Global Information Assurance Certification Paper

The OpenBSD Invisible Firewall

Gerald Comeaux

GSEC Practical Assignment 1.4b, option 1

INTRODUCTION

Firewalls are a very important part of network security. There are many options out there, and many different factors to consider. An option that many people are unaware of, and therefore can never consider, is the OpenBSD invisible, firewall. OpenBSD is able, without any kernel modification, to set up a bridge between two network interfaces. Coupled with pf, the firewall software that is shipped with OpenBSD, this makes for a powerful firewall.

Could a bridging firewall be a good option for you? The aim of this paper is to explain how it works, go over advantages and disadvantages, and give some examples of how it could be put to use. With this information you should be better suited to make the decision for yourself.

SETUP

As this paper is intended to be more of a research paper on the bridging aspects of the OpenBSD/pf firewall, it will not go very deep into the setup of the firewall itself. There are already numerous web sites available with information on how to install OpenBSD and set up pf, and references will be provided for further information on the process.

For OpenBSD, the disks can be purchased or the setup files can be downloaded to create CD's (note that an ISO image is not available for download, but the files necessary to create a bootable CD are). It is also possible to download a floppy image to begin the setup and have the machine connect to an ftp site to download the necessary setup files, but it should go without saying that putting a newly set up machine on the Internet without having up to date patches could turn out bad. This is not a chance you want to take with a firewall.

OpenBSD's setup is very straightforward. After partitioning the disk, there is a selection of packages to choose for installation. Of the eleven packages, four are for X11 and one is a games package; these packages are unnecessary and can be left uninstalled. When the packages are done, remove the setup CD, reboot the machine and you should now have a computer running OpenBSD.[1] If there are any other packages that you might want to have installed on your firewall, such as Snort (Intrusion detection, www.snort.org) or NTOP (a network traffic monitor, www.ntop.org), this would

be the time to do it. Double check that there are no patches or updates that need to be installed (a quick visit to www.openbsd.org/errata.html), and you can move on to setting up pf.

Pf is OpenBSD's packet filtering software. From OpenBSD's web site: pf "performs two tasks: dealing with packet level forwarding permissions, and mapping hosts/subnets to a range of external addresses."[2] In layman's terms, it was built to filter traffic and to do network address translation (NAT). Pf must be enabled by making a change to the /etc/rc.conf.local file.[2] The firewalls rules are set up in the pf configuration file, /etc/pf.conf.[2] Since this is a text file, and the rules must all be written by hand it may seem a bit daunting at first, but actually proves to be quite intuitive.

To learn more about the pf.conf file and writing pf rules, check out www.benzedrine.cx/pf.html and www.inebriated.demon.nl/pf-howto. Some good examples of basic pf configurations can be found at ezine.daemonnews.org/200207/transpfobsd.html,

Setting up the firewall to pass traffic from one interface to the other is a simple matter of configuring the 2 network cards to be up without IP addresses, and setting up a third bridge interface between the two.[3]

HOW IT WORKS

To understand why the firewall is called invisible, one must have at least a rudimentary understanding of the seven layer ISO OSI model, and IP packets and headers. Therefore, before we get into the specifics of why the firewall is called invisible we shall embark on a brief refresher of the seven layer model.

There are seven layers of networking defined by the International Standard Organization's Open System Interconnect model. From bottom to top they are Physical (wires), Data Link (Ethernet), Network (IP), Transport (TCP, UDP), Session, Presentation, and Application. To cover how these work together, let's imagine that the Application Layer has some information that it needs to get to another computer on the network. The Application Layer will take the data and pass it down to the Presentation Layer. The Presentation Layer will then tack on a small amount of information, called a header, to the front of the data that it just received. The header is there so that when the packet arrives at its destination, the receiving Presentation Layer will know what to do with the data. The Presentation Layer passes it down to the Session Layer, who will do the same thing. As far as the Session Layer is concerned, the entire package that it receives from the Presentation Layer, header and all, is regarded as a single entity of data. Continuing on down the line, each layer will affix its own header, and each layer will regard the information it receives from the layer above as a single piece of data. When the data gets to the bottom, it will be sent to its destination, where the process

will happen in reverse. Each layer on the receiving end will look at its header (the one that was put there by the corresponding layer on the sending side) to determine what to do, remove its header (only), and pass the data to the layer above it.[6] This is not a very exact description of the process, and there are a few discrepancies, (for example, in addition to a header sometimes a trailer is attached) but an in depth look at the model is beyond the scope of this paper and none of the topics discussed here require any deeper understanding.

