

1 Introduction

In the first and second chapters we dealt with Mobile agents and in this chapter, we will examine intrusion detection. Our contribution will consist in using different tools NetBeans, jpcap, winpcap, Mysql and Aglets; we will create Mobile agents in the environment Aglets and start capturing packets. After a while, a decision will be made, the database will be updated and the attacker PC will be shut dawn.

2 Description of our work

Our System is composed of mobile agents. Each one migrates to each PC to capture packets. If the agent detects an intrusion, it updates the database and the other agent, “Shut dawn Agent” will shut dawn the attacker PC.

3 Used tools

To realize our intrusion detection system, we have used some tools; Netbeans to create Java Classes, Aglets for the environment of mobile agent, Winpcap and Jpcap two cooperative tools allowing IDE Netbeans to capture Packets from the network and MySQL to create and update a database of intruder IPs.

3.1 NetBeans

The NetBeans Platform is a broad Java framework on which you can base large desktop applications. NetBeans IDE itself is one of the hundreds applications based on the NetBeans Platform. The NetBeans Platform contains APIs that simplify the handling of windows, actions, files, and many other things typical in applications.

Each distinct feature in a NetBeans Platform application can be provided by a distinct NetBeans module, which is comparable to a plugin. A NetBeans module is a group of Java classes that provides an application with a specific feature.[33]

3.1.1 Java

Java is a computer programming language. It enables programmers to write computer instructions using English based commands, instead of having to write in numeric codes. It is known as a “high-level” language because it can be read and written easily by humans. Like English, Java has a set of rules that determine how the instructions are written. These rules are known as its “syntax.” Once a program has been written, the high-level instructions are translated into numeric codes that computers can understand and execute. [31]

3.2 Aglets

An aglet is a mobile Java object that visits aglet-enabled hosts in a computer network. It is autonomous, since it runs in its own thread of execution after arriving at a host, and reactive, because of its ability to respond to incoming messages. [4]

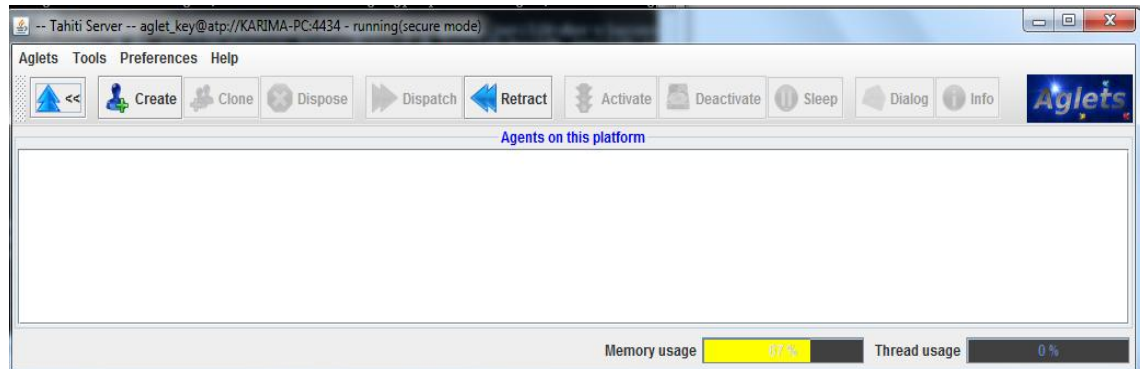


Figure3.1 Tahiti Server

Below we summarize the aglets' fundamental operations, namely, creation, cloning, dispatching, retraction, deactivation, activation, disposal, and messaging:

3.2.1 Creation

The creation of an aglet takes place in a context. The new aglet is assigned to an identifier, inserted (Layout) into the context, and initialized. The aglet starts executing as soon as it has been successfully initialized. [4]

3.2.2 Cloning

The cloning of an aglet produces an almost identical copy of the original aglet in the same context. The only differences are the assigned identifier and the fact that the execution restarts in the new aglet. Note that execution threads are not cloned. [4]

3.2.3 Dispatching

Dispatching an aglet from one context to another will remove it from its current context and insert it into the destination context, where it will restart execution (execution threads do not migrate). We say that the aglet has been "pushed" in its new context. [4]

3.2.4 Retraction

The retraction of an aglet will pull (remove) it from its current context and insert it into the context from which the retraction was requested. [4]

3.2.5 Activation and deactivation

The deactivation of an aglet is the ability to temporarily halt its execution and store its state in secondary storage. Activation of an aglet will restore it in a context. [4]

3.2.6 Disposal

The disposal of an aglet will halt its current execution and remove it from its current context. [4]

3.2.7 Messaging

Messaging between aglets involves sending, receiving, and handling messages synchronously as well as asynchronously. [4]

