (4.16) is appeared, Administrator click on **Configuration** button then **Configuration** frame shown in Figure (4.17) will appear.

**Step2:** Receive a list of shared resources from each client that must be protected on the network by clicking on **Start** button.



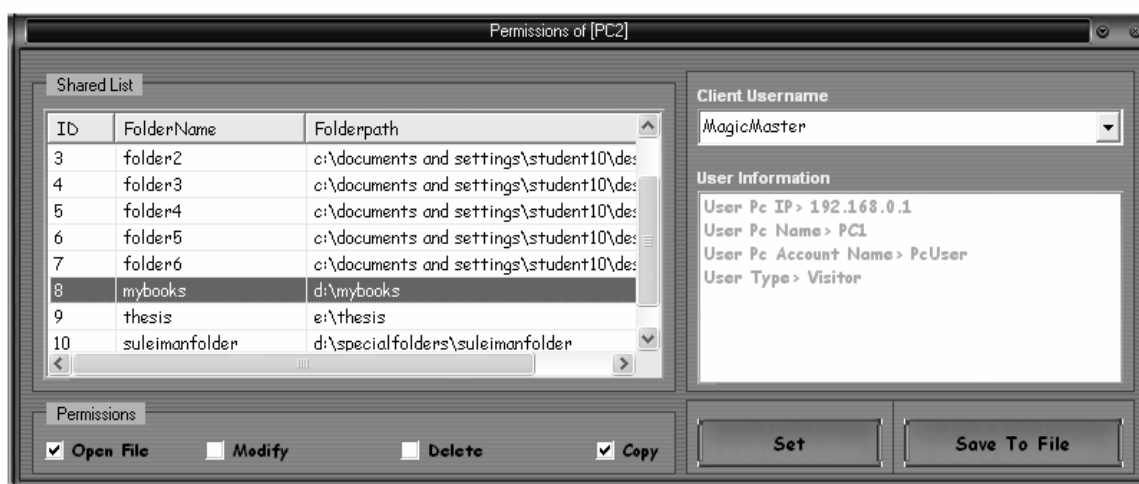
**Figure (4.17)** Administrator Configuration frame

**Step3:** Create the ACL (Access Control List) for each received shared resources list by clicking on *Set Permission* button. After, the click on Set Permission button, the frame shown in Figure (4.18) will appear. This frame contains list of the received shared resources lists. Each list named as [*client computer name\_*"shared.bin"] corresponds to the client that sent its shared resources list as a file.



**Figure (4.18)** Clients' shared resources lists frame

Administrator then can select one of the lists and then click on *Set/Update* button to set permissions or update if the permissions were set before. When Administrator clicks on Set/Update button, the frame shown in Figure (4.19) will appear.



**Figure (4.19)** Administrator set permissions frame

In the Administrator set permission frame, Administrator can set permissions for a specified shared resources list as explained in the following iterative steps:

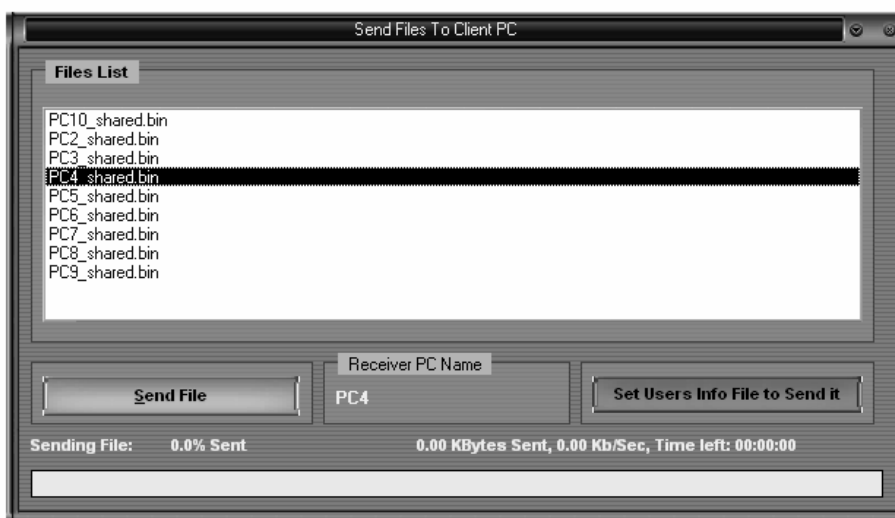
1. Select shared resource from Shared resources list by selecting an item from *Shared List*.
2. Select user from *Client Username* ComboBox.
3. Choose one or more of the Checkboxes that named *Open*, *Modify*, *Delete*, and *Copy* with respect to the types of permissions. The type of permission depends on the shared resource and user's information that appears on frame with *User information* label.
4. Click on *Set* button to add the permission of the selected shared resource to ACL. If the permission is already exists in ACL, then an error message will appear to avoid duplicate in ACL.
5. When ACL creation or updating is completed, Administrator should click on *Save to File* button to save the ACL into file to be send to the client who sent the shared list.

**Step4:** Send files to the FMS Client subsystems. These files represent the pre information of FMS Client subsystems. These files are ACL file and list of users' information (that gathered during users' registration and stored in "users\_info\_list.bin" file).

When *Send Users Info/ACL Lists* button, on the frame shown in Figure (4.17), is clicked on; the frame shown in Figure (4.20) will appear. This frame contains on *Files list*, *Send File* button, and *Set Users Info File to Send it* button. Initially, File list contains list of ACL files, then Administrator can select ACL and click on Send File button to send it to client who its PC name appear in *Receiver PC Name* label. When Administrator aims to send the of users information list to Client subsystem, Administrator should click on a button named *Set Users Info*



**File to Send it**, and then click on Send File button to send it to same receiver (Client subsystem).



**Figure (4.20)** Send Files to Client Subsystem frame

**Step5:** Configure users registrations database by clicking on **Configure Users RegDB** button on frame shown in Figure (4.17). When click on this button, frame **Configure RegDB** shown in Figure (4.21) will appear.



**Figure (4.21)** Configuration RegDB frame

To configure employees' information, click on **Set Employees** button, then **Employees Configuration** frame, as shown in Figure (4.22), will appear. This frame consists of the following:

- a. **Employee Full name** textbox: to allow Administrator to enter employee name.
- b. **Employee ID** textbox: to allow Administrator to enter employee identification card number.

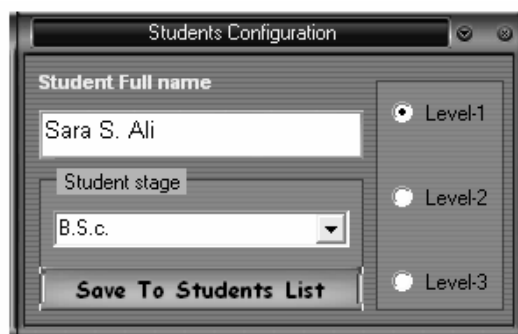
- c. **Employee Level** option: to define the level of the employee that is used to define the level of his/her permissions.
- d. **Save To Employee List** button: when clicking on this button, the entered employee name, employee ID, and employee level will be saved into employees information file.



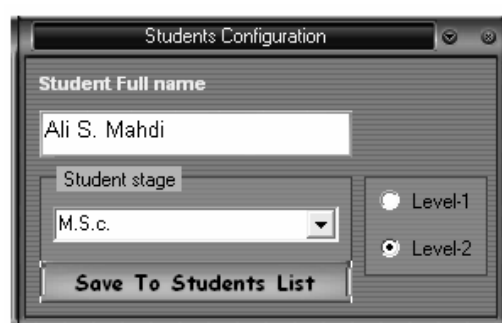
**Figure (4.22)** Employees Configuration frame

To configure students' information, click on **Set Students** button, then **Students Configuration** frame will appear. This frame consists of the following:

- a. **Student Full name** textbox: allow Administrator to enter student name.
- b. **Student stage** ComboBox: select the stage of student that could be either M.Sc. or B.Sc.
- c. **Level Option**: this option is depend on the student stage. If student stage is B.Sc., then he/she has three levels represented by classes, first class, second class, and third class as shown in Figure (4.23). While if student stage is M.Sc., then he/she has two levels; which are courses year and project year as shown in Figure (4.24).
- d. **Save To Student List** button: when this button is clicked on, the entered student name, student stage, and student level will be saved into students information file.



**Figure (4.23)** Students Configuration frame with B.S.c. student



**Figure (4.24)** Students Configuration frame with M.S.c. student

## **B. Monitoring the Events**

On the Main Control frame shown in Figure (4.16), when **Monitoring** button is clicked on, then the Monitoring frame is appeared as shown in Figure (4.25). This frame has four functions, they are:

1. Select the monitoring process mode either to be online or offline mode. When **Enable Monitoring** button is clicked on, each alarm message received from FMS Client subsystem will be displayed in the *Online Messages* list. Whereas, when clicking on **Disable Monitoring** button is done, each alarm message received will be stored temporary into offline messages file.
2. View offline messages that received in offline monitoring mode, by click on **View Offline Messages** button, *Offline Messages* frame will appear, as shown in Figure (4.26).

Monitoring

Enable Monitoring

Monitoring Mode : monitoring disabled.

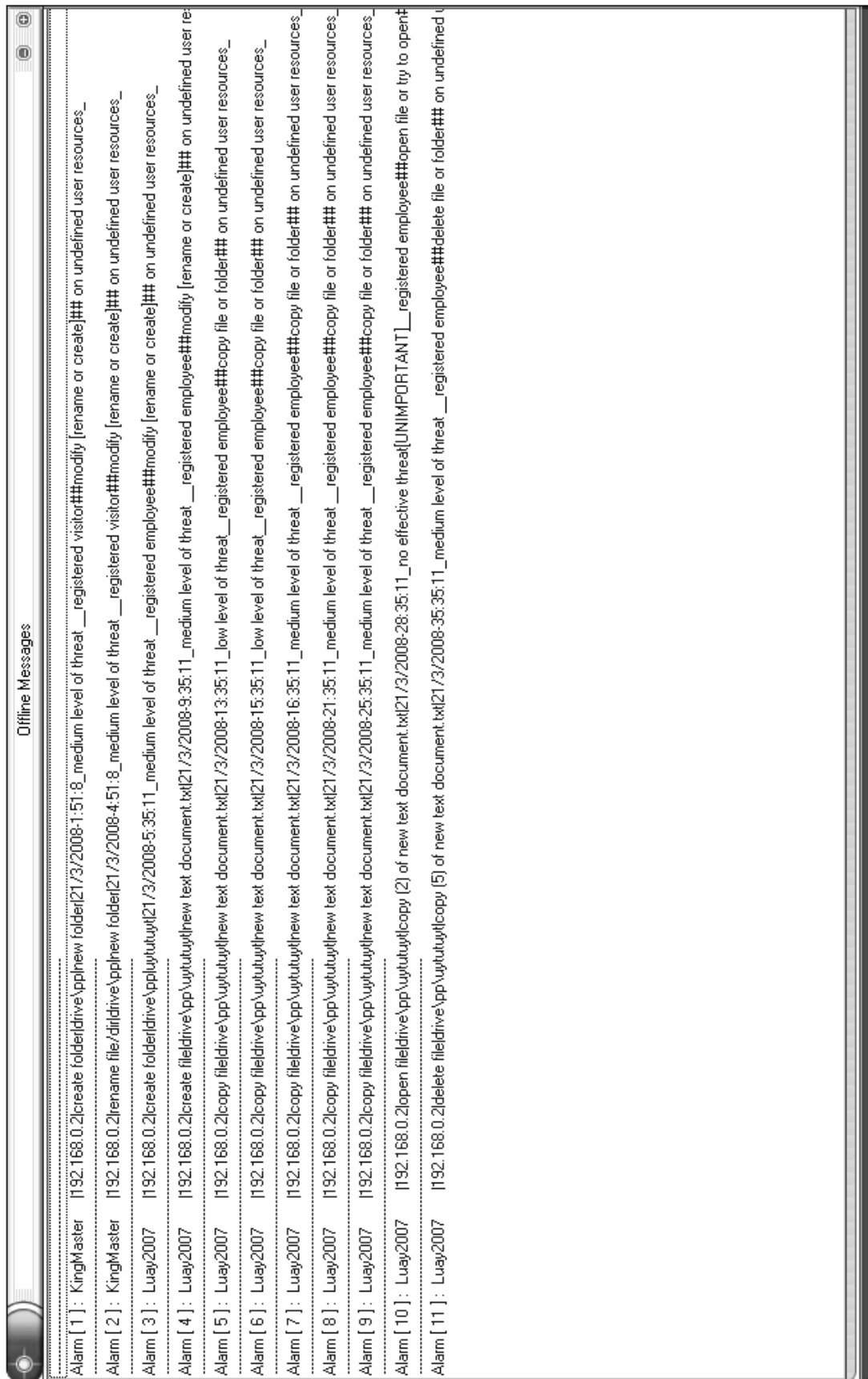
ID	Suspicious User	Victim P.c IP	Action	Path	File Name	Date / Time	Alarm Message	Notification Message
15	Luy2007	192.168.0.2	create file	D:\rootfolder\My F...	new microsoft word do...	21/3/2008-11:37:56	medium level of threat	registered employee###
16	<b>Luy2007</b>	<b>192.168.0.2</b>	<b>open file</b>	<b>D:\rootfolder\VM...</b>	<b>new microsoft word...</b>	<b>21/3/2008-11:38:8</b>	<b>no effective threat[UNIMPOR...</b>	<b>registered employee###</b>
17	Luy2007	192.168.0.2	delete file	D:\rootfolder\My F...	new microsoft word do...	21/3/2008-11:38:20	medium level of threat	registered employee###
18	Luy2007	192.168.0.2	write	2008\rootfolder\VM...	new text document.txt	22/3/2008-1:53:17	medium level of threat	registered employee###
19	<b>Luy2007</b>	<b>192.168.0.2</b>	<b>open file</b>	<b>2008\rootfolder...</b>	<b>new text document...</b>	<b>22/3/2008-1:53:37</b>	<b>no effective threat[UNIMPOR...</b>	<b>registered employee###</b>
20	Luy2007	192.168.0.2	open file	2008\rootfolder\VM...	new text document.txt	22/3/2008-1:53:47	low level of threat	registered employee###
21	Luy2007	192.168.0.2	open file	2008\rootfolder\VM...	new text document.txt	22/3/2008-1:53:49	low level of threat	registered employee###
22	Luy2007	192.168.0.2	open file	2008\rootfolder\VM...	new text document.txt	22/3/2008-1:54:2	medium level of threat	registered employee###
23	Luy2007	192.168.0.2	open file	2008\rootfolder\VM...	new text document.txt	22/3/2008-1:54:57	medium level of threat	registered employee###
24	Luy2007	192.168.0.2	open file	2008\rootfolder\VM...	new text document.txt	22/3/2008-1:55:16	medium level of threat	registered employee###
25	Luy2007	192.168.0.2	open file	2008\rootfolder\VM...	new text document.txt	22/3/2008-1:55:30	high level threat	registered employee###
26	Luy2007	192.168.0.2	open file	2008\rootfolder\VM...	new text document.txt	22/3/2008-1:55:42	high level threat	registered employee###
27	<b>Luy2007</b>	<b>192.168.0.2</b>	<b>open file</b>	<b>2008\rootfolder...</b>	<b>new text document...</b>	<b>22/3/2008-1:55:57</b>	<b>very high level threat[RISK]</b>	<b>registered employee###</b>
28	<b>Luy2007</b>	<b>192.168.0.2</b>	<b>open file</b>	<b>2008\rootfolder...</b>	<b>new text document...</b>	<b>22/3/2008-1:56:9</b>	<b>very high level threat[RISK]</b>	<b>registered employee###</b>
29	Luy2007	192.168.0.2	rename file/dir	\rootfolder\My Fol...	oipol	22/3/2008-1:59:43	medium level of threat	registered employee###
30	Luy2007	192.168.0.2	create file	D:\rootfolder\My F...	new text document.txt	22/3/2008-1:59:56	medium level of threat	registered employee###
31	<b>Luy2007</b>	<b>192.168.0.2</b>	<b>open file</b>	<b>D:\rootfolder\VM...</b>	<b>new text document...</b>	<b>22/3/2008-2:0:9</b>	<b>no effective threat[UNIMPOR...</b>	<b>registered employee###</b>
32	Luy2007	192.168.0.2	open file	D:\rootfolder\My F...	new text document.txt	22/3/2008-2:0:15	low level of threat	registered employee###
33	Luy2007	192.168.0.2	open file	D:\rootfolder\My F...	new text document.txt	22/3/2008-2:1:4	low level of threat	registered employee###
34	Luy2007	192.168.0.2	open file	D:\rootfolder\My F...	new text document.txt	22/3/2008-2:1:25	medium level of threat	registered employee###
35	Luy2007	192.168.0.2	open file	D:\rootfolder\My F...	new text document.txt	22/3/2008-2:1:33	medium level of threat	registered employee###
36	Luy2007	192.168.0.2	open file	D:\rootfolder\My F...	new text document.txt	22/3/2008-2:1:59	medium level of threat	registered employee###
37	Luy2007	192.168.0.2	open file	D:\rootfolder\My F...	new text document.txt	22/3/2008-2:2:6	high level threat	registered employee###
38	Luy2007	192.168.0.2	open file	D:\rootfolder\My F...	new text document.txt	22/3/2008-2:2:13	high level threat	registered employee###
39	<b>Luy2007</b>	<b>192.168.0.2</b>	<b>open file</b>	<b>D:\rootfolder\VM...</b>	<b>new text document...</b>	<b>22/3/2008-2:2:39</b>	<b>very high level threat[RISK]</b>	<b>registered employee###</b>

View Offline Messages

View Users Profiles

Response to User

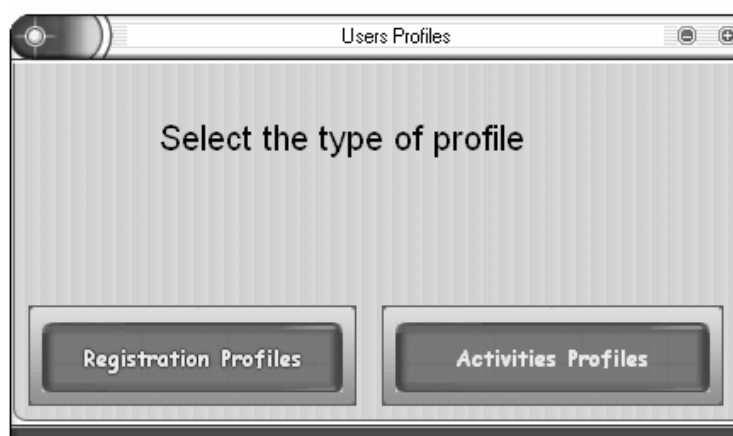
Figure(4.25) Administrator Monitoring frame



Figure(4.26) Offline Messages frame

3. View the users' profiles corresponding to users registrations processes and FMS Clients alarm messages. Users' profiles is divided into two parts, users information profiles and users activities.

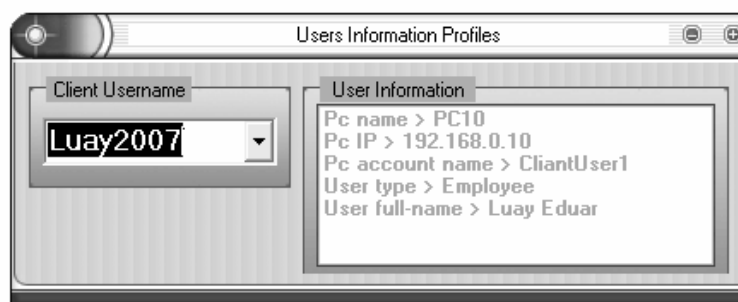
When click on **View Users Profiles** button on Monitoring frame, **Users Profiles** frame will appear, as shown in Figure (4.27). This frame has two buttons, **Registration Profiles** and **Activities Profiles**.



**Figure (4.27)** Users Profiles frame

When click on **Registration Profiles** button, the **Users Information Profiles** frame, shown in Figure (4.28), will appear. This frame consists of:

- a. **Client Username** ComboBox: to allow Administrator to select Client's username.
- b. **User Information** list: to display short information about selected user.



**Figure (4.28)** Users Registration Profiles frame

When click on **Activities Profiles** button, *Users Activities* frame, as shown in Figure (4.29), will appear. This frame contains a list of user's activities in details, and user threat level that computed according to each alarm message with high threat level.

The screenshot shows a window titled 'Users Activities'. At the top, there is a 'Select User' dropdown menu with 'Luay2007' selected, a 'Search' button, and a 'User Threat Level' field showing '14'. Below this is a table with the following columns: ID, Action, Victim P..., File Name, Threat Level, and Date / Time. The table contains 29 rows of activity data.

ID	Action	Victim P...	File Name	Threat Level	Date / Time
1	delete file	192.168...	1.txt	medium level of threat	21/3/2008-11:34:38
2	delete file	192.168...	copy of 1.bmp	medium level of threat	21/3/2008-11:34:39
3	delete file	192.168...	copy (6) of 1.bmp	medium level of threat	21/3/2008-11:34:39
4	delete file	192.168...	copy (8) of 1.bmp	medium level of threat	21/3/2008-11:34:39
5	delete file	192.168...	copy (16) of 1.rar	medium level of threat	21/3/2008-11:34:39
6	create folder	192.168...	My Folder	medium level of threat	21/3/2008-11:35:5
7	create file	192.168...	new text document.txt	medium level of threat	21/3/2008-11:35:9
8	copy file	192.168...	new text document.txt	low level of threat	21/3/2008-11:35:13
9	copy file	192.168...	new text document.txt	low level of threat	21/3/2008-11:35:15
10	copy file	192.168...	new text document.txt	medium level of threat	21/3/2008-11:35:16
11	copy file	192.168...	new text document.txt	medium level of threat	21/3/2008-11:35:21
12	copy file	192.168...	new text document.txt	medium level of threat	21/3/2008-11:35:25
13	open file	192.168...	copy (2) of new text docu...	no effective threat[UNIMPOR...	21/3/2008-11:35:28
14	delete file	192.168...	copy (5) of new text docu...	medium level of threat	21/3/2008-11:35:35
15	create file	192.168...	new microsoft word docu...	medium level of threat	21/3/2008-11:37:56
16	open file	192.168...	new microsoft word docu...	no effective threat[UNIMPOR...	21/3/2008-11:38:8
17	delete file	192.168...	new microsoft word docu...	medium level of threat	21/3/2008-11:38:20
18	write	192.168...	new text document.txt	medium level of threat	22/3/2008-1:53:17
19	open file	192.168...	new text document.txt	no effective threat[UNIMPOR...	22/3/2008-1:53:37
20	open file	192.168...	new text document.txt	low level of threat	22/3/2008-1:53:47
21	open file	192.168...	new text document.txt	low level of threat	22/3/2008-1:53:49
22	open file	192.168...	new text document.txt	medium level of threat	22/3/2008-1:54:2
23	open file	192.168...	new text document.txt	medium level of threat	22/3/2008-1:54:57
24	open file	192.168...	new text document.txt	medium level of threat	22/3/2008-1:55:16
25	open file	192.168...	new text document.txt	high level threat	22/3/2008-1:55:30
26	open file	192.168...	new text document.txt	high level threat	22/3/2008-1:55:42
27	open file	192.168...	new text document.txt	very high level threat[RISK]	22/3/2008-1:55:57
28	open file	192.168...	new text document.txt	very high level threat[RISK]	22/3/2008-1:56:9
29	rename file/dir	192.168...	oipoi	medium level of threat	22/3/2008-1:59:43

Figure (4.29) Users Activities frame

### C. Response to Monitoring

To response to the monitoring, a click on *Response to User* button is made on the Monitoring frame, then *Response* frame will appear, as shown in Figure (4.30).

The Administrator response is a warning message, which is send to the suspicious user. Response frame consists of the following:

1. **User Name** ComboBox: to allow Administrator to select one of the suspicious users to warn it.
2. **User Information** list: to display short information about selected user.
3. **Message** textbox: to allow Administrator to enter suitable warning message to be sent to the suspicious user.
4. **Clear** button: to clear the Message textbox.
5. **Send** button: when administrator click on Send button, FMS Administrator subsystem will send the written message (in the Message textbox) to the selected user (which is assigned by the User Name ComboBox).

