CSN10105 - Advanced Security and Digital Forensics

Enhanced Agent-based Intrusion Detection System

Napier Edinburgh University

May 2009

Student mat.

07010875

Table of content

1 INTRODUCTION

Following an increasing demand in systems to be more pro-active in terms of security, it is required to create a system able to detect malicious activity within a network. The system is an agent-based intrusion detector, able to generate alerts when a policy violation occurs. The systems should be able to present the alerts clearly to the administrator, and store them securely.

This report will present the work undertaken in order to realise this intrusion detector.

The **first part** will make an analysis about the requirement, and outline the proposed system. The goals will be clearly presented, and the main specifications and features will be detailed.

The **second part** of this report will present the implementation of the system, how the system has been created, in what extend it will be useful and efficient regarding the policy presented in the first part. Some technical details will be presented, like the creation (pieces of programming) of the system.

The **third part** will present some tests realised, using the created system. This part will show that the system is fully functional, efficient and easy to use. This step will prove that the system meet the requirements.

Then the **conclusion** will present the global aspects of the project, the justification of the choices made for the system, in what extend it could have been different if the requirements were different, the weaknesses and goods points of the implementation.

2 REQUIREMENT ANALYSIS

The requirement analysis will present the main requirements of the system for it to meet the requirements. A study will present the requirements, how the system could provide the required service and how it could do so.

2.1 Topology layout

As the system will be a network analyser, the application should be able to retrieve all the packets between the internal network (private LAN) and the external network (Internet). The global network topology is presented in Illustration 1. The network analyser should be placed in a privileged place within the network, to allow to retrieve all packets routed by Router1.

The IDS should be connected to a port configured to all the traffic from all its ports. This can be configured onto CISCO devices using the SPAN configuration :

```
monitor session 1 source vlan 1 rx
monitor session 1 destination interface Fa0/24
```

This configuration allows the redirection of all traffic from VLAN 1 (default VLAN) to the interface FastEthernet 0/24, in reception only (using both emission and reception create duplicate packets within the interface).

2.2 Incoming connection on port 1234, and use encryption tunnels

The use of encrypted tunnels can be a threat in the extend that the traffic going through the tunnel cannot be checked by the firewalls: some viruses can use this to connect on an external server, and send / receive, for example, some sensitive informations about the company work. All the traffic (incoming as out coming) should be checked by the firewall.

Most of encryption tunnels have to use IKE (Internet Key Exchange) to exchange their keys, in order to set up the encrypted tunnel. This protocol uses port 500 with UDP. A common tunnelling encryption uses ESP (Encapsulating Security Payload), on port 50 or AH encryption (Authentication Header) on port 51. The use of SSL VPN are widely deployed thanks to the application OpenVPN (using port 1194). PPTP tunnels (Point-to-Point Tunnelling Protocol) uses port 1723 (TCP).

The definition to detect clients within the private network to connect through tunnels can be defined as:

```
Source: any
Destination: any
Protocol: TCP or UDP
Destination ports: 50, 51, 500, 1194
Source port: any
```

2.3 Incoming ping of a host and port scan

The use of port scan onto a network can be a threat: if some ports are open, this can lead to an attack attempt, if the application using this port is not properly secured. The port-scans are usually performed thanks to the “SYN scanning”, also known as “half-open scanning”.

To do so, an attacker sends a SYN packet to a computer, trying to open a TCP connection. If a service is running on the attacked computer, it will reply by a SYN-ACK. The attacker will be informed that a port is open on the machine.

A port-scan will try to send SYN packet to a range of port, on the same machines. This can be detected by a load of SYN packets, with same IP destination but similar ports.

2.4 Incoming MSN Messenger connection request

The use of MSN messenger service can be prohibited within the company: employees can spend more time chatting, playing or other activities than working. The use of MSN service, whatever the client is, needs a connection on the authentication server from MSN, service which uses port 1863, both TCP and UDP (http://support.microsoft.com/kb/927847).

A lookup on Internet allowed to be informed that none other application is using this port (http://portforward.com/cports.htm). The use of this port should only be used by MSN messenger. The attempt of connection onto a machine using the port 1863 should match to an attempt of connection to MSN services.

The detection of outgoing MSN Messenger connections can be defined by:

Source: Local Network Destination: any Protocol: TCP or UDP Destination port: 1863 Content: MSN

2.5 BBC content streaming

The BBC website (www.bbc.co.uk) allow some streaming content for free, by the use of a web browser. The two streaming content proposed by the website are a video player (iPlayer) and a web radio. The abusive use of this service can lead to excessive bandwidth usage, slow down the global network and so loose in profitability for the company. Both of the content should be detected by the system.

Video streaming content

As there is no documentation available concerning the player, some packet captures allowed to find out the specific requests, from the client, to launch the play of a video. After a long period of tests (using different videos, loading / refreshing the web page), the only common point which could identify the lecture of a video has been found.

