Global Information Assurance Certification Paper

How to fight P2P in a corporate environment

GSEC Practical V1.4b
Option 2
Case study in information security
25/07/2004

[Mar 8, 2004, Sep 17, 2004]

By Eric ANDRE

Table of contents

1. ABSTRACT

A large number of definitions can be found to describe exactly what peer to peer is. The following definition seem to be sufficient accurate to define this concept[1]: A distributed network architecture may be called “peer to peer” network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity …)

Theses networks have become increasingly popular over the past five years and the history of P2P cannot be recounted without a reference to Napster. Nevertheless, since Napster, some distributed network architectures have become very popular (FastTrack, eDonkey 2000 …) with the growth of the xDSL and the cable connections.
Each network has its own features (pure and hybrid peer to peer networks), however it is not the objective of this document to describe them in details.
Nowadays, the use of P2P clients in corporate environments raises number of matters and drawbacks that I will present in this document:

-In most of cases, the files shared are mostly audio and video contents and lots of them are originally illegal copies. To resume, this activity is illegal in most countries. In some cases, the people in charge of information system can be prosecuted.

-The files downloaded, sometimes, contain viruses, Trojan horses and backdoors. Theses threats can cause a lot of damages to corporate information systems.

-The use of P2P will impact the performance of the corporate network, as well the WAN link as that internal network.
This misuse will decrease the performance of critical applications…

Anyway, this use in corporate environment must be forbidden or limited. In this study, I shall present a range of possibilities in order to stop or limit this illegitimate traffic.

My analysis is directed around three axes: Workstation level, network level and application level.

Once treated through these three elements, this malicious traffic can be immediately stopped. Yet, this fight requires a technology watch effort, because more and more peer to peer application developers try to find technical responses to legitimate this traffic (using well-know port numbers for example) or to make it encrypted, and then impossible to identify as such.

[1] http://csdl.computer.org/comp/proceedings/p2p/2001/1503/00/15030101.pdf

2. OVERVIEW ABOUT P2P APPLICATIONS

Before describing the different ways to block P2P applications, I will present to you the most important applications with theirs functions.
I will not precisely describe how each application operate but will focus on a short description by taking in account network flows.

2.1. EDonkey Network:

The eDonkey network is a peer to peer network that relies on servers to connect users.
It runs multiple international servers, so it is unlikely that it will ever be shut down as the centralised, US-based Napster servers were. The use of this P2P network relies on eMule and eDonkey 2000 clients and servers.

The following chart indicates the different default services used by application:

Services / Rules	Description
TCP port 4661 (dst) [2]	To connect to servers
TCP port 4662 (dst)	To connect to other clients
UDP (dst)	To search sources, queue rating…
TCP port 4662 (src) [3]	To permit others to download our downloaded sources

Note: if a client doesn't authorize incoming connections for any reason, the peer to peer network will allocate it a low ID and it will not be able to achieve full transfer speed.
Nevertheless, theses ports bindings can be modified in using the graphical user interface.
To summarise, a connection, by default, proceeds in the following way:

-IP client -> IP Server: 4661 (TCP)
Client connects on eDonkey server to get list of shared files on different peers.

-IP client -> IPs Clients: 4662 (TCP)
Downloads between different peers use by default TCP port 4662.

-IP client->IPs clients: 4672 (UDP)
In function of versions: to search some sources and recently some mechanisms to optimize downloads.

-IP client-> IPs servers: 4665 (UDP)
Source asking on eDonkey servers.

Note: The following link gives in detail all connections:
http://www.emule-project.net/home/perl/help.cgi?l=1&rm=show_topic&topic_id=122

[2] dst : Destination
[3] src : Source

Note: In the last release of software, the source exchange between peers used TCP port 4662. In fact, it's important to check the evolutions of released softwares (to block the flows).

2.2. FastTrack:

The FastTrack architecture is composed of two sub-systems in which the first tier consists of fast connections to the network and the second tier consists of slower connections. Clients on the first tier are known as **SuperNodes** and clients on the second tier are known as **Nodes**.
Thus, SuperNodes (clients with fast connection) become searching servers for others Nodes.

