



C·CURE

Information Security Architects

Introduction into Linux Firewalling

Jan Vanhercke
OTA president



- w** What this presentation is not about (and why)
- w** Why is this only an introduction
- w** How complex is the problem
- w** Different approaches to firewalls
- w** Sample setup of a Linux secured environment
- w** Questions and Answers



What this presentation is not about

- w** I will not tell you:
 - TO USE GOOD PASSWORD
 - NOT TO USE BUGGY SOFTWARE
 - TO READ YOUR LOGS
 - NOT TO COMPILE OR RUN UNSIGNED OR UNKNOWN APPLICATIONS
- w** If you really care about security follow the mailing lists (CERT, BUGTRACK, X-FORCE, etc...)
- w** Well managed systems are less likely to be vulnerable
- w** Visit the hackers/info sites and get a reality check:
 - www.insecure.com
 - www.anticode.com
 - www.rootshell.com



Why is this only an Introduction

- w** Security is not just a matter of 'is it safe ?'
- w** There is no such thing as a zero risk
- w** Evaluate risks versus efforts
- w** Security can only be achieved by
 - Defining a **policy** (what you want to achieve)
 - Defining a supporting **architecture** (how you organize the environments)
 - Defining the proper **technology** (the products or freewares.... :-)
- w** Looking just at the technology does not eliminate the risks (I.e. is it really what you are looking for)



How complex is the problem

- w** IP has authentication built in, so anyone can imitate somebody. e.g.:
 - spoofing: faking the address
 - session hijacking taking over the communication
- w** Most traffic (99,99%) travels in clear text. Passwords can be read over the net.
 - Sniffing: a simple 'tcpdump' can reveal passwords
 - Switched networks can be fooled using ARP cache poisoning
- w** Most systems have errors in their IP stack, in the OS handling of incoming requests, or in the reassembly of packets.
 - denial of service against M\$-Windows systems
 - TCP session spoofing in Linux 2.0.35



How complex is the problem

- w** Not only the kernel is to blame, but the libraries or applications may contain errors, cause stack overruns (core dumps).
 - For instance IQUERY attack against DNS.
- w** Some protocols go in special cases into 'degraded' security mode, an opportunity for a hacker.
 - For instance the SMB protocol may downgrade to unencrypted passwords.
- w** Sometimes the design of the security is simply bad.
 - Encryption may lead to information hijacking... 'give me 1Mio for the key to your newly encrypted data, or loose it forever'.



How complex is the problem

- w** Some measures are so transparent you wouldn't notice you were missing them...
- w** Sample policy:
 - encrypt all packets between A and B !!!
- w** But is the transfer of mail encrypted between A and B encrypted ?
 - The mail server may decided to use a smart relay going via system C (not encrypted) or use the system referred to by the MX record
- w** So, encrypt the e-mail!!!
 - But what if you loose the key, if you want someone to read you mail while you are away, or if you want to check on virusses (obviously only for M\$).



How complex is the problem

- w** The ultimate challenge... knowledge!!!
- w** Understanding
 - routing
 - ip features
 - OS security
 - Application security
- w** Firewalls typically do not eliminate all of these risks, they prevent a system from being accessed beyond what is desirable.
- w** Hence, the less you need, the fewer things can go wrong
- w** Furthermore, hacking has never been easier... 'script kiddies'.



Different approaches to firewalls

- w** Firewalling using proxy technology:
 - works at the application level only
 - is generally quite secure (if well configured)
 - but is quite slow, and often not transparent for the application
- w** Linux solutions may include:
 - apache (for web) <http://www.apache.org>
 - delegate (for pop, telnet, ftp, etc...) <http://www.delegate.org>
 - socks (generic IP proxy) <http://www.socks.nec.com/>
 - tis <http://www.tis.com>
- w** Common errors
 - Proxies that work in both directions
- w** Mostly used if NAT and IP filtering is not possible, or if strict access control is needed



Different approaches to firewalls

- W** Firewalls using IP filtering
 - standard feature of Linux kernel
 - mostly used in combination with 'masquerading'
 - is fast and very lightweight
- W** Linux solutions may include:
 - using ipfwadm for 2.0.x kernels
 - using ipchains for 2.1 and 2.2 kernels
 - use of defragmentation and masquerading
- W** Common errors
 - lack of understanding of IP operations
 - wrong placement of the filters