Sent message will be received by FMS Client subsystem to warn the user for its suspicious action(s).



**Figure (4.30)** Administrator Response frame

When Administrator aim to terminate the run of FMS Administrator subsystem, FMS will ask Administrator to enter his/her password, to verify Administrator identity. If the password is valid then FMS execution is terminated. Otherwise, an error message appears.



## 4.2 FMS Evaluation

The evaluation of FMS has three perspectives; *Accuracy*, *Time Consuming*, and *Memory Consuming*. The accuracy evaluation of such intrusion detection system depends on two factors; *false positive* and *false negative* measures. While the time consumption of FMS is computed according to the execution time of system initialization stage and detection stage. The initialization time is spent for once to construct ACL by Client subsystem, while detection time is spent to search ACL each time by Client subsystem to find the permission of an accessed object. Memory consumption is the size of memory, which is used by FMS to construct the ACL into binary tree.

### 4.2.1 Accuracy

The accuracy of FMS will be assessed using the measures: the false positive and false negative values. Both measures depend on: (1) number of detected attacks, (2) number of alerts that generated by Client subsystem. To measure the accuracy of FMS, a simulation was conducted by applying different types of file attacks on the network resources and the number of false positive and false negative alarms were computed.

In addition to false positive and false negative, during the tests, a score is given to each detected threat.

The types of the applied file attacks are those associated with the file access types: **Open**, **Modify** (i.e. write, rename or create objects), **Delete**, and **Copy**. These file access types lead to nine types of file application services: *open file*, *write to file*, *rename file*, *rename folder*, *create file*, *create folder*, *delete file*, *delete folder*, and *copy file*.

The choice of test case was depended on the following criteria:

- (1) Number of occurrences of each event.

- (2) Number of executed action for specific subject.
- (3) Variety of user authentication (i.e. allowed or denied).
- (4) Type of object classes.
- (5) Time of event occurrence.

In the test cases, the authorized users will be referred as **Auth**, and the unauthorized users will be referred as **Unauth**.

**Test Case One:**

Each of the four types of actions have been applied for ten times, by various cases of user authentication, user class, and object class, to compute the false positive and false negative values.

Table (4.1) shows the false positive values for test case one, and Table (4.2) shows the false negative values for the same case.

**Table 4.1** False Positive values of the Test Case one

User Class	Action Type	System Class		Application Class		User Class	
		Auth	Unauth	Auth	Unauth	Auth	Unauth
Student	Open	1	0	2	0	1	0
	Modify	0	0	0	0	1	0
	Delete	1	0	0	0	0	0
	Copy	0	0	0	0	2	0
Employee	Open	0	0	1	0	0	0
	Modify	1	0	2	0	1	0
	Delete	1	0	0	0	0	0
	Copy	3	0	0	0	2	0
Visitor	Open	2	0	0	0	2	0
	Modify	1	0	0	0	0	0
	Delete	1	0	0	0	0	0
	Copy	0	0	0	0	1	0

From Table (4.1) several points were noticed, which are:

1. The cases of attacks that are sent by unauthorized users didn't present any false positive values because false positive depends on the number of authorized accesses which are described as attacks.
2. When the class of the object is **System class** and the type of action are either *Modify* or *Delete* then some false positive hits have been registered because detection rule decision is made according to the number of the occurrences of those actions by the same user.
3. Other cases of false positives alarms are obtained when *Open* and *Copy* actions are performed for all resources classes. Also, false positive alarms are obtained when *Modify* and *Delete* actions are performed for **User** and **Application** classes. Those cases are registered because the complete path of the accessed object isn't received by detection engine correctly due to the network traffic.

**Table (4.2)** False Negative values of the Test Case one

User Class	Action Type	System Class		Application Class		User Class	
		Auth	Unauth	Auth	Unauth	Auth	Unauth
Student	Open	0	0	0	1	0	0
	Modify	0	1	0	0	0	0
	Delete	0	0	0	0	0	0
	Copy	0	0	0	0	0	2
Employee	Open	0	0	0	0	0	0
	Modify	0	1	0	0	0	0
	Delete	0	0	0	0	0	0
	Copy	0	2	0	2	0	1
Visitor	Open	0	0	0	1	0	0
	Modify	0	0	0	0	0	0
	Delete	0	0	0	0	0	0
	Copy	0	2	0	0	0	1

From Table (4.2) several points have been noticed, they are:

1. The cases of attacks which are sent by authorized users haven't caused any false negative alarm, because false negative depends on the number of unauthorized accesses (attacks) which are undetected.
2. There are few attacks cases that FMS may fail to detect them. These cases may occur due to one of the following reasons:
  - a. The Client subsystem didn't generate events for these actions,
  - b. The Administrator subsystem didn't receive the alarm messages under heavy traffic, or
  - c. The rules decided that these actions are authorized with a particular permission because the rules follow a target path similar to the path of other target in ACL.

All of the above-mentioned reasons due to network traffic that may not pass the occurred event or the object path isn't determined correctly.

As result, the accuracy of FMS is measured in percent as follow:

1. The number of executed actions in Table (4.1) are 360 actions, they imply 26 cases of false positive. Therefore, the accuracy of FMS in the term of false positive is %92.7.
2. The number of executed actions in Table (4.2) are 360 actions, they imply 14 cases of false negative. Therefore, the accuracy of FMS in the term of false negative is %96.1.

**Test Case Two:**

The four types of actions have been applied, ten times for each one, different cases of user authentication, user class, and object class, at each test case the alarm level value was determined.

Table (4.3) shows the alarm level values for Student class. Table (4.4) shows the alarm level values for Employee class. And Table (4.5) shows the alarm level values for Visitor class. Each arrow symbol (i.e. →) represents the increase of the alarm level for its primitive value. The bold numbers represent the false positive cases.

**Table 4.3** Alarm level values for *Student* user class

	System Class		Application Class		User Class	
	Auth	Unauth	Auth	Unauth	Auth	Unauth
Open	<b>3</b>	3→6	0	2→5	0	1→2
Modify	0→5	9→9	0	8→9	<b>7</b>	7→9
Delete	0→5	9→9	<b>8</b>	8→9	0	7→9
Copy	0	3→6	0	2→5	0	1→2

**Table 4.4** Alarm level values for *Employee* user class

	System Class		Application Class		User Class	
	Auth	Unauth	Auth	Unauth	Auth	Unauth
Open	0	3→6	0	2→5	0	1→2
Modify	0→5	9→9	<b>8</b>	8→9	<b>8</b>	7→9
Delete	0→5	9→9	0	8→9	0	7→9
Copy	0	3→6	0	2→5	0	1→2

**Table 4.5** Alarm level values for *Visitor* user class

	System Class		Application Class		User Class	
	Auth	Unauth	Auth	Unauth	Auth	Unauth
Open	3	3→6	0	3→6	0	2→4
Modify	0→5	9→9	0	8→9	0	7→9
Delete	0→5	9→9	0	8→9	7	7→9
Copy	0	3→6	0	3→6	0	2→4

The discussion of results of test case two can be summarized as follows:

1. The attack cases that sent by authorized user, except Modify and Delete on system resources, don't present threat according to rules. Nevertheless, some cases have been presented as alarms and these cases are false positive cases. The generated alarm for the false positive case is equal to the primitive value because the error in detection didn't occurred again in the next occurrence and the rule decided these actions as authorized actions.
2. According to rules the cases of Modify and Delete actions on system resources class, which are sent by all types of authorized users, were presented by alarms when these actions occurred more than one times only.
3. Alarm level value, which is generated, begin with the primitive value according to Table (3.2) and then increased according to Algorithm (3.19).
4. According to Table (3.2), the primitive values of the alarm level for Student user class are similar to alarm level values for Employee user class in cases of unauthorized users only. Therefore, one can notice this similarity in Tables (4.3) and (4.4).

## 4.2.2 Time and Memory Consumption

The efficiency of the monitoring systems is evaluated primarily by the Accuracy, but the Accuracy is often depends on the consumed time to execute some stages in the systems.

In FMS, there is a time required to construct the ACL during the initialization stage, and additional time is required to search the ACL for permission of an accessed object during the detection stage.

The above consumed time depends on the number of shared files and folders which are indexed into the ACL (i.e. number of nodes in the ACL).

In addition to the consumed time, the used memory to construct the ACL is an important efficiency factor it depends on the number of shared files and folders.

Table (4.6) presents the performance of the system in terms of time and memory consumption for certain cases of the numbers of shared files and folders. Memory consumption is the size of used memory to load the ACL as a link list into the computer RAM.

Both, the consumed time and memory depend on some of the H/W specifications of the computer that runs FMS. FMS was installed in computer which has the following specification according to CPU and RAM:

1. CPU – 1.7 GHz.
2. RAM – 512 MB.

Table (4.6) shows the Time and Memory consumption for different cases of numbers of shared resources.

**Table 4.6** Time and Memory consumption

<b>Size of shared resources (Files/Folders)</b>	<b>Number of shared resources (Files/Folders)</b>	<b>Time to construct ACL</b>	<b>Time to search ACL</b>	<b>Size of used RAM</b>
Measured in GB		Measured in second		Measured in MB
≈ 4.21	8,579	16	0.009	4.936
≈ 7.69	13,317	32	0.011	7.352
≈ 11.62	19,855	42	0.015	13.328
≈ 48.10	21,900	42	0.021	16.188
≈ 10.72	41,756	94	0.081	31.348
≈ 39.09	62,254	118	0.200	59.104
≈ 32.01	78,481	155	0.406	82.413
≈ 89.77	140,735	300	0.912	112.932

The discussion of results can be summarized as follows:

1. The numbers of shared resources depends on the contents of the shared resources of students, employees and visitors, which turn depend on applied environment. For example, the number 41756 represents files and folders all have size around 10 Gigabytes. As for the largest case (i.e. 140735 files and folders), it represents the greatest number of the test cases, the files and folders size is about 90 Gigabytes. The size of shared resources is ineffective on the size of ACL; because ACL deals with object's name only.
2. The consumed time for constructing the ACL in some cases may be too long (e.g. 300 seconds ≈ 5 minutes), but this time will be spend for once, only during initialization stage, and hence it is reasonable time for a system tries to index shared resources size about 90 Gigabytes or more.



3. The consumed time to search ACL is short time when the number of objects is not relatively big. And this is the advantage of using the Binary Tree structure to index the shared resources.
4. The size of used memory in some cases may be considered big, but this is, today, not a problem with available computer's memory size is more than 512 MB.

## 1.1 Introduction

In the last decade, networks have grown in both size and importance. In particular TCP/IP networks, and most notably the world-wide Internet, have become the main means to exchange data and carry out transactions. They have also become the main means to attack hosts.

The growing importance of the Internet and of network security issues has shifted security concerns from the operating system domain to the network itself. Security efforts are experiencing a similar shift. Local, centralized approaches are evolving into distributed and networked approaches, in an effort to cope with the interconnected heterogeneous platforms and to provide scalable solutions.

*Intrusion detection* (ID) is an approach to security that is also evolving towards networked environments. Intrusion detection began as a technique for detecting masqueraders and misfeasors in standalone systems, but in the last few years the focus of intrusion detection has moved towards network [Vig97].

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of *intrusions*, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network. Intrusions are caused by attackers accessing the systems from the network, authorized users of the systems who attempt to gain additional privileges for which they are not authorized, and authorized users who misuse the privileges given them. *Intrusion Detection System* (IDS) is a software or hardware product that automates this monitoring and analysis process [Bac00].

Intrusion detection system allows organizations to protect their systems from the threats that come with increasing network connectivity

and reliance on information systems. Given the level and nature of modern network security threats, the question for security professionals should not be *whether* to use intrusion detection, but *which* intrusion detection features and capabilities to use. There are several compelling reasons to acquire and use IDSs [Bac00]:

1. To detect attacks and other security violations those are not prevented by other security measures.
2. To act as quality control for security design and administration, especially of large and complex enterprises.
3. To provide useful information about intrusions that do take place by documents the existing threat to an organization, allowing improved diagnosis, recovery, and correction of causative factors.

## 1.2 Taxonomy of Intrusion Detection Systems

Intrusion Detection Systems can be classified into three categories with respect to: 1) Where data is collected; 2) Where and how data is processed; and 3) The way that analyze the collected data (analyze strategy) [Spa00].

In the following sections, these categories will be explained.

### 1.2.1 Data Collection Mechanisms

Intrusion Detection Systems can be divided into two main types according to way data is collected: *Host-Based Intrusion Detection System* and *Network-Based Intrusion Detection System* [Spa00].

#### A. Host-Based Intrusion Detection Systems

Host-based IDSs (HIDSs) operate on information collected from within an individual computer system. This vantage point allows host-based IDSs to analyze activities with great reliability and precision,

determining exactly which processes and users are involved in a particular attack on the operating system. Furthermore, unlike network-based IDSs, host-based IDSs can “see” the outcome of an attempted attack, as they can directly access and monitor the data files and system processes usually targeted by attacks.

Host-based IDSs normally utilize information sources of two types, operating system audit trails, and system logs. Operating system audit trails are usually generated at the innermost (kernel) level of the operating system, and are therefore more detailed and better protected than system logs. However, system logs are much less obtuse and much smaller than audit trails, and are furthermore far easier to comprehend. Some host-based IDSs are designed to support a centralized IDS management and reporting infrastructure that can allow a single management console to track many hosts. Others generate messages in formats that are compatible with network management systems [Bac00].

Host-based intrusion detection systems have many advantages, including [Axe98]:

1. Host-based IDSs, with their ability to monitor events local to a host, can detect attacks that cannot be seen by a network-based IDS.
2. Host-based IDSs can often operate in an environment in which network traffic is encrypted, when the host-based information sources are generated before data is encrypted and/or after the data is decrypted at the destination host
3. Host-based IDSs are unaffected by switched networks.
4. When Host-based IDSs operate on OS audit trails, they can help to detect Trojan Horse or other attacks that involve software integrity breaches. These appear as inconsistencies in process execution.

While the disadvantages of host-based intrusion detection systems are [Axe98]:

1. Host-based IDSs are harder to manage, as information must be configured and managed for every host monitored.
2. Since at least the information sources (and sometimes part of the analysis engines) for host-based IDSs reside on the host targeted by attacks, the IDS may be attacked and disabled as part of the attack (denial-of-service attacks).
3. Host-based IDSs are not well suited for detecting network scans or other such surveillance that targets an entire network, because the IDS only sees those network packets received by its host.
4. When host-based IDSs use operating system audit trails as an information source, the amount of information can be immense, requiring additional local storage on the system.
5. Host-based IDSs use the computing resources of the hosts they are monitoring, therefore inflicting a performance cost on the monitored systems.

## **B. Network-Based Intrusion Detection Systems (NNIDS)**

NIDS deals with the traffic that is passed over the network between hosts. A NIDS "sniffs" packets from off the network and compares the traffic to signature of known attacks or normal behaviour depends on analyses method. NIDS designed to be able to look at the information that is in the packets and keep their focus on what traffic is traveling over the wire on a subnet.

Network-based IDSs often consist of a set of single-purpose sensors placed at various points in a network. These units monitor network traffic,

performing local analysis of that traffic and reporting attacks to a central management console [Isa04].

Advantages of network-based IDSs are [Lyd04]:

1. A few well-placed network-based IDSs can monitor a large network.
2. The deployment of network-based IDSs has little impact upon an existing network. Network-based IDSs are usually passive devices that listen on a network wire without interfering with the normal operation of a network. Thus, it is usually easy to retrofit a network to include network-based IDSs with minimal effort.
3. Network-based IDSs can be made very secure against attack and even made invisible to many attackers.

The disadvantages of network-based IDSs are [Lyd04]:

1. Network-based IDSs may have difficulty processing all packets in a large or busy network and, therefore, may fail to recognize an attack launched during periods of high traffic.
2. Many of the advantages of network-based IDSs don't apply to more modern switch-based networks. Switches subdivide networks into many small segments (usually one fast Ethernet wire per host) and provide dedicated links between hosts serviced by the same switch. Most switches do not provide universal monitoring ports and this limits the monitoring range of a network-based IDS sensor to a single host. Even when switches provide such monitoring ports, often the single port cannot mirror all traffic traversing the switch.
3. Network-based IDSs cannot analyze encrypted information. This problem is increasing as more organizations (and attackers) use Virtual Private Networks (VPNs).
4. Most network-based IDSs cannot tell whether or not an attack was successful; they can only discern that an attack was initiated. This

means that after a network-based IDS detects an attack, administrators must manually investigate each attacked host to determine whether it was indeed penetrated.

5. Some network-based IDSs have problems dealing with network-based attacks that involve fragmenting packets. These malformed packets cause the IDSs to become unstable and crash.

### **C. Hybrid Intrusion Detection System**

Hybrid Intrusion Detection is the fusion of both HIDS and NIDS so that they both work together by combine the ability of NIDS to detect an attack and then correlate that with the HIDS that will tell if the security has been breached. Hybrid intrusion detection systems combine the goods (and some the bads) of both HIDS and NIDS [Roe02].

There are some disadvantages though in using hybrid IDS:

1. This is an infant technology and some commercial products could be expensive.
2. They can only monitor one host just like the HIDS.

### **D. Network-Node Intrusion Detection Systems**

Network-Node Intrusion Detection System performs a task similar to traditional NIDS by monitoring network packets for signs of misuse. However, NNIDS another approach to NIDS by singling out the data that is coming into a single host and analyzing it. NNID software is installed on each computer to be protected. This software does not watch all traffic sent through the network; rather, it only monitors packets sent to and from the particular computer on which it reside. This eliminates the problems that network intrusion detection traditionally has had by collecting and analyzing the packets at their final destination. This allows the system to analyze the traffic at its destination but it take the focus away from the

traffic that is going to over the subnet allowing the detector to see specifically where the attack was going [Isa04].

This system resolves the three major problems mentioned previously in NIDS, which are [Smi02]:

1. Since the NNID software resides on each computer, it has access to unencrypted network traffic and thus is not blinded by VPNs and other forms of network encryption.
2. NNIDS eliminates throughput concerns because it only monitors traffic sent to and from one computer.
3. Since the NNID software is specifically installed on those computers designated to be protected, switched networks and other variations of network topologies no longer provide barriers through which the intrusion detection software cannot see.

## 1.2.2 Data Processing

IDSs can be classified according to where and how data is processed into *Distributed-Based Intrusion Detection System* and *Centralized-Based Intrusion Detection System*.

### A. Distributed-Based Intrusion Detection System

The Distributed-Based Intrusion Detection System (DIDS) is where data is collected and analyzed in multiple hosts [Spa00]. Gathering and analyzing data from many distributed systems in order to obtain a more complete picture of the attackers' activities. Monitoring and detection is done using an agent-based approach, where response decisions are made at central point of analysis [Bur03].



## **B. Centralized-Based Intrusion Detection System**

This system may collect data in a distributed fashion, but processed centrally. A centralized IDS analyses its data at a fixed number of locations. These locations are independent of the number of hosts begin monitored. Where distributed IDS analyses its data at a number of locations proportional to the number of hosts that are begin monitored.

In this classification, only the locations and the number of the data analysis computers are considered; the data collection components are not considered [Zie04].

### **1.2.3 Analyze Strategy**

With respect to the method of analysis the collected data by IDS, IDSs can be classified into two types: *Misuse-Intrusion Detection System* and *Anomaly-Intrusion Detection System* [Bac00].

#### **A. Misuse-Intrusion Detection System**

Misuse detection essentially checks for "activity that's bad" with comparison to abstracted descriptions of undesired activity. This approach attempts to draft rules describing known undesired usage (based on past penetrations or activity which is theorized would exploit known weaknesses) rather than describing historical "normal" usage. Rules may be written to recognize a single auditable event that in and of itself represents a threat to system security, or a sequence of events that represent a prolonged penetration scenario. The effectiveness of provided misuse detection rules is dependent upon how knowledgeable the developers are about vulnerabilities. Misuse detection may be implemented by developing expert system rules, model based reasoning, state-transition analysis systems, or neural nets [Ory06].

Advantages of Misuse-Intrusion Detection System are [Ham06]:

1. Misuse detectors are very effective at detecting attacks without generating an overwhelming number of false alarms.
2. Misuse detectors can quickly and reliably diagnose the use of a specific attack tool or technique. This can help security managers prioritize corrective measures.
3. Misuse detectors can allow system managers, regardless of their level of security expertise, to track security problems on their systems, initiating incident handling procedures.

Disadvantages of Misuse-Intrusion Detection System are [Ham06]:

1. Misuse detectors can only detect those attacks they know about. Therefore, they must be constantly updated with signatures of new attacks.
2. Keeping the knowledge base of such intrusion detection system up to date is not easy. Even after gathering information about the attacks, it is time consuming to analyze them and update the knowledge base of the IDS.

## **B. Anomaly-Intrusion Detection System**

Anomaly detectors identify abnormal unusual behavior (anomalies) on a host or network. They function on the assumption that attacks are different from “normal” (legitimate) activity and can therefore be detected by systems that identify these differences. Anomaly detectors construct profiles representing normal behavior of users, hosts, or network connections. These profiles are constructed from historical data collected over a period of normal operation. The detectors then collect event data and use a variety of measures to determine when monitored activity deviates from the norm [Axe98].

Advantages of Anomaly-Intrusion Detection System are [Bac00]:

1. IDSs based on anomaly detection detect unusual behavior and thus have the ability to detect symptoms of attacks without specific knowledge of details.
2. Anomaly detectors can produce information that can in turn be used to define signatures for misuse detectors.

Disadvantages of Anomaly-Intrusion Detection System are [Bac00]:

1. Anomaly detection approaches usually produce a large number of false alarms due to the unpredictable behaviors of users and networks.
2. Anomaly detection approaches often require extensive “training sets” of system event records in order to characterize normal behavior patterns.

As a result, Intrusion detection systems (IDS) are defined by both the method used to detect attacks and the placement of the IDS on the network. An IDS may perform either misuse detection or anomaly detection and may be deployed as either a network-based system or a host-based system. These results in four general groups: misuse-host, misuse-network, anomaly-host and anomaly-network. Some IDS combine qualities from these categories (usually implementing both misuse and anomaly detection) and are known as hybrid systems in their methods for analyzing collected data [Bur03].

### **1.3 Literature Survey**

Various efforts in the field of Intrusion Detection were introduced. The literature survey presented in this section will be mainly concerned with design and implementation of intrusion detection systems. Some of the related published literatures are summarized bellow:

1. *Snapp, Brentano, and et al [Sna91]*: They designed and implemented a prototype Distributed Intrusion Detection System (DIDS) that combines distributed monitoring and data reduction (through individual host and LAN monitors) with centralized data analysis (through the DIDS director) to monitor a heterogeneous network of computers. A main problem considered in their paper was the Network-user Identification problem, which is concerned with tracking a user moving across the network, possibly with a new user-id on each computer. Initial system prototypes had provided quite favorable results on this problem and the detection of attacks on a network. This paper provides an overview of the motivation behind DIDS, the system architecture, and capabilities, and a discussion of the early prototype.

2. *Chen, Cheung, and et al [Che96]*: This paper presented *GrIDS* (Graph-Based IDS) which is attempted to fuse together the information from various nodes into a graph of the entire activity of the system. This allows large-scale attacks to be easily noticed. Because of the similarity of this approach to network management, it can had an easy to understand user interface and would be straightforward to integrate with network visualization and monitoring tools. Unfortunately, the framework is not powerful enough to build an IDS with a high level of coverage. Nor is it useful for most automated monitoring tasks, with a few exceptions: the detection of worms, the detection of network scans, and the detection of telnet chains. Extensive scalability was not been explicitly addressed.

3. *Vigna and Kemmerer [Vig97]*: This paper presented a new approach that applies the State Transition Analysis Technique (STAT) to network intrusion detection. Network-based intrusions are modeled using state transition diagrams in which state and transition are characterized in a

network environment. The target network environment itself is represented using a model based on hyper-graphs. By using a formal model of both the network to be protected and the attacks to be detected, the approach is able to determine which network events have to be monitored and where they can be monitored, providing automatic support for configuration and placement of intrusion detection components.