3.3 WinPcap

WinPcap is an open source library for packet capture and network analysis for the Win32 platforms. Most networking applications access the network through widely used operating system primitives such as sockets. It is easy to access data on the network with this approach since the operating system copes with the low-level details (protocol handling, packet reassembly, etc.) and provides a familiar interface that is similar to the one used to read and write files. Sometimes, however, the 'easy way' is not up to the task, since some applications require direct access to packets on the network. That is, they need access to the "raw" data on the network without the interposition of protocol processing by the operating system. [10]

3.4 Jpcap

Jpcap is an open source library for capturing and sending network packets from Java applications. It provides facilities to:

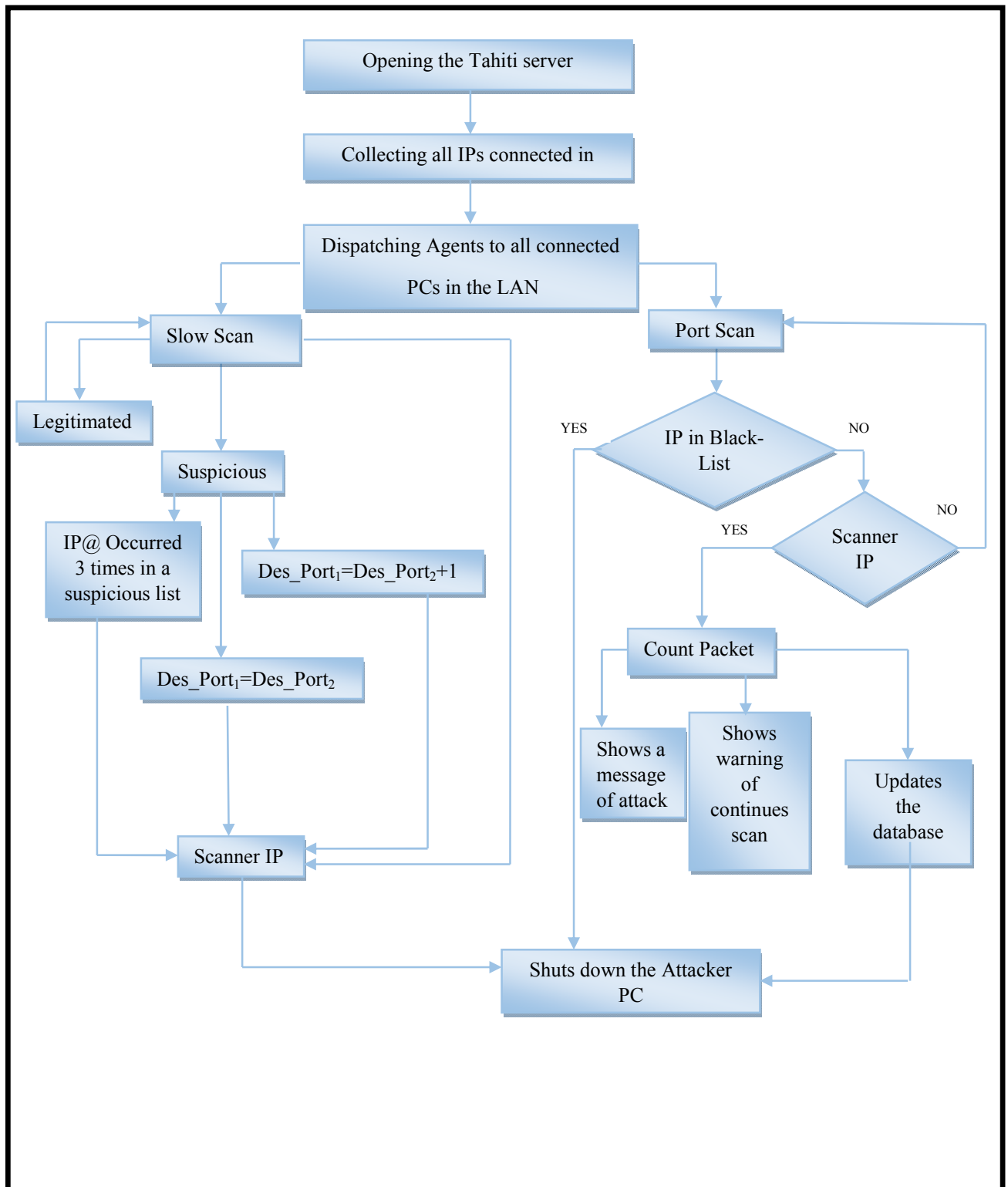
-  Capture raw packets live from the wire.
-  Save captured packets to an offline file, and read captured packets from an offline file

- ✚ Filter the packets according to user-specified rules before dispatching them to the application.
- ✚ send raw packets to the Jpcap network based on libpcap/winpcap, and implemented in C and Java.[10]