Different approaches to firewalls

W Firewall using tcpd wrapper

- restricts at the application level
- can be used as an intermediate daemon between the application and inetd
 - only usable for not pre-spawned applications, mostly TCP based
- can be built in the application by using the hosts_access API
 - interesting for UDP based communications (like RPC)
 - built in:
 - portmapper
 - mountd
 - nfsd

W GET THE RIGHT VERSION!!! TROJAN INFECTED IMPLEMENTATIONS ARE CIRCULATING ON THE NET

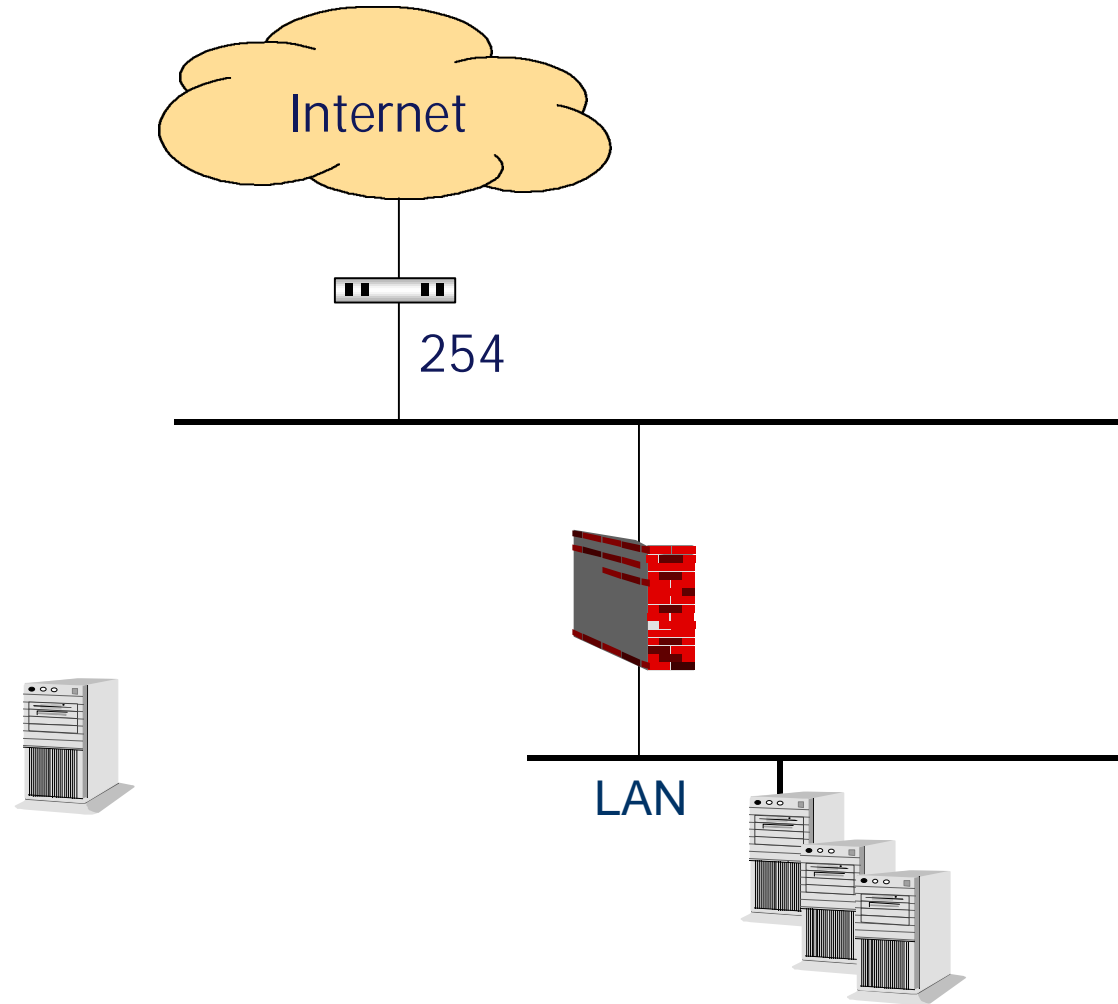


Practical situation

- w** Provider gives us an IP range (C-class) 1.2.3.0/24.
- w** We will subdivide this into a DMZ and an external (unprotected) network segment
 - 1.2.3.1 - 1.2.3.126 = DMZ
 - 1.2.3.129 - 1.2.3.254 = External network
- w** We have an internal network using private address space = 192.168.1.0
- w** Allocated addresses:
 - Router = 1.2.3.254
 - Firewall = 1.2.3.129, 1.2.3.126 and 192.168.1.254
 - Webserver/Mailrelay/DNS = 1.2.3.1