There is one more consideration that must be taken into account in this example, though, when these two computers talk to each other. If the two computers are not on the same network then the data will not have such an easy journey. Computers on a network only know how to communicate directly with other computers on the same network; all other traffic goes to a router. The first router that a computer uses to get out to the rest of the world is called a gateway. Imagine a network as a neighborhood, and the gateway router as a postman. If you wanted to give a package to your neighbor you could just walk over and drop it off yourself. If you wanted to get something to a friend in a different country you could give it to the postman, and he will make sure that it gets there. He will not deliver it himself, and would most likely get lost in the neighborhood that your far-off friend lives in, but he does know at least how to get it to the postman in that area, who does know the neighborhood there. This is not too very different from how routers work. Routers use headers to tell where a data packet is supposed to go. Two things that are included in every IP (Network Layer) header are the source and destination addresses. Routers only 'unpack' the data up to the IP header, where it makes a small change, packs it back up, and sends it on its way.

The change that the router makes to the IP header is to something called a TTL, or time-to-live, field. The TTL value (stored in the TTL field), also stored in the IP header, is established where the packet originates and is decremented each time that packet gets handled by a router. This brings us to a key point: routers function at the Network Layer (Layer 3), while bridges function at the Data Link Layer (Layer 2). Routers have IP addresses, and decrement the TTL value in the IP header as they pass the packet along. Bridges and Hubs do not. Since they operate at the Data Link Layer, they neither have IP addresses nor do they decrement the TTL value.

The TTL value is primarily there to tell routers when to 'kill' a packet. When a router handles a packet whose TTL value has been whittled down to zero, it drops the packet and sends a “time exceeded” message to the originating host, informing it of the untimely death of it's packet. Without this TTL value, it would be possible for an improperly routed packet to loop around infinitely. Eventually, with enough packets looping around out there with no destination things would start to get pretty clogged up.[4][5]

A secondary use of the TTL value is to map networks. Traceroute, a very

popular network utility, uses the TTL value to determine the path taken and how many 'hops' it takes to get from one spot on a network to another. A packet is sent with a TTL value of one first, so that the very first machine (unless the machine being searched for is on the same network this machine will be the gateway) to see this packet drops it and sends a time exceeded message back to the originator. Concerning the time exceeded message; it also has an IP header. Remembering that the IP header also contains the source and destination IP addresses, in order for the router to send the time exceeded message it is also sending it's own IP address. When the originating host gets the time exceeded message and sees the IP address of the router that sent the message, it now knows the address of the first 'hop' between it and the destination of the traceroute. The originating host repeats this process, incrementing the TTL value by one each time so that the packet will go just one step further and every router down the line will return a time exceeded message to the host. This will show the IP address of every router along the path from the originating host to the destination. The way traceroute is able to differentiate between routers along the way and the destination that it's looking for is quite clever. It sends a UDP (Network Layer protocol) message to the target machine with the target port set very high. These higher numbered ports are known as ephemeral ports, and there are no standard services that accept connections on these ports. When the message reaches the target machine, it will make a request to a port that (in all likelihood) is not accepting request. The target machine will refuse the request ("Hey, I don't have any services running on that port!"), and when the message that the originating host receives is "connection refused" instead of "time exceeded" it knows that that's the machine that it was looking for and can now stop the process. [5]

Setting up the two network cards as a bridge allows all traffic above the Data Link Layer on one of the firewall's network interfaces to be forwarded to the other network interface. They do not decrement the TTL. This is why the firewall is called invisible; It has no IP and it operates as a bridge (Data Link Layer, below the Network Layer) instead of as a router (Network Layer, where IP operates).[7] While the information is getting hustled through from one side of the bridge to the other, PF is able to peek inside each packet and filter the traffic according the rules set in the pf.conf file. The network cards may not be able to see anything above the Data Link Layer, but PF can and it filters on the Network and Transport layers.[8]

IV. Comparison with most conventional firewalls

The OpenBSD/PF bridging firewall is quite a different beast than many other firewalls. Compared to most vendors' products, it is extremely inexpensive. It can be made with an old computer that may be lying around. The minimum hardware requirements are a 133MHz processor, 64MB of RAM, and a 1 GB hard drive.[3] (these are for the i386 platform, for a complete list of supported hardware check www.openbsd.org/faq/faq1.html#Platforms). Doing a quick search on eBay (www.ebay.com) shows that a computer that meets these minimum

requirements can be had for as little as US $20.00. Personally I have yet to pay for the hardware needed to build one, but realistically not everybody has old computers lying around waiting to be resurrected into firewalls.