**4. *Porras and Neumann [Por97]:*** This paper presented EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances) which was a new generation of distributed IDS in the development of IDDES (Intrusion Detection Expert System) and NIDES (Next-generation Intrusion Detection Expert System). EMERALD was arguably the most complete advanced intrusion detection system. Much of the professional team has extensive experience in intrusion detection. It uses both signatures in an expert system and statistics to detect attacks. Considerable portions of EMERALD have a well-defined API (Application Programming Interface) with multiple modules. Indeed, the EMERALD software has been significantly developed and even deployed in a few instances. EMERALD has been used as the core IDS for later research resulting in IDS software by SRI. For example, the alert correlation research performed by SRI was added to the EMERALD software for testing.

**5. *Barrus and Rowe [Bar98]:*** They proposed a distributed architecture with autonomous agents to monitor security-related activity within a network. Each agent operates cooperatively yet independently of the others, providing for efficiency, real-time response, and distribution of resources. This architecture provides significant advantages in scalability, flexibility, extensibility, fault tolerance, and resistance to compromise. They also proposed a scheme of escalating levels of alertness, and a way to notify

other agents on other computers in a network of attacks so they can take preemptive or reactive measures. They designed a neural network to measure and determine alert threshold values. A communication protocol is proposed to relay these alerts throughout the network. They illustrated their design with a detailed scenario.

6. *Yang, Ning, and et al* [Yan00]: This paper presented CARDS (Distributed System for *Detecting Coordinated Attacks*) was the implementation of ideas in "Abstraction based Intrusion Detection in Distributed Environments". The main idea in this signature based approach is to automatically break the signature down into smaller components that can be distributed and recombined to match the signature. The approach is interesting, but the technical treatment is difficult to follow. It may be a promising approach, when and if many issues get resolved. It is not yet sufficiently developed to facilitate an analysis of network performance or other distributed performance issues. It is not clear how it performs for denial of service attacks or other overload attacks. The implementation as described does not appear to be complete; indeed a paragraph in the back of the CARDS paper implies that it is only underway, but not finished. The interconnections between nodes are glossed over.

7. *Roesch* [Roe02]: *Roesch* showed that there are a few basic concepts one should understand about Snort. Snort is an open source network intrusion prevention and detection system utilizing a rule-driven language, which combines the benefits of signature, protocol and anomaly based inspection methods. There are three main modes in which Snort can be configured: sniffer, packet logger, and network intrusion detection system. Sniffer mode simply reads the packets of the network and displays them for user in a continuous stream on the console. Packet logger mode logs the packets to

the disk. Network intrusion detection mode is the most complex and configurable configuration, allowing Snort to analyze network traffic for matches against a user defined rule set and performs several actions based upon what it sees.

**8. *Smith* [Smi02]:** *Smith* illustrated that the learning abilities of neural networks can serve to detect the malicious activity on network environment. These systems are tested against denial of service attacks, distributed denial of service attacks, portscans, and even some doorknobs attacks. In addition, showed how these systems detect long-term attacks, which occur when attackers space out their efforts to evade detection. In this work, the network based intrusion detection explained using a Perception-based, feed-forward neural network system and a system based on classifying, self-organizing maps. Both of these systems are tested on data provided from the DARPA intrusion detection evaluation program as well as live attacks in an isolated computer network.

**9. *Maselli, Deri, and Suin* [Mas03]:** Goal of this paper is to show that in every network there are some global variables that can be profitably used for detecting network anomalies, regardless of the type of network users and equipment. As most of the relations among these variables are fixed, this paper shows that it is possible to define generic network rules aimed to automatically detect selected network anomalies. Finally, it covers the design and implementation of an open-source application used to effectively validate this work on a large campus network.

**10. *Li* [Wei04]:** This paper described a technique of applying Genetic Algorithm (GA) to Network Intrusion Detection Systems (NIDSs). A brief overview of the intrusion detection system, genetic algorithm, and related detection techniques are presented. Parameters and evolution process for

GA are discussed in detail. Unlike other implementations of the same problem, this implementation considers both temporal and spatial information of network connections in encoding the network connection information into rules in IDS. This is helpful for identification of complex anomalous behaviors. This work focused on the TCP/IP network protocols.

## 1.4 Aim of Thesis

This research aims to monitor security-related activity within a network. It should monitor a variety of hosts and operating system platforms to detect an intruder attack before an intruder can do more damages or accesses to a sensitive information. The proposed research is applied by explicitly looking for the filing system attack as a result of previous analysis (predefined suspicious behaviour) or due to some triggering event of suspicious behavior. Also, it proposes a scheme of escalating levels of alertness so that the system administrator can take preemptive action against the attack, such as warning the suspicious users for their misuse actions.

## 1.5 Thesis Layout

Beside chapter one, the remaining part of this thesis consists of other four chapters:

- ✓ **Chapter Two** (*Network Monitoring*): Discuss the concepts of network categories, network protocols. Also, it discusses network monitoring for intrusion detection systems and their component and efficiency.
- ✓ **Chapter Three** (*FMS Design and Architecture*): Presents the structure of the established system such that it explains the main



stages of the system, with some explanation to the implementation steps of each stage.

- ✓ **Chapter Four** (*FMS Interfaces and Evaluation*): Presents the main interfaces of the established system and discusses the results of the conducted tests to detect network intrusions. The weakness and success cases of the proposed system are presented.
- ✓ **Chapter Five** (*Conclusions and Suggestions for Future Work*): Presents a list of derived conclusions and suggestions for future work.

### 3.1 Introduction

This chapter is dedicated to investigate the design considerations and implementation requirements, which are considered throughout the design stage of the proposed intrusion detection system. The scheme of the proposed system and its implementation will be presented in details.

In this chapter, the words "User", "Administrator", and "system manager" are used as indication to persons who have different responsibilities. Client Subsystem and Administrator Subsystem indicate the software programs that run by User and Administrator, respectively. The word "Client" refers to the host on the network.

### 3.2 Design Considerations and Requirements

The aim of the thesis is to design and implement an intrusion detection system that protect all shared resources (i.e. shared files and folders) on each host in the network from attacks.

The name of the proposed system is chosen to be **FMS** (the acronym for **File Monitoring System**). FMS should monitor all the packets coming to each host in the network and detect the intrusion cases by reporting the Administrator with alarm messages.

To achieve the above aim, different features of intrusion detection systems were studied. The study had guide to point the following considerations during designing stage of FMS:

1. FMS will be designed as a Distributed-based Intrusion Detection System, i.e. the data will be collected and analyzed on each host in the network individually. Since, the shared resources of each host will be protected independently.
2. The analysis strategy of FMS was chosen to be Anomaly, where the profiles that represent the normal behaviour of each user per each host will be initialized first by Administrator, then stored on each host lately.

The reason for choosing the Administrator to initialize and update each profile of each user is to offer more confidentiality and integrity level to the system.

3. The intrusion that will be detected by FMS is concerned with checking illegal user's actions that are associated with shared files and folders, such as open file, delete folder, etc.

To implement the above designing considerations, the following hardware and software requirements will be in need:

1. Ethernet LAN, the most widely deployed protocol in network layer in the TCP/IP protocol suite is Ethernet. So, Ethernet LAN type will be chosen instead of other types (Token Ring, Arcnet). Ethernet network connects the clients using either HUB or Switch HUB.
2. Dedicated network category, the most suitable one for the thesis is Peer-to-Peer category. Because each host can share its resources to other hosts as Server, and can access the shared resources on other hosts as Client.
3. Operating Systems to run Ethernet network such as Windows XP or Server 2003.
4. Programming language that support network operations like connecting, receiving incoming network packets and sending messages from host to another using socket API functions. Visual Basic (VB) is chosen to implement this thesis.

### 3.3 FMS Architecture

FMS consists of two main subsystems; *Administrator Subsystem* and *Client Subsystem* as illustrated in Figure (3.1).

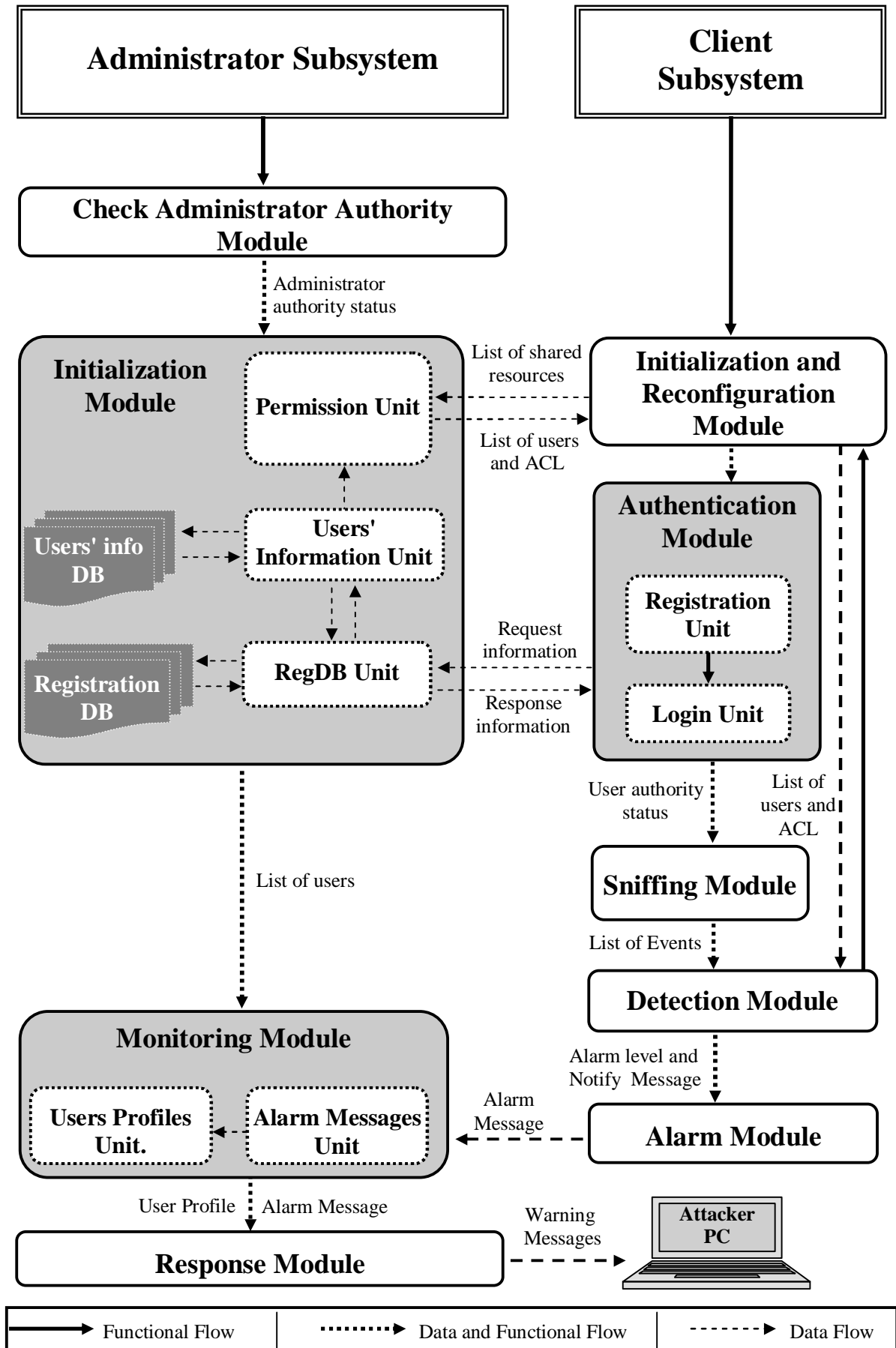


Figure (3.1) FMS Architecture

Each subsystem as shown in Figure (3.1) implies different stages, and each stage is performed by specific units. Some stages execute units in both subsystems independently, while other stages, imply overlapping between executing jobs of units of both subsystems when some shared results are needed. Client subsystem modules are:

1. **Initialization and Reconfiguration Module:** it initializes Client subsystem information. This module performs the following functions:
  - a. Identify the important user shared resources which must be protected with a particular level of protection.
  - b. Determine the Administrator's PC.
  - c. Assemble the local shared resources into a list  $L$ .
  - d. Send the list  $L$  to the Administrator subsystem to set/update permissions.
  - e. Get list of shared/permissions (or ACL) from the Administrator subsystem.
  - f. Get list of users from the Administrator subsystem.
  - g. Determine Time threshold value ( $t$ ) that will be used in Detection Module.
2. **Authentication Module:** it identifies user to the system. This module consists of two units, as follow:
  - a. **Registration Unit:** it registers the allowed user into the system depending on user's information.
  - b. **Login Unit:** it log-in any user into the system depends on user's registration
3. **Sniffing Module:** it sniffs incoming packets to the current host, and then generates list of occurring events.
4. **Detection Module:** it detects malicious events by applying specific rules during traffic analysis, and then generates an alarm code which

represents the level of threat that occurred, and a notify message that describes the occurred threat. These information will be send to the Alarm Module.

5. **Alarm Module:** it generates alarm messages which represent users suspicious activities and then sending these messages to the Administrator. The alarm message depends on alarm code and notify message that generated in the Detection Module.

On the other hand, Administrator subsystem consists of the following modules:

1. **Check Administrator Authority Module:** it checks the authenticity of Administrator identity.
2. **Initialization Module:** it initializes pre-system information. This module consists of three units, as follows:
  - a. **Permission Unit:** it Sets/Updates the permissions (i.e. Access Control List - ACL) for all shared resources of each user, and then sends them to the Client subsystem.
  - b. **Users Information Unit:** it creates Database contains list of users with some required information to the system after registration process. These information are PC name, PC local IP, PC account name, user name, and user type.
  - c. **Registration Database (RegDB) Unit:** it creates Database contains all identification information of all users identification whom allowed to use the system.
3. **Monitoring Module** This module consists of two units, as follow:
  - a. **Alarm Messages Unit:** it selects the case of monitoring process either to be online or offline modes.

b. **Users Profiles Unit:** it creates profile for any misbehavior user to be utilized in the future detection.

4. **Response Module:** it generates a proper response (Warning messages) in the case of detecting an intrusion.

### 3.3.1 Client Subsystem

The client subsystem is the main part of FMS that is installed on the remote computers to be protected from the intruders. The users of these computers must be legal ones to access the shared resources on the network. The client subsystem will disable any access to the shared resources unless user's identity is authorized to the system. To check the user's authentication and start monitoring process, the client subsystem will run at windows startup.

The client subsystem is divided into five modules, as it will be explained bellow.

#### A. Initialization Module

The main goal of this module is to initialize all necessary Client subsystem information. The following six functions of this module will be executed once at the first run of the system.

**1. Identify the user shared resources:** The system manager identifies the important user shared resources which must be protected with a particular level of *protection*. The shared resources are classified into classes according to their applicability (e.g. administration, books, thesis, etc). Classes are stored in classes file. Algorithm (3.1) explains how to add new class to the classes file, whereas, Algorithm (3.2) explains how to add new shared resource to an existing class.

### Algorithm (3.1) Add Shared Resources Class

**Goal:** Add a new class to the classes' types file.

**Input:**

object\_classes\_file<sub>in</sub>: file contains a List of all classes types with their shared resources.

Class\_Type: Class of the shared resources.

**Output:**

object\_classes\_file<sub>out</sub>: updated file contains a List of all classes types with their shared resources.

**Step1:** System manager enters the new Class\_Type.

**Step2:** **If** (Class\_Type does **NOT** exist in object\_classes\_file<sub>in</sub>) **Then**

1. *Call Algorithm (3.2)* with Class\_Type for each addition of shared resource into Class\_Type.

2. *Add Class\_Type* to object\_classes\_file<sub>out</sub>

3. **Return** (object\_classes\_file<sub>out</sub>)

**Else Return** (object\_classes\_file<sub>in</sub>)

**Step3:** **End.**

### Algorithm (3.2) Update Shared Resources Class

**Goal:** Add a new shared resource to existing class (i.e. update existing class).

**Input:**

Class\_Type<sub>in</sub>: Class of the shared resources.

Shared\_Name: String represents a shared resource.

**Output:**

Class\_Type<sub>out</sub>: Class of the shared resources after updating it.

**Step1:** System manager select shared resource that named a Shared\_Name

**If** (Shared\_Name does **NOT** exist in Class\_Type<sub>out</sub>) **Then**

1. *Increase* the number of defined shared resources in Class\_Type<sub>in</sub>

Class\_Type<sub>out</sub>. Index = Class\_Type<sub>in</sub>. Index + 1

2. Class\_Type<sub>out</sub>. Shared (Index) = Shared\_Name

**Else** Class\_Type<sub>out</sub> = Class\_Type<sub>in</sub>

**Step2:** **Return** (Class\_Type<sub>out</sub>).



**2. Determining PC Name of the Administrator:** the Name of the Administrator's PC is used for sending and receiving specific data to/from Administrator subsystem during the interaction processes between the two subsystems. By using the *Broadcasting technique*, the system will gain the Administrator PC name by sending Client request message and wait for the Administrator to answer the Client request with Administrator PC name.

**3. Assembling the shared resources into a list, and send it to the Administrator:** This unit assembles the local shared resources from the local *Microsoft Management Console* (MMC). Microsoft Management Console is an application executed at a host for an administrative job. Microsoft Management Console uses the extension *msc* which stands for Microsoft Common Console Document (MCCD). Therefore, files which use the .msc extension open with the application Microsoft Management Console. One of the console tools is Computer Management Console (its console file compmgmt.msc). This Console lists many consoles such as Event Viewer console, Shared Folders console, Local Users and Groups, etc.

Shared Folders console (its console file fsmgmt.msc) lists the local shared resources, current users' sessions, and open files through the network on the current computer. Shared resources on the current computer represented as a list of records; each record is formatted, as depicted in Figure (3.2).

Shared Folder Name	Shared Path	Type	#Client Connections	Comment
--------------------	-------------	------	---------------------	---------

**Figure (3.2)** Structure of shared resources list as stored in Shared Folder Console

After collecting the list of shared resources of each client, these resources are sent to the Administrator to set their permission. Setting

permissions depends on the type of shared resource and the class type of the user. This list will be used later by the Client system to build Access control list. Algorithm (3.3) presents assemble and sends shared resources units.

**Algorithm (3.3) Assemble the Shared Resources into List**

**Goal:** Assemble the local shared resources into a list from the local Microsoft Management Console (MMC) and send it to the Administrator.

**Const:**

MMC = Microsoft Management Console.

**Input:**

AdminPcName: Administrator PC Name.

**Output:** List\_Sh: List of shared resources.

**Step1:** Access the computer shared resources list using fsmgmt.msc console file with MMC.

**Step2: Set** L = MMC.SharedFolders.SharedResourcesList

**Step3: For each Item In L do**

List\_Sh. Index = List\_Sh. Index + 1

List\_Sh. Items (Index) = L.Item

**End For Item**

**Step4:** Send List\_Sh list to the Administrator's computer that named AdminPcName.

**Step5: End.**

**4. Construct the Access Control List:** This unit receives a list called ACL of shared resources and their permissions for each user, from Administrator subsystem in order to organize these items in a suitable structure. This structure will be accessed later on by Detection Module to determine the access validity of each user. The selection of a suitable structure wasn't easy since it must meet the following criteria:

- It should be flexible in size to contain all shared resources but at the same time it should reserve a less memory capacity. Since there is

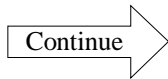
dependability between the permissions of the parent resource and their children, the redundancy must be removed.

- The access time to the object (i.e. shared resource) should be as minimum as possible to satisfy fast execution.
- Any modification to the structure (i.e. changing an exist shared resource, inserting new one, etc) should be as fast as possible.

According to the above criteria, the structure is selected to be **Binary Tree** that satisfies acceptable execution time. The tuple of the binary tree is shown in Figure (3.3). This tree is ordered according the path of the shared resource where the left child represents **Brother** file or folder of the shared resource and the right child represents the **Son** of it. Algorithm (3.4) shows how to construct the ACL.

<b>Shared File/Folder</b>	<b>Permission</b>	<b>Brother</b> for shared file/folder	<b>Son</b> of shared folder only
---------------------------	-------------------	---------------------------------------	----------------------------------

**Figure (3.3)** Tuple's format for ACL

<b>Algorithm (3.4) Construct ACL</b>
<p><b>Goal:</b> Construct the ACL received from the administrator into Binary Tree tructure.</p> <p><b>Input:</b> <math>ACL_{in}</math>: List of local shared resources with their permissions received from Administrator.</p> <p><b>Output:</b> <math>ACL_{out}</math>: Binary tree representing the local shared resources with their permissions.</p>
<p><b>Step1:</b> Index = 1</p> <p><b>Step2: While Not EOF (<math>ACL_{in}</math>) Do</b></p> <p style="padding-left: 20px;">(1)Get the Indexed object from <math>ACL_{in}</math> and store it in tree <math>ACL_{out}</math>.</p> <p style="padding-left: 40px;"><math>ACL_{out}.Shared = ACL_{in}.Item(Index).Shared\_Name</math></p> <p style="padding-left: 20px;">(2)Copy permissions of all registered users of the indexed object to <math>ACL_{out}</math>.</p> <p style="padding-left: 40px;"><b>For</b> i = 1 <b>To</b> No of Registered User <b>do</b></p> <p style="padding-left: 60px;"><math>ACL_{out}.Permission.User(i) = ACL_{in}.Item(Index).Permission(i)</math></p> <p style="padding-left: 40px;"><b>End For</b> i</p> <div style="text-align: right; margin-top: 10px;">  </div>

(3) *Construct* list of indexed object's brothers and organize them into  $ACL_{out}$ .

**For** each Folder **In** SubFolders(Parent ( $ACL_{in}$ .Shared)) **do**

*Create* New Node of ACL.

**Set** NewNode = New ACL

**Set** NewNode.Shared = Folder

**Set**  $ACL_{out}$ .Brother = NewNode

**Set**  $ACL_{out}$  = NewNode

Index = Index + 1

**GOTO** (1)

**End For** Folder

**For** each File **In** SubFiles(Parent ( $ACL_{out}$ .Shared)) **do**

*Create* New Node of ACL.

**Set** NewNode = New ACL

**Set** NewNode.Shared = File

**Set**  $ACL_{out}$ .Brother = NewNode

**Set** NewNode.Son = Null

**Set**  $ACL_{out}$  = NewNode

**GOTO** (1)

**End For** File

(4) *Construct* list of indexed object's children and organize them into  $ACL_{out}$

**a. Set** SubFolders =SubFolders( $ACL_{in}$ .Item (Index))

**b. Set** SubFiles =SubFiles( $ACL_{in}$ .Item (Index))

**c. For** each SubFolder **In** SubFolders **do**

**Set**  $ACL_{out}$ .Son = SubFolder

**Set**  $ACL_{out}$ . Shared = SubFolder

**GOTO** (1)

**End For** SubFolder

**d. For** each SubFile **In** SubFiles **do**

**Set**  $ACL_{out}$ .Son = SubFile

**Set**  $ACL_{out}$ .Shared = SubFile

**GOTO** (1)

**End For** SubFile

**End Of While**

**Step3: End.**

**5. Get the list of registered users from the Administrator subsystem:** The list of users will be used in Detection module. This list contains full information of each user such *PC name, PC's IP, PC user account name, username, and user Type*. This information is aggregated during registrations processes of each user.

**6. Determine Time threshold value ( $t$ ):** This value represents the maximum accepted period of time to compute the repetitions of each action. System manager will define the threshold value then it will be used in Detection module later on to compute the threat.

After executing the previous units, the job of Initialization module is finished and user Authentication module is started. Initialization module is executed at the first run of the system to initialize the system information, but it may re-run again to re-configure the system if there is a need. Reconfiguration the system may be required, for example, if there is new user was registered into the system, or updating the current ACL.