Typical configuration





What do we want to achieve

- w** The firewall is protected and inaccessible for the outside world.
- w** The webserver communicates with in the internal network, and must be able to deliver and receive e-mail.
- w** The webserver must be able to accept public DNS queries, HTTP requests and operate as a mail relay.
- w** The internal users should be able to use the Internet as if they were directly connected.
- w** Spoofing controls are placed on the router (CISCO)



What we want to achieve

- w** Systems on the DMZ should not be allowed to go to the internal network (except for delivering mail).
- w** The webserver runs the DNS server for the Internet.
- w** The mailserver should only relay message to *ourdomain.be* when accessed from the outside.
- w** The mailserver must refuse mails from spammers on the realtime blacklist.



What we will not cover

- W** Time is limited for this presentation, so we will not cover:
 - Configuring the CISCO securely:
 - anti-spoofing,
 - attack protection
 - Setup of the mail relay. We can talk about sendmail for hours, or on the Linux SIG anniversary :-)
 - How you configure Linux routing (hey, this is an introduction into firewalling, not networking)
 - Dual DNS setup using BIND-8
- W** We will cover the details of
 - Kernel configuration (2.0.36 based)
 - IP filters (ipfwadm based)



Configuring the Linux kernel

- W** All features needed are located in the 'networking options' when building the kernel.
- W** Needed are:
 - **Network firewalls**
 - **IP forwarding/gatewaying**: enables routing between interfaces
 - **IP firewalling**: enables filtering capabilities
 - **IP firewall packet logging (opt.)**: eventually logs communications
 - **IP masquerading**: Network Address Translation (Hiding)
 - **ICMP masquerading**: NAT Hiding for ICMP (IP control information)
 - **IP defragment**: needed for masquerading
 - **IP accounting (opt.)**: counts the traffic
 - **DROP source routed frames**: disables the ability to hard code a route in the IP packet. SOURCE ROUTED PACKETS ARE A DANGER TO SECURITY!!!



About IP forwarding

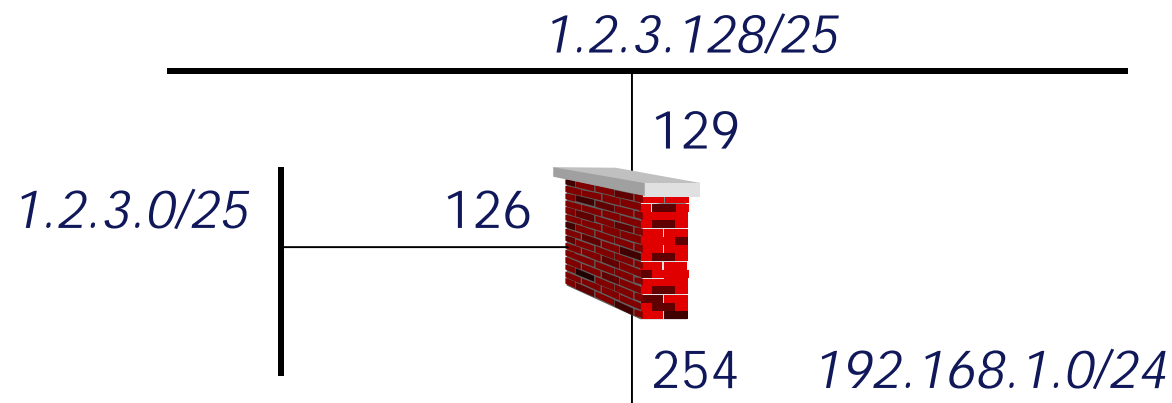
- w** When a packet enters the firewall on a system, the kernel will check if it is targeted at one of its addresses.
- w** If this is not the case, the packet is normally dropped.
- w** With forwarding, the kernel will look at the routing tables and send it back to the interface that the routes dictate.
- w** On later versions of the kernel this can be activated/deactivated in general by setting `/proc/sys/net/ipv4/ip_forward`
 - e.g. `echo 1 > /proc/sys/net/ipv4/ip_forward` will enable forwarding
- w** Firewall rules may disable routing as well