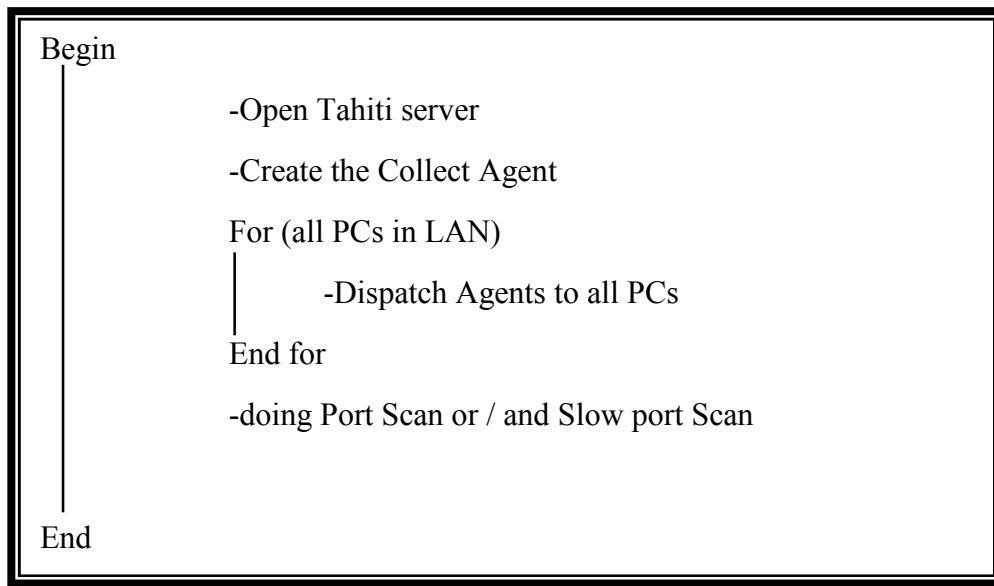
3.5 MySQL

MySQL is a full-featured relational database management system (RDBMS) that competes with the likes of Oracle DB and Microsoft's SQL Server. MySQL is sponsored by the Swedish company MySQL AB, which is owned by Oracle Corp. However, the MySQL source code is freely available because it was originally developed as freeware. MySQL is written in C and C++ and is compatible with all major operating systems. [35]

4 Organigramme System



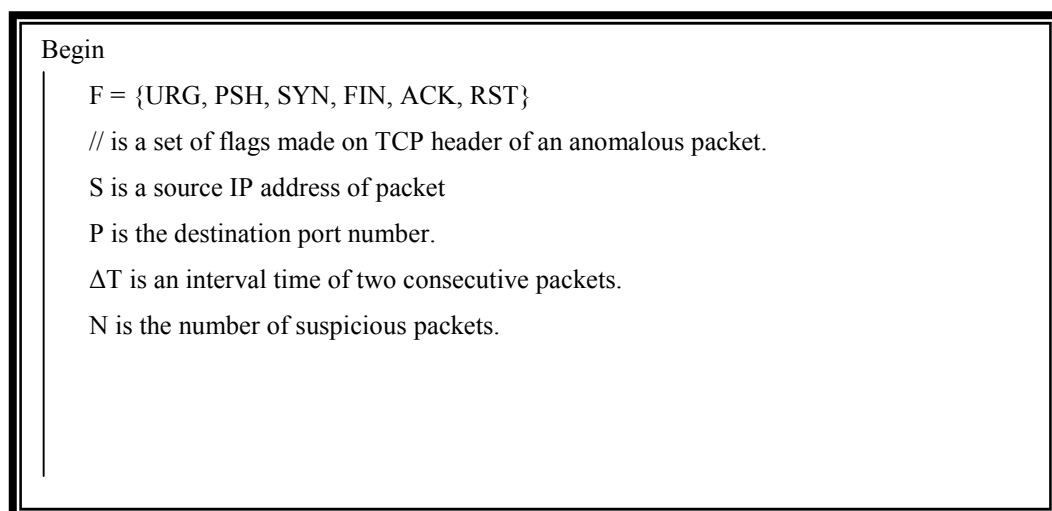
5 Principal algorithms



The capturing agent follows two different algorithms that will be discussed next:

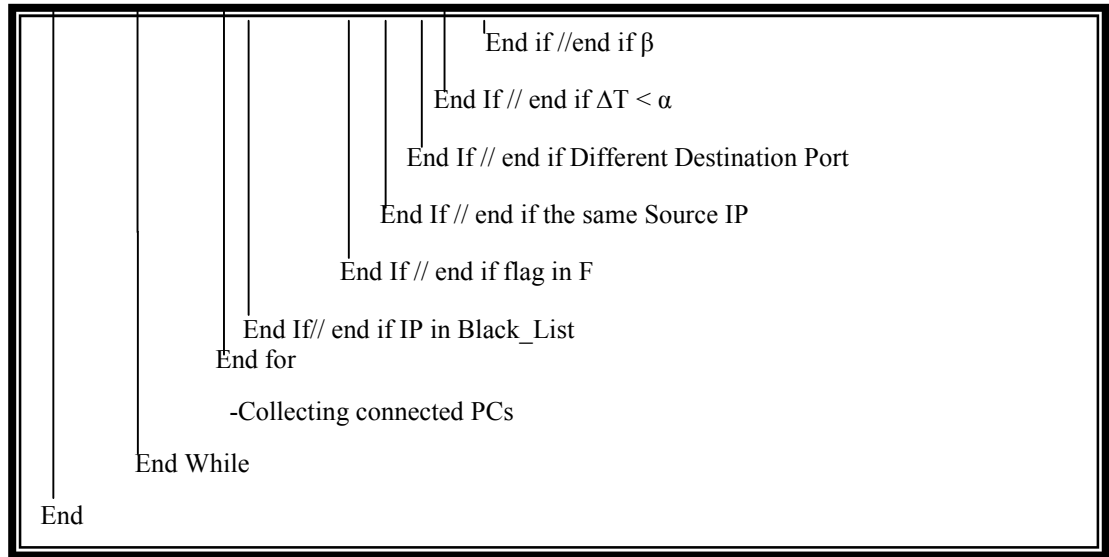
5.1 Port Scan

This algorithm is based on U. Kanlayasiri, S.Sanguanpong and W.Jaratmanachot [25], but its limitation forced us to change it, based on their perspectives which are changing the refinement of decision techniques and the host's profile history [25]. To realize that, we chose to create a database to store all the scanner IPs and give them a dangerous rate beginning from one star to 4 stars; So, if the IP has four stars, it will be added to Blacklist-IP.



```

While (true)
    -Start Capturing
    While (time of capturing<Giving Time)
        -Keep Capturing Packets
    End while
    For (All IPs in LAN)
        If (IP is in Blacklist) Then
            -Show message to continue receiving Packets
            If (decision==no) Then
                -Shut down the attacker PC
            Else
                IF flag is in F Then
                    IF S of each Packet is the same Then
                        IF P each Packet is different Then
                            IF  $\Delta T < \alpha$  Then
                                IF  $N = \beta$  Then
                                    -Showing Intrusion alerts
                                    IF  $N=2\beta$  Then
                                        -Showing a 2nd Intrusion Alert
                                    End if
                                    IF  $N=3\beta$  Then
                                        IF (IP  $\exists$  in Data Base) Then
                                            -Update-Database (IP, Rate)
                                        Else -Create new IP in Database
                                        End If
                                        -Shut dawn the Attacker PC
                                    End if //end if 3 $\beta$ 
                                End if
                            End if
                        End if
                    End if
                End if
            End if
        End if
    End For
End While
    
```



Following the perceptive of the algorithms of U.Kanlayasiri, S.Sanguanpong and W.Jaratmanachot [25], we create A database to store Scanner IPs and rate them according to the number of scans. If the IPs have four in their rating, they will be in the Black-List. After 3 Scannings of the victim PC with the same IP, the database will be updated. If the IP already exists, we will create a new element. This is considered the History of host. When we talk about the second perspective, which is changing the refinement of a decision respecting the algorithm, the alarm will started after 20 Packets. We add another refinement which alerts the victim PC after 20 Packets and waits for other 20 packets. If the packets equal 2β , warning message will show. In addition, if the number of packets equals 3β , the database will create a new element or will update if the IP already exists.

5.2 Slow Port Scan

The three most common TCP scanning techniques that an attacker may follow in order to know which ports are open and which ports are closed are: connect scan, half-connect scan and FIN scan. Next, we explain how these scans are performed and for each scan, we present the features that will be collected in order to distinguish legitimate users from scanners. [16]

5.2.1 Connect Scan

The scanner sends a SYN, and determines whether the port is open or closed based on the server's response. If the server replies by a SYN-ACK, then, the port is open and the scanner continues the three-way handshake and establishes a connection by sending an ACK to the server. Otherwise, the server replies by a RST to declare a closed port. [16]

The first feature that distinguishes legitimate users from scanners is the number of

connections made to closed ports (N_{closed}). Legitimate users are aware of the offered services and are not expected to try to establish connections to closed ports. On the contrary, there is a great chance that an attacker while scanning the ports, will try, unknowingly, to connect to closed ports. For each IP, the number of connections attempted to closed ports (N_{closed}) is determined by calculating the difference between the incoming SYN from each IP and the outgoing SYN ACK from the server of each IP. The number of connections to closed ports that are made by an IP (IP) can be determined as shown in equations (1): [16]

$$SYN_{Closed}^{IP} == SYN_{in}^{IP} - SYN_{Out}^{IP} \quad (1)$$

Where $SYN_{in,ip}$ is the number of incoming SYN segments from (IP) to the server, and $SYN_{out,ip}$ is the number of outgoing SYN-ACKS that are sent from the server to (IP).