When the user clicks on “play” within the player, the following request is sent to the server:

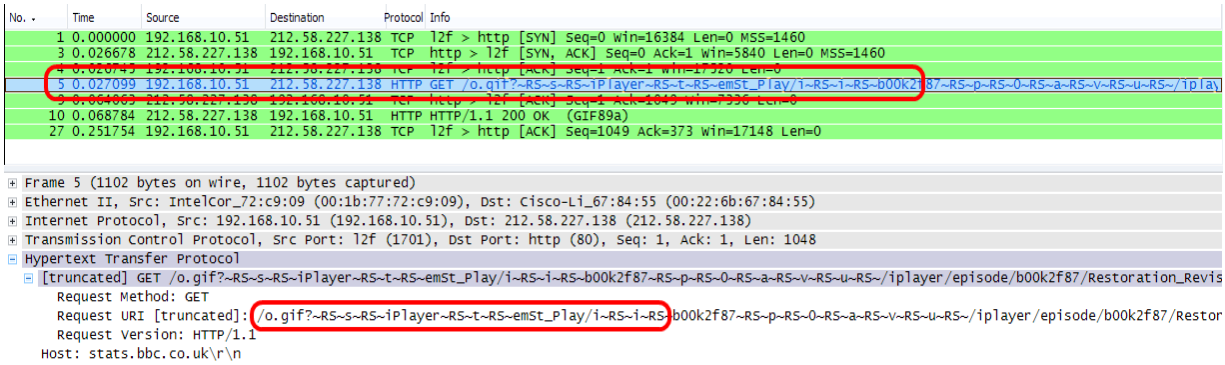

Illustration 1: BBC streaming capture

This capture has been made thanks to Wireshark, at the moment when the page was loaded but the video not started yet. The user clicked on “PLAY” button, and here is part of the request which is common, whatever the video is:

```
/o.gif?~RS~s~RS~iPlayer~RS~t~RS~emSt_Play/i~RS~i~RS[...]
```

This is an HHTP request (destination port 80) and retrieving the file /o.gif, with many arguments (following the ? within the request). Neither destination IP address or host can be trust, as there are probably many servers for one domain name (to balance the load).

The rule to detect the BBC video streaming can now be defined by:

Source: Local Network Destination: any Protocol: HTTP (destination port = 80) Content: /o.gif?~RS~s~RS~iPlayer~RS~t~RS~emSt_Play/i~RS~i~RS

Radio streaming content

The radio player on the BBC website can be retrieved directly from http://www.bbc.co.uk/radio1/. To be able to play the radio, a streaming-content player should be used such realPlayer. Once installed, some captures allowed to detect the establishment of the media flux:

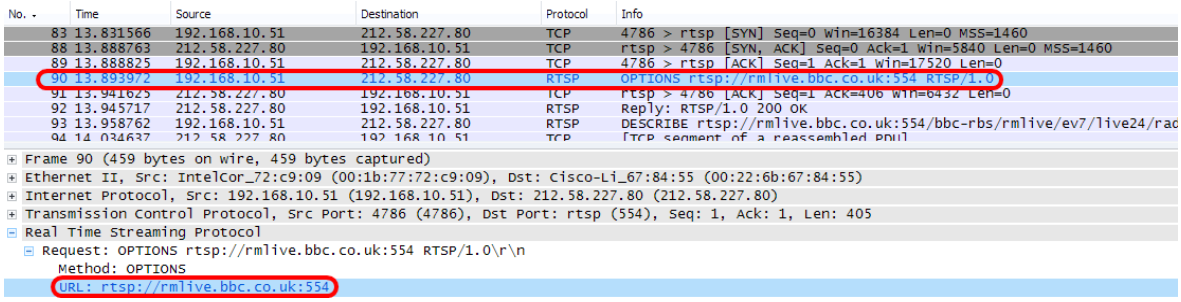

Illustration 2: BBC radio content capture

The flux use RTSP protocol, with the destination port 554. The establishment of the connection can be seen thanks to the three first packets (SYN handshake, SYN, SYN-ACK and ACK). However the streaming content is retrieved from the url rtsp://rmlive.bbc.co.uk:554. This is a reliable information to detect the streaming.

The definition of a connection to the radio streaming content is:

Source: Local Network Destination: any Protocol: RTSP (destination port = 554) Content: rtsp://rmlive.bbc.co.uk:554

2.6 Botnet activity

The Botnet can be retrieved from http://www.dcs.napier.ac.uk/botnet.zip. It is an example of an application that is prohibited within the company network.

The provided Botnet tries to connect to a distant machine on port 1001 (Illustration 2) The bot sends a SYN flag, in order to open a connection. The distant node should have a service running onto its 1001 port, to reply by an SYN-ACK and then establish a connection.

As the use of this bot is prohibited by the security policy, the system should be able to detect the fist attempt of connection, both from Internet to the local network than from Internet to private network.

The detection of Botnet activity can be defined by two ways: incoming and outgoing connection attempt. The outgoing connection can be defined by:

Source: Local Network Destination: any Protocol: TCP (destination port = 1001) Content: SYN flag

And the incoming by:

Source: any Destination: Local Network any Protocol: TCP (destination port = 1001) Content: SYN flag