Downloads on FastTrack based on HTTP requires following the steps below:

"-Once the location of the file has been found, the client that wants the file connects to the client that hosts the file and sends a HTTP request for the file.

-The client hosting the file interprets the request and sends a HTTP response. The HTTP headers used in FastTrack have been modified to accommodate extra information such as meta-data but standard HTTP/1.1 headers are supported which means that files can be downloaded from KaZaA clients through a web-browser such as Internet Explorer or Mozilla.**"**[4] (see reference)

To resume, downloading on FastTrack (via KaZaA 2.6.3) can used the TCP port 80 via option on client.

2.3. BitTorrent:

BitTorrent is a protocol designed for file transfer. It is peer-to-peer by nature, as users connect to each other directly to send and receive portions of the file. However, there is a central server (called a tracker) which coordinates the action of all peers.
In order to describe the mechanism used by BitTorrent, I used a sniffer to analyse the different generated requests.
In fact, there are 4 main steps when a user executes a torrent link. Nevertheless, the third step is the most important to block this traffic because the downloading is occurring during this step.

-IP client -> IP server (hosting torrent file)
This step allows getting a torrent file. In this file, we will get the tracker server.

-IP client -> Name server
To resolve a tracker name (tracker**.***torrent.com)

-IP client -> Tracker Server: port TCP (usually 2004)

[4] http://ntrg.cs.tcd.ie/undergrad/4ba2.02/p2p/

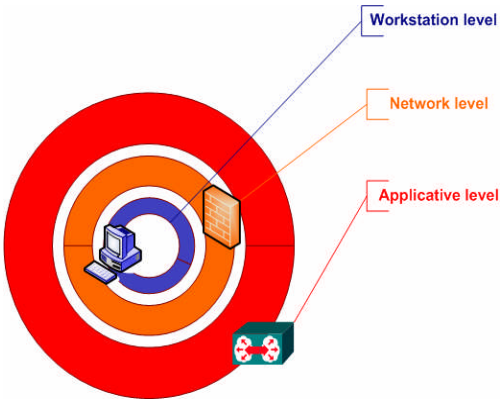
-IP client -> Poll of IP addresses
Get sources from a pool of IP addresses (actually a range of TCP Ports is the following 6881-6899)

Note: The TCP port 2004 can vary according to the trackers (TCP port 6969 is often used).
Currently, the last release of software is 3.4.2. Moreover, prior to version 3.2, BitTorrent by default uses ports in the range of **6881-6889**. As of 3.2 and later, the range has been extended to **6881-6999.**

Note: We can obtain more informations by using the following link:
http://btfaq.com/serve/cache/25.html

3. How to limit & block P2P

The drawing below illustrates the different levels where we can operate to block P2P traffic.



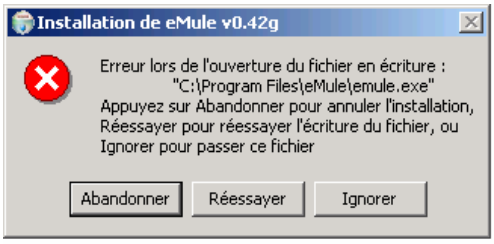
Indeed, my analysis is based on theses three directions to block this traffic.
Acting on each level has its advantages and drawbacks according to the situation of the corporate information system. Nevertheless, several methods can be used at the same time to obtain better results.

Note: To execute the different tests, I will use various P2P clients (KaZaA 2.6.3 and eMule 0.42g).

3.1. Workstation level:

The first level is to limit the rights of the users in order to prevent the installation of clients (eMule, KaZaA…) on their workstations.
In order to illustrate the "Workstation Level", I have created a simple user on Windows 2000 Professional with service pack 4 (French release) in the following group "utilisateurs".

Once logged with this user, I have downloaded the eMule client (to download in using eDonkey network).
The following screenshot shows that the user does not have the sufficient rights.