About IP masquerading

- W** If a packet should be masqueraded then the system will do the following:
- it will look at the destination IP address, and look for the route to that system.
 - Based on the route to follow, the kernel decides what interface, and IP address (*fip*) is needed to send the packet to
 - The kernel will replace the source address to the IP address (*sip*) of the requested interface
 - The kernel will replace the source port number (*sp*) to a value ≥ 61000 and < 65096 (*fp*).
 - The kernel registers the mapping (*sip*, *sp*) -> (*fip*, *fp*).
 - When packets return from the host, the kernel looks up the mapping (*fip*, *fp*) and replaces the destination address with the correct number.

- w** Let's assume that all packets from the 192.168.1.0/24 network are to be masqueraded.
- w** Then packets sent to:
 - 1.2.3.0/25 network will be seen as being sent from 1.2.3.126
 - 1.2.3.128/25 network, and the internet, will be seen as being sent from 1.2.3.129





Masquerading implications

- W** Sometimes you may want to have packets forwarded, not masqueraded.
 - A system on 1.2.3.0/25 may need to access a server on 192.168.1.0/24. If the return packet is masqueraded then the communication will never succeed!!!
- W** The masquerading information is stored in a kernel table. Entries in this table will timeout (15 minutes for TCP, 5 minutes for UDP) if no traffic is using that entry. Hence a telnet will close down after 15 minutes of inactivity. This can be modified with 'ipfwadm -s '.
- W** Masquerading does by default translate the content. So if the communication contains IP addresses, or if back connections are needed then the protocol can/will fail.



There is something wrong in protocolland

w DNS protocol

- The content of DNS packets obviously contain IP addresses. They will not be translated (and you probably would not want that anyhow)

w FTP protocol

- FTP establishes a connection to port 21.
- When retrieving data, the server opens a connection for that data to the client coming from port 20.
- The address the server must use is transferred in the first connection.
- So enabling masquerading will not allow FTP to work (unless using the PASSIVE mode)

w Many other protocols are in the same situation.



Installing support for specific protocols

W When you build the kernel, then also build and install the modules:

- *make modules*
- *make modules_install*

W Install the support for specific protocols via `insmod`. Example:

```
for module in /lib/modules/`uname -r`/ipv4/*_masq_*  
do  
    insmod $module  
done
```

W The above shell script will install all protocol supported for the running kernel.

W For 2.0.36 these are: Cuseeme, ftp, irc, quake, raudio, vdolive



Masquerading and the impact on filtering

- w** Normally the firewall should not accept packets from the internet.
- w** If masquerading is activated then packets will arrive for the IP address of the firewall and must be let in to become de-masqueraded.
- w** This could become a security risk, if the system would have applications accepting packets, hence on the packet coming in for a portnumber higher than 61000 and less than 65096 should be allowed. Since the range is 'blocked' for masquerading.



Masquerading and filtering packets in general

- w** When the firewall sends out packets, they are never masqueraded (not even if it is a number needing masquerading).
- w** The communications will either use a fixed port number (in case of daemons listening to certain communications), or will get an number assigned by the kernel.
- w** In the latter case that number will be greater or equal to 1024.
- w** If a system needs to communicate to the outside world it must be able to receive packets targeted at these ports.
- w** Some system daemons run on ports ≥ 1024 : nfsd, X11, oracle, apache.... So these systems would become accessible



Masquerading and filtering packets in general

- W** Filtering on the 'state of the communication' is not a wise idea, since that would disable FTP, and does not work well for UDP communications
- W** Hence, since a firewall is reasonable static, a `netstat -a` will show what daemons are listening for network traffic.
- W** **MAKE CERTAIN THAT THESE DAEMONS ARE NOT ABLE TO RECEIVE TRAFFIC UNLESS NEEDED.**

Important notice:

Some firewall administrator argument that normal traffic can be distinguished from FTP back-connection since they originate from port# 20. Filtering based upon source port, for a public IP address, is very dangerous since a hacker has the choice of picking any port# he likes. Therefore we argument against this practice. Allowing all traffic coming from port #20 is the same as saying, you can attack our systems for as long as you pretend to be an FTP data connection! Remember there is no authenticity in IP packets



Using ipfwadmin

- w** The firewall code can filter packets upon entry (INPUT) or exit (OUTPUT) of an interface.
- w** The firewall code can decide just to forward, or to masquerade and forward the packets.
- w** The firewall also has a default policy that will be applied when packet are not matched against any of the filters available.
- w** Multiple rules can be defined. They will be verified sequentially in the firewall.
- w** Actions that can be taken when a filter matches against a specific packet are:
 - accept: allow the packet to get through
 - deny: ignore/drop the packet
 - reject: send an ICMP message back to the client that this packet is not allowed.