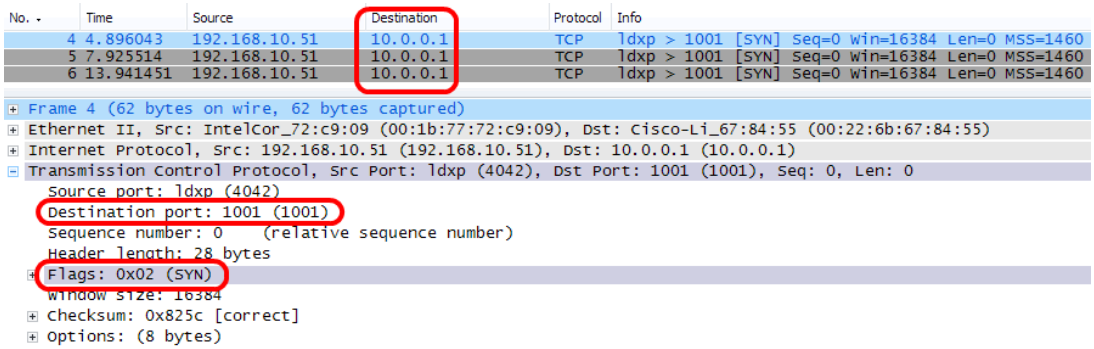

Illustration 3: Botnet activity onto the network

3 OUTLINE IMPLEMENTATION

Now that the system requirements have been defined and detailed, the outline implementation will present a description about the creation of the IDS. The main points about the system structure will be presented.

3.1 Snort

To meet the systems requirements, the IDS should be reliable and efficient. Snort is a leader in terms of intrusion detection systems, it is multi-platform, free of use, widely used and well documented. Snort uses WinPCAP and libPCAP to analyse packets.

Snort is an agent application, which analyses the packets and creates alerts when a sequence defined by rules is found. The rules are defined by the user as text files. Snort can be run using the following command:

```
C:\snort\bin\snort.exe -dev -i 2 -C -l C:\intrusions -K ascii -c C:\snort\rules.txt
```

Snort.exe can have many arguments, such *-dev –i <device-number>* which specify the network interface to use, *-l <path>* defines the log directory and *-c <path>* gives the path of the rule file to use.

The rules allow the definition of the packets / traffic that snort should intercept, log or simply alert the user. They define the rule which the analysed packet should match to create an alert.

The syntax for each rule is :

```
alert proto machine1 port direction machine2 port (msg:"Message here" ; options ; sid;)
```

The *proto* parameter defines the protocol, then *machine1* / *machine2* parameters define the IP addresses (or network) of the nodes, and their ports is specified (can be *any*). The direction is specified to control in which way the packet should go to match the rule (incoming or outgoing packet). Each rule uses a *sid* parameter, which is a unique identifier.

3.2 Snort rules definitions to meet the requirements

The traffic type which should be logged by the IDS has been defined in the requirement analysis part. Their conversion from a user-friendly text to Snort rules will be presented.

First the networks should be defined within the rule file. A network is defined by its IP address and subnet. Some variables are defined to facilitate reading and eventual changes (one line should be modified in the case of addresses modification within the LAN).

In the case of this study, only a local network (private LAN) and Internet have to be defined. In this case the HOME_NET defined as the 192.16.10.0/24 is the private network, and any address which is not part of HOME_NET is called EXTERNAL_NET (representing Internet).

```
var HOME_NET 192.168.10.0/24
var EXTERNAL_NET !$HOME_NET
```

Once the variables are defined, they can be used within the rest of the rule file as $HOME_NET and $EXTERNAL_NET. The rules can be now defined in accordance to the system requirement.

- Incoming connection on port 1234, and use encryption tunnels

```
alert tcp $EXTERNAL_NET any <> $HOME_NET 1234 (msg:"Incoming on 1234"; flags: S; sid:1001;)
alert tcp $HOME_NET any -> $EXTERNAL_NET 22 (msg:"SSH tunnel"; flags: S; sid:1002;)
alert tcp $HOME_NET any -> $EXTERNAL_NET 443 (msg:"HTTPS tunnel"; flags: S; sid:100;)
alert tcp $HOME_NET any -> $EXTERNAL_NET 465 (msg:"SMTP SSL tunnel"; flags: S; sid:1004;)
alert tcp $HOME_NET any -> $EXTERNAL_NET 563 (msg:"NNTPS tunnel"; flags: S; sid:1005;)
alert tcp $HOME_NET any -> $EXTERNAL_NET 993 (msg:"IMAP SSL tunnel"; flags: S; sid:1006;)
alert tcp $HOME_NET any -> $EXTERNAL_NET 995 (msg:"POP3 SSL tunnel"; flags: S; sid:1007;)
alert udp $HOME_NET any <> $EXTERNAL_NET 1194 (msg:"OpenVPN tunnel"; sid:1008;)
```

The detection of any traffic using SSL encryption should be detected. This is considered as a tunnel encryption as the traffic cannot be checked by the firewalls.

- Incoming ping of a host and port scan

Ping detection :

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any(msg:"Incoming PING"; itype: 8 ; sid:1010;)
```

The ping is using ICMP (Internet Control Message Protocol) protocol,

Port scan (these lines should be at the top of the file):

```
preprocessor flow: stats_interval 0 hash 2
preprocessor sfportscan: proto { all } scan_type { all } sense_level { low }
```

- Incoming MSN Messenger connection request

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 1863 (msg:"MSN chat connection"; flow: established;
content: "MSG"; sid:1003;)
```

- BBC video streaming

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 80 (msg:"BBC Iplayer content"; content: "/o.gif?
~RS~s~RS~iPlayer~RS~t~RS~emSt_Play/i~RS~i~RS"; depth: 112; sid:1020;)
```

- BBC radio streaming

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 554 (msg:"BBC radio content" ; sid:1021;)
```

- Botnet activity

```
alert tcp $HOME_NET any -> any 1001 (msg:"BotNet outgoing activity"; flags: S; sid:1030;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 1001 (msg:"BotNet incoming activity"; flags: S; sid:1031;)
```

3.3 Application

Snort itself can produce only text-base output: it is convenient when using command line, but not user-friendly. To meet the requirements, the system should be user friendly and provide a clear evidence of the alerts.