## **B. User Authentication Module**

The job of this module is to ensure that users of FMS are authorized to access the shared resources. To reach this goal, two steps are needed: first, registration of the new allowed users to the system (the word "allowed" means all users can use FMS and already have saved information in its database). Second, login the predefined users by checking their identify information. After that, the network-shared resources will be visible to the users. The above two steps are performed by the following units:

### **1. Registration Unit**

In this unit, if the user is not registered before, he/she must be registered as a new member in the system. Any allowed user can register into the system by performing the following steps:

- a. Entering username: username has maximum length 15 characters and should not contain any special characters.
- b. Entering password: password has maximum length 8 characters. The user is requested to enter a password twice to confirm its password.
- c. Selecting user type: user type identifies the level of that user and his/her permissions. User type could be Student, Employee, or any user type according to the system environment.
- d. User may be requested to enter additional information depending on the user type. These information is to identifying the user according to its type. For example, when an employee (i.e. user type is employee) aims to register into the system, he/she asked to enter his/her employee name and identification card number.

After acquiring the registered information by Client subsystem, these information will be verified by Administrator subsystem by checking the registration database (RegDB). Administrator subsystem will inform the user about registration status (i.e., either accepted or rejected). If it is accepted, the registration information will be added to the RegDB. If it is not, then another trial will be made. Algorithm (3.5) shows how the Registration unit works.

## **2. Login Unit**

After finishing registration, Login unit is called when the user needs to use the system. To login, the user must present its own correct username and password. Client subsystem will verify the entered username and password by matching them with the registered username and password. If the user enters invalid information, the system will prompt him/her to re-enter correct information, for three times as maximum. When User is logged into the system; the shared resources will be visible to him/her and

Client subsystem will start execute its monitoring functions. Algorithm (3.6) shows how the Login unit works.

### **Algorithm (3.5) Register Process**

**Goal:** Register new member into the Client subsystem.

**Input:**

Username: the name of Client's user.

Pass: the password of Client's user.

Confirm\_Pass: the password of Client's user.

User\_Type: type of Client's user.

Additional\_Info: Additional information related to User\_Type.

**Output:** Flag: Boolean represents the state of registration (True or False).

**Step1:** **Input** Username, Pass, Confirm\_Pass, and User\_Type.

**Step2:** **If** Pass = Confirm\_Pass **Then**

**If** User\_Type = "Student" **Then**

**Input** full name and store it into Additional\_Info

**Else If** User\_Type = "Employee" **Then**

**Input** full name and ID number and store it into Additional\_Info

**Else**

        Flag = False

**Step3:** *Connect* to Administrator subsystem using socket API functions.

**Step4:** *Send* (Username,Pass, User\_Type, and Additional\_Info) to the Administrator subsystem to check the validity of information.

*Call Algorithm (3.21)* by Administrator subsystem and put the result in Flag.

**Step5:** **Return** (Flag).

### **Algorithm (3.6) Login Process**

**Goal:** Login system's member into the Client subsystem.

**Input:**

Username: the name of Client's user.

Pass: the password of Client's user.

**Output:** Flag: Boolean represents the state of login process (True or False).

**Step1:** *Call Algorithm (3.7)* to make shared resources invisible.

**Step2: Input** Username and Pass.

**Step3:** *Connect* to Administrator subsystem using socket API functions.

**Step4:** *Send* (Username and Pass) to the Administrator subsystem to check the validity of information.

*Call Algorithm (3.22)* by Administrator subsystem and put result in Flag.

**Step5: If** (Flag is True) **Then**

*Call Algorithm (3.8)* to make shared resources visible.

**Step6: Return** (Flag).

### **Algorithm (3.7) Make Share Invisible**

**Goal:** Make the shared resources invisible to the Client's user.

**Const:** Path\_S = indicates the path of the folder where network's shared resources are stored as list of folders.

**Input:** None.

**Output:** None.

**Step1: For** each subfolder **In** Path\_S **do**

**Set** attribute of the subfolder = system + hidden

**End For** subfolder

**Step2: End.**



### **Algorithm (3.8) Make Share Visible**

**Goal:** Make the shared resources visible to the Client's user.

**Const:** Path\_S = indicates the path of the folder where network's shared resources is stored as list of folders.

**Input:** None.

**Output** None.

**Step1:** For each subfolder In Path\_S do

**Set** attribute of the subfolder = normal

**End For** subfolder

**Step2:** End.

## **C. Sniffing Module**

Sniffer function is listening to the network segment or sniffs packets on a specific network segment as explained in chapter two. Network segment determines what the Sniffer is looking for. The Sniffer module in this thesis sniffs packets on a three-layer segment (Application, Transport, and Internet layers) of TCP/IP model.

Sniffer module will receive all incoming window messages, and then filters them to reduce the amount of processing by ignoring unimportant packets to its purpose. Filtering process depends on the type of incoming messages; meaningful messages are from the type of **Window Socket Message** (WinSockMsg), which are specified to network packets. In addition, WinSockMsg will be filtered to keep only the messages that are related to file system operations.

After finishing filtering process, Sniffer module will generate a sequence of events that represent the executed operations by the sniffed network segment.

Sniffer functions can be summarized into three parts: 1- Initialize the socket function to prepare the system to connect over LAN and retrieve

specific network segment, 2- Subclassing function to process the incoming network packets in the defined message handling procedure, and 3- Filtering function to generate events that represent file system application services only. To do these three functions, Sniffer module consists of three units, which are:

- 1- **Initialize socket unit** which initialize the Socket.
- 2- **Subclass unit** which subclassing the window that receives the socket message. In other words, Hook the window to sniffs only the Windows Socket Message (WinSockMsg).
- 3- **Filter unit** which filtering the WinSockMsg.

## 1. Initialize Socket Unit

The function of this unit is to initialize the network socket using some API declarations and functions to prepare the system to connect over local network and retrieve only 3-layer segment of the TCP/IP model. Algorithm (3.9) shows how this unit works.

### Algorithm (3.9) Initialize Socket

**Goal:** Initialize the network socket using some API functions and prepare the socket to retrieve the full TCP/IP packet without the network layer protocol information (header and footer).

**Input:** None.

**Output:** None.

**Step1:** Load windows socket API library using **WSAStartUP** API function to initialize the underlying windows socket DLL, this step must success before using API socket declarations and functions.

**Step2:** Create the socket S using **Socket** API function

```

AF_INET = 2           // TCP/IP Protocols.
Sock_RAW = 3         // Access to internal network protocol type.
IPPROTO_IP = 0      // determine which segment of S will be processed.
S = Socket (AF_INET , Sock_RAW , IPPROTO_IP)
    
```

Continue 

```
// S represents the endpoint for communication
```

**Step3:** Set socket option using **Setsockopt** API function.

```
SOL_SOCKET = 0xFFFF // this value means option will be set
SO_RCVTIMEO = 0x1006 // Set time-out value
RCVTIMEO = 0x1388 // option value = 5000 seconds
Call Setsockopt (S,SOL_SOCKET, SO_RCVTIMEO , RCVTIMEO,_)
```

**Step4:** Bind the socket using **Bind** API function.

```
// Binds defined IP and port number with S
Current_Socket.IP = Local IP // PC's LAN IP
Current_Socket.PortNumber = 0x1B58 // port : 7000
Call Bind (S, Current_Socket , Len(Current_Socket))
```

**Step5:** Specify mode of operation on the socket using **WSAIoctl** API function.

```
SIO_RCVALL = 0x98000001 / S can receiving all IPv4 and IPv6 packets.
Call WSAIoctl (S, SIO_RCVALL, _, _, _, _, 0, 0)
```

**Step6:** Specify the type of the operation that will be done on the socket message using **WSAAsyncSelect** API function.

```
HWnd = CurrentWindow.HWnd // Handle of current window
WSM = 0x401 // 0x401 mean window socket message
Ops = 1 // READ incoming packet (i.e. RECV function will be used).
Call WSAAsyncSelect (S, HWnd, WSM , Ops)
```

**Step7: End.**

## 2. Hook the window – Subclass Unit

Unit function is a subclass the processing of window message or hooks the window using some API functions.

Hook is a point in the system message-handling mechanism where an application can install a subroutine to monitor traffic in the system (e.g. network traffic) and process particular types of messages (such *window socket message*) before they reach the target window procedure.

Subclass unit changes the processing of certain type of window message, which is WinSockMsg, from the default Window procedure to

new defined procedure called Filtering\_Procedure. Filtering\_Procedure is the next unit of the Sniffer module. Algorithm (3.10) explains how to subclass the window.

### Algorithm (3.10) Subclassing Window

**Goal:** Subclass the window using some API functions and process particular types of Window messages in new procedure.

**Const:** WinSockMsg: Window Socket Message type

**Input:** WinMsg: Window Message.

W, L: additional parameters that describe the window message.

Current\_Window: Window which receives the message.

**Output:** Flag represents Success or Fail.

**Step1:** Get address of the new window message procedure that handle the window message using **AddressOf** API function:

Filter\_Address = **AddressOf** (Filtering\_Procedure)

**Step2: If** WinMsg is WinSockMsg **Then**

1. *Specify* the operation of the **SetWindowLong** function:

Opt = GWL\_WNDPROC // mean that new address sets as window

2. Use **SetWindowLong** API function to causes the system to call the new window procedure instead of the previous one and the previous window procedure address will be stored in Add\_Old:

Addr\_Old=**SetWindowLong**(Current\_Window,Opt,Filter\_Address )

**Return** (Success).

**Else**

Cause the system to restore the processing of window message to the default window procedure using **CallWindowProc** API function:

Call **CallWindowProc** (Addr\_Old, Current\_Window ,WinMsg , W, L)

**Return** (Fail).

**Step3: End.**

### 3. Filter Unit

The previous units determine *which* type of Window messages will be processed and *where* it will be processed. Filter unit, on the other hand, presents *how* to process the WinSockMsg.

In the new message procedure (Filtering\_Procedure), system will receive the window socket message (WinSockMsg) and store it into buffer with length 1500 bytes as maximum packet size according to Maximum Transmission Unit (MTU) of TCP/IP for the Ethernet. MTU is the maximum size of a single data unit (e.g., IP datagram) of digital communications depending on the properties of the network. The content of buffer represents the packet that travels over network to the current host as shown in Figure (3.4). This packet is constructed from 3-layers headers that encapsulate the application data.

IP Header	TCP Header	NetBIOS Header	SMB Header	Data of application service
20 bytes	20 bytes	4 bytes	32 bytes	<= 1424

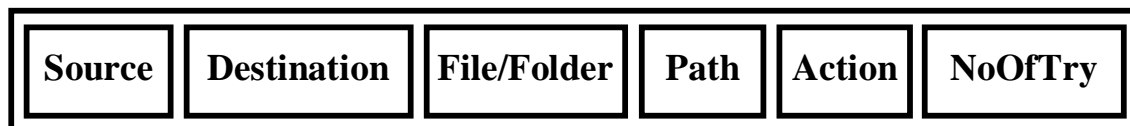
**Figure (3.4)** Buffer content

The intended application service at the data of application layer is one of the file system application services such as open file, delete file, copy file, etc. These applications deal with Server Message Block protocol (SMB) of the application layer protocols. SMB protocol uses to request file and print services from server systems over network. SMB is based on NetBIOS session type as explained in chapter two.

Filtering Unit is classified into two parts; first part is to filter the WinSockMsg and aggregate the values of protocols headers into IP, TCP, NetBIOS, and SMB records respectively. The second part is to generate events that will be used later in Detection module. These events represent

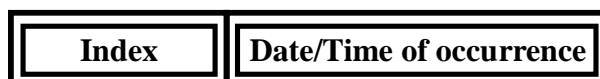
the file system events which are occurred on the local host through the network.

Each event is represented as a record of six fields as illustrated in Figure (3.5), these fields are:



**Figure (3.5)** Event format

- a. **Source:** this field contains full information about the subject of action. These information are: FMS username, user type, source PC name, PC's IP, PC user account name, and PC description.
- b. **Destination:** this field contains full information about the accessed shared file/folder, these information are: Destination PC name, PC's IP, and PC description.
- c. **File/Folder:** this field contains the accessed shared file/folder name.
- d. **Path:** this field contains the path of the accessed shared file/folder.
- e. **Action:** this field contains the action type which is one of the file system application services (actions or operations). File system operations are: **open file, delete folder, delete file, rename folder, rename file, create folder, create file, write to file, and copy file.**
- f. **NoOfTry:** this field represents list of occurrences of event. Each item in the list is represented by date/time of event, as shown in Figure (3.6).



**Figure (3.6)** NoOfTry List format

Algorithm (3.11) represents the steps of filtering network messages process, while Algorithm (3.12) represents the steps of generating events process.

### Algorithm (3.11) Filtering Network Messages

**Goal:** Filtering WinSockMsg and extracting the information of one traveled packet

**Input:**

WinSockMsg: Window Socket Message represents the incoming packet from the network to current Window

S: Current Socket.

**Output:**

IPrec: IP Header Record of IP Header.

TCPrec: TCP Header Record of TCP Header.

NTBrec: NetBIOS Header Record of NetBIOS Session Header.

SMBrec: SMB Header Record of SMB Message Header.

State: Boolean flag that represents if the packet is filtered (state=True) or discarded (state=False).

**Step1:** *Sniff* the incoming packet using **RECV** API function and store it into Buffer with length 1500 Bytes.

**RECV** (S, Buffer, 1500, \_)

**Step2:** *Copy* the first 20-bytes of **Buffer** to the IP Headers Record.

IPrec = Buffer [0 . . 19]

**Step3:** *Check* the protocol field of IP Header if it is TCP protocol.

**If** Protocol Field of IP Header  $\diamond$  6 **Then**

TCP protocol doesn't exit at Transport layer.

State = False.

**GOTO Step12**

**Step4:** *Copy* the next 20-bytes of **Buffer** to the TCP Headers Record.

TCPrec = Buffer [20 . . 39]

**Step5:** *Check* the port number field of TCP Header that determines which process runs at Application layer.

**If** TCPrec.SourcePort  $\diamond$  139 *And*

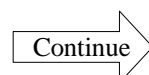
TCPrec.SourcePort  $\diamond$  445 *And*

TCPrec.DestinationPort  $\diamond$  139 *And*

TCPrec.DestinationPort  $\diamond$  445

**Then**

SMB protocol doesn't exit at Application layer.



State = False.

**GOTO Step12**

**Step6:** Copy the next 4-bytes of Buffer to the NetBIOS Header Record because SMB depends on NetBIOS protocol.

NTBrec = Buffer [40 . . 43]

**Step7:** Check the type of NetBIOS session to achieve SMB deals with file system application services.

**If** NTBrec.SessionType  $\neq$  0 **Then** // i.e. not session message

Traveled messages aren't client request or server response of file system application services.

State = False.

**GOTO Step12**

**Step8:** Copy the next 32-bytes of Buffer to the SMB Headers Record.

SMBrec = Buffer [44 . . 75]

**Step9:** Check the first 4-bytes that represent the ID of SMB header which mean that SMB defined at Application layer.

**If** (SMBrec.ID[1] = 0xFF) **And** (SMBrec.ID[2] = 'S') **And**  
(SMBrec.ID[3] = 'M') **And** (SMBrec.ID[4] = 'B') **Then**

SMB is defined at Application layer as client request or server response to file system application services.

**Else**

State = False.

**GOTO Step12**

**Step10:** Check the type of SMB message (or traveled packet) if it is client request or server response.

**If** (SMBrec.FLAG & 128) = 0 **Then**

SMB message is client request

Copy the remaining of Buffer to the SMB Data Portion.

SMBrec.DataPortion = Buffer [76 . . 1499]

**Else**

State = False.

**GOTO Step12**

**Step11:** State = True

**Step12: End.**



From Algorithm (3.11), one can notice, the current detection engine depends on client request message only. The client request packet contains or passes the needed information (key information) about the request, while server response message represents the state of client request.

### Algorithm (3.12) Generate Events

**Goal:** generates list of events that represent the occurred actions.

**Input:** IPrec: IP Header Record.

    TCPrec: TCP Header Record.

    SMBrec: SMB Header Record.

**Output:** Events[]: List of events.

    Index: number of occurred events.

**Step1:** *Initialize* the flag that indicates the state of determination of the file system application service. tmpEvent will be used as temporary event to aggregate the information before adding it to Event[] list.

    ActionDetectedFlag = True

**Step2:** *Check* if the Command field of SMB Header is delete directory.

**If** (SMBrec.COMMAND = 1) **Then**           //SMB\_COM\_DEL\_DIR

        tmpEvent.Action = "\_delete folder\_"

**For** i = 0 **To** SMBrec.Data.ByteCount **do**

            tmpEvent.File/Folder= tmpEvent.File/Folder+SMBrec.Data.DirectoryName[i]

**End For** i

**GOTO Step9**

**Step3:** *Check* if the Command field of SMB Header is rename file or folder.

**If** (SMBrec.COMMAND = 7) **Then**           //SMB\_COM\_Rename

        tmpEvent.Action = "\_rename file/folder\_"

**For** i = 0 **To** SMBrec.Data.ByteCount **do**

            tmpEvent.File/Folder= tmpEvent.File/Folder+ SMBrec.Data.OldFileName[i]

**End For** i


**GOTO Step9**

**Step4:** *Check* if the Command field of SMB Header is NT\_CreateAndX.

**If** (SMBrec.COMMAND = 162) **Then**       //SMB\_COM\_NT-CreateAndX

**If** (SMBrec.Data.CreateDisposi = 0x40) *And*

            (SMBrec.Data.CreateOptions = 0x02) *And*

Continue 

```

(SMBrec.Data.ShareAccess = 0x02) And
(SMBrec.Data.Flags = 0x16) Then
tmpEvent.Action = "_create file_"
  Call Algorithm(3.13) and put result into tmpEvent.File/Folder
Else If (SMBrec.Data.CreateDisposi = 0x200040) And
  (SMBrec.Data.CreateOptions = 0x02) And
  (SMBrec.Data.ShareAccess = 0x01) Then
tmpEvent.Action = "_delete file_"
  Call Algorithm(3.13) and put result into tmpEvent.File/Folder
Else If (SMBrec.Data.CreateDisposi = 0x000001) And
  (SMBrec.Data.CreateOptions = 0x02) And
  (SMBrec.Data.ShareAccess = 0x02) Then
tmpEvent.Action = "_create Folder_"
  Call Algorithm(3.13) and put result into tmpEvent.File/Folder
Else If (SMBrec.Data.CreateDisposi = 0x200044) And
  (SMBrec.Data.CreateOptions = 0x02) And
  (SMBrec.Data.ShareAccess = 0x01) Then
tmpEvent.Action = "_copy file_"
  Call Algorithm(3.13) and put result into tmpEvent.File/Folder
ElseIf (SMBrec.Data.CreateDisposi = 0x000040) And
  SMBrec.Data.CreateOptions = 0x02) And
  (SMBrec.Data.ShareAccess = 0x03) And
  (SMBrec.Data.AllocationSize = 0x80) And
  (SMBrec.Data.DesiredAccess = 0x2019F) Then
tmpEvent.Action = "_write to file_"
  Call Algorithm(3.13) and put result into tmpEvent.File/Folder
Else
tmpEvent.Action = "_open file_"
  Call Algorithm(3.13) and put result into tmpEvent.File/Folder
GOTO Step9

```


**Step5:** Check if the Command field of SMB Header is TreeConnectAndX.

```

If (SMBrec.COMMAND = 117) Then // SMB_COM_TreeConnectAndX
//This Command passes the current reached path.
  Call Algorithm(3.14) and put result into tmpEvent.Path
GOTO Step9

```

**Step6:** Check if the Command field of SMB Header is Transaction.

Continue 

**If** (SMBrec.COMMAND = 37) **Then** //SMB\_COM\_Transaction

*Check* the bytes of SMB Data portion to aggregate the needed info.

// When this command is occurred, many types of information may be passed depending on bytes values of SMB Data portion. These information may be one of the following at a time:

1. Source computer name that will be stored into tmpEvent.SourcePC
2. Destination computer name that will be stored into tmpEvent.DestinationPC
3. Source PC description that will be stored into tmpEvent.SourcePC\_Desc
4. Destination PC description that will be stored in tmpEvent.DestinPC\_Desc
5. Source PC account name that will be stored into tmpEvent.SourcePC\_Acc //

**GOTO Step9**

**Step7:** *Check* if the Command field of SMB Header is SessionSetupAndX.

**If** (SMBrec.COMMAND = 115) **Then** //SMB\_COM\_SessionSetupAndX

*Check* the bytes of SMB Data portion to aggregate the needed info.

// When this command is occurred, many types of information may be passed depending on bytes values of SMB Data portion. These information may be one of the following at a time:

1. Source PC account name that will be stored into tmpEvent.SourcePC\_Acc.
2. Source computer OS version.
3. Source computer LAN type. //

**GOTO Step9**

**Step8:** *Ignore* the incoming packet.

ActionDetectedFlag = False

**GOTO Step13**

**Step9:** tmpEvent.SourcePC\_IP = IPrec.SourceIP

**Step10:** tmpEvent.DestinationPC\_IP = IPrec.DestinationIP

**Step11:** tmpEvent.Date&Time = Date&Time of the current computer

**Step12:** **If** (ActionDetectedFlag = True) **Then**

*Call Algorithm (3.15)* to add tmpEvent to Event[].

*Clear* tmpEvent

**Step13: End**

### Algorithm (3.13) Get Name SMB\_COM\_162

**Goal:** Get full name (or path) of the File/Folder that processed by the action that had been detected in SMB\_COM\_NT\_CreateAndX.

**Const:** SMB\_COM\_NT\_CreateAndX record

**Input:** SMBrec: SMB Header Record.

**Output:** File/Folder\_Name: Represents File/Folder name of particular Action.

**Step1:** *Initialize* the output string.

File/Folder\_Name = ""

**Step2:** *Get* full name of the File/Folder..

**For** i=Len (SMB\_COM\_NT\_CreateAndX)+1 **To** SMBrec.Data.ByteCount **do**

File/Folder\_Name = File/Folder\_Name + SMBrec.Data [i]

**End For** i

**Step3:** **End.**

### Algorithm (3.14) Get Path SMB\_COM\_117

**Goal:** Get path of the File/Folder that processed by the action that had been detected in SMB\_COM\_TreeConnectAndX.

**Const:** SMB\_COM\_TreeConnectAndX record

**Input:** SMBrec: SMB Header Record.

**Output:** Path: Represents Path of File/Folder name of particular Action.

**Step1:** *Initialize* the output string.

Path = ""

**Step2:** *Get* path of the File/Folder.

**For** i=Len(SMB\_COM\_TreeConnectAndX)+1 **To** SMBrec.Data.ByteCount **do**

Path = Path + SMB.Data [i]

**End For** i

**Step3:** **End.**

### Algorithm (3.15) Add Event to the List of Events

**Goal:** Update the existing Event[] List by adding new event.

**Input:** tmpEvent : Represents new event was occurred.

Event[] : List of Events.

Index<sub>in</sub> : Number of Events.

**Output:** Index<sub>out</sub> : Number of Event after update if not Exist.

Event[] : List of Events after updated.

**Step1:** Call Algorithm(3.16) with Index<sub>in</sub> to check if tmpEvent is exist in Event[], then, put results into Exist Flag and IndexE.

**Step2:** If Exist = True Then //Update the NoOfTry field of the exist event.

Number = Even[IndexE].NoOfTry.No

Even[IndexE].NoOfTry.No = Number + 1

Event[IndexE].NoOfTry.Times (Number+1) = tmpEvent.Date&Time

**Else** // Increase the number of events then add the new event.

Index<sub>out</sub> = Index<sub>in</sub> + 1

Event[Index<sub>out</sub>] = tmpEvent

**Step3:** End.

### Algorithm (3.16) Check the List of Events

**Goal:** Check if the new event was added to Event[] List before, or not.

**Input:** tmpEvent : Represents new event was occurred.

Event[] : List of Events.

Index : Number of Events in Event[].

**Output:** Exist : Boolean flag (True or False) to indicate if the event exist or not.

IndexE : Index of Event that match tmpEvent.