Defining an ipfwadm access list

- w** Some 'die hard' rules:
 - Default policy should be to deny any access.
 - Define the rules from least significant to most general.
 - Always take care of the protection of the firewall FIRST!
- w** The most important instructions for a completely closed firewall:
 - `ipfwadm -I -p deny`
 - `ipfwadm -O -p deny`
 - `ipfwadm -F -p deny`
 - `ipfwadm -I -f`
 - `ipfwadm -O -f`
 - `ipfwadm -F -f`
- w** This will clear the firewall tables and reject all incoming and outgoing information. Don't try this over the network :-)



Is ICMP really needed ?

- w** Some communications must be able to pass through the firewall.
- w** In case of masquerading the firewall must be able to accept masquerading.
- w** ICMP packets have been abused in the past, and make many administrators doubt if they would accept it or not.
- w** ICMP is needed for IP protocol management, so it is desirable to have support for it.
- w** ICMP, when well implemented, should not pose a bigger risk than a correctly implemented IP stack



What filter is needed ?

- W** When a client (C) talks to a server (S) for protocol (P), the client will use a random (*) portnumber.
- W** The rules that must be supported are:
 - allow {C;*} -> {S;P}
 - allow {S;P} -> {C;*}
- W** This can be put into the following ipfwadm rules:
 - ipfwadm -I -a accept -S **C** -D **S P**
 - ipfwadm -I -a accept -S **S P** -D **C**
- W** ipfwadm has a -b option for bi-directional traffic that is the equivalent of:
 - allow {C;*} -> {S;P}
 - allow {S;*} -> {C;P}
- W** It must not be confused with the connections needed for network replies.



Where to drop what traffic

- w** There is much debate about what list is best used to filter the traffic efficiently.
- w** As most traffic will pass through the firewall, the traffic will hit the INPUT filter first, so that may be the best place to do filtering.
- w** For traffic that may originate from the firewall, and that must be restricted, the OUTPUT filter is possibly the best place.
- w** In many cases the firewall is a hard system to access from the outside, but must be capable of communicating to the outside world, hence in most cases the OUTPUT filter is empty and allows all traffic.



The policy for our environment

The firewall

- W** First, we need to clear the firewall policy as explained previously, but we will allow outbound communication:
 - ipfwadm -O -p accept
- W** Secondly, we need to allow communications for the firewall that are strictly needed and reject all the rest. We have to do this for all IP addresses that the firewall has !!!
 - ipfwadm -I -a accept -P icmp -S 0.0.0.0/0 -D \$IP
 - ipfwadm -I -a accept -P udp -S 0.0.0.0/0 -D \$IP 61000:65096
 - ipfwadm -I -a accept -P tcp -S 0.0.0.0/0 -D \$IP 61000:65096
 - ipfwadm -I -a reject -S 0.0.0.0/0 -D \$IP -o



The policy for our environment

The webserver

- w** The internal webserver should be able to perform DNS queries and to send and receive e-mail.
- w** Ident as a protocol is often used with STMP and FTP, so we will support it as well. Ping(ICMP) is sometime used to verify if a server is active
- w** The rules go as follows:
 - ipfwadm -I -a accept -P **icmp** -S 0.0.0.0/0 -D 1.2.3.1
 - ipfwadm -I -a accept -P tcp -S 0.0.0.0/0 -D 1.2.3.1 **25 53 80 113**
 - ipfwadm -I -a accept -P udp -S 0.0.0.0/0 -D 1.2.3.1 **53**
 - ipfwadm -I -a accept -S 1.2.3.1 -D 0.0.0.0/0
- w** The rules above appear to be ok, but will not work, Because inbound packets from an outbound SMTP connection can not get in!
- w** The same problem exists for zone transfers in DNS or even for traffic generated by a DNS lookups on the webserver (for security BIND 8 uses a non-53 port for DNS queries)



There is a problem with 'state'

- w** As the previous example shows, there is a practical problem with a system in the DMZ that acts as a 'client'.
- w** Since its source port is random the firewall must allow a large range of packets to get in (port > 1024).
- w** Some system services run unprotected in the area.
- w** One solution is to look in the IP packet if it is part of an established connection (ACK). But this only works for TCP.
- w** I don't like this, since it doesn't solve the FTP issue, and doesn't work for UDP.
- w** Furthermore if a hacker creates an IP packet with the ACK bit activated, it will get through the firewall. Although currently harmless, there may be side effects in the future.