If no connections are made to closed ports, then, the difference equals zero. This is based on the fact that when the server receives a connection establishment to a closed port, it responds by an RST indicating that the destination port is closed. Whereas, the server responds by a SYN ACK segment if the port is open. [16]

5.2.2 Half-Connect Scan

This technique is also known by the SYN scan. It is similar to the connect scan except that after receiving a reply from the server that the port is open, the scanner does not continue the three-way handshake and does not establish a full connection. This type of scan is usually more difficult to detect, since establishing the connection is not completed and is not logged. [16]

The number of half connections (N_{hc}^{ip}) that are made by an IP (IP) can be directly determined by calculating the number of SYN-ACKs that were sent from the server to (IP) and that were not followed by an ACK.[16]

5.2.3 FIN Scan

As discussed before, the FIN bit is set when the two parties want to terminate a pre-established connection. A scanner determines whether a port is open or closed by sending a TCP segment with the FIN bit set to a port. Since no connection to that port was established in the first place, the server responds by RST if the port is closed or the server does not respond at all if the port is open. Based on whether the scanner receives an RST or not, it can determine whether the port is closed or open. [16]

The number of FIN probes that were not preceded by establishing a connection (N_{Fin}) can be calculated for each IP by calculating the difference between the incoming FINs from the IP to the server and the outgoing FINs from the server to the IP. The number of FIN probes that

is made by an IP (IP) and that was not preceded by establishing a connection can be determined as shown in equation (2):

$$N_{Fin}^{IP} == FIN_{in}^{IP} - FIN_{Out}^{IP} \quad (2)$$

We calculate these features ($N_{Closed}, N_{hc}, N_{Fin}$) for each IP at the end of each window time. Then, we decide whether each IP represents a scanner or a legitimate user based on these features. The rules for classifying an IP (IP) based on the values of its features are shown as follows: [16]

$$\left\{ \begin{array}{l} \textit{Legitimate} \quad N_{Closed}^{ip} = 0 \text{ and } N_{hc}^{ip} = 0 \text{ and } N_{Fin}^{ip} = 0 \\ \textit{Suspicioius} \quad N_{Closed}^{ip} = 1 \text{ or } N_{hc}^{ip} = 1 \text{ or } N_{Fin}^{ip} = 1 \\ \textit{Scanner} \quad N_{Closed}^{ip} > 1 \text{ or } N_{hc}^{ip} > 1 \text{ or } N_{Fin}^{ip} > 1 \end{array} \right.$$

```

Begin
  While (true)
    For (Scan-Time = 0; Scan-Time < 3; Scan-Time++)
      While (time of capturing < Giving Time)
        For (all IPs in LAN)
          -Calculate NIpClosed
          -Calculate Nhc
          -Calculate NFin
          If (NIPClosed > 1 OR Nhc > 1 OR NFin > 1)
            -Shut down the Attacker PC
          End If
          If (NIPClosed == 1 OR Nhc == 1 OR NFin == 1)
            -Add IP to Suspicious list
            If ((Packet1.desPort == Packet2.desPort)
              || (Packet1.desPort == Packet2.desPort + 1)
              || (IP occurred 3 times in the Suspicious List))
              -Remove IP from Suspicious List
              -Add IP to Scanner list
            End If
          End If
        End for
      End while
    End for
    -Clear Suspicious-IP-List
    -Collecting connected PCs
  End while
END

```

6 The results

The opening of the Tahiti server to create the first mobile agent which is responsible for collecting all the IPs connected in LAN as shown in figure 3.1:

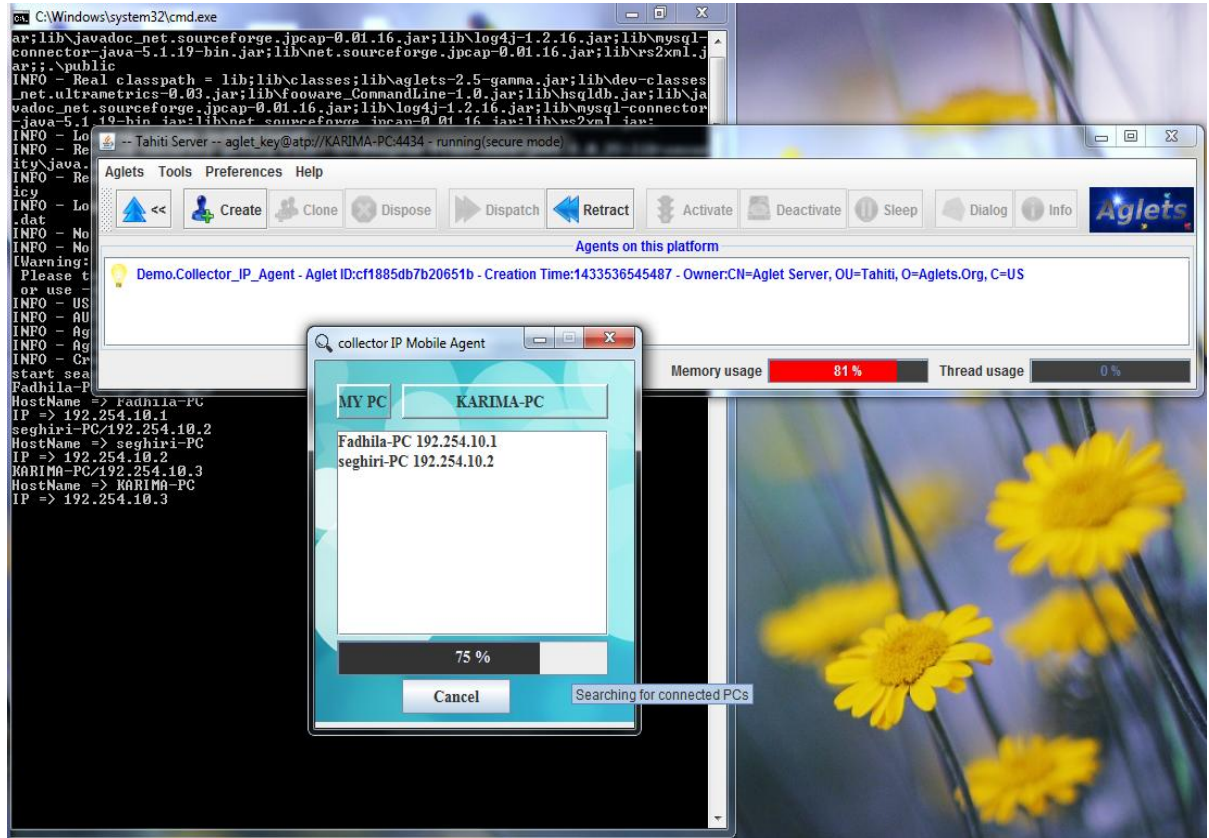


Figure 3.1 Mobile agent to collect all connected PCs in LAN

After the Collector agent finished searching for connected PCs, a window will show that it is finished searching and all mobile agents dispatched to PCs in LAN Figure3.2, Figure3.3

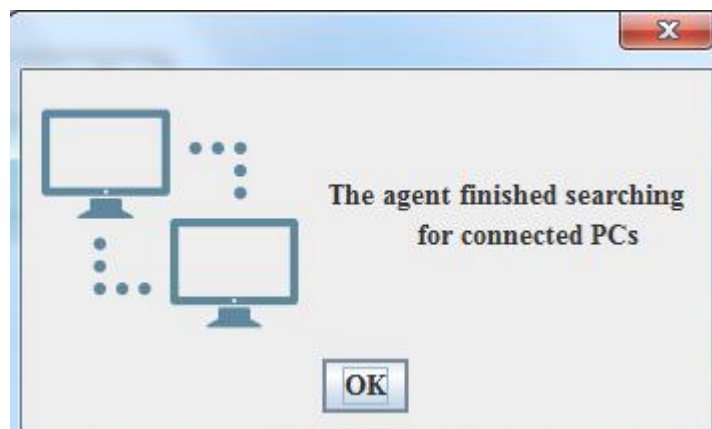


Figure3.2 The mobile agent finished searching for connected PCs



Figure3.3 The mobile agent finished dispatching agents to all PCs dispatching
After all mobile agents have arrival to remote hosts (Figure3.4), the user chooses one of the two algorithms or uses both of them.

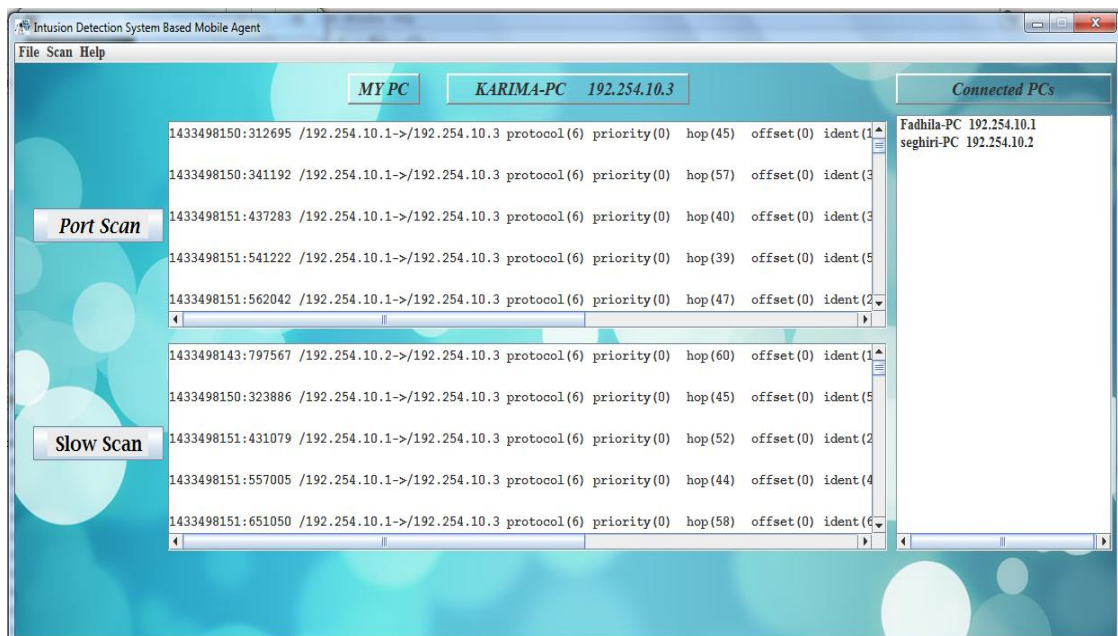


Figure3.4 The main interface for each mobile agent

6.1 Results of Port Scan algorithm

This algorithm will show three different windows after the 20 first packets, 40 and 60 packets. After the latter, the attacker PC will be shut down by the mobile agent.