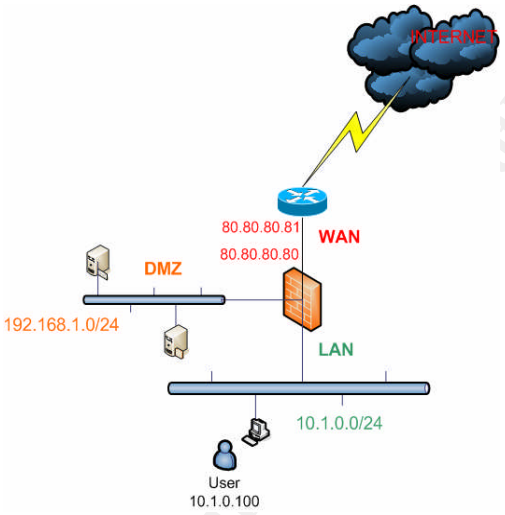


Of course, this method works in many of cases.

In the facts, the system administrators in corporate environment have a more flexible policy of rights. Indeed, the workload in order to manage authorizations may make this activity difficult and the load attached to it particularly heavy. Finally, it depends of the corporate environment size.

3.2. Network Level:

To deal with this chapter, we will consider the following network:



Inside Network (LAN): 10.1.0.0 (subnet 255.255.255.0)
Outside Network (WAN): 80.80.80.80 – 80.80.80.81 (arbitrary)

We assume that workstation (IP address 10.1.0.100) has the firewall as default gateway. The servers in DMZ have a limited impact about traffic of users (an impact about bandwidth but in fact users reach Internet in direct without proxy). The firewall makes a network address translation to access to Internet.

Note: The different ranges of IP addresses are totally fictive.

Firewall:

Nowadays, most Information Systems are protected by an active network device called firewall which filters packets according to a rule base.

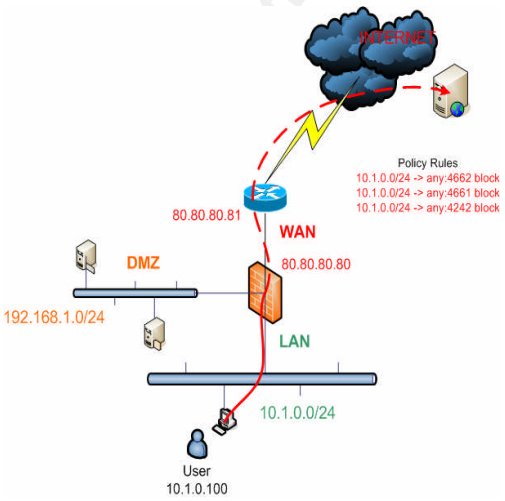
Firewalls provide a variety of services to networks in terms of security. The most important in our case is the filtering of traffic.
Currently, there are lots of methods to filter the network traffic.
This table is an overview about firewall and theirs functions:

Methods	Presentation
Packet filtering	Packet filtering is the process of deciding the disposition of each packet that can possibly pass through a firewall. The criteria used in each filtering rules are numerous (source address, destination address, protocol, source and destination port …)
Stateful inspection	Stateful inspection is a firewall architecture that works at the transport layer. It refers to the ability to track connection "state information". It means that the firewall has the capacity to base control decisions based on previous communication.
Deep inspection	This method describes the capabilities to look within the application payload of a packet or traffic stream and make decisions on the significance of that data based on the content of that data.

Nevertheless, in our situation, we assume that the Stateful inspection method is being used.
In the following examples, I have tried to identify the main traffic flows bound to applications in order to block it at network layer.

Firewall and eMule

I assume that my client (4662 TCP port local reachable from Internet) is not available from Internet. That means that when I connect to the eDonkey network, I will have a low ID (because I do not contribute to the community by putting my files online and available to other users, my speed of download will be low).
In order to block or at least to limit eMule traffic, it's possible to block some TCP ports on the gateway.