Solutions for problems of state

- w** So normally, the firewall will need to block all service ports ≥ 1024 that have services running to prevent system services from being used.
- w** The other ports ≥ 1024 should be open to allow packets to get in for client applications.
- w** Commercial firewalls have a mechanism of 'state management' so that they know where the first packet of a session came from.
- w** With a little twist we can use the masquerading features of Linux to achieve the same effect.



(Ab)Using Masquerading for minimal stateful inspection

- w** We provide the webserver with a new IP address (1.2.3.125). And we define an alias for the official webserver address.
- w** We make the configuration in such a way that all default traffic goes via the address 1.2.3.125. This does not affect queries to 1.2.3.1 since that connection is bound to another address.
- w** We can now configure the firewall so that packets from the address 1.2.3.125 get masqueraded.
- w** Masquerading will only allow packets through that are part of a connection (tcp, udp or icmp)



The policy for our environment

The webserver

- w** The webserver is allowed to connect to any system on the internet. We may want to change this so that only e-mail and DNS is allowed.
- w** The complete list for inbound traffic is the following (don't forget to add the rules for the protection of the firewall first):
 - ipfwadm -l -a accept -P **icmp** -S 0.0.0.0/0 -D 1.2.3.1
 - ipfwadm -l -a accept -P **tcp** -S 0.0.0.0/0 -D 1.2.3.1 **25 53 80 113**
 - ipfwadm -l -a accept -P **udp** -S 0.0.0.0/0 -D 1.2.3.1 **53**
 - ipfwadm -l -a accept -P **tcp** -S 1.2.3.1 **25 53 80 113** -D 0.0.0.0/0 **-y**
 - ipfwadm -l -a accept -P **udp** -S 1.2.3.1 **53** -D 0.0.0.0/0
 - ipfwadm -l -a accept -P **tcp** -S 1.2.3.125 -D \$MAILSERVER **25**
 - ipfwadm -l -a accept -P **tcp** -S \$MAILSERVER **25** -D 1.2.3.125 **-y**
 - ipfwadm -l -a accept -P **tcp** -S \$MAILSERVER -D 1.2.3.125 **113**
 - ipfwadm -l -a accept -P **tcp** -S 1.2.3.125 **113** -D \$MAILSERVER **-y**
 - ipfwadm -l -a reject -b -S 1.2.3.0/25 -D 192.168.1.0/24 -o
 - ipfwadm -l -a accept -S 1.2.3.125 -D 0.0.0.0/0
 - ipfwadm -l -a accept -S 192.168.1.0 -D 0.0.0.0/0
 - ipfwadm -l -a reject -S 0.0.0.0/0 -D 0.0.0.0/0 -o



The rules for masquerading

- W** We do not want to do masquerading between the internal network and the DMZ.
- W** We need to do masquerading for the 1.2.3.125 address when going to the internet, and for 192.168.1.0/24 going to the internet.
- W** The complete rules are:
 - ipfwadm -F -a accept -b -S 192.168.1.0 -D 1.2.3.0/25
 - ipfwadm -F -a accept -m -S 1.2.3.125 -D 0.0.0.0/0
 - ipfwadm -F -a accept -m -S 192.168.1.0/24 -D 0.0.0.0/0



Some final remarks

- w** This overview is not complete. We could improve the policy so the the policy also controls if the IP addresses are coming from the right network card.
- w** The instructions should be put into a script.
- w** If using a dial-up link (PPP), it is possible to call that script from `/etc/ppp/ip-up`, and obtain a secured dial-up link.



C-CURE

Information Security Architects

**K. Rogierstraat 27
B-2000 Antwerpen**

Tel: +32 (0)3 216.50.50

Fax: +32 (0)3 216.50.51

**Email: info@c-cure.be
<http://www.c-cure.be>**