**Step1:** Initialize the output flag.

Exist = False

**Step2:** Sort the Event[] list according the subject name.

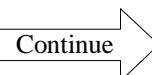
**For** i = 1 **To** Index **do**

**For** j = 1 **To** index – i **do**

**If** (Event[j].Subject name > Event[j+1].Subject name ) **Then**

temp = Event ( j )

Event ( j ) = Event ( j + 1 )



```
Event ( j + 1 ) = temp
```

```
End For j
```

```
End For i
```

**Step3:** Use Binary search technique to Search sorted list of events to find tmpEvent.

```
(1) Low = 1
```

```
High = Index
```

```
(2) If (High < Low) Then // not found
```

```
    GOTO Step4
```

```
Mid = (Low + High) / 2
```

```
If (Event[Mid].Subject name > tmpEvent.Subject name) Then
```

```
    High = Mid - 1
```

```
    GOTO (2)
```

```
Else If (Event[Mid].Subject name < tmpEvent.Subject name) Then
```

```
    Low = Mid + 1
```

```
    GOTO (2)
```

```
Else // Found
```

```
    IndexE = Mid
```

```
    Exist = True
```

**Step4: End.**

From Algorithm (3.12), one can notice that many commands may pass the same information like PC account name, which are required for generating an event. The problem is how to know which command is passing the information, and this depends on the network traffic and the time of the executed command. The solution is implemented in Algorithm (3.12), which deals with all possible cases.

## D. Detection Module

The main function of this module is to detect the malicious events and reports the Alarm module with *alarm level* (which is also called *threat level*) by applying particular rules on occurring events depending on information that provided by Initialization and Sniffing modules.

In addition to alarm level, this module supplies the Alarm module with *Notification Message* that contains some information that guides the Administrator to identify the threat.

This module is classified into two units; *Aggregation* unit and *Decision Making* unit. The function of Aggregation unit is to collect the information that is provided by Initialization and Sniffing modules plus information provided by this unit. Decision Making unit is the place where decisions are made to distinguish which event(s) cause the threat.

## 1. Aggregation Unit

This unit receives information from previous modules (Initialization and Sniffing modules) and computes other to prepare them to be utilized by Decision Making unit.

Initialization module provides *ACL* and *list of FMS registered users*. ACL will be utilized to determine user's permission of specific object. List of registered users will be utilized to identify class type of the user as recorded during user registration process. The implemented user classes in the FMS are *Student*, *Employee*, and *Visitor* according to university environment.

Sniffing module provides *the list of occurred events*. Each event is presented by subject, action, object, and number of occurrences data types.

The computed information by this unit depends on the list of occurred events. This information is the class type of the object. The accessed objects are classified into *System*, *Application*, and *User* classes depending on object path, and object applicability.

System and Application classes are defined automatically by the system while User class resources are defined by the system manager during initialization stage as explained in section 3.3.1 - A.

System and application classes have defined the shared files/folders that related to Windows resources and application resources, respectively. Examples of the system class path are "C:\WINDOWS" or "C:\WINDOWS\SYSTEM32". Example of application class path is "D:\Program Files\Microsoft Office".

Table (3.1) presents the summary of the above mentioned assembled information.

**Table (3.1)** Aggregated information by Aggregation unit

The providing module	Incoming information	Computed information	Description
Initialization module	ACL	Permission	User access type for specific shared file/folder.
Initialization module	List of users	User class	User type to identify the user privilege.
Sniffing module	List of Events	Object class	File/Folder type.
Sniffing module	List of Events	Action type	One of the file system application services.
Sniffing module	List of Events	NoOfTry value	Number of event occurrences.

Algorithm (3.17) presents the function of get permission from ACL.



**Algorithm (3.17) Get Permission**

**Goal:** determine the access type of particular object (file or folder) depend on ACL.

**Input:** UsersList: List of registered users.

ACL: Access Control List.

Event: represents an occurred event.

**Output:** Perm: Indicator to user access type which is either **1** (allowed) or **0** (denied).

**Step1:** ActionType is indicating to the type of event action.

**If** Event.Action is *Open* **Then**                    ActionType = 1

**Else If** Event.Action is *Modify* **Then**                    ActionType = 2

**Else If** Event.Action is *Delete* **Then**                    ActionType = 4

**Else If** Event.Action is *Copy* **Then**                    ActionType = 8

**Step2:** *Split* Event.Path using **Split** function according to "\" character into array of folders' names called PathArr[] with length PathL , and then add the object name to the PathArr, then add object name to the array.

PathL = **Split** (Event.Path , "\" , PathArr)

PathArr[PathL] = Event.File/Folder

PathArr[PathL] = Event.File/Folder

**Step3:** *Initialize* the variables

**Set** TempL = ACL

i = 1

**Step4:** **While** (i <= PathL *AND* TempL <> Null) **Do**

**If** (TempL.ObjectName = PathArr[i] )**Then**

**If** i = PathL **Then Goto Step5**

**Set** TempL = TempL.Son

i = i +1

**Else Set** TempL = TempL.Brother

**End Of While**

**Step5:** *Get* the index of the Event.Subject from UsersList.

InX = UserList ( Event.subject)

**Step6:** *Get* the permission type of InX user for TempL node in the ACL.

Perm = (TempL.Permission(InX) *AND* ActionType)

**If** Perm = ActionType **Then** Perm = 1

**Else** Perm = 0

**Step7:** **Return** (Perm)

## 2. Decision Making Unit

The main function of this unit is to compute an alarm level based on saved primitive values, and generates a notification message that will be sent to Alarm module, by applying specific rules depending on the provided information from Aggregation unit. Rules are the core of the Detection module. Rules are defined as conditional statements that decide if specific event considered as threat (intrusion) or not.

The primitive values of the alarm level are range from 1 to 9 as illustrated in Table (3.2) depending on three factors which are *user class*, *object class*, and *action type*. These values are estimated according to the problem environment needs. For example, any one shouldn't access system class resources. Therefore, the alarm level for any threat may damage system resources is set to 9 (i.e. maximum alarm level).

The primitive values will be increased depending on the number of occurrences, which is represented by NoOfTry field, of the same event within the period  $t$ .

**Table (3.2)** Alarm level values

User Object Class	Student				Employee				Visitor			
	Open	modify	Delete	Copy	open	modify	Delete	Copy	open	modify	Delete	Copy
<b>System</b>	3	9	9	3	3	9	9	3	3	9	9	3
	2	8	8	2	2	8	8	2	3	8	8	3
<b>Application</b>	1	7	7	1	1	7	7	1	2	7	7	2
	1	7	7	1	1	7	7	1	2	7	7	2

In addition to alarm level, detection rules supply notification message. Notification message has information that guide the Administrator to identify the threat. Notification message describes the occurred threat by providing information related to the threat. These information are: user class, action type, and object class. In some cases, additional message is considered. This case is occurred when the same user performs different types of attacks on the same object. Algorithm (3.18) presents the work of the detection rules.

### Algorithm (3.18) Decision Making

**Goal:** Decide if the Event(s) represent a threat or not, depends on the pre-information.

**Input:**

Index: Index to the current event.

Event[]: List of occurred events.

Permission: Represents access type of particular object (0 or 1).

ResourceClass: String represents file/folder class.

UserClass: String represents type of user depends on users information list.

UsersList: List of registered users.

AlarmLevels: 3-dimensional array of Alarm level that represents **Table (3.2)**.

First index represents the ResourceClass, second index represents the UserClass, and third index represent the action type.

UserResources: List of user resources, which are defined in initialization stage.

**Output:**

Alarm\_Level : Threat level that focused on host.

Notify\_Message : description of the threat.

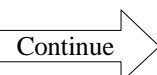
**Step1:** Initialize output.

Alarm\_Level = 0

Notify\_Message = ""

**Step2:** Get UserClass depending on UsersList and Event[index].Source username.

**Step3:** Get ResourceClass depending on Event[index].Path,Event[index].File/Folder, and UserResources.



**Step4:** Get the permission of Event[index]

Call **Algorithm (3.17)** and put result into Permission.

**Step5:** If Permission = 1 **Then GOTO Step10** // Authorized user

**Step6:** Get Alarm\_Level value from AlarmLevels array

(1) Determines ResourceClass number (R\_C).

**If** (ResourceClass = "System") **Then** R\_C =1

**Else If** (ResourceClass = "Application") **Then** R\_C =2

**Else** R\_C = 3

(2) Determines UserClass number (U\_C).

**If** (UserClass = "Student") **Then** U\_C =1

**Else If** (UserClass = "Employee") **Then** U\_C =2

**Else** U\_C = 3

(3) Determines Action type according to Event[Index].Action field.

**If** (Event[Index].Action = "\_open file\_") **Then**

Action =1 // open

**Else If** (Event[Index].Action = "\_rename directory\_") **OR**

(Event[Index].Action = "\_rename file\_") **OR**

(Event[Index].Action = "\_create directory\_") **OR**

(Event[Index].Action = "\_create file\_") **OR**

(Event[Index].Action = "\_write to file\_") **Then**

Action = 2 // modify

**Else If** (Event[Index].Action = "\_delete directory\_") **OR**

(Event[Index].Action = "\_delete file\_") **Then**

Action = 3 // delete

**Else If** (Event[Index].Action = "\_copy file\_") **Then**

Action = 4 // copy

(4) **Set** alarm level value from its position in AlarmLevels.

Alarm\_Level = AlarmLevels [R\_C , U\_C].Values[Action]

**Step7:** Check Event[Index].NoOfTry field.

**If** (Event[Index].NoOfTry > 1) **Then**

Call **Algorithm (3.19)** and put result into Alarm\_Level.

**Step8:** Generate notification message depending on UserClass, ResourceClass, and Event[Index].Action field.





**Algorithm (3.19) Update Alarm Level Value**

**Goal:** update the value of Alarm level depends on NoOfTry field.

**Const:**

t : represent the time threshold value for counting lowest difference to the Event.Dates&times between times of occurrences.

**Input:**

Index: Index to the current event.

Event[] : List of occurred Event.

Alarm\_Level<sub>in</sub> : Threat level that focused on host.

UserClass class of user

ResourceClass: class of shared resource

**Output:**

Alarm\_Level<sub>out</sub> : Alarm level which updated.

**Step1:** Count = 0

**For** i = Event[Index].NoOfTry.No **DownTo** 2

**If** ABS (Event[Index].NoOfTry.Times(i) -Event[Index].NoOfTry.Times(i-1)) < t

**Then** Count = Count + 1

**Else**

**Exit For** loop

**End For** i

**Step2:** // Depends on the following points, increase the value of Alarm\_Level<sub>in</sub>:

(1) Value of Count that represent the active repetition of Event

(2) Event[Index].Action

(3) UserClass

(4) ResourceClass

(5) Relationship between values of alarm level in **Table (3.1)** //

// check the deletion and modification actions.

**If** Event[Index].Action = "\_delete directory\_" *OR*

Event[Index].Action = "\_delete file\_" *OR*

Event[Index].Action = "\_rename directory\_" *OR*

Event[Index].Action = "\_rename file\_" *OR*

Event[Index].Action = "\_create directory\_" *OR*

Event[Index].Action = "\_create file\_" *OR*

Continue 

```

Event[Index].Action = "_write to file_"
Then Modify_Delete = True
Else Modify_Delete = False
If ResourceClass = "System" Then
    If Modify_Delete= True Then
        Alarm_Levelout = Alarm_Levelin + Count
    Else // open file or copy actions
        Alarm_Levelout = Alarm_Levelin + int (Count / 3)
Else If ResourceClass = "Application" Then
    If Modify_Delete= True Then
        Alarm_Levelout = Alarm_Levelin + int (Count * (8/9))
    Else // open file or copy actions
        If (UserClass = "Student") Or (UserClass = "Employee") Then
            Alarm_Levelout = Alarm_Levelin + int ((Count * (8/9)) / 4)
        Else
            Alarm_Levelout= Alarm_Levelin +int((Count* (8/9)) / (8/3))
Else If ResourceClass = "User" Then
    If Modify_Delete= True Then
        Alarm_Levelout = Alarm_Levelin + int (Count * (7/9))
    Else // open file or copy actions
        If (UserClass = "Student") Or (UserClass = "Employee") Then
            Alarm_Levelout = Alarm_Levelin + int ((Count * (7/9)) / 7)
        Else
            Alarm_Levelout = Alarm_Levelin + int ((Count * (7/9)) / (7/2))
Step3: Return (Alarm_Levelout).

```

## E. Alarm Module

The main function of Alarm module is to generate *Alarm Message* for the malicious event(s) and send it to the Administrator subsystem.

The generation of the alarm messages is done according to some rules, which are called *alarm rules*. These rules depend on three factors, which are: alarm level value, notification message, and the information of the

occurred event. These factors come from Detection module. The format of Alarm rule is illustrated in Figure (3.7).

Input	Input	Input	Output
<b>Alarm level</b>	<b>Occurred Event</b>	<b>Notification Message</b>	<b>Alarm Message</b>

**Figure (3.7)** Alarm rule format

Alarm level value is used to represent the type of primitive alarm message, see Table (3.3).

**Table (3.3)** Types of Primitive Alarm Messages

<b>Alarm level value</b>	<b>Primitive Alarm Message</b>
1	"_not effective threat_"
2 , 3	"_low threat level_"
4 , 5 , 6	"_medium threat level_"
7 , 8	"_high threat level_"
9	"_computer risk_"

The Event fields provide many important information about the threat that will be attached to the primitive alarm message. These information are: subject username, destination PC's IP, object path, object name, and date & time of threat.

Notify\_Message contains user class, action type, object class, and additional message for special case of attack. Notify\_Message will be attached to the alarm message as follow:

$$Alarm\_Message_{out} = Alarm\_Message_{in} + Notify\_Message$$



### 3.3.2 Administrator Subsystem

The Administrator subsystem is the control unit of the monitoring system. It is installed on one of the LAN computers from which the Administrator monitors the LAN. The Administrator subsystem is divided into four modules as follow:

#### A. Check Administrator Authority Module

This module is used to check the Administrator authenticity before opening the main window of Administrator subsystem. This module, as explained in Algorithm (3.20), gives the Administrator three trails to enter the correct username and password. If the user fails, the Administrator subsystem execution will terminated.

#### Algorithm (3.20) Check Administrator Authenticity

**Goal:** Check the Administrator authenticity.

**Const:** Administrator\_ID : Administrator username.

          Password : Administrator password.

**Input:** Admin\_ID: the name of Administrator.

          Pass: the password of Administrator.

**Output:** Flag: Flag represent Success or Fail.

**Step1:** *Initialize* the variables

          i = 1

          Flag = False

**Step2:** *Check* if the Administrator exceeds his/her three trails.

**If** i > 3 **Then GOTO Step5**

**Step3:** **Input** Admin\_ID and Pass.

**Step4:** **If** (Administrator\_ID = Admin\_ID) *And* (Password = Pass) **Then**

          Flag = True

**Else**

          i = i + 1

**GOTO Step2**

**Step5:** **Return** (Flag).

## B. Initialization Module

This module executed synchronously with the Initialization module of the Clients subsystems. This module has 3-functions, which are: permission list generation, clients list generation, and registration configuration. Therefore, this module is classified into three units according to their functions: *Permission Unit*, *Users Information Unit*, and *Registration Database Unit (RegDB Unit)*.

### 1. Permission Unit

The Administrator subsystem receives the shared resources lists from each client subsystem. Each list contains paths and names of shared resources. These lists will be stored in temporary files on Administrator PC. The Administrator subsystem allows the Administrator to set the permissions of these shared resources.

The permission types are four; *open*, *modify*, *delete*, and *copy*. These four types include inherently 9 actions which could be performed by any network message as shown in Table (3.4). Each permission is coded with a number that indicate the permission type. These numbers are represented by 4-bits, each bit indicates to a specific permission type.

**Table (3.4)** Permissions types

Permission		File system application services
Code	Type	
0000	Not set	all operations are unauthorized
0001	Open	open file
0010	Modify	write to file, rename file, rename folder, create file, and create folder
0100	Delete	delete file and delete folder
1000	Copy	copy file

The Administrator follows the following steps to set a permission for any object:

- (1) Choose the list of shared resources from the shared lists that arrived from the Clients subsystems.
- (2) Choose an object from the above selected list.
- (3) Choose user from list of registered users.
- (4) Choose particular type(s) of permission. (i.e. access type)
- (5) Repeat steps 3-4 to all registered users.

The Administrator determines type of permission depends on both **object class** and **user class**. The Administrator may set one or more permissions for specified object. If the Administrator does not determine the type of permission for specific object, its permission will set to zero, which means all users are denied and not allowed access this object.

For example, if the Permission of Object-A is [9 for User1] , [11 for User2] and Permission of Object-B is 0. That is mean:

User1 is authorized to *Open* and *Copy* Object-A

User2 is authorized to *Open*, *Modify*, and *Copy* Object-A

All Users is **un**authorized to access Object-B

When the permissions of a particular shared resources list are set, the Permission unit saves this list with their permissions into new file, which is called Access Control List (ACL) for these shared resources. ACL is list of all users-objects permissions. Each item in this list formatted as shown in Figure (3.8).

Object		List of authorized users (N users) with their permissions							
Path	Name	User-1	Permission-1	User-2	Permission-2	...	...	User-N	Permission-N

**Figure (3.8)** ACL format

Each ACL will be sent to its Client subsystem sender, during initialization stage, to be utilized later by Detection module.

## 2. Users Information Unit

The main function of this unit is to generate list of FMS registered users depending on information of the successful user registration process.

The list of users contains detailed information about each user. These information are: *PC name, PC's IP, PC user account name, username, and user type*. These information comes from the Registration Database unit after the success of the registration process.

The following modules and units utilize the list of users:

- (1) Permission Unit: to identify the information of each user during permission evaluation.
- (2) RegDB Unit: to check if the user aims to register with existing username.
- (3) Response Module: to present the attacker information such as PC name and PC's.
- (4) Initialization and Reconfiguration module of Client subsystem: to send the list of users to the Detection module to identify the user class during detection process.

This list will be updated after each successful registration process, so it will be stored in a database that called *Users' Info DB*.

## 3. Registration Database Unit

Registration Database holds all identification information of all users. Registration database represent list of names, IDs, user classes, or any information can be used during the registration stage. This database created, and updated at this Unit.

When allowed user aimed to register into FMS in Register Unit of Authentication Module in the Client subsystem; his/her registration information will received here. Algorithm (3.21) explains how to check these information to accept the registration or deny it.

If the user registration information are accepted, then, these information will be sent to Users Information unit to add the user to the list of registered users.

**Algorithm (3.21) Check Registration**

**Goal:** Check the registration information received from the Client subsystem at Register process.

**Const:** UserinfoDB: Database contains list of registered users information.

RegDB: Database contains lists of users identification information.

**Input:** Register\_information: represent Username, Pass, User\_Type, and Additional\_Info.

**Output:** Flag: Flag represent Success or Fail.

**Step1:** **If** (Username doesn't exist in UserinfoDB) *And*  
(User\_Type and Additional\_Info match the information saved at  
Registration DB) **Then**  
    Add Register\_information (Username, Pass, User\_Type) to UserinfoDB  
    **Return** (Success)  
**Else**  
    **Return** (Fail)

**Step2:** End.

Also, when user tries to login into FMS through the Login Unit; his/her login information will received here. Algorithm (3.22) explains how to check the integrity of the user.

### Algorithm (3.22) Check Login Process

**Goal:** Check the Login information that received from the Client subsystem at Login process.

**Const:** UserinfoDB : Database contains List of registered users' information.

**Input:** Username: Entered Username at Login Process.

Pass: Entered Password at Login Process.

**Output:** Flag: Flag represents Success or Fail.

**Step1:** **If** (Username and Pass are match the saved information at UserinfoDB) **Then**

**Return**(Success)

**Else**

**Return** (Fail)

**Step2:** **End.**

## C. Monitoring Module

This module has two main functions; receive the alarm messages and create users profiles according to the alarm messages. To perform these functions, this module consists of two units; *Alarm Messages* and *Users Profiles* units.

### 1. Alarm Messages Unit

The function of this unit is to receive the alarm messages from each Client subsystem. These messages may be displayed as list of alarms to the Administrator as *Online* mode of monitoring or may be saved into status file as *Offline* mode. FMS allows the Administrator to select which mode of monitoring to be executed (which is either Online or Offline modes). This characteristic adds more flexibility to the system.

Each alarm message contains level of the occurred threat. Hence, not all received messages should be displayed in the same style. The message

of the highest level of threat will be displayed with more active style, like play sound to notify the Administrator for this case of threat.

The received messages will be utilized by the Users Profiles unit to find out the suspicious behaviour for each attacker. Therefore, this unit will send a copy from each received alarm message to the Users Profiles unit.

## 2. Users Profiles Unit

Each alarm message contains details about the threat subject. These details are username, user type, PC name, user PC account name, and PC's IP. Alarm messages are classified into groups according to the subjects of threats. These groups, which are called Users profiles, show the suspicious behaviors of users due to their suspicious actions.

Each user's profile contains *user threat level*, and description of his/her attacks. User threat level is computed according the alarm messages of highest threat level that related to this user.

Users' profiles are utilized by Response module to warn misbehavior users.

## D. Response Module

The main function of this module is taking a response in case of detecting intrusion. In FMS, the response is represented as *Warning Message* that warns the suspicion user for his/her misuse. The Warning Message depends on the received alarm messages and users profiles.

Alarm message describes the occurred threat, while user profile describes the suspicious user behaviour. Depending on these information, the Administrator chooses a suitable Warning Message to be sent to the suspicion user, as shown in Algorithm (3.23).

### **Algorithm (3.23) Response**

**Goal:** Send warning messages to suspicious user as result for its misuse.

**Input:**

Alarm\_MSG: Alarm messages from Alarm Module at Client subsystem.

UsersList: List of registered users.

User\_Profile: suspicious user profile.

**Output:**

Warn\_Msg: Warning Message.

**Step1:** Depending on the Alarm\_MSG and User\_Profile, the Administrator, *Choose* a proper message to be sent to the suspicious user and set this message in Warn\_MSG.

**Step2:** Send Warn\_MSG to the user that has been identified from Alarm\_MSG and Users\_List.

**Step3:** End.



## 2.1 Introduction

The world dictionary defines a network as a "group of computers and associated devices that are connected by communications facilities".

There are three roles for computers in networks [Jai02]:

1. Clients, which use but don't provide network resources.
2. Servers, which provide network resources.
3. Peers, which both use and provide network resources.

In this chapter, network categories, protocols, and network monitoring for intrusion detection systems that are published in the literature have been exposed.

## 2.2 Network Categories

Based on the roles of the computers attached to them, networks are divided into three types [Jai02]:

1. Server-based (also called client-server).
2. Peer (also called peer-to-peer).
3. Hybrid network (is a client server network that also has peers sharing resources and most networks are hybrid networks).

The above mentioned types of networks will be described in the following sections.

### 2.2.1 Server-Based Networks

Server-based networks are defined by the presence of dedicated servers on a network that provide security and resources to the network.

Server-based (or client-server) networks divide processing tasks between clients and servers. Clients (often called "front end") request services, such as file storage and printing, and servers (often called "back end") deliver them. Server computers typically are more powerful than client computers, or are optimized to function as servers [Zac01].

### 2.2.2 Peer Networks

In a peer-to-peer network, there are no dedicated servers, and there is no hierarchy among the computers. All the computers are equal and therefore they are known as peers. Each computer functions as both a client and a server, and there is no administrator responsible for the entire network. The user at each computer determines what data on that computer is shared on the network. Peer-to-peer networks are also called *workgroups*. The term "workgroup" implies a small group of people.

Peers are also not optimized to share resources. Generally, when a number of users are accessing resources on a peer, the user of that peer will notice significantly degraded performance. Peers also generally have licensing limitations that prevent more than a small number of users from simultaneously accessing resources [She01].

### 2.2.3 Hybrid Networks

Hybrid networks have all three types of computers operating on them and generally have active directory domains and workgroups. This means that while most shared resources are located on servers, network users still have access to any resources being shared by peers in your workgroup. It also means network users do not have to log on to the domain controller to access workgroup resources being shared by peers [She01].

## 2.3 Network Protocol Model

A *protocol* is a set of rules for communication. A protocol defines the shape of a *packet* (Packet refers to the information that is sent over a network through network communications which originate at a source, and sent to a destination) that will be transmitted across the network, as well as all the fields within the packet and how they should be interoperated.