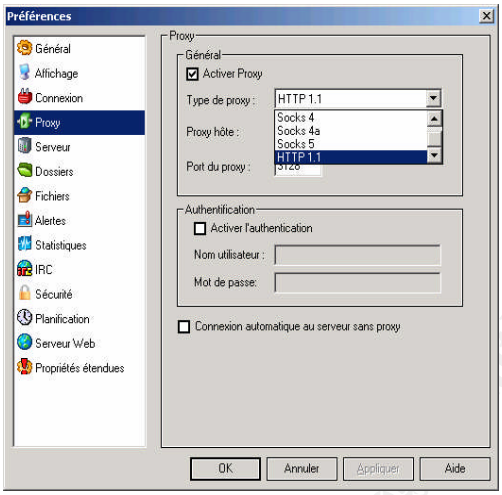


Finally as the graph described it, we will use this following policy rule (to block):

Inside Network -> Any: 4662 TCP (transfer file) Inside Network -> Any: 4661, 4242… TCP (connection servers) Inside Network -> Any: 4672, 4665 UDP (search sources)

Note: Some eMule servers from Internet bind different TCP Ports than 4661 and 4242.

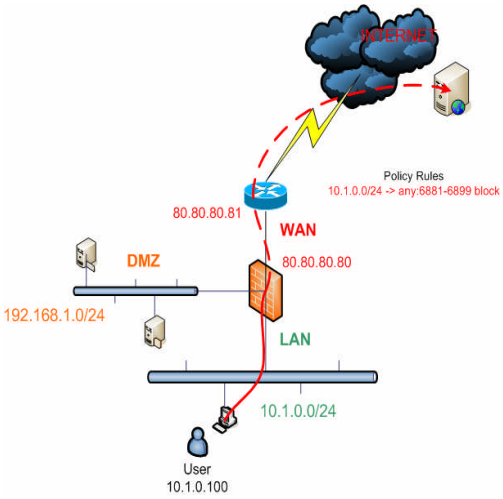
In many cases, some users will be able to use a proxy server in order to bypass the security policy. In its last release, the eMule client can select a proxy server in function.



From Internet, a large number of sites propose some open proxy lists.
http://www.openproxies.com/

Firewall and BitTorrent

According to the presentation of this application, we have to block a range of TCP ports in order to forbidden this traffic. In general this rule blocks most of downloads.



Inside Network -> Any: 6889-6899 TCP (transfer file)

Nevertheless, in some cases, we can change the range of TCP ports while creating of a "downloader" via the following options: minport 6881 --maxport 7000.
This program ("original" downloader) has to run on the server which hosts the file to download.

Firewall and KaZaA

With a very strict policy in order to block this traffic, this flow can't be stopped by a device working at layers 3 and 4.
Without taking into account the customer requirements (about flows), a strict policy block all incoming traffic from Internet and authorize outcoming only some services as http, dns, smtp, pop3…

Despite this, in last resort, the P2P traffic will use port 80 used also by http protocol.

Conclusion

The main weapon, that a network administrator has to block peer to peer, is to deny services in function of applications.
When you are in charge of monitoring a security policy, it's crucial to authorize only the flows that are necessary for operations of information system.
Moreover, some firewalls are able to limit the number of source IP based sessions (to limit the number of sessions generated by P2P client) and to make traffic shaping to limit impact on bandwidth.
Nevertheless, despite of this method (allow only defined and useful services), some applications such as KaZaA have possibilities to function through the 80 port or to use proxies.

In theses conditions a network device working at network layer cannot differentiate a legitimate HTTP request (in accordance with RFC 1945[5]) and KaZaA flows.

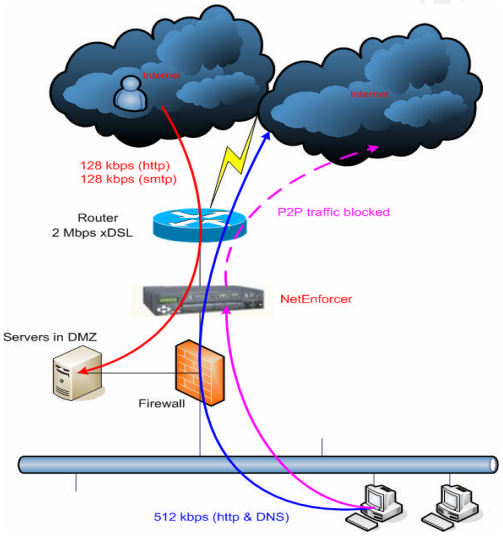
3.3. Application Level

A good solution to analyse the use of access point of internet is to deploy a device which has the capacities to analyze the payload of IP packets.