An application will be created, to manage Snort program. The system will be able to start Snort, retrieve the alerts and display them to the end-user. The application will be created using Visual Basic .NET programming language. As the system will run onto a machine running a Windows system, the .NET applications are a good solution, as they can be re-used by other applications using .NET technologies (such Office suite).

The application is presented as a single executable file and an associated library file, named dotnetWinpcap.dll, which allow the use of WinPcap resources for .NET programming(Victor Tan http://www.codeproject.com/KB/IP/dotnetwinpcap.aspx). The application uses the library to retrieve the network interfaces from the machine.

Once launched, the application will automatically check if Snort is installed, and then ask the user to chose a directory where the log files will be stored. The user will have the choice to create the rules (Illustration 3). To finish the configuration, the user will choose the interface to listen to, and validate by clicking on OK.

Then the main window appear, using a clear layout to show easily the alerts. The user can start / stop the captures, show the alerts in two modes:

- Show simplified alerts: the *alerts* window will present all the retrieved alerts, where one line present one alert. The presentation is simplified, to allow a global view of all the alerts.
- Show detailed alerts: all the alerts will be shown on the *alerts* window, with their details. The time, IP and MAC addresses, ports, protocol, TTL and other IP headers.

There is the possibility for the user to retrieve the full packets which occurred an alert. By using the detailed view of the alerts, the source IP address is displayed. Then, the mini browser allow to retrieve the associated file (each folder is associated to the source IP address), and then the file-names

Some captures of the program are available in appendices.

4 TESTS

After the presentation and definition of the requirements and the system itself, some tests are performed in order to verify the validity of the undertaken work. For each of the requirements, some packets-capture will be presented and the associated output of the system.

Note that for the tests the variable $HOME_NET (private network) is defined by the IP address of the test machine. Any other IP address is considered as external network. The HOME_NET is defined as the 192.16.10.51/32 network, which includes only one node (net mask is on 32 bytes).

```
var HOME_NET 192.168.10.51/32
```

The topology implemented consist in using two laptops (192.168.10.51 and 192.168.10.52) connected thanks to wifi to a Linksys router (192.168.10.1) running the DD-WRT firmware (to allow the remote access by SSH and so perform some of the test).

4.1 Incoming connection on port 1234, and use encryption tunnels

The tests performed to detect connection on the port 1234 has been made thanks nmap, which is a port-scanner, and will be presented more in details later. Nmap has been configured to send a SYN on the port 1234. This is a attempt to establish a connection. For the tests of tunnels encryption, only the tests about SSH and HTTPS have been performed. To do so, a remote connection has been established to the router (192.168.10.1), and the web-browser has retrieved a secure web page (https://pod51002.outlook.com/owa/, the email service of Napier Edinburgh University). Some alerts have been displayed by the system, in accordance to the configured rules.

```
---Displaying all alerts (advanced format)---
[**] [1:1002:0] SSH tunnel [**]
[Priority: 0]
05/11-21:10:53.692696 0:1B:77:72:C9:9 -> 0:22:6B:67:84:55 type:0x800 len:0x3E
192.168.10.51:2638 -> 82.41.25.143:22 TCP TTL:128 TOS:0x0 ID:62193 IpLen:20 DgmLen:48 DF
******S* Seq: 0x38F8494D  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

[**] [1:100:0] HTTPS tunnel [**]
[Priority: 0]
05/11-21:11:24.304231 0:1B:77:72:C9:9 -> 0:22:6B:67:84:55 type:0x800 len:0x3E
192.168.10.51:2639 -> 213.199.174.116:443 TCP TTL:128 TOS:0x0 ID:62249 IpLen:20 DgmLen:48 DF
******S* Seq: 0x7A49375A  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
```

Illustration 4: alerts displayed by use of tunnels

The detection of the tunnel allowed to log only the establishment of the connection, instead of logging all the encrypted packets (which would be useless because of their encryption, their content cannot be read).

Unfortunately non of the other rules have been testes by lack of equipments (routers dealing with VPN). We can consider that the other rules can detect the encrypted traffic as well, as the protocol used is TCP and the detection of SYN is based on the same principle.

4.2 Incoming ping of a host and port scan

A requirement for the system is to detect incoming ping. A ping has been realised from the node 192.168.10.1 to the client (192.16.10.51). The output of the system display an alert:

```
[**] [1:1010:0] Incoming PING [**]
```

If the user click on the "Show complete alert", then more details will be provided:

```
[**] [1:1010:0] Incoming PING [**]
[Priority: 0]
05/10-15:39:49.174558 0:22:6B:67:84:55 -> 0:1B:77:72:C9:9 type:0x800 len:0x62
192.168.10.1 -> 192.168.10.51 ICMP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:39253   Seq:0  ECHO
```

However, only the packet header are displayed. If the user need to display the whole packet content, this is possible by using the mini browser. Then all the packet headers and the payload is logged.