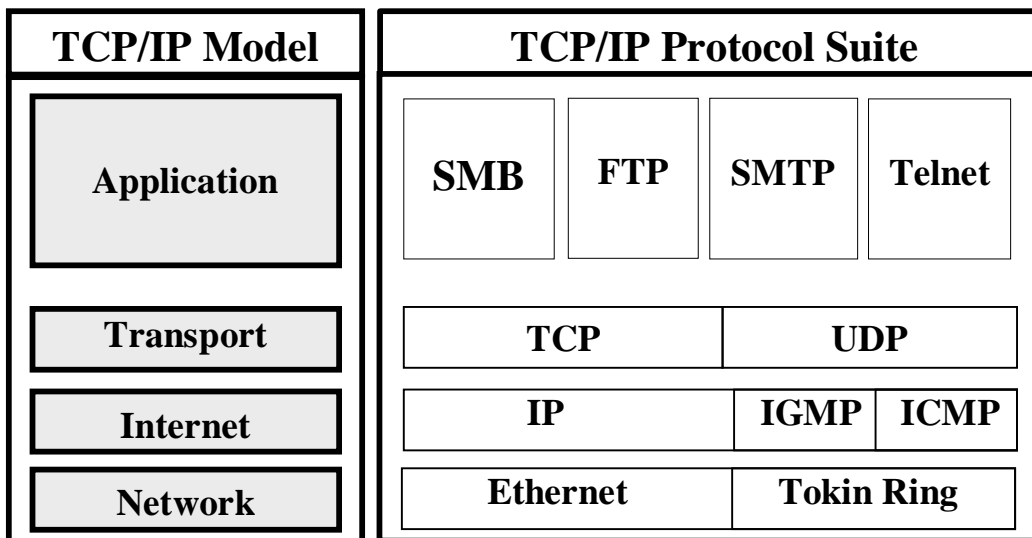
A *protocol suite* is a group of protocols that evolves together, that used in the same environment or created in the same company, such as the Internet Protocol suite and NetWare suite.

Each protocol suite has many protocols each performs a specific function. Some of these protocols perform the same function as another protocol in another protocol suite [Dul98].

The most commonly referenced protocol model is the OSI (Open Systems Interconnect) model, it was developed by the ISO (International Standardization Organization). The OSI model referred to as network protocol model [Dav04].

Internet is the ultimate example of an internetwork (collection of interconnected LANs). The protocols that make up the internet protocol suite, the best known being TCP (Transmission Control Protocol) and IP (Internet Protocol), have become de facto standards because of the success of the Internet. The entire protocol suite is sometimes referred to as TCP/IP suite as shown in Figure (2.1).

The four layers of the TCP/IP model, as shown in Figure (2.1), are as follow [Dul98]:



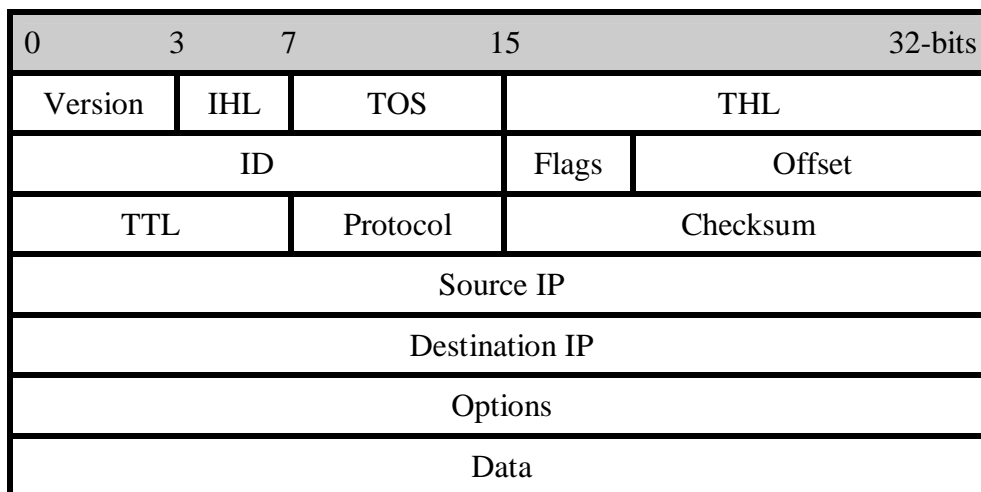
**Figure (2.1)** TCP/IP Model

**1. Network (Link) Layer:** The network layer is responsible for communication directly with the network. It must understand the network architecture being used, such as token ring or Ethernet, and provide an interface allowing the Internet layer to communicate with it. The Internet layer is responsible for communication directly with the network layer.

**2. Internet Layer:** The Internet layer is primarily concerned with routing and delivery of packets through the Internet Protocol (IP). All the protocols in the Transport layer must use IP to send data. The Internet Protocol includes rules for how to address and direct packets, fragment and reassemble packets, provide security information, and identify the type of service being used.

IP is not a connection-based protocol, however, it does not guarantee that packets transmitted on to the wire will not be lost, damaged, duplicated, or out of order because these are the responsibility of higher layers of the networking model, such as the Transport layer or the Application layer.

The header that IP applies to the data (it receives from the transport layer protocol) is typically 20 bytes long. The datagram format is shown in Figure (2.2) [Mic03].



**Figure (2.2)** IP header format

The most commonly used fields of IP header are:

- a) **Version (4- bits):** This field specifies the version of the IP protocol used to create the datagram, which are either 4 or 6. The version in current use is 4, which can support up to 2<sup>32</sup> addresses (32 bits), while 6 is developed as a replacement that can support up to 2<sup>128</sup> addresses (128 bits) to fulfill future needs with better security and network related features.
- b) **Internet Header Length (IHL, 4 bits):** This field specifies the length of the datagram's header, in 32-bit (4-byte) words. The typical length of a datagram header is five word (20 bytes), but if the datagram includes additional options, it can be longer, which is the reason for having this field.
- c) **Type Of Service (TOS, 1 byte):** This field contains a code that specifies the services priority for the datagram. This is a rarely used feature that enables a system to assign a priority to a datagram that routers observe while forwarding it through an internetwork. The values provide a trade-off a many delay, throughput, and reliability.
- d) **Total Header Length (THL, 2 bytes):** This field specifies the length of datagram, including that of the datagram fields and all of the header fields, in bytes.
- e) **Identification (ID, 2 bytes):** This fields contains a value that an equally identifies the datagram. The destination system uses this value to reassemble datagram that have been fragmented during transmission.
- f) **Flags (3 bits):** This field contains bits used to regulate the datagram fragmentation processes.
- g) **Fragment Offset (Offset, 13-bits):** When a datagram is fragmented, the system inserts a value in this field that identifies this fragment's place in the datagram.

- h) **Time To Live (TTL, 1 byte):** This field specifies the number of networks that the datagram should be permitted to travel through on the way to its destination. Each router that forwards the datagram reduces the value of this field by one. If the value reaches zero, the datagram is discarded.
- i) **Protocol (1 byte):** This field contains a code that identifies the protocol that generated the information found in the data field such as 6 represents TCP and 17 represents UDP.
- j) **Header Checksum (Checksum, 2 bytes):** This field contains a checksum value computed on the IP header fields only (and not the contents of the data field) for the purpose of error detection.
- k) **Source IP Address (Source IP, 4 bytes):** This field specifies the IP addresses of the system that generated the datagram.
- l) **Destination IP Address (Destination IP, 4 bytes):** This field specifies the IP addresses of the system for which the datagram is sent.
- m) **Options (variable):** This field is present only when the datagram contains one or more of the 16 available IP options. The size and content of the field depends on the number and the nature of the options.
- n) **Data (variable):** This field contains the information generated by the protocol specified in the protocol fields. The size of the field depends on the data link layer protocols used by the network over which the system will transmit the datagram.

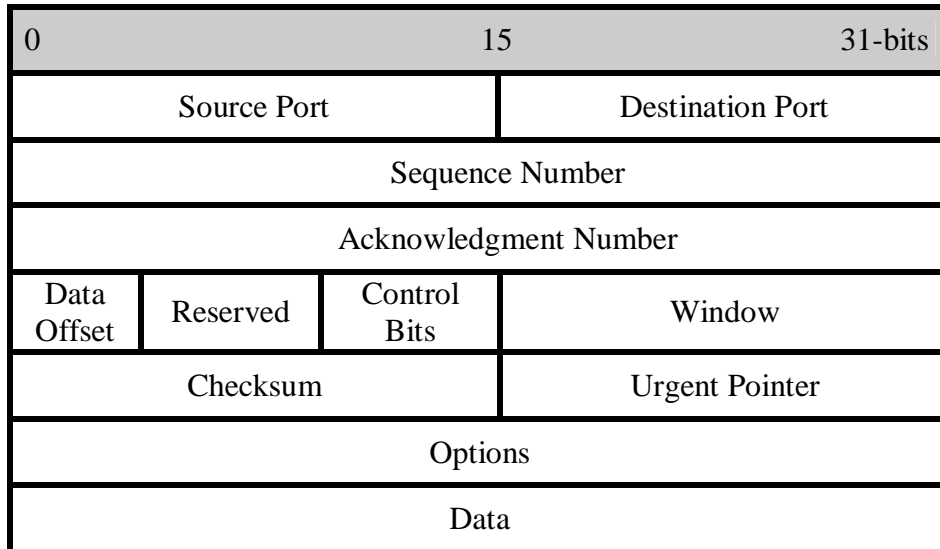
Other protocols that exist in the Internet layer are the Internet Control Messaging Protocol (ICMP) that is used for network error reporting and generating messages that require attention, and Internet Group Management Protocol (IGMP) that is used to support multicasting.

**3. Transport Layer:** The transport layer provides service of transporting application layer data between the client and server sides of an application. The TCP/IP suite includes two protocol at this layer, the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP), which provide connection-oriented and connectionless data transfer, respectively.

TCP is the main transport layer protocol of the Internet protocol suite. It also provides addressing (with service addresses) services at the network layer. TCP provides reliable, full-duplex, connection-oriented transport service to upper-layer protocols.

In many cases, the Application layer protocol passes more data to TCP than can fit into a single packet, so TCP splits the data into smaller pieces. Each piece is called a *segment*, and the segments that comprise a single transaction are known collectively as a *sequence*. Each segment receives its own TCP header and is passed down to the Internet layer for transmission in a separate datagram. When all of the segments arrive at the destination, the receiving computer reassembles them into the original sequence [Mic03].

The header that TCP applies to the data (it receives from the Application layer protocol) is typically 20 bytes long. The datagram format is shown in Figure (2.3).



**Figure (2.3)** TCP header format [Mic03]

The most commonly used fields of TCP header are:

- a) **Source Port (2 bytes):** This field identifies the process on the transmitting system that generated the information carried in the data filed.
- b) **Destination Port (2 bytes):** This field identifies the processes on the receiving system for which the information in the data filed is intended.
- c) **Sequence Number (4 bytes):** This field identifies the location of the data in this segment in relation to the entire sequence.
- d) **Acknowledgment Message (4 bytes):** In acknowledgment (ACK) messages, this field identifies the sequence number of the next segment expected by the receiving system.
- e) **Data Offset (4 bits):** This field identifies the number of 4-byte words in the TCP header.
- f) **Reserved (6 bits):** This field unused.
- g) **Control Bits (6 bits):** This field contains 6 flag bits that identify the functions of the message.
- h) **Window (2 bytes):** This field identifies how many bytes the computer capable of accepting from the connected system.



- i) **Checksum (2 bytes):** This field contains the results of the Cyclical Redundancy Check (CRC) computation performed by the transmitting system, and used by the receiving system to detect error in the TCP header, data, and parts of the IP header.
- j) **Urgent Pointer (2 bytes):** When the urgent (URG) control bit is present, this field indicates which part of the data in the segment the receiver should be treated as urgent.
- k) **Options (variable):** This field may contain information related to optional TCP connection configuration features.
- l) **Data (variable):** This field may contain one segment of an information sequence generated by an application layer protocol.

**4. Application Layer:** The Application layer handles details of the particular end-user applications. The TCP/IP protocols at this layer can take several different forms. Some protocols, such as the File Transfer Protocol (FTP), can be applications in themselves, whereas others, such as Hypertext Transfer Protocol (HTTP), provide services to applications.

Additionally, in this layer there are Network File System (NFS). Also, may be called as the Distributed File System (DFS) or Internet File System (IFS). The most common Network File Systems are: 1-Andrew File System (AFS), 2-Distributed File System (DFS), and 3-*Server Message Blocks (SMB)*, which is explained in the next section.

## 2.4 Server Message Block Protocol (SMB)

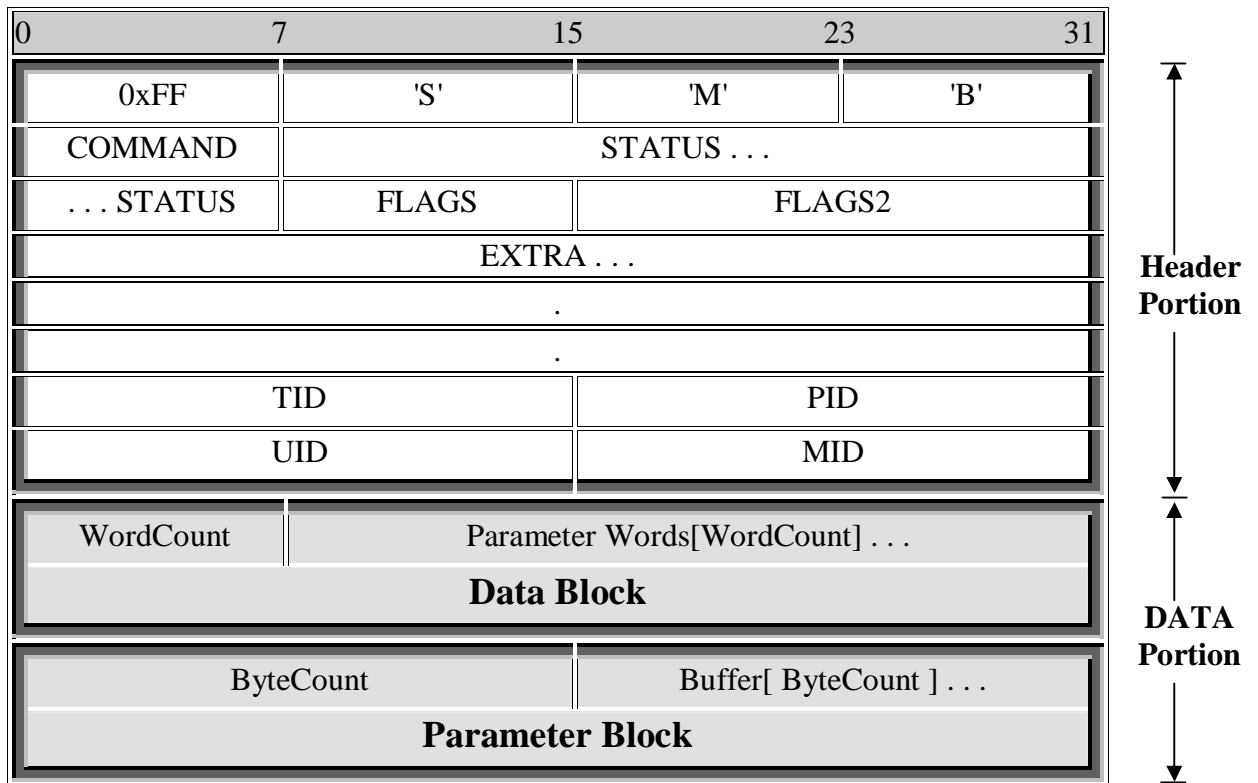
Server Message Block (SMB) is a network protocol whose most common used in sharing files on a LAN at application layer. This protocol allows a client to manipulate files just as if they were on the local computer.

The SMB is a client-server or request-response protocol. Operations such as read, write, create, delete, and rename are supported. The only difference being that the files are not on the local computer and are actually on a remote server [You01].

The SMB protocol works by sending packets from the client to the server, each packet is typically a basic request of some kinds of file processes, such as open file, close file, or read file. The server then receives the packets, checks to see if the request is legal, verifies the client has the appropriate file permission, and finally executes the request and returns a response packet to the client. The client then parses the response packet and can determine whether or not the initial request was successful.

SMB is a fairly hi-level network protocol. In the OSI (Open Systems Interconnect) model, it is probably best described at the Application/Presentation layer. This mean SMB relies on other protocols for transport. The most common protocol used for reliable transport is NetBIOS (refer to Network Basic Input/Output System). SMB message format will be illustrated in Figure (2.4), which is composed of three basic parts [Mic04]:

1. The header,
2. The parameter Block, and
3. The data block.



**Figure (2.4)** SMB Message format

### 2.4.1 SMB Header

SMB header consists of a number of fields which are explained in the following sections.

#### A. SMB Header Fields

The most common used fields of SMB header are [Mic04]:

- **Protocol ID:** The first four bytes are the protocol identifier, which always has the same value as shown in Figure (2.5).

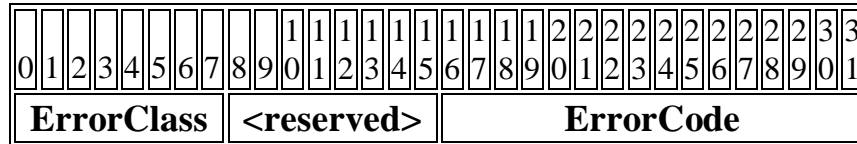
1 <sup>st</sup> Byte	2 <sup>nd</sup> Byte	3 <sup>rd</sup> Byte	4 <sup>th</sup> Byte
0xFF = 255	ASCII of 'S' = 83	ASCII of 'M' = 77	ASCII of 'B' = 66

**Figure (2.5)** SMB Protocol ID field

- **COMMAND Field:** tells us what kind of SMB we are looking at, for example: when rename a file or directory client request or server response

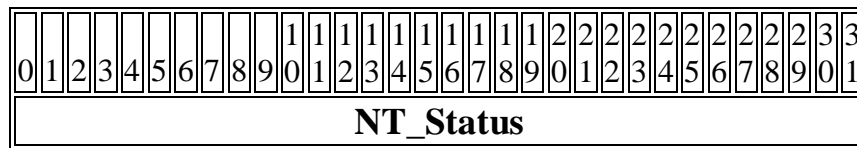
(SMB message), COMMAND field take the value 7. This field will be discussed in section 2.4.1.2.

- **STATUS Field:** There are two possible error code formats, DOS & OS/2 and NT-STATUS. DOS & OS/2 use 16-bits error codes, grouped into classes, as presented in Figure (2.6).



**Figure (2.6)** DOS & OS/2 error code format

While WindowsNT introduced a new set of 32-bit error codes, known as NT\_STATUS codes. These use the entire status field to hold the NT\_Status value, as shown in Figure (2.7).



**Figure (2.7)** NT\_STATUS error code format

With two errors code formats from which to choose, the client and server must confer to decide which set will be used.

- **FLAGS Field** that characterizing the SMB request/response, if bit 7 is set, this packet is a server response.
- **FLAGS2 Field:** 16-bit flag field defining the capabilities of the client/server transaction.
- **EXTRA Field:** this field takes up most of the remaining formerly reserved bytes. It contains 12-bytes filled by zeros.
- **TID:** acronym to "Tree ID". TID field is used to identify connections to shares once they have been established.

- **PID:** acronym to "Process ID". This value is set by the client, and is intended as an identifier for the process sending the SMB request.
- **UID:** acronym to "User ID". The server assigns it after the user has authenticated and is valid until the user logs off.
- **MID:** acronym to "Multiplex ID". This is used by the client to keep track of multiple outstanding requests.

## B. SMB Command Field

The COMMAND field determines the DATA portion of the SMB header, so according to the COMMAND field value, the DATA portion will be structured. Therefore, DATA portion may take several structures. These structures are called *Formats* of the SMB commands. The first data portion field is WORDCOUNT, being the same field as WordCount in the SMB Header.

The most widely SMB *commands* that carry more significant information about most common file system operations are listed bellow [You01]:

- **Delete Directory Command:** This command is send to delete a directory and the directory name with its Path is passed. The abbreviation of this command is SMB\_COM\_DEL\_DIR. When SMB Command field takes the value 0x01, the data portion of SMB header will be formatted as Table (2.1).

**Table (2.1)** Data Portion fields of SMB\_COM\_DEL\_DIR

Field name	Displacement (bytes)	Length (bytes)	Description
WordCount	0	1	The value is zero.
ByteCount	1	2	Count of data bytes The value is greater than 0.
BufferFormat	3	1	The value is 0x04.
DirectoryName[]	4	Variable	Directory name.

- **Rename File/Directory Command:** This command is send to changes the name of file or directory. The abbreviation of this command is SMB\_COM\_RENAME. When SMB Command field takes the value 0x07, the Data Portion of SMB header will be formatted as Table (2.2).

**Table (2.2)** Data Portion fields of SMB\_COM\_Rename

Field name	Displacement (bytes)	Length (bytes)	Description
WordCount	0	1	Count of parameter words. The value is 1.
SearchAttributes	1	2	Target file attributes
ByteCount	3	2	Count of data bytes. The value is greater than 4.
BufferFormat1	5	1	The value is 0x04.
OldFileName	6	Variable	Old file name.
BufferFormat2	*	1	Second buffer format. The value is 0x04.
NewFileName	*	Variable	New file name.

- **NT\_CreateAndX Command:** When SMB command took the value 0xA2, the SMB command is NT\_CreateAndX. The abbreviation of this command is SMB\_COM\_NT\_CreateAndX. The data portion of the SMB request at this command will pass many of the file system application operations depend on the fields values of SMB\_COM\_NT\_CreateAndX header. These operations may be Create file, Delete file, Copy file, Open file, or Create folder. Table (2.3) illustrates SMB data portion of this command.

**Table (2.3)** Data Portion fields of SMB\_COM\_NT\_CreateAndX

Field name	Displacement (bytes)	Length (bytes)	Description
WordCount	0	1	Count of parameter words. The value is 24.
AndXCommand	1	1	Secondary command. If no secondary command exists, the value is 0xFF.

Field name	Displacement (bytes)	Length (bytes)	Description
AndXReserved	2	1	Reserved, this value must be zero.
AndXOffset	3	1	Offset to WordCount location for the following command.
Reserved	4	1	Reserved, the value must be zero.
NameLength	5	1	Length in bytes of the Filename[] field.
Flags	6	4	Flags, see <b>Table (2.4)</b> .
RootDirectory	10	4	If non-zero, the open command is relative to this directory.
DesiredAccess	14	8	Access desired.
AllocationSiz	22	8	Initial allocation size.
FileAttribute	30	4	Flags and attributes for the file.
ShareAccess	34	4	Type of share access, see <b>Table (2.5)</b> .
CreateDisposi	38	4	Flags defining the action to take if the file already exists, see <b>Table (2.6)</b> .
CreateOptions	42	4	File create options.
Impersonation	46	2	Security QOS information. This field specifies the client impersonation level, see <b>Table (2.7)</b> .
SecurityFlags	50	1	Security tracking mode flags, see <b>Table (2.8)</b> .
ByteCount	51	2	Count of data bytes.
Filename[ ]	53	Variable	Name of file to open or create.

**Table (2.4)** SMB\_COM\_NT\_CreateAndX Flags field values

Value	Meaning
0x02	Request a dynamic lock.
0x04	Request a batch dynamic lock.
0x08	Target of open command must be a directory.
0x10	Request extended response.

**Table (2.5)** SMB\_COM\_NT\_CreateAndX ShareAccess field values

Value	Meaning
0x00000000	Prevents file sharing.
0x00000001	Can be opened for read access.
0x00000002	Can be opened for write access.
0x00000004	Can be opened for delete.

**Table (2.6)** SMB\_COM\_NT\_CreateAndX CreateDisposition field values

Value	Meaning
FILE_SUPERSEDE 0x00000000	If the file already exists, supersede it by the specified file. Otherwise, create the file.
FILE_OPEN 0x00000001	If the file already exists, return success; otherwise, fail the operation.
FILE_CREATE 0x00000002	If the file already exists, fail the operation; otherwise, create the file.
FILE_OPEN_IF 0x00000003	Open the file if it already exists; otherwise, create the file.
FILE_OVERWRITE 0x00000004	Overwrite the file if it already exists; otherwise, fail the operation.
FILE_OVERWRITE_IF 0x00000005	Overwrite the file if it already exists; otherwise, create the file.