Should you choose to support OpenBSD, the latest stable release (with pf included) can be ordered on CD for US $40.00. There are no licensing restrictions that would prevent you from installing this copy of OpenBSD on multiple computers, so after purchasing the software you can install it on as many machines as you like. For the extremely frugal, the necessary files can be downloaded for free from the Internet. Potentially, one of these firewalls can be built for no cost other than the time it takes to set up.

As discussed on the first page, the setup is relatively easy. Even for someone with no Unix or Linux experience at all the setup is as simple as following the directions at the www.openbsd.org/faq/faq4.html. The instructions are very detailed and every step in the setup is covered there.

As explained earlier, since this firewall is acting as a bridge it is invisible to IP traffic. It is very difficult to detect. It cannot be pinged, and does not show up in a traceroute. It is likely that an attacker will think that any machines behind this firewall are using personal firewall software (such as Black Ice, Kerio, or ZoneAlarm), rather than thinking they are behind a firewall. Any communication that is allowed through the firewall will seem to be able to communicate directly with the target machine, while anything that is not will seem to just fall away. Since any machine with an IP address on a network has the potential to be attacked directly, removing its IP address makes it infinitely more difficult to compromise.

For computers using firewall software running on top of Windows or Unix/Linux, the administrator must pay careful attention to both operating system and product updates to prevent an attacker from gaining access to the firewall. It would be trouble enough if someone were able to get past the firewall, much less get root or admin access on the firewall itself. In addition to keeping the firewall up to date on software patches, the administrator also has to concern him/herself with the possibility of applied software patches and updates breaking things. Since a downed firewall usually means that any machines it's guarding have lost connectivity until the problem is fixed, this can be a rather large concern. Administrators must walk a fine line between waiting long enough after a new patch is out to make sure it is properly tested and won't break anything, and patching things fast enough to not allow anyone to exploit the firewall. Along with this, not every patch is of the same severity. Some patches must be applied immediately, perhaps due to a rather critical bug that is currently being exploited, while others are relatively small bug fixes. With a bridging firewall this is one less thing that a network administrator has to worry about. This does not mean that you won't have to apply any updates, only that the issue won't be as pressing as if the machine were readily accessible to attackers. This is a good time to clarify a certain point: this firewall is not impervious to attack. While not having an IP address

makes it much harder to detect and attack, and greatly reduces the risk of a compromise on the machine itself, there is still the chance. Never be complacent with updates and patches, even with the bridging firewall.

Of course, there are also drawbacks to a machine without an IP address. The other side of the address-less coin is that without an IP address not only can it not be rooted, it cannot be connected to at all from anywhere on the network. This means that you want to be pretty sure of your firewall rules before you set this machine up, because any changes to the them will have to be made via console. Another drawback of having no remote access is that there is really no easy way to check the logs on the firewall.

Probably the biggest drawback to having no IP, however, is the inability to do network address translation. In order to understand fully what is lost by using a bridging firewall, a brief explanation of NAT is necessary.

Network address translation, or NAT, is when a firewall or router masks the addresses of the sending and/or receiving computers. This is done both for security purposes and to allow more computers to connect to the Internet than a site has registered IP addresses. For example, most high-speed home Internet connections only offer a single IP address to the home user, say 68.11.xxx.xxx. Using NAT, home users can connect as many computers to the Internet as they like while still only having one assigned IP address. Each computer can be given it's own IP address in a private network, say 192.168.1.0. The firewall or router is given two addresses, one on the outside interface with the address assigned by the ISP, the 68.11.xxx.xxx number, and the second interface will have an address in the private network, likely 192.168.1.1. The firewall or router is set to be the default gateway on all of the computers in the private network, and it handles all of the work from there.

NAT works by having the device doing the translation “remember” where all of the traffic coming in and going out is supposed to go. Continuing our example from earlier, if machine A on the private network with address 192.168.1.10 wanted to look at www.openbsd.org it would pass the request to it's gateway, the firewall, to get out to the real world. The firewall would pass the request to www.openbsd.org *using its own address, 68.11.xxx.xxx*. Then, when machine B on the private network with the address 192.168.1.20 wants to check out www.benzedrine.cx the firewall sends the request out again using its own address, 68.11.xxx.xxx. As far as the rest of the Internet can tell, all of the communications from that location are coming from a single computer, the firewall. When the communications come back from www.openbsd.org the firewall remembers that it was machine A that wanted that information and forwards it appropriately. The same will happen with the information from www.benzedrine.cx going to machine B. With NAT, you can put as many computers as you like behind that one firewall. Also, since the rest of the world can only see the address of the firewall, and not the computers behind it, it's more difficult to attack them directly. For security

purposes, it's much easier to protect a single point of entry than it is to try to lock down each computer on the network individually.