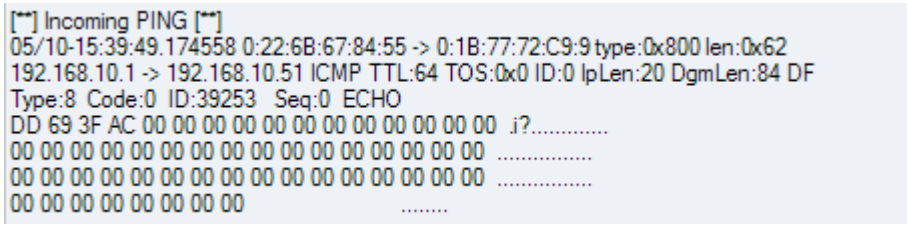

Illustration 5: incoming PING detected

The payload is empty (only 0 are displayed), but this can change depending of the systems used (the first letters of the alphabet are often used). In this case this ping is from a Unix embedded machine (BusyBox on a Linksys router).

The detection of port scan can be tested thanks to the software *nmap* (www.insecure.org). This allows to perform some post-scan on designed machines. For the tests the use of standard scan has been used: this send SYN flags to a range of ports, and wait for a SYN-ACK reply. If a reply is received, this mean that the port is open.

An port-scan has been performed between the machines: once the machine 192.168.10.51 has been scanned (by 192.168.10.52) and after the machine 192.168.10.51 performed the scan to the other one. Both of the scans have been intercepted by the system:

```
[**] [122:1:0] (portscan) TCP Portscan [**]
[Priority: 3]
05/11-12:28:35.732623 4D:41:43:44:41:44 -> 4D:41:43:44:41:44 type:0x800 len:0xAC
192.168.10.52 -> 192.168.10.51 PROTO:255 TTL:0 TOS:0x0 ID:1666 IpLen:20 DgmLen:158

[**] [122:1:0] (portscan) TCP Portscan [**]
[Priority: 3]
05/11-12:29:28.428914 4D:41:43:44:41:44 -> 4D:41:43:44:41:44 type:0x800 len:0xAD
192.168.10.51 -> 192.168.10.52 PROTO:255 TTL:0 TOS:0x0 ID:2023 IpLen:20 DgmLen:159
```

Illustration 6: port scan in both ways (incoming and outgoing)

The detection of both way port scan is a good policy, as it is not in a normal use that user scan each other's computers. This can be a sign of potential threat such worm or virus.

4.3 Incoming MSN Messenger connection request

The test about the incoming MSN messenger connection has been performed with the client Windows Live Messenger version 2008 (Messenger 8.5). However, the result should be the same with any MSN client (Pidgin, Trillian) as the alert is based on the MSN protocol, and any client needs to use it to connect to the service.

The IDS has been started, while the client was disconnecter. Then the user / password have been typed, to start the connection. After a successful authentication, the MSN client display the contacts: the connection has been established.

The system has detected the connection: an alert is logged:

```
[**] [1:1003:0] MSN chat connection [**]
```

When retrieved thanks to the mini-browser, the packet is displayed: it is possible to know the login of the user who is using the service. However, for legal reason, it may be against the law to store these data without the contentment of the users.

```
[**] MSN chat connection [**]
05/10-15:41:20.847762 0:1B:77:72:C9:9 -> 0:22:6B:67:84:55 type:0x800 len:0x98
192.168.10.51:3619 -> 207.46.96.153:1863 TCP TTL:128 TOS:0x0 ID:58137 IpLen:20 DgmLen:138 DF
***AP*** Seq: 0x1D366BE5  Ack: 0xC5767886  Win: 0x4462  TcpLen: 20
43 56 52 20 32 20 30 78 30 34 30 63 20 77 69 6E  CVR 2 0x040c win
6E 74 20 35 2E 31 20 69 33 38 36 20 4D 53 4E 4D  nt 5.1 i386 MSNM
53 47 52 20 38 2E 35 2E 31 33 30 32 20 6D 73 6D  SGR 8.5.1302 msm
73 67 73 20 7A 65 62 69 6B 65 72 40 66 72 65 65  sgs [illegible]@free
2E 66 72 0D 0A 55 53 52 20 33 20 53 53 4F 20 49  .fr..USR 3 SSO I
20 7A 65 62 69 6B 65 72 40 66 72 65 65 2E 66 72   [illegible]@free.fr
0D 0A                                            ..
```

Illustration 7: alert and packet from a MSN connection

4.4 BBC content streaming

The policy does not allow any streaming content from BBC website. Two rules has been established to detect both radio and video content. For the test, a web browser opened the BBC website and listened to the radio, and then watched an video. The system has been able to detect the play for each service:

```
[**] [1:1019:0] BBC radio content [**]
[Priority: 0]
05/11-20:32:13.644365 0:1B:77:72:C9:9 -> 0:22:6B:67:84:55 type:0x800 len:0x20B
192.168.10.51:2448 -> 212.58.227.96:554 TCP TTL:128 TOS:0x0 ID:5838 IpLen:20 DgmLen:509 DF
***AP*** Seq: 0x98186B72  Ack: 0x99D93C2C  Win: 0x4470  TcpLen: 20

[**] [1:1010:0] BBC Iplayer content [**]
[Priority: 0]
05/11-20:36:01.785388 0:1B:77:72:C9:9 -> 0:22:6B:67:84:55 type:0x800 len:0x439
192.168.10.51:2523 -> 212.58.227.137:80 TCP TTL:128 TOS:0x0 ID:8217 IpLen:20 DgmLen:1067 DF
***AP*** Seq: 0xAE5EF7DD  Ack: 0xA85D746D  Win: 0x4470  TcpLen: 20
```

Illustration 8: advanced display alert for radio and video content from BBC

The rule can be considered as efficient in the extend where only one alert is displayed, and not one for each packet. This result one alert per video / radio started.