In this document, I will present to you three products which may help to differentiate flows in using TCP port number and the patterns of IP packet payload.

NetEnforcer

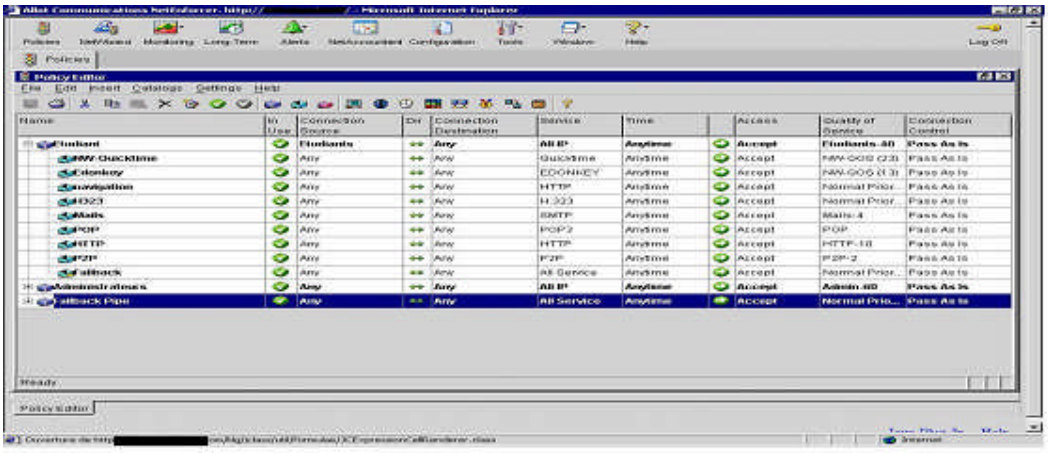
NetEnforcer[6] is an appliance which offers a traffic shaping functionality in order to allocate bandwidth for each application and service.
In fact, the NetEnforcer identifies and classifies traffic up to the application layer. Indeed, the traffic shaping method can identify a protocol in using some ports pre-defined and to analyse the payload of a packet. This lets you distinguish between multiple applications using the same protocol. For example, you could differentiate between a legitimate http requests and a KaZaA traffic using port 80…
Thus, it's possible to block dynamic port applications (P2P, H.323…)
This appliance is deployed in a bridge mode to be transparent in the network architecture.



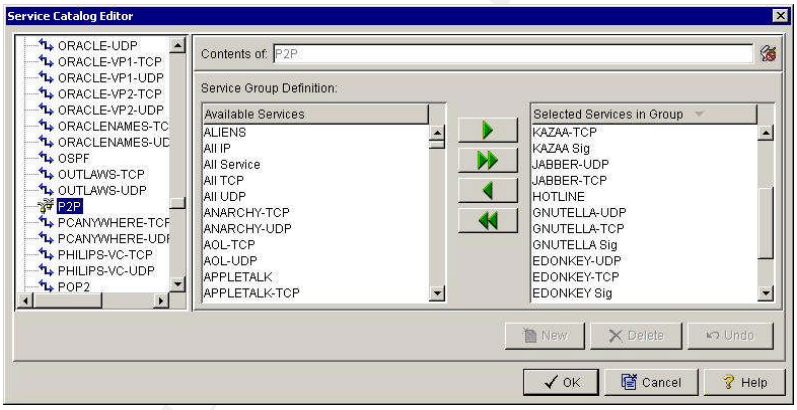
In most of cases, NetEnfocer has to be deployed just before the device which in charge of translating addresses (sources addresses) to generate some reports with real (internal) IP addresses.
Moreover, this device offer a protection from denial of service (Dos/Ddos) attacks by limiting the number of connections per second and by defining a