**Table (2.7)** SMB\_COM\_NT\_CreateAndX ImpersonationLevel field values

Value	Meaning
0x0000	Anonymous
0x0001	Identification
0x0002	Impersonation
0x0003	Delegation



**Table (2.8)** SMB\_COM\_NT\_CreateAndX SecurityFlags field values

Value	Meaning
SECURITY_DYNAMIC_TRACKING 0x01	Security tracking mode is dynamic.
SECURITY_EFFECTIVE_ONLY 0x02	Only the enabled aspects of the client security context are available to the server.

- **TreeConnectAndX Command:** When SMB command took the value 0x75, the SMB command is TreeConnectAndX. The abbreviation of this command is SMB\_COM\_TreeConnectAndX. The client request for this command will pass the current path of the file/folder that processed by the one of the file system application operation or services. Table (2.9) illustrates SMB data portion of this command.

**Table (2.9)** Data Portion fields of SMB\_COM\_TreeConnectAndX

Field name	Displacement (bytes)	Length (bytes)	Description
WordCount	0	1	Count of parameter words. The value is 4.
AndXCommand	1	1	Secondary command. If no secondary command exists, the value is 0xFF.
AndXReserved	2	1	Reserved. The value must be zero
AndXOffset	3	2	Offset in bytes to the WordCount location for the following command.
Flags	5	2	Additional information. When Bit0 of the 2-bytes present, the Tree connection to <b>Tid</b> should be disconnected.
PasswordLength	7	2	Length of Password[].
ByteCount	9	2	Count of data bytes. The value is greater than 2.
Password[]	11	Variable	This is a variable length field with the length specified by PasswordLength. If a password is not used, the value is a null.

Field name	Displacement (bytes)	Length (bytes)	Description
Path[]	*	Variable	Server name and/or share name.
Service[]	*	Variable	Type of component requested, see <b>Table (2.10)</b> .

**Table (2.10)** SMB\_COM\_TreeConnectAndX Service field values

Value	Meaning
A	Disk share for PC NETWORK PROGRAM 1.0 or later.
LPT1	Printer for PC NETWORK PROGRAM 1.0 or later.
IPC	Named pipe for MICROSOFT NETWORKS 3.0 or later.
COMM	Communications device for MICROSOFT NETWORKS 3.0 or later.
?????	Any device type for MICROSOFT NETWORKS 3.0 or later.

- **SessionSetupAndX Command:** When SMB command took the value 0x73, the SMB command is SessionSetupAndX. The abbreviation of this command is SMB\_COM\_SessionSetupAndX. The client request for this command continues the user session definition begun by the request of NEGOTIATE command (SMB\_COM\_NEGOTIATE) at the 3-hand checking request and response.

The packet of this command defines the data portion of the SMB client request and response packets. Table (2.11) shows the data portion fields for request message for this command.

**Table (2.11)** Data Portion fields of SMB\_COM\_SessionSetupAndX

Field name	Displacement (bytes)	Length (bytes)	Description
WordCount	0	1	Count of parameter words. The value is 10.
AndXCommand	1	1	Secondary command. If no secondary command exists, the value is 0xFF.

Field name	Displacement (bytes)	Length (bytes)	Description
AndXReserved	2	1	The value must be zero.
AndXOffset	3	2	Offset in bytes to the WordCount location for the following command.
MaxBufferSize	5	2	Client maximum buffer size.
MaxMpxCount	7	2	Maximum count of pending multiplexed requests.
VcNumber	9	2	If this is the first VC number, the value is zero.
SessionKey	11	4	The value is valid only if VcNumber is non-zero.
PasswordLength	15	2	Length of account password.
Reserved	17	4	The value must be zero.
ByteCount	21	2	Count of data bytes.
AccountPassword[]	23	Variable	Account password.
AccountName[]	*	Variable	Name of account.
PrimaryDomain[]	*	Variable	Client primary domain.
NativeOS[]	*	Variable	Client native operating system.
NativeLANMan[]	*	Variable	Client native LAN Manager type.

For more information about the SMB commands, Appendix A present list of most common SMB commands with their codes.

## 2.4.2 Parameters Block

In the middle of the SMB message are two fields labeled WordCount and Parameter Words as shown in Figure (2.4), which are:

1. WordCount is the number of words in the Words array.
2. Words[WordCount] is SMB parameters; varies with SMB command.

The Words field is simply a block of data that is  $2 \times \text{WordCount}$  bytes in length. Each SMB message type has a different record structure that is carried in the Words block. Think of that structure as representing the

parameters passed to a function (the function identified by the SMB command code listed in the header) [Mic04].

### 2.4.3 Data Block

Following the SMB\_PARAMETERS is another block of data, the content varies in structure depending on a per-SMB header as explained in section 2.4.1.2. SMB Data Block consist of two fields as shown in Figure (2.4), they are:

1. ByteCount is the number of bytes in the Bytes field.
2. Bytes[ByteCount] is the contents varies with SMB command.

The Bytes field holds the data to be manipulated. For example, it may contain the data retrieved in response to a READ operation, or the data to be written by a WRITE operation.

## 2.5 NetBIOS Over TCP/IP

The SMB protocol is supposed to be transport independent. That is, SMB should work over any reliable transport that meets a few basic criteria. NetBIOS (Network Basic Input/Output System) is one such transport.

Port number of TCP protocol determines how the SMB works over TCP/IP. Port 139 refers to SMB is working over NetBIOS, and then NetBIOS over TCP/IP. While port 445 refers to SMB is working over TCP/IP directly.

NetBIOS is probably best described at the session layer in the OSI model. NetBIOS mainly used by Microsoft to transport SMB file service messages as shown in Figure (2.8) [Dav04].

OSI	TCP/IP	Protocols		
Application	Application	SMB		
Presentation		NetBIOS		
Session		Name Service	datagram service	Session Service
Transport	Transport	UDP		TCP
Network	Internet	IP		
Data link	Network	Ethernet		
Physical				

**Figure (2.8)** NetBIOS over TCP/IP

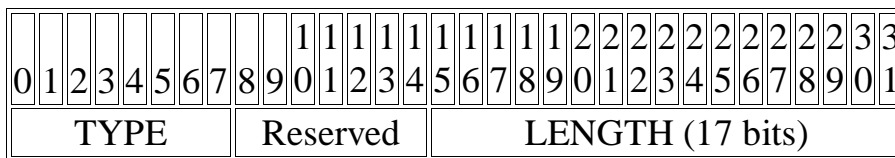
NetBIOS provides three basic services over TCP/IP suite, which are [Dav04]:

1. *NetBIOS Name service*: Name service is used to map NetBIOS names (addresses) to IP addresses in the underlying IP network. NetBIOS addressing is dynamic. Application may add names as needed, and remove those names when finished. Each node on the LAN will also have a default name, known as the Machine Name or the Workstation service name, which is typically added when NetBIOS starts. Name service uses UDP protocol at the Transport layer in the TCP/IP suite.

2. *NetBIOS Datagram service*: Datagram service provides for the delivery of NetBIOS datagram via UDP protocol. NetBIOS provides connectionless communications to handle UDP datagram. When the main function of NetBIOS is transport SMB file service messages; datagram service is

probably the second least well understood aspect of NetBIOS because correct implementation is not critical to file sharing especially.

**3. NetBIOS Session service:** Session service is used to establish and maintain point-to-point, connection-oriented NetBIOS session over TCP. Session service is the traditional transport for SMB. Applications that use NetBIOS session service run on SMB protocol at the Application layer, TCP protocol at the Transport, and IP protocol at the Internet layer in the TCP/IP model. Examples of these applications that use NetBIOS session service are file and printer sharing. Type of session is determined depend on the values of NetBIOS Session header fields. Figure (2.9) represents the NetBIOS Session header format.



**Figure (2.9)** Session Service header format

The LENGTH field contains the number of bytes of payload, and the TYPE field is used to distinguish between the six different Session Service message types, which are [RFC87]:

1. 0x00 == Session Message
2. 0x81 == Session Request
3. 0x82 == Positive Session Response
4. 0x83 == Negative Session Response
5. 0x84 == Retarget Session Response
6. 0x85 == Session Keepalive

## 2.6 Network Monitoring for Intrusion Detection

In the world of intrusion detection, researchers tend to focus on detecting attacks and clearly anomalous activity. An important component of a complete intrusion detection solution is basic network monitoring and traffic analysis. Network monitoring collects information on connections, while traffic analysis allows to see what services are being used on a network and to compare that against the activity that we should be seeing. This allows to identify unauthorized services being used within a network [Mar02].

### 2.6.1 Network Monitoring

In order to perform basic network monitoring, there is a need to collect information on traffic at various points within a network.

As networks become larger, the deployment of a single IDS could not satisfy the need for ubiquitous protection. A potentially large number of IDSs (called IDS sensors in this case) have to act in concert to trace suspicious activity through all domains of the network. Events that might indicate an intrusion have to be aggregated and analyzed on a central host. Aggregation encompasses collecting and normalizing diverse information from multiple sources providing information in an unified data format [Sol04].

One of the important methods that collect information is packet sniffing. A packet *sniffer* is computer software that can intercept and log traffic passing over a digital network or part of a network. As data streams travel back and forth over the network, the sniffer captures each packet and eventually decodes and produces it to the analyzer to analyze its content according to the appropriate specifications [Ory06].

Sniffed segment was chosen as allow to examination of the traveled

packets and in particular to filtering these packets and extract needed information to detect the threat. This segment can be shown on the TCP/IP suite, see Figure (2.10) [Moo02].

Layer Number	Layer Name	Sample Protocol
Layer1	Application	SMB, HTTP, FTP, Telnet
Layer2	Transport	TCP, UDP
Layer3	Internet	IP, IPX
Layer4	Network	Ethernet, Arcnet

**Figure (2.10)** Sniffed segment of TCP/IP (represented by shadow area)

### 2.6.2 Traffic Analysis

Before an actual analysis can take place, a correlation engine has to process the events and find relationships between them. The actual correlation can be based on a combination of time stamps, source or destination addresses, and other logical relationships established between the events [Sol04].

Protocol analysis means analyzing the behavior of protocols to determine whether one host is communicating normally with another. For example, the TCP handshake is initiated by sending a TCP SYN packet to another host. The other host responds with a SYN ACK packet, to which the originating host responds with an ACK packet. Suppose that a host sends nothing but SYN packets to another host. This is an indication of a “SYN flood” attack designed to deplete memory and other resources in the receiving host [End04].



## 2.7 Measuring the Efficiency of an IDS

The efficiency of an Intrusion Detection System can be measured by the number of *false positives* and *false negatives*. A false positive is produced if the IDS claims there has been an intrusion when there has not. A false negative occurs if the IDS fails to detect an intrusion.

This measurement determines how accurately the IDSs diagnose an attack. The impact of an IDS' accuracy is going to be organization specific. Some organizations may have high tolerance of false positive because they have staff and time to investigate them, while other organizations would rather have a system that misses attacks as long as it does not raise false alarms. Therefore, false positive may be standard measurement to the IDS in some environments, considering that the false negative is not found for perfect detection [BEn02].

False negatives can be computed by subtracting the number of accurately detected attacks from the total number of attacks sent. Whereas, false positives can be computed by subtracting the number of accurately detected attacks from the total number of alerts raised.

As result, to do measuring of efficiency to an IDS in an environment, there are some requirements that should be achieved. The following requirements should be exist in the IDS environment [Ran01]:

1. **Attacker** hosts that attacks are lunched from, and
2. **Target** hosts that the attacks are directed at, and that run the intrusion detection software and monitor a specified network segment.

The following requirements should be computed by applying IDS [Ran01]:

1. Total number of attacks sent.
2. Number of accurately detected attacks.
3. Total number of alerts raised.

# *Abstract*

Computer network technologies have grown rapidly in the last few decades. With the increased use of networked computers for critical applications, computer intrusions have been increased and became a significant threat to these systems and, thus Intrusion Detection Systems (IDS) have become essential addition to security infrastructure of most organizations.

This thesis presents the design and implementation of a Network Node Intrusion Detection System (NNIDS) that support IPv4 protocol. The name of the proposed system is chosen to be *FMS* (the acronym for **F**ile **M**onitoring **S**ystem). It detects a variety of attacks which are directed to the resources of filing system. The implied detection rules are based on matching the predefined normal behaviour of the system with the characteristics of the detected users' events.

The primary constituting system modules are: logging module which defines the users allowed to access shared resources; sniffing module that captures and decodes packets and generates a list of events; detection module that analyzes the list of events and determines the suspicious activity; and alarming module that generates alarm messages to the Administrator in case of attacks.

The system has been evaluated according to three factors accuracy, time, and memory consumption. Several simulated attacks have been sent to the proposed system to test it. Test shows that most of the attacks of the filing system can be detected with acceptable ratios of false positive and false negative values.

## *Acknowledgment*

*First, I would like to thank God, for all the blessings that have given us.*

*I would like to express my sincere gratitude and appreciation to my supervisor Dr. Abeer M. Yousif for her valuable guidance, supervision and untiring efforts during the course of this work.*

*There are no enough words to be wonderfully eloquent to thank Dr. Luay Adwar Jorj for his continuous supports and valuable guidance during period of my studies.*

*Grateful thanks for the Head of Department of Computer Science Dr. Taha S. Bashaga, staff and employees.*

*Finally, special thanks to my family especially my parents, and my friends for their continuous encouragement during the period of my studies.*

Suleiman

## A.1 Common SMB Commands Codes

The most common commands of the SMB COMMAND values represented in Table (A.1).

**Table (A.1)** Most Common SMB Commands

SMB Command	Code
SMB_COM_CREATE_DIRECTORY	0x00
SMB_COM_DELETE_DIRECTORY	0x01
SMB_COM_OPEN	0x02
SMB_COM_CREATE	0x03
SMB_COM_CLOSE	0x04
SMB_COM_FLUSH	0x05
SMB_COM_DELETE	0x06
SMB_COM_RENAME	0x07
SMB_COM_QUERY_INFORMATION	0x08
SMB_COM_SET_INFORMATION	0x09
SMB_COM_READ	0x0A
SMB_COM_WRITE	0x0B
SMB_COM_LOCK_BYTE_RANGE	0x0C
SMB_COM_UNLOCK_BYTE_RANGE	0x0D
SMB_COM_CREATE_TEMPORARY	0x0E

<b>SMB Command</b>	<b>Code</b>
SMB_COM_CREATE_NEW	0x0F
SMB_COM_CHECK_DIRECTORY	0x10
SMB_COM_PROCESS_EXIT	0x11
SMB_COM_SEEK	0x12
SMB_COM_LOCK_AND_READ	0x13
SMB_COM_WRITE_AND_UNLOCK	0x14
SMB_COM_READ_RAW	0x1A
SMB_COM_READ_MPX	0x1B
SMB_COM_READ_MPX_SECONDARY	0x1C
SMB_COM_WRITE_RAW	0x1D
SMB_COM_WRITE_MPX	0x1E
SMB_COM_WRITE_COMPLETE	0x20
SMB_COM_SET_INFORMATION2	0x22
SMB_COM_QUERY_INFORMATION2	0x23
SMB_COM_LOCKING_ANDX	0x24
SMB_COM_TRANSACTION	0x25
SMB_COM_TRANSACTION_SECONDARY	0x26
SMB_COM_IOCTL	0x27
SMB_COM_IOCTL_SECONDARY	0x28

<b>SMB Command</b>	<b>Code</b>
SMB_COM_COPY	0x29
SMB_COM_MOVE	0x2A
SMB_COM_ECHO	0x2B
SMB_COM_WRITE_AND_CLOSE	0x2C
SMB_COM_OPEN_ANDX	0x2D
SMB_COM_READ_ANDX	0x2E
SMB_COM_WRITE_ANDX	0x2F
SMB_COM_CLOSE_AND_TREE_DISC	0x31
SMB_COM_TRANSACTION2	0x32
SMB_COM_TRANSACTION2_SECONDARY	0x33
SMB_COM_FIND_CLOSE2	0x34
SMB_COM_FIND_NOTIFY_CLOSE	0x35
SMB_COM_TREE_CONNECT	0x70
SMB_COM_TREE_DISCONNECT	0x71
SMB_COM_NEGOTIATE	0x72
SMB_COM_SESSION_SETUP_ANDX	0x73
SMB_COM_LOGOFF_ANDX	0x74
SMB_COM_TREE_CONNECT_ANDX	0x75
SMB_COM_QUERY_INFORMATION_DISK	0x80

<b>SMB Command</b>	<b>Code</b>
SMB_COM_SEARCH	0x81
SMB_COM_FIND	0x82
SMB_COM_FIND_UNIQUE	0x83
SMB_COM_NT_TRANSACT	0xA0
SMB_COM_NT_TRANSACT_SECONDARY	0xA1
SMB_COM_NT_CREATE_ANDX	0xA2
SMB_COM_NT_CANCEL	0xA4
SMB_COM_NT_RENAME	0xA5
SMB_COM_OPEN_PRINT_FILE	0xC0
SMB_COM_WRITE_PRINT_FILE	0xC1
SMB_COM_CLOSE_PRINT_FILE	0xC2
SMB_COM_GET_PRINT_QUEUE	0xC3

## *Supervisor Certification*

We certify that this thesis was prepared under our supervision at the Department of Computer Science/College of Science/Al-Nahrain University, by **Suleiman Sa'adon Fawzy** as partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

### *Supervisors*

Signature:

Name: **Abeer M. Yousif**

Title: **Lecturer**

Date: / / **2008**

### *The Head of the Department Certification*

In view of the available recommendations, I forward this thesis for debate by the examination committee.

Signature:

Name: **Dr. Taha S. Bashaga**

Title: **Head of the department of Computer Science,  
Al-Nahrain University.**

Date: / / **2008**



## *Examining Committee Certification*

---

---

We certify that we have read this thesis and as an examining committee, examined the student in its content and what is related to it, and that in our opinion it meets the standard of a thesis for the degree of Master of Science in Computer Science.

### *Examining Committee Certification*

Signature:

Name: **Dr. Loay E. George**

Title: **Assist. Prof. (Chairman)**

Date:    /    / **2008**

Signature:

Name: **Dr. Bara'a A. Attea**

Title: **Assist. Prof. (Member)**

Date:    /    / **2008**

Signature:

Name: **Dr. Jamal M. Kadhum**

Title: **Lecturer (Member)**

Date:    /    / **2008**

### *Supervisors Certification*

Signature:

Name: **Abeer M. Yousif**

Title: **Lecturer**

Date:    /    / **2008**

### *The Dean of the College Certification*

Approved by the Council of the College of Science

Signature:

Name: **Dr. LAITH ABDUL AZIZ AL - ANI**

Title: **The Dean of College of Science, Al-Nahrain University.**

Date:    /    / **2008**

Dedication . . .

To

My Dear Parents

Suleiman



## **List of Tables**

- Table(2.1) Data Portion fields of SMB\_COM\_DEL\_DIR*
- Table(2.2) Data Portion fields of SMB\_COM\_Rename*
- Table(2.3) Data Portion fields of SMB\_COM\_NT\_CreateAndX*
- Table(2.4) SMB\_COM\_NT\_CreateAndX Flags field values*
- Table(2.5) SMB\_COM\_NT\_CreateAndX ShareAccess field values*
- Table(2.6) SMB\_COM\_NT\_CreateAndX CreateDisposition field values*
- Table(2.7) SMB\_COM\_NT\_CreateAndX ImpersonationLevel field values*
- Table(2.8) SMB\_COM\_NT\_CreateAndX SecurityFlags field values*
- Table(2.9) Data Portion fields of SMB\_COM\_TreeConnectAndX*
- Table(2.10) SMB\_COM\_TreeConnectAndX Service[] field values*
- Table(2.11) Data Portion fields of SMB\_COM\_SessionSetupAndX*
- Table(3.1) Aggregated information by Aggregation unit*
- Table(3.2) Alarm level values*
- Table(3.3) Types of primitive Alarm Messages*
- Table(3.4) Permissions types*
- Table(4.1) False Positive values of Test Case One*
- Table(4.2) False Negative values of Test Case One*
- Table(4.3) Alarm level values for Student user class*
- Table(4.4) Alarm level values for Employee user class*
- Table(4.5) Alarm level values for Visitor user class*
- Table(4.6) Time and Memory consumption*

## **List of Abbreviations**

<b>ACK</b>	<i>Acknowledgment</i>
<b>ACL</b>	<i>Access Control List</i>
<b>AFS</b>	<i>Andrew File System</i>
<b>API</b>	<i>Application Programming Interface</i>
<b>CARDS</b>	<i>A Distributed System for Detecting Coordinated Attacks</i>
<b>CRC</b>	<i>Cyclical Redundancy Check</i>
<b>DFS</b>	<i>Distributed File System</i>
<b>DIDS</b>	<i>Distributed-based Intrusion Detection System</i>
<b>EMERALD</b>	<i>Event Monitoring Enabling Responses to Anomalous Live Disturbances</i>
<b>FMS</b>	<i>File Monitoring System</i>
<b>FTP</b>	<i>File Transfer Protocol</i>
<b>GA</b>	<i>Genetic Algorithm</i>
<b>GrIDS</b>	<i>Graph based Intrusion Detection System</i>
<b>HIDS</b>	<i>Host-based Intrusion Detection System</i>
<b>HTTP</b>	<i>Hypertext Transfer Protocol</i>
<b>ICMP</b>	<i>Internet Control Messaging Protocol</i>
<b>ID</b>	<i>Intrusion Detection</i>
<b>IDES</b>	<i>Intrusion Detection Expert System</i>
<b>IDIOT</b>	<i>Intrusion Detection In Our Time</i>
<b>IDS</b>	<i>Intrusion Detection System</i>
<b>IFS</b>	<i>Internet File System</i>
<b>IGMP</b>	<i>Internet Group Management Protocol</i>
<b>IHL</b>	<i>Internet Header Length</i>
<b>IP</b>	<i>Internet Protocol</i>
<b>ISO</b>	<i>International Standardization Organization</i>
<b>LAN</b>	<i>Local Area Network</i>
<b>MCCD</b>	<i>Microsoft Common Console Document</i>

<b>MMC</b>	<i>Microsoft Management Console</i>
<b>MID</b>	<i>Multiplex ID</i>
<b>MIDAS</b>	<i>Multics Intrusion Detection and Alerting System</i>
<b>MTU</b>	<i>Maximum Transmission Unit</i>
<b>NADIR</b>	<i>Network Anomaly Detection and Intrusion Reporter</i>
<b>NetBIOS</b>	<i>Network Basic Input/Output System</i>
<b>NFS</b>	<i>Network File System</i>
<b>NIDS</b>	<i>Network-based Intrusion Detection System</i>
<b>NNIDS</b>	<i>Network Node Intrusion Detection System</i>
<b>NSM</b>	<i>Network Security Monitor</i>
<b>OSI</b>	<i>Open Systems Interconnect</i>
<b>PID</b>	<i>Process ID</i>
<b>RegDB</b>	<i>Registration Database</i>
<b>SMB</b>	<i>Server Message Block</i>
<b>SRI</b>	<i>Stanford Research Institute</i>
<b>STAT</b>	<i>State Transition Analysis Technique</i>
<b>TCP</b>	<i>Transmission Control Protocol</i>
<b>TCP/IP</b>	<i>Transmission Control Protocol / Internet Protocol</i>
<b>THL</b>	<i>Total Header Length</i>
<b>TID</b>	<i>Tree ID</i>
<b>TOS</b>	<i>Type Of Service</i>
<b>TTL</b>	<i>Time To Live</i>
<b>UDP</b>	<i>User Datagram Protocol</i>
<b>UID</b>	<i>User ID</i>
<b>URG</b>	<i>Urgent Pointer</i>
<b>VB</b>	<i>Visual Basic</i>
<b>VPN</b>	<i>Virtual Private Network</i>
<b>WinSockMsg</b>	<i>Window Socket Message</i>