4.5 Botnet activity

As presented in the *requirement analysis* part, the Botnet activity should be detected. The tests performed allowed to show that the rule created before is efficient. A Botnet has been started onto the node itself (considering that is it part of the private network), to test the outgoing Botnet activity. An attempt of connection from the external network to the internal network onto port 1001 has been performed, thanks to *nmap* (presented before). The output of the system has logged both activities :

```
---Displaying all alerts (advanced format)---
[**] [1:1031:0] BotNet incoming activity [**]
[Priority: 0]
05/11-12:17:02.670251 0:1B:38:38:DA:3E -> 0:1B:38:17:EF:3D type:0x800 len:0x3C
192.168.10.52:43499 -> 192.168.10.51:1001 TCP TTL:47 TOS:0x0 ID:44698 IpLen:20 DgmLen:44
******S* Seq: 0xB596CF62  Ack: 0x0  Win: 0x1000  TcpLen: 24
TCP Options (1) => MSS: 1460

[**] [1:1031:0] BotNet outgoing activity [**]
[Priority: 0]
05/11-12:17:02.670251 0:1B:38:38:DA:3E -> 0:1B:38:17:EF:3D type:0x800 len:0x3C
192.168.10.51:43499 -> 192.168.10.52:1001 TCP TTL:47 TOS:0x0 ID:44698 IpLen:20 DgmLen:44
******S* Seq: 0xB596CF62  Ack: 0x0  Win: 0x1000  TcpLen: 24
TCP Options (1) => MSS: 1460
```

Illustration 9: botnet activity alerts

Both incoming and outgoing activity is detected, however there is one alert for each attempt of connection, which can be an issue regarding the storage of logs.

5 CONCLUSION

The presentation of the requirements and the implementation part of the study allowed a precise definition of the created advanced IDS. The use of Snort gives a reliable base for the system, and the definition of the rules can be modified easily at any time.

The application has been developed to display alerts in a user-friendly way, instead of retrieving text files from direct output of snort. The user is able to display easily and quickly the alerts, and so be aware of any violation of access policy. The files are stored in the directory which can be chosen by the user, and can be retrieved at any time for a need of prove evidence.

However the application suffers of a major default: the user can see the global alerts only when the process is not running. But the alert files can be retrieved thanks to the mini-browser.

This system has another aspect which can present some issues: if the detected traffic generate an alert each time that a packet matching the rule is detected, the alert file-size will be consequent and can create some latency within the system. Also, the number of file alerts (content of the packets) will increase considerably and can lead to a system failure. This is why the rules have been tested on a long-term connections, but new ones should be well tested as well.

The performance of an IDS can be measured thanks to its results, retrieved after a period of tests. Some key metrics defines the success of the system, like False positives, the number of intrusions that the system failed to detect, False negatives, the alerts generated that were not intrusions. Thanks to the tests realised, there are some False negative (MSN connection generate sometimes some alerts during a conversation) but the false positives are quasi-non-existent, in the limit of the tests performed.

Limitations

The system has some limitations: at any time the BBC streaming protocol could change, and the rules will be obsoletes. The MSN Messenger protocol can change as well, but not on a short-term because of compatibility. Also, the same mechanism has been used since the first versions of the protocol (http://www.hypothetic.org/docs/msn/general/connections.php). There is also the possibility for users to use web-based messenger interfaces (such http://webmessenger.msn.com or www.meebo.com), which is more difficult to trace.

Another limitation has been detected during the tests, this is the use of content-streaming (in the case of this project, BBC streaming was concerned). There is a possibility for the players using RTSP to encapsulate the content within HTTP traffic, and so work trough proxies and firewall (QuickTime Streaming Server Modules Programming Guide, 2005).

Further work

An enhancement of the system could store the alerts using a database, and show some statistics detailed, depending of the level of the alert, the type of content and the time. All these informations are very useful for any network administrator. It helps to detect example suspicious activities within the network (e.g. MSN messenger alerts during the night is suspicious: no employees are supposed to be here, this could be a worm using MSN client).

The use of graphics diagrams and statistics helps to retrieve easily the suspicious activities. Also, coupled to a firewall, a system could adapt automatically the firewall configuration depending of the type of alerts from the IDS. For example the use of Real Time Protocol onto another port than the one blocked could result of a new rule for the firewall, and block this traffic.