Figure3.5 The window after 20 packets



Figure3.6 The window after 40 packets



Figure3.7 The window after 60 packets

If the IP has four in its dangerous rate in its database, that means it is in black list (Figure3.8). This window allows the user to choose whether to continue receiving packets or not. The last choice will make the Mobile agent shut down the attacker PC. If the user does not choose the mobile agent, it will automatically shut down the attacker PC without waiting for the user's decision.



Figure3.8 IP in the black list

6.2 Results of Slow Port Scan algorithm

This algorithm will show many different windows according to the type of scanned or suspicious IP; if it is a suspicious IP, we mentioned three ways to treat this IP according to [1] and if the destination port of two consecutive packets are equal, then, this IP is removed from the suspicious IP list to the scanner IP list. If the destination port of two consecutive packets are consecutive too, than this IP is removed from the suspicious IP list to the scanner IP list. Finally, if these two conditions are not verified, we will wait. If this IP is repeated three times in the suspicious IP list, then, we removed it from the suspicious IP list and add it to the scanner IP list.



Figure3.9 The window shows after the system detect a scanner IP



Figure3.10 The window shows after the system detects a suspicious IP

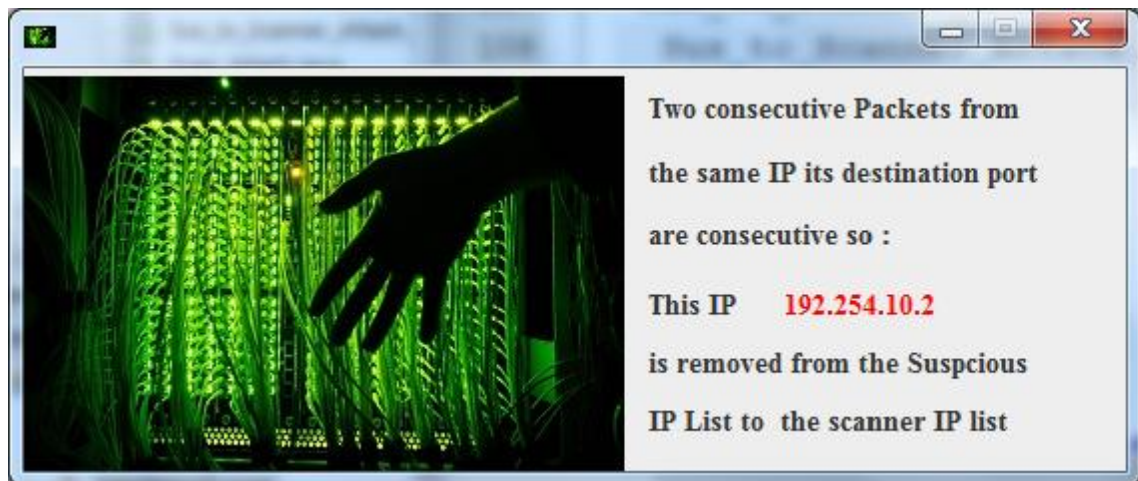


Figure3.11 Suspicious IP sends two packets with consecutive destination ports

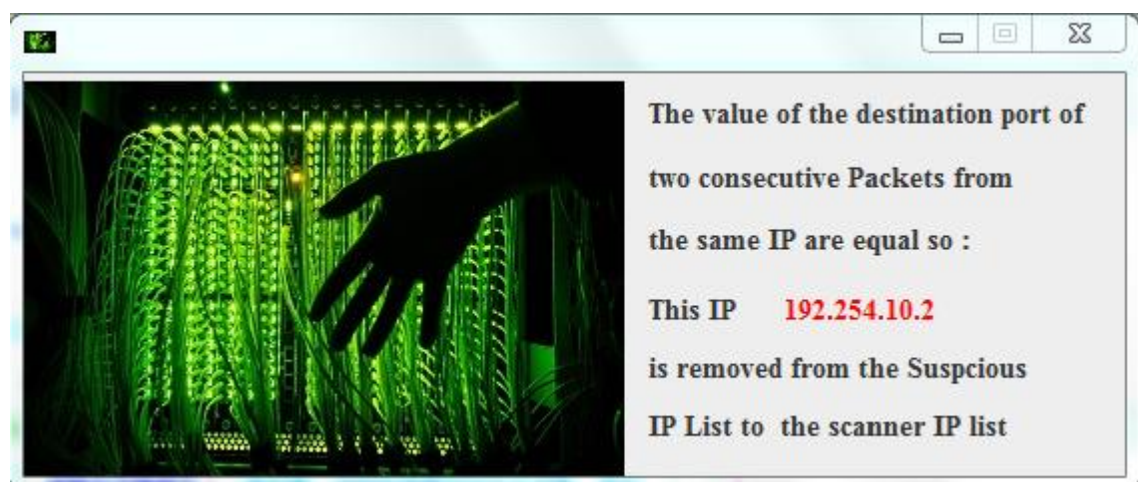
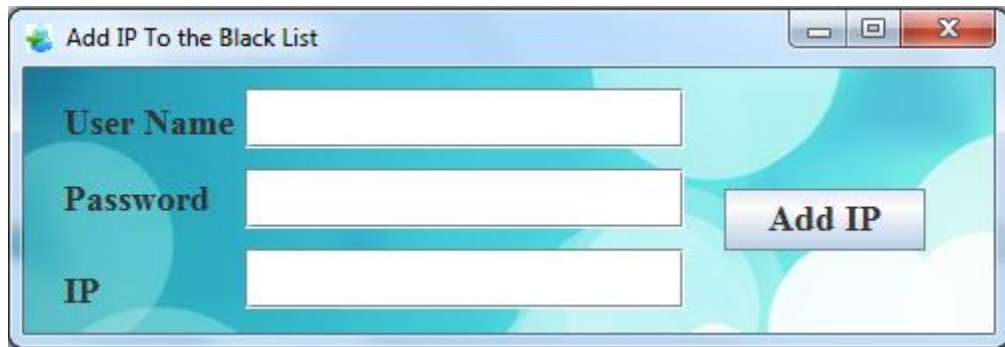


Figure3.12 Suspicious IP sends two packets with the same destination ports



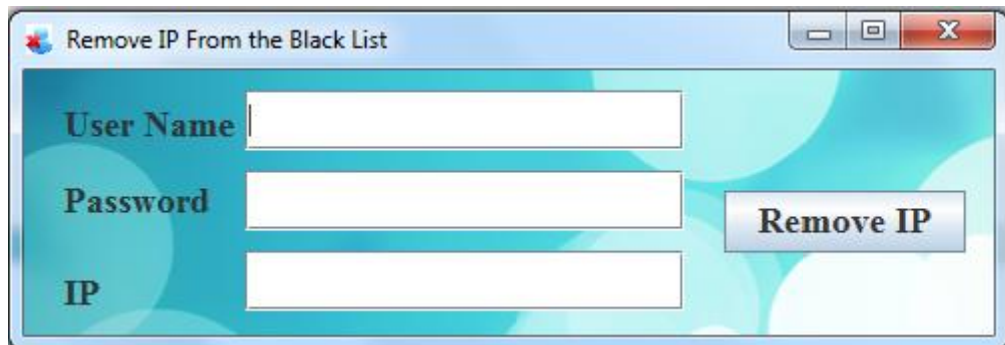
Figure3.13 Suspicious IP repeated 3 times in a suspicious list