## List of Algorithms

- Algorithm (3.1) Add Shared Resources Class*
- Algorithm (3.2) Update Shared Resources Class*
- Algorithm (3.3) Assemble the Shared Resources into List*
- Algorithm (3.4) Construct ACL*
- Algorithm (3.5) Register Process*
- Algorithm (3.6) Login Process*
- Algorithm (3.7) Make Share Invisible*
- Algorithm (3.8) Make Share Visible*
- Algorithm (3.9) Initialize Socket*
- Algorithm (3.10) Subclassing Window*
- Algorithm (3.11) Filtering Network Messages*
- Algorithm (3.12) Generate Events*
- Algorithm (3.13) Get Name SMB\_COM\_162*
- Algorithm (3.14) Get Path SMB\_COM\_117*
- Algorithm (3.15) Add Event to the List of Events*
- Algorithm (3.16) Check the List of Events*
- Algorithm (3.17) Get Permission*
- Algorithm (3.18) Decision Making*
- Algorithm (3.19) Update Alarm level value*
- Algorithm (3.20) Check Administrator Authenticity*
- Algorithm (3.21) Check Registration*
- Algorithm (3.22) Check Login Process*
- Algorithm (3.23) Response*

## List of Figures

- Figure(2.1) TCP/IP Model*
- Figure(2.2) IP header format*
- Figure(2.3) TCP header format*
- Figure(2.4) SMB Message format*
- Figure(2.5) SMB Protocol ID field*
- Figure(2.6) DOS & OS/2 error code format*
- Figure(2.7) NT\_STATUS error code format*
- Figure(2.8) NetBIOS over TCP/IP*
- Figure(2.9) Session Service header format*
- Figure(2.10) Sniffed segment of TCP/IP*
- Figure(3.1) FMS Architecture*
- Figure(3.2) Structure of shared resources in Shared Folder Console*
- Figure(3.3) Tuple's format for ACL*
- Figure(3.4) Buffer content*
- Figure(3.5) Event format*
- Figure(3.6) NoOfTry List format*
- Figure(3.7) Alarm rule format*
- Figure(3.8) ACL format*
- Figure(4.1) FMS Main frame*
- Figure(4.2) FMS Setting frame*
- Figure(4.3) User Shared Resources Configuration frame*
- Figure(4.4) Configuration ACL frame*
- Figure(4.5) FMS Register frame*
- Figure(4.6) Main Control frame*
- Figure(4.7) Tools menu of Main Control frame*
- Figure(4.8) Main Control frame with disabled Monitoring menu*
- Figure(4.9) Tools menu after add FMS to start up menu*
- Figure(4.10) FMS Client subsystem icon*
- Figure(4.11) Monitoring menu of Main Control frame*
- Figure(4.12) How to stop monitoring*
- Figure(4.13) Help menu on Main Control frame*

- Figure(4.14)** *About message*
- Figure(4.15)** *Administrator Access frame*
- Figure(4.16)** *Administrator Main Control frame*
- Figure(4.17)** *Administrator Configuration frame*
- Figure(4.18)** *Clients' shared resources lists frame*
- Figure(4.19)** *Administrator Set Permission frame*
- Figure(4.20)** *Send File to Client subsystem frame*
- Figure(4.21)** *Configuration RegDB frame*
- Figure(4.22)** *Employees Configuration frame*
- Figure(4.23)** *Students Configuration frame with B.Sc. student*
- Figure(4.24)** *Students Configuration frame with M.Sc. student*
- Figure(4.25)** *Administrator Monitoring frame*
- Figure(4.26)** *Offline Messages frame*
- Figure(4.27)** *Users Profiles frame*
- Figure(4.28)** *Users Registration Profiles frame*
- Figure(4.29)** *User Activities frame*
- Figure(4.30)** *Administrator Response frame*



# بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

بِـرَفْعِ اللّٰهِ الرَّحْمٰنِ الرَّحِیْمِ  
اٰمَنُوْا مِنْكُمْ وَالْحٰمِیْنَ اٰمَنُوْا

اَلْعِلْمِ كَارِجَاتٍ وَاللّٰهُ بِمَا تَعْمَلُوْنَ خَبِیْرٌ

# صَلَّىٰ اللَّهُ عَلَيْهِ وَسَلَّمَ

(المجاهدنة ( ١١ )

- [Abr01] Wanderley J. Abren, "*NIDS on Mass Parallel Processing Architecture*", Jr, Phrack 57, 2001.
- [And80] James. P. Anderson, "*Computer Security Threat Monitoring and Surveillance*", Technical Report Contract 79F26400, Anderson Co., Box 42, Fort Washington, PA, 19034, USA, April 1980.
- [And98] R. Anderson and A. Khattak, "*The Use of Information Retrieval Techniques for Intrusion Detection*", In Proc. of the 1<sup>st</sup> International Workshop on the Recent Advances in Intrusion Detection (RAID), Louvain-la-Neuve, Belgium, September 1998, pp. 441-448.
- [Axe98] S. Axelsson, "*Research in Intrusion Detection Systems: A Survey*", TR: 98-17, Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden, 1998.
- [Bac00] Rebecca Bace and Peter Mell, "*Intrusion Detection Systems*", National Institute of Standards and Technology, NIST Special Publication on Intrusion Detection System, 2000.  
<http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf>
- [Bar98] Joseph Barrus and Neil C. Rowe, "*A Distributed Autonomous-Agent Network-Intrusion Detection and Response System*", In Proc. of the 1998 Command and Control Research and Technology Symposium, Monterey CA, June-July 1998, pp. 1-12.
- [BEn02] James Fell BEng, "*Combination of Misuse and Anomaly Network Intrusion Detection Systems*", Kaleton Internet, Unit 205, 57 Great George Street, Leeds, LS1 3AJ, United Kingdom, March 2002.  
<http://www.kaleton.com/infosec/idspaper.html>

- [Bur03] Daniel J. Burroughs, Linda F. Wilson, and George V. Cybenko, "*Analysis of Distributed Intrusion Detection Systems Using Bayesian Methods*", Thayer School of Engineering, Dartmouth College, Hanover, NH 03755, 2003.
- [Che96] S. Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "*GrIDS - A Graph Based Intrusion Detection System for Large Networks*", In Proc. of the 19th National Information Systems Security Conference, October 1996, pp. 361-370.
- [Cro94] M. Crosbie and E. Spafford, "*Defending a Computer System using - Autonomous Agents*", Technical Report No. 95-022, COAST Laboratory, Department of Computer Sciences, Purdue University, 1994.
- [Das99] Luiz A. DaSilva, "*Network Management paper*", Center for Wireless Telecommunication, Virginia Polytechnic Institute and State University, 1999.  
[http://www.ee.vt.edu/~prangapr/documants/rangaprabhu-Qos\\_paper.doc](http://www.ee.vt.edu/~prangapr/documants/rangaprabhu-Qos_paper.doc)
- [Dav04] Joseph Davies, "*TCP/IP Fundamentals for Microsoft Windows*", Online Book, Technical Writing for Microsoft Corporation, Published in November 2004 and Updated in January 2007.  
<http://technet.microsoft.com/en-us/library/bb727013.aspx>
- [Den87] D. Denning, "*An Intrusion-Detection Model*", IEEE Transactions on Software Engineering, 13(2):222-232, 1987.
- [Dul98] Emmett Dulaney, "*MCSE FAST TRACK – TCP/IP*", Book, New Riders Publishing, First Edition, September 1998.
- [Edm00] Edmund W., "*Network Monitoring fundamentals and standards report*", College of Engineering and Technology of Ohio University, 2000.  
<http://www.cis.ohio-state.edu/~jain/cis788-97/net-monitoring/index.htm>

- [End04] Carl Endorf, Eugene Schultz, and Jim Mellander, "*Intrusion Detection & Prevention*", Book, published by McGraw-Hill, March 2004.
- [Ham06] Lawrence R. Halme and R. Kenneth Bauer, "*[Intrusion Detection FAQ] AINT Misbehaving: A Taxonomy of Anti-Intrusion Techniques*", Arca Systems, Inc., 2540 North First St., Suite 301, San Jose, CA 95131-1016, May 2006. <http://www.sans.org>
- [Heb90] T. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "*A Network Security Monitor*", In Proc. of the IEEE Symposium on Research in Security and Privacy, 1990, pp. 296-304.
- [Her04] Christopher R. Hertel, "*Implementing CIFS*", Online Book, 2004.  
<http://ubiqx.org/cifs>
- [Hoc93] J. Hochberg, K. Jackson, C. Stallings, J. McClary, D. Dubois, and J. Ford, "*NADIR: An Automated System for Detecting Network Intrusion and Misuse*", Computers & Security, Elsevier Science Publishers, Volume 12, 235-248, May 1993.
- [Isa04] Richard C. Isaacson, "*Small Environment Network Intrusion Detection*", Paper, 2004. <http://www.beldurnik.com>
- [Jac91] K. Jackson, D. Dubois, and C. Stallings, "*An Expert System Application for Network Intrusion Detection*", In Proc. of the 14th National Computer Security Conference, Washington, D.C., October 1-4, 1991, pp. 215-225.
- [Jai02] Manish Jain, "*Networking Complete: Second Edition*", Book was Published for BPB Publications, B-14, Connaught Place New Delhi-110001, 2002.
- [Jou97] F. Jou, F. Gong, C. Sargor, S. Wu, and C. Rance, "*Architecture Design of a Scalable Intrusion Detection System for the Emerging Network*

*Infrastructure*", Technical Report CDRL A005, Department of Computer Science, North Carolina State University, 1997.

- [Jou00] Jou, Y., Gong, F., Sargor, C., Wu, X., Wu, S., Chang, H., and Wang, F., "*Design and Implementation of a Scalable Intrusion Detection System for the Protection of Network Infrastructure*", In DARPA Information Survivability Conference and Exposition, January 2000.
- [Kum94] S. Kumar and E. Spafford, "*An Application of Pattern Matching in Intrusion Detection*", Technical Report CSD- TR-94-013, The COAST Project, Department of Computer Sciences, Purdue University, 1994.  
<http://citeseer.ist.psu.edu/kumar94application.html>
- [Kum95] Sandeep Kumar, "*Classification and Detection of Computer Intrusions*", Department of Computer Sciences, Purdue University, Ph.D. Dissertation, 1995.
- [Lun88] T. Lunt, R. Jagannathan, R. Lee, S. Listgarten, D. Edwards, P. Neumann, H. Javitz, and A. Valdes, "*IDES: The Enhanced Prototype. A Real-Time Intrusion Detection System*", Technical Report SRI Project 4185-010, SRI-CSL-88-12, Computer Science Laboratory, SRI International, 1988.
- [Lyd04] Andrew Lydon, "*Compilation for Intrusion Detection Systems* ", Master Thesis of Science, College of Engineering and Technology of Ohio University, March 2004.
- [Mar02] Kelly Martin, "*Security Focus, <http://www.securityfocus.com>*", 2002.  
<http://www.securityfocus.com/infocus/1220>
- [Mas03] Gaia Maselli, Luca Deri, and Stefano Suin, "*Design and Implementation of an Anomaly Detection System: an Empirical Approach*", Centro Serra, Pisa University, Pisa, Italy, 2003.

- [Mic03] Microsoft, "*How TCP/IP Works*", March 2003.  
<http://technet2.microsoft.com/WindowsServer/en/library/3a9b874b-188a-4352-b542-27f433db07b01033.mspx?mfr=true>
- [Mic04] Microsoft, "*CIFS Packet Formats*", 2004.  
<http://msdn2.microsoft.com/en-us/library/aa302213.aspx>
- [Moo02] Andrew Moore, James Hall, Christian Kreibich, Euan Harris, and Ian Pratt, "*Architecture of a Network Monitor*", University of Cambridge Computer Laboratory, JJ Thomson Avenue, Cambridge CB3 0FD, United Kingdom, 2002.
- [Muk94] Biswanath Mukherjee, Todd L. Heberlein, and Karl N. Levitt, "*Network Intrusion Detection*", IEEE Network, 8(3):26-41, May-June 1994.
- [Ory06] Oryspayev D. Oryspayuli, "*What intrusion detection approaches work well if only TCP/IP packet header information is available?*", Master Thesis, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Netherlands, August 2006.
- [Pri97] Katherine E. Price, "*Host-Based Misuse Detection and Conventional Operating Systems' Audit Data Collection*", Master Thesis of Science, Purdue University, December 1997.
- [Por97] P. Porras and P. Nenmann, "*EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances*", In Proc. of the Nineteenth National Computer Security Conference, Baltimore, Maryland, October 22-25, 1997, pp. 353-365.
- [Ran01] Marcus J. Ranum. "*Experiences Benchmarking Intrusion Detection System*", NFR Security Technical Publication, December 2001. <http://www.nfr.com>
- [RFC87] NetBIOS Working Group RFC 1002 (Request For Comment), "*Protocol*

*Standard for a NetBIOS Service on a TCP/IP Transport Detailed Specification*", March 1987. <ftp://ftp.rfc-editor.org/in-notes/rfc1002.txt>

- [Roe02] Martin Roesch, "*Snort Users Manual: Snort Release: 1.9.x*", 26th, depends on, *Snort Documentations*, Online, Internet", April 2002. <http://www.snort.org>
- [Seb88] M. Sebring, E. Shellhouse, M. Hanna, and A. Whitehurst, "*Expert Systems in Intrusion Detection: A Case Study*", In Proc. of the 11th National Computer Security Conference, Baltimore, Maryland, October 17-20, 1988, pp. 74-81.
- [Sha02] Richard Shape, "*Just what is SMB?: V1.2*", 8 October 2002.  
<http://samba.anu.edu.au/cifs/docs/what-is-smb.html>
- [She01] Tom Sheldon, "*Encyclopedia of networking and Telecommunications*", Pan American and International Copyright Conventions, 2001.  
[http://www.linktionary.com/f/file\\_systems.html](http://www.linktionary.com/f/file_systems.html)
- [Sma88] S. Smaha, "*Haystack: An Intrusion Detection System*" , In Fourth Aerospace Computer Security Applications Conference, Tracor Applied Science Inc., Austin, Texas, December 1988, pp. 37-44.
- [Smi02] Rasheda Smith, "*Network-Based Intrusion Detection Using Neural Networks*", Department of Computer Science, Rensselaer Polytechnic Institute, Troy, New York 12180-3590, 2002.
- [Sna91] Steven R. Snapp, James Brentano, Gihan V. Dias, Terrance L. Goan, Todd L. Heberlein, Che-Lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, and Doug Mansur, "*DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype*", Computer Security Laboratory, Division of Computer Science, University of California Davis, Davis, California 95616, 1991.

- [Sna92] S. Snapp, S. Smaha, D. Teal, and T. Grance, "*The DIDS (Distributed Intrusion Detection System) prototype*". In Proc. of the Summer USENEX Conference, San Antonio, Texas, June 8-12, 1992, pp. 227-233.
- [Sol04] Rene Soltwisch and Dieter Hogrefe, "*A Survey on Network Security - 2004*", Institute for Informatics, George-August-University Contingent, Germany, ISSN 1611-1044, Number IFI-TB-2004-04, October 2004.  
<http://www.ifi.informatik.uni-goettingen.de>
- [Spa00] E. Spafford and D. Zamboni, "*Data collection mechanisms for intrusion detection systems*", CERIAS Technical Report, Center for Education and Research in Information Assurance and Security, 1315 Recitation Building, Purdue University, West Lafayette, In 47907-1315, June 2000.
- [Vac89] H. Vaccaro and G. Liepins, "*Detection of Anomalous Computer Session Activity*", In Proc. of the 1989 IEEE Symposium on Security and Privacy, Oakland, California, May 1-3, 1989, pp. 280-289.
- [Vig97] Giovanni Vigna and Richard A. Kemmerer, "*NetSTAT: A Network-based Intrusion Detection System*", Reliable Software Group, Department of Computer Science, University of California Santa Barbara, 1997.
- [Wei04] Wei Li, "*Using Genetic Algorithm for Network Intrusion Detection*", Department of Computer Science and Engineering, Mississippi State, University, Mississippi State, MS 39762, 2004.
- [Yan00] Jiahai Yang, Peng Ning, X. Sean Wang, and Sushil Jajodia, "*CARDS: A Distributed System for Detecting Coordinated Attacks*", In Proc. of Sixteenth Annual Working Conference on Information Security (SEC 2000), S. Qing and J. H. P. Eloff, Eds., Kluwer Academic, August 2000, pp. 171-180.
- [You01] Eric Young, "*CIFS Explained*", Code FX, Software solutions for applications and appliances, 4545 Campus Ave., California, San Diego, CA 92116, 2001.  
<http://www.codefx.com>



- [Zac01] Craig Zacker, "*Network+ Certification Training Kit: Second Edition*", published by Microsoft Press, A Division of Microsoft Corporation, One Microsoft way, 2001.
- [Zie04] Marek P. Zielinski, "*Applying Mobile Agents in an Immune-System-Based Intrusion Detection System*", Master Thesis, Computer Science, University of South Africa, November 2004.

# List of Contents

## **Chapter One: Intrusion Detection**

<i>1.1 Introduction</i>	(1)
<i>1.2 Taxonomy of Intrusion Detection Systems</i>	(2)
<i>1.2.1 Data Collection Mechanisms</i>	(2)
<i>A. Host-Based Intrusion Detection Systems</i>	(2)
<i>B. Network-Based Intrusion Detection Systems</i>	(4)
<i>C. Hybrid Intrusion Detection System</i>	(6)
<i>D. Network-Node Intrusion Detection Systems</i>	(6)
<i>1.2.2 Data Processing</i>	(7)
<i>A. Distributed-Based Intrusion Detection System</i>	(7)
<i>B. Centralized-Based Intrusion Detection System</i>	(8)
<i>1.2.3 Analyze Strategy</i>	(8)
<i>A. Misuse-Intrusion Detection System</i>	(8)
<i>B. Anomaly-Intrusion Detection System</i>	(9)
<i>1.3 Literature Survey</i>	(10)
<i>1.4 Aim of Thesis</i>	(15)
<i>1.5 Thesis Layout</i>	(15)

## **Chapter Two: Network Monitoring**

<i>2.1 Introduction</i>	(17)
<i>2.2 Network Categories</i>	(17)
<i>2.2.1 Server-Based Networks</i>	(17)
<i>2.2.2 Peer Networks</i>	(18)
<i>2.2.3 Hybrid Networks</i>	(18)
<i>2.3 Network Protocol Model</i>	(18)
<i>2.4 Server Message Block Protocol (SMB)</i>	(25)
<i>2.4.1 SMB Header</i>	(27)
<i>A. SMB Header Fields</i>	(27)
<i>B. SMB Command Field</i>	(29)
<i>2.4.2 Parameters Block</i>	(35)
<i>2.4.3 Data Block</i>	(36)
<i>2.5 NetBIOS Over TCP/IP</i>	(36)
<i>2.6 Network Monitoring for Intrusion Detection</i>	(39)
<i>2.6.1 Network Monitoring</i>	(39)
<i>2.6.2 Traffic Analysis</i>	(40)
<i>2.7 Measuring the Efficiency of an Intrusion Detection System</i>	(41)

## Chapter Three: System Design and Architecture

3.1	Introduction	(42)
3.2	Design Considerations and Requirements	(42)
3.3	FMS Architecture	(43)
3.3.1	Client Subsystem	(47)
A.	Initialization Module	(47)
B.	User Authentication Module	(53)
1.	Registration Unit	(53)
2.	Login Unit	(54)
C.	Sniffing Module	(57)
1.	Initialize Socket Unit	(58)
2.	Hook the window – Subclass Unit	(59)
3.	Filter Unit	(61)
D.	Detection Module	(70)
1.	Aggregation Unit	(71)
2.	Decision Making Unit	(74)
E.	Alarm Module	(79)
3.3.2	Administrator subsystem	(81)
A.	Check Administrator Authority Module	(81)
B.	Initialization Module	(82)
1.	Permission Unit	(82)
2.	Users Information Unit	(84)
3.	Registration Database Unit	(84)
C.	Monitoring Module	(86)
1.	Alarm Messages Unit	(86)
2.	Users Profiles Unit	(87)
D.	Response Module	(87)

## Chapter Four: FMS Interfaces and Evaluation

4.1	FMS Interfaces	(89)
4.1.1	FMS Client Subsystem Interfaces	(89)
A.	Initialization of FMS Client subsystem	(90)
B.	Registering into FMS	(92)
C.	Login into FMS	(94)
D.	Controlling FMS	(94)
4.1.2	FMS Administrator Subsystem Interfaces	(98)
A.	Configuration FMS Administrator subsystem	(99)
B.	Monitoring the Events	(104)
C.	Response to Monitoring	(108)
4.2	FMS Evaluation	(110)
4.2.1	Accuracy	(110)
4.2.2	Time and Memory Consumption	(116)

## **Chapter Five: Conclusions and Suggestions for Future Work**

*5.1 Conclusion* -----(119)

*5.2 Future Work*-----(121)

**References**-----(122)

**Appendix (A): Most Common SMB Commands**----- (1-A)



جمهورية العراق  
وزارة التعليم العالي و البحث العلمي  
جامعة النهرين  
كلية العلوم

# نظام تعقب المتطولين نوع محقة شبكة

رسالة مقدمة الى كلية العلوم، جامعة النهرين كجزء من متطلبات نيل شهادة  
الماجستير في علوم الحاسوب

من قبل

**سليمان سعدون فوزي**

(بكالوريوس ٢٠٠٤)

المشرفة

**د. عبير متي يوسف**

ربيع الأول ١٤٢٩

نيسان ٢٠٠٨

*Republic of Iraq  
Ministry of Higher Education and scientific research  
Al-Nahrain University  
College of Science*



# *Network Node Intrusion Detection System*

*A Thesis  
Submitted to the College of Science, Al-Nahrain University  
In Partial Fulfillment of the Requirements for  
The Degree of Master of Science in Computer Science*

*By  
Suleiman Sa'adon Fawzy  
(B.Sc. 2004)*

*Supervisor  
Dr. Abeer M. Yousif*

*April 2008*

*Rabee' Al-Awwal 1429*

# الخلاصة

نمت تقنيات شبكة الحاسوب بسرعة في العقود القليلة الماضية. وبالإستعمال المتزايد للحاسبات المُشبَّكة (شبكة حاسبات – Network Computers) للتطبيقات الحرجة أو الخاصة، تدخلات الحاسوب ازدادت وأصبحت تهديد خطر إلى هذه الأنواع من الأنظمة، وبهذا ، أنظمة تعقب المتطفلين أصبحت إضافة ضرورية إلى البنية الأمنية التحتية للكثير من المنظمات.

تُقدّم هذه الأطروحة تصميم وتطبيق نظام تعقب المتطفلين نوع عقدة شبكة (NNIDS) الذي يدعم بروتوكول IPv4. أن اسم النظام المُقترح مُختارٌ لكي يكون *FMS* (المختصر لنظام مراقبة الملف). هذا النظام يكتشف مجموعة من الهجمات المسلطة على المصادر المشتركة من نوع نظام الملفات (Filing System). يحوي النظام المقترح على قواعد تستند على مطابقة السلوك الطبيعي المُعرّف للنظام مع خصائص أحداث المستخدمين المُكتشفة.

هناك وحدات أساسية يتكون منها النظام المقترح: وحدة تسجيل الدخول التي تُميزُ المستخدمين الذين سَمَحَ لهم بالوصول الى المصادر المشتركة، ووحدة الشّم التي تلتقط وتترجم رزم الشبكة المنقولة وتولّد قائمة بالأحداث الخاصة بها، ووحدة الكشف التي تُحلّل قائمة الأحداث وتقرر أي منها يمثل نشاط مريباً ومشكوكٌ فيه، ووحدة الأذار التي تُولّد رسائل إنذار إلى المدير في حالات تحديد الهجمات.

تم تقييم النظام وفقاً لثلاثة عوامل: الدقة، الوقت، واستهلاك الذاكرة. عدّة هجمات مُصطنعة أرسلت إلى النظام المُقترح لإختباره. نتائج الإختبار بيّنت بأن أغلب هجمات نظام الملفات يُمكن أن يُكتشفَ بالنظام المُقترح بنسب مقبولة من قيم الأخطاء الإيجابية والسلبية.