6 REFERENCES

Buchanan, B. (2009). Security and Forensic Computing. Edinburgh

Mintz, M. (2004). *MSN Messenger Protocol*. Retrieved May 2009, from http://www.hypothetic.org/docs/msn/

Microsoft © (2009). *Network ports and URLs that are used by Windows Live Messenger*. Retrieved May 2009 from http://support.microsoft.com/default.aspx/kb/927847

Sturges, S. (2009). *Snort™ Users Manual.* Retrieved May 2009 from http://snort.org/docs/snort_htmanuals/htmanual_284/

April 2005, QuickTime Streaming Server Modules Programming Guide: Concepts, Tunnelling *RTSP and RTP Over HTTP*, http://developer.apple.com/documentation/QuickTime/QTSS/Concepts/QTSSConcepts.html#/apple_ref/doc/uid/TP30000245-TPXREF143. Retrieved May 2009

2009, Welcome to the Netgroup at Politecnico di Torino. http://winpcap.polito.it/. Retrieved April 2008

March 2009, V. Tan, Packet Sniffing with WinPCAP Functions Ported to a .NET Library, http://www.codeproject.com/KB/IP/dotnetwinpcap.aspx. Retrieved April 2009

7 APPENDICES

7.1 Screen shots of the system

Some screen shots are presented to allow the audience to have an idea of the system layout. It has been programmed using VB.NET and Framework 3.5. This program has been developed by Florian Berthelot, largely inspired from the version written in C# by B. Buchanan.

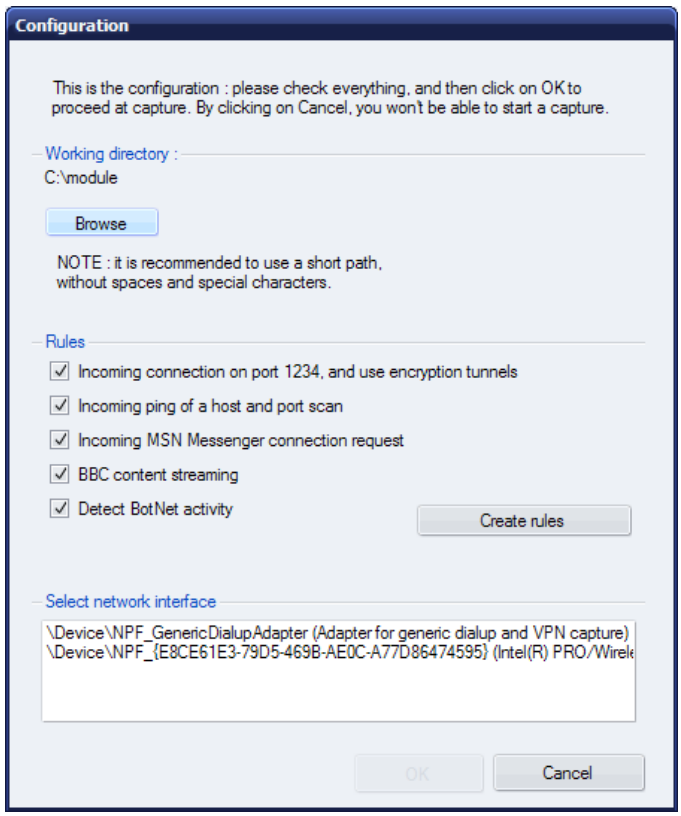

Illustration 10: configuration of the program

The user is able to chose in which directory the alerts will be stored. Then the creation of the rule file can be modified, in order to stick to the requirements of the systems. To finish the configuration, an API allows the retrieval of network interfaces, and the user can chose on which one the program should listen to.