[5] http://www.faqs.org/rfcs/rfc1945.html
[6] http://www.allot.com/pages/products_index.asp?intGlobalId=2

policy in case of DOS attacks: either admitted traffic without QoS or dropped the traffic.
To summarise we can define a real traffic shaping policy in the same way as when you define a policy of security with IP addresses sources, IP addresses destinations, applications, TCP and UDP ports…



We can observe a rule which limit P2P traffic. This rule is based on the following P2P services.

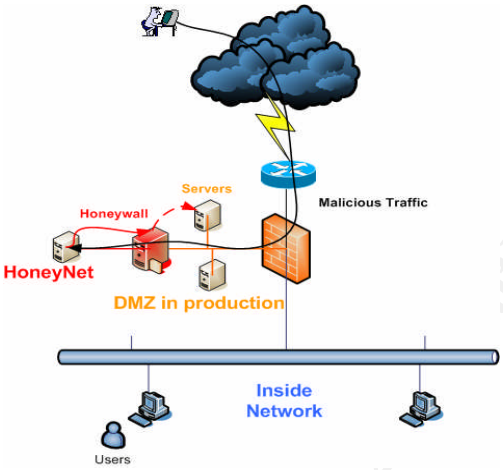


The definition of each application is based at the same time on TCP and UDP ports and protocol signatures. Yet, I didn't present you a fingerprint of signature, the NetEnforcer don't offer this option (to edit the different signatures and to make it).
To conclude about this device, it offers an excellent way of blocking P2P traffic or of limiting through its traffic shaping capability. One of the most important criteria (of this type of product) is the update of signatures and reactivity to edit new signatures as soon as a new P2P network appeared.

Note: The market offers a great number of competitors such as PacketShaper of Packeteer.

Snort_inline

The aim of this project is to offer an IPS (Intrusion Prevention System) based on snort.
Snort is a famous IDS (Intrusion Detection System) which makes it possible to treat the payloads of packets in order to detect malicious traffic as intrusion attempts.
To resume briefly, an IDS is a network device that observes traffic and attempts to identify and notify dangerous traffic while an IPS is an in-line system that block malicious traffic before it reach its targets.
In order to test this method, I have used Honeywall CDROM. The main goal of this bootable CDROM is to set up a Honeywall gateway.
To describe the operation of this Honeywall, the following diagram explains to you the direction of different flows.



Thus, all traffic from Internet is allowed to HoneyNet. Nevertheless, once that hacker has a complete control on HoneyServer, his actions are limited. Without being exhaustive:
- The firewall iptables ensure a connection limiter. For example, the hacker will be limited in a number of icmp, udp and tcp packets.
- Honeywall supports a snort-inline to forbid malicious traffic and intrusion attempts…
Of course, the second point interest us to block P2P traffic.

Snort-inline in combination with iptables as a bridging firewall will send packets to “userspace” for processing.
Snort-inline (userspace application) can receive and possibly manipulate the packets traversing the bridge.

Snort_inline supports three rules:

Rules	Presentation
Drop	The drop rule tells iptables to drop the packet and log it.
Sdrop	The sdrop rule tells iptables to drop the packet. Nothing is logged.
Reject	The reject rule type tells iptables to drop the packet, log it and send a TCP reset if the protocol is TCP or an ICMP port unreachable if the protocol is UDP.

Snort_inline use some filters to determine malicious traffic and in our case, support a rules base to detect the following P2P networks:
- P2P applications detected: Napster, GNUTella, Fastrack and BitTorrent.
(See p2p.rules – see annexes). An important advantage is the possibility of writing snort rules[7] to be more reactive.