The user can manage the blocked IP by adding new IP or remove it from the black-list. But, the system requires a user name and a password (Figure3.11) and (Figure3.12) in addition it can see the state of the database (Figure3.13).



A screenshot of a Java Swing window titled "Add IP To the Black List". The window has a light blue background with a subtle pattern. It contains three text input fields labeled "User Name", "Password", and "IP" stacked vertically on the left. To the right of these fields is a single button labeled "Add IP". The window has standard Windows-style title bar controls (minimize, maximize, close) in the top right corner.

Figure3.14 Add IP to the Black-List



A screenshot of a Java Swing window titled "Remove IP From the Black List". The window has a light blue background with a subtle pattern. It contains three text input fields labeled "User Name", "Password", and "IP" stacked vertically on the left. To the right of these fields is a single button labeled "Remove IP". The window has standard Windows-style title bar controls (minimize, maximize, close) in the top right corner.

Figure3.15 Remove IP from the black-List



A screenshot of a Java Swing window titled "IP DataBase". The window displays a table with four columns: "IP_adress", "MAC_adress", "HostName", and "DangerousRate". The table contains two rows of data. The window has standard Windows-style title bar controls (minimize, maximize, close) in the top right corner.

IP_adress	MAC_adress	HostName	DangerousRate
192.254.10.1	20-cf-30-44-c0-53	Fadhila-PC	4
192.254.10.2	70-5a-b6-e3-52-30	seghiri-PC	1

Figure3.16 IPs statistics

7 Conclusion

This chapter describes our implementation of the intrusion detection system based on mobile agents. We realized this system that used Aglets; environment of mobile agent NetBeans, Jpcap, Winpcap and MySQL to code our system and store data in the database.