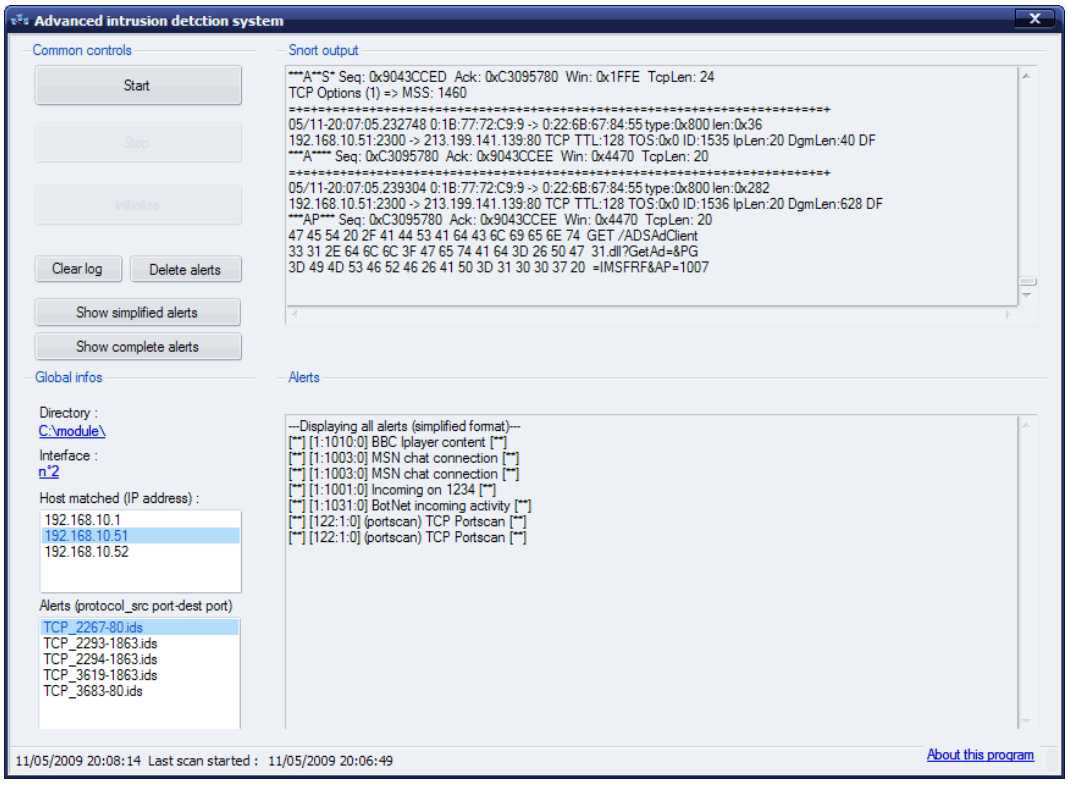

Illustration 11: main windows with simple display of alerts

The mini-browser displays the content of the *Directory*, defined by the user during the configuration. One click in the first field allows the display of all the alert files within the second field. A double click on the alert file allow the display of its content in the *Alerts* field.

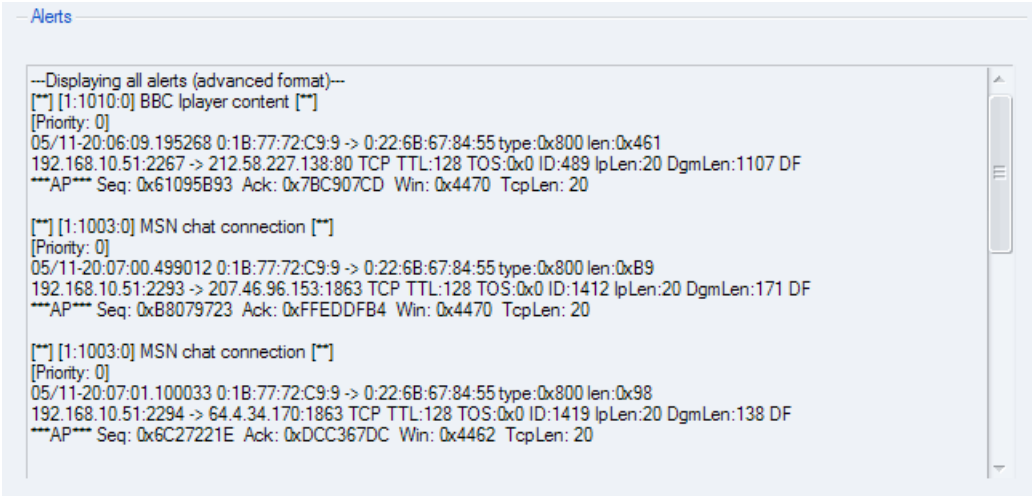

Illustration 12: advanced display of the alerts

7.2 Final rule

```
var HOME_NET 192.168.10.51/32
var EXTERNAL_NET !$HOME_NET

# Loading of the preprosessors

# Flow preprocessor allow to make stateful analysis
preprocessor flow: stats_interval 0 hash 2

# sfportscan allow the detection of port-scanning
preprocessor sfportscan: proto { all } scan_type { all } sense_level { low }

# Detection of incoming traffic on the port 1234
alert tcp $EXTERNAL_NET any <> $HOME_NET 1234 (msg:"Incoming on 1234"; flags: S; sid:1001;)

# detection of tunnels
alert tcp $HOME_NET any -> $EXTERNAL_NET 22 (msg:"SSH tunnel"; flags: S; sid:1002;)
alert tcp $HOME_NET any -> $EXTERNAL_NET 443 (msg:"HTTPS tunnel"; flags: S; sid:100;)
alert tcp $HOME_NET any -> $EXTERNAL_NET 465 (msg:"SMTP SSL tunnel"; flags: S; sid:1004;)
alert tcp $HOME_NET any -> $EXTERNAL_NET 563 (msg:"NNTPS tunnel"; flags: S; sid:1005;)
alert tcp $HOME_NET any -> $EXTERNAL_NET 993 (msg:"IMAP SSL tunnel"; flags: S; sid:1006;)
alert tcp $HOME_NET any -> $EXTERNAL_NET 995 (msg:"POP3 SSL tunnel"; flags: S; sid:1007;)
alert udp $HOME_NET any <> $EXTERNAL_NET 1194 (msg:"OpenVPN tunnel"; sid:1008;)

# Detection of radio content streaming (from BBC)
alert tcp $HOME_NET any -> $EXTERNAL_NET 554 (msg:"BBC radio content" ; content: "SETUP
rtsp://rmlive.bbc.co.uk:554" ; sid:1019;)


# Detection of incoming ping
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"Incoming PING"; itype: 8; sid:1010;)

# Detection of MSN connection
alert tcp $HOME_NET any -> $EXTERNAL_NET 1863 (msg:"MSN chat connection"; flags: S; sid:1020;)

# Detection of any botnet activity
alert tcp $HOME_NET any -> any 1001 (msg:"BotNet outgoing activity"; flags: S; sid:1030;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 1001 (msg:"BotNet incoming activity"; flags: S; sid:1031;)

# Detection of iPlayer content streaming (from BBC)
alert tcp $HOME_NET any -> $EXTERNAL_NET 80 (msg:"BBC Iplayer content"; content: "/o.gif?
~RS~s~RS~iPlayer~RS~t~RS~emSt_Play/i~RS~i~RS"; depth: 112; sid:1040;)
```