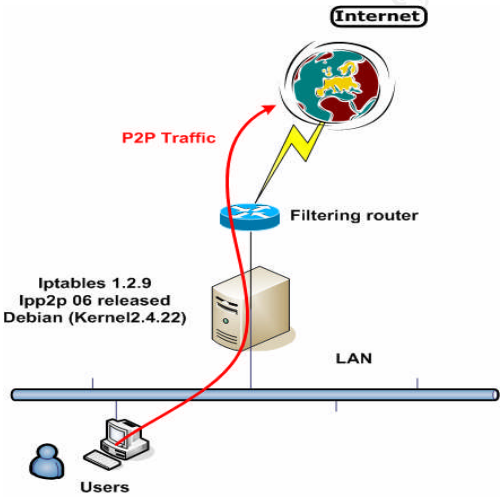
Honeywall CDROM is not an operational solution to block P2P in the direction where honeywall doesn't permit all inbound and outbound connections. Yet, it's a good start to test snort-inline without to compile the sources.

IPP2P

The last method that I present to you is an extension for the firewall iptables.
IPP2P is a module which analyse the payload of TCP packets for detecting patterns of P2P networks. It doesn't work for UDP packets.
The last release has able to detect the following P2P Applications:

- eDonkey
- Gnutella
- FastTrack
- DirectConnect…

This combination IPP2P/iptables can support at the same time bridge mode or routing mode.
The following diagram describes where the couple Iptables/IPP2P has been deployed:



The Kernel has to be compiled with netfilter enabled. The bridge mode is an option in the kernel setup. Moreover, using netfilter on a bridge requires a kernel patch available at the ebtables homepage (in routing mode, no patch is necessary).
This module is installed by compiling the sources. I don't describe exactly the method to install this module.

[7] http://packetstormsecurity.nl/papers/IDS/snort_rules.htm

An example is the following line to forbidden p2p traffic:

```
# iptables –A forward –p tcp –m ipp2p –ipp2p –j drop
```

Note: the option –ipp2p is used to drop all P2P traffic. Nevertheless, you can setup policy rules via an options base (at this time).

To improve your knowledge about this solution, you should visit the official site which describe of many examples.
http://rnvs.informatik.uni-leipzig.de/ipp2p/index_en.html

4. CONCLUSION

Since the beginning of file sharing using P2P Architecture, we have observed an evolution:
First these networks are now based on distributed architectures and finally are more and more difficult to block by simply shutting down centralised servers (as at the time of the Napster network).
After this first evolution, some P2P networks have tried to handle their traffic using well-know ports and protocols such as the KaZaA network.
Consequently applications such as KaZaA can't be stopped by blocking a range of ports at level 4, without stopping legitimate traffic as http requests.
Nevertheless some traffic shaping appliances (which are able to analyse the payload of packets – at Level 7) can identify P2P traffic via protocol signatures (protocol analysis). Using the methods described in this document, P2P networking can be prohibited on a corporate network.

However, in the next future with more and more restrictive policies in place (ISP filtering, legal proceedings…), it is foreseeable that some projects will develop new architectures using ssl encryption. Thus, with encrypted traffic, the analysis of payload will become impossible and some new ways to fight P2P applications will have to be found.

5. ANNEXES

5.1. P2P.rules (Snort Rule)

```
# © Copyright 2001-2004, Martin Roesch, Brian Caswell, et al.
#    All rights reserved.
# $Id: p2p.rules,v 1.15 2004/06/15 13:39:49 cazz Exp $
#-------------
# P2P RULES
#-------------
# These signatures look for usage of P2P protocols, which are usually
# against corporate policy
alert tcp $HOME_NET any -> $EXTERNAL_NET 8888 (msg:"P2P napster login";
flow:to_server,established; content:"|00 02 00|"; depth:3; offset:1; classtype:policy-violation;
sid:549; rev:8;)
alert tcp $HOME_NET any -> $EXTERNAL_NET 8888 (msg:"P2P napster new user login";
flow:to_server,established; content:"|00 06 00|"; depth:3; offset:1; classtype:policy-violation;
sid:550; rev:8;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 8888 (msg:"P2P napster download attempt";
flow:to_server,established; content:"|00 CB 00|"; depth:3; offset:1; classtype:policy-violation;
sid:551; rev:7;)
alert tcp $EXTERNAL_NET 8888 -> $HOME_NET any (msg:"P2P napster upload request";
flow:from_server,established; content:"|00|_|02|"; depth:3; offset:1; classtype:policy-violation;
sid:552; rev:7;)
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"P2P GNUTella client request";
flow:to_server,established; content:"GNUTELLA"; depth:8; classtype:policy-violation; sid:1432;
rev:6;)
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"P2P Outbound GNUTella client
request"; flow:to_server,established; content:"GNUTELLA CONNECT"; depth:40;
classtype:policy-violation; sid:556; rev:5;)
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"P2P GNUTella client request";
flow:to_server,established; content:"GNUTELLA OK"; depth:40; classtype:policy-violation;
sid:557; rev:6;)
alert tcp $HOME_NET any <> $EXTERNAL_NET 6699 (msg:"P2P Napster Client Data";
flow:established; content:".mp3"; nocase; classtype:policy-violation; sid:561; rev:6;)
alert tcp $HOME_NET any <> $EXTERNAL_NET 7777 (msg:"P2P Napster Client Data";
flow:to_server,established; content:".mp3"; nocase; classtype:policy-violation; sid:562; rev:5;)
alert tcp $HOME_NET any <> $EXTERNAL_NET 6666 (msg:"P2P Napster Client Data";
flow:established; content:".mp3"; nocase; classtype:policy-violation; sid:563; rev:6;)
alert tcp $HOME_NET any <> $EXTERNAL_NET 5555 (msg:"P2P Napster Client Data";
flow:established; content:".mp3"; nocase; classtype:policy-violation; sid:564; rev:7;)
alert tcp $HOME_NET any <> $EXTERNAL_NET 8875 (msg:"P2P Napster Server Login";
flow:established; content:"anon@napster.com"; classtype:policy-violation; sid:565; rev:6;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 1214 (msg:"P2P Fastrack kazaa/morpheus
GET request"; flow:to_server,established; content:"GET "; depth:4;
reference:url,www.kazaa.com; reference:url,www.musiccity.com/technology.htm;
classtype:policy-violation; sid:1383; rev:6;)
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"P2P Fastrack kazaa/morpheus
traffic"; flow:to_server,established; content:"GET"; depth:3; content:"UserAgent|3A|
KazaaClient"; reference:url,www.kazaa.com; classtype:policy-violation; sid:1699; rev:7;)
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"P2P BitTorrent announce request";
flow:to_server,established; content:"GET"; depth:4; content:"/announce"; distance:1;
content:"info_hash="; offset:4; content:"event=started"; offset:4; classtype:policy-violation;
sid:2180; rev:2;)
alert tcp $HOME_NET any -> $EXTERNAL_NET 6881:6889 (msg:"P2P BitTorrent transfer";
```

```
flow:to_server,established; content:"|13|BitTorrent protocol"; depth:20; classtype:policy-
violation; sid:2181; rev:2;)
```

5.2. Sources

From Internet:

-Emule project
http://www.emule-project.net/
http://www.emule-project.net/home/perl/help.cgi?l=1&rm=show_topic&topic_id=122

-eDonkey project
http://www.eDonkey2000.com/

-IPP2P Project
http://rnvs.informatik.uni-leipzig.de/ipp2p/index_en.html

-Allot Communications website
http://www.allot.com/
http://www.allot.com/pages/products_index.asp?intGlobalId=2

-Writing Snort Rules
http://packetstormsecurity.nl/papers/IDS/snort_rules.htm

-Honeywall project website
http://www.honeynet.org/tools/cdrom/

-P2P Definition
http://csdl.computer.org/comp/proceedings/p2p/2001/1503/00/15030101.pdf

-BitTorrent Informations
http://btfaq.com/serve/cache/25.html

-RFC 1945 HTTP 1.0
http://www.faqs.org/rfcs/rfc1945.html

-Peer to Peer technologies and protocols
http://ntrg.cs.tcd.ie/undergrad/4ba2.02/p2p/