Finally, aside from the differences between routing firewalls and bridging firewalls, there are some things that some things that other firewalls offer that OpenBSD and PF simply do not. An example of this would be filtering at layers above the Transport Layer. There are some firewalls that have the ability to act as a proxy server, so that the computers being guarded actually make requests *directly to* the firewall for information. This is different than just acting as a gateway. For example, when using a web proxy the computers on the private network must have their web browsers configured to send HTTP requests to the proxy server. We can expand on our example from earlier to help explain this. If we were using a proxy firewall then when machine A, with it's IP address of 192.168.1.10, wants to look at a web site (say www.sans.org) it will send a request to 192.168.1.1 (it is *asking the firewall* for the page, rather than *passing the request through the firewall*). If the firewall does not have that web site cached, it will make the request to www.sans.org itself and then give the information to machine A *if it passes the rules*. When behind a firewall that is doing network address translation but not acting as a proxy, if machine A wants to look at www.sans.org it will send the request to 65.173.218.106 (the IP address of the web server, in this case the request is being passed through the firewall). Because the web server's address is not on the same network as machine A, it will forward the request to its gateway router (which also happens to be filtering packets in the case of a firewall doing NAT). PF can only check for malicious code in the Transport and Network Layers. A proxy firewall has the ability to go a bit further with the ability to check for malicious code that might be contained in a web page, but it is not transparent to the client computers it is protecting.[9] This is something that should be taken into account when defending a network. If this level of protection is something that you feel might be necessary then by all means use a proxy firewall, but don't rule out the possibility of including a bridging firewall as well. This will be covered more in the next section.

Setting up an OpenBSD bridging firewall is almost as inexpensive as the cheapest available (potentially free), and pf is usually more versatile than anything in that price range. It may even rival some higher-end firewall software. I have been part of OpenBSD and pf replacing a somewhat high end firewall product due to pf's ability filter certain Network Layer protocols (AH and ESP, for example) that the current older version of installed software didn't recognize. To be fair, the firewall software now recognizes and supports filtering of these protocols, but even this could be a small advantage for OpenBSD and pf as the upgrades will always come at a cost even after the initial purchase, and OpenBSD and pf do not cost to upgrade.

APPLICATIONS

One of the nice things about a box without an IP address is that it can be installed on a network without having to worry about making a lot of administrative changes around the network itself, such as setting up gateways

on client machines, changes to router tables, or in the case of a proxy server setting up every client to use the proxy. In any setting where a single computer needs to be defended it will fit nicely. It can be used by itself to protect a home computer or in front of a web server, but it's not limited to guarding just one computer. It can be used to protect whole portions of networks without having to subnet and route between them. It cannot do network address translation, but it can still be used to protect any number of computers. It also can be used to defend a network DMZ (Demilitarized Zone, servers that must be accessed by the outside world are usually put here), guarding perhaps the web and mail servers.

Many times, companies will use 'layers' of firewalls to protect their information (or that of their customers and clients). For instance, the most likely place for a customer's credit card information to be stolen is not in transit, going across the web as many people might think. It is in fact much more likely to be stolen from a server, where it is stored along with possibly tens of thousands of other credit card numbers. You can see why companies would want to go to great lengths to protect these servers. An ideal place to put this firewall would be between a web server (which could be behind a firewall of its own) and the database server that stored the customer credit information, making sure that the only traffic getting through to the database server is coming directly from the web server, and only on certain ports. To an intruder, even though the web and database servers might appear to be on the same network, any traffic that did not originate from the web server and that was not destined for those certain ports would simply get lost for no apparent reason.

It can also be useful for organizations that have temporary users. Some companies will hire a large amount of contractors, or some places will have kiosks set up for clients to check email or browse the web. Since it can filter outgoing traffic as well as incoming, it can be used to defend a network not only from outside threats but from inside threats as well. A computer or group of computers that are on the private network can be put behind a bridging firewall to restrict access from them to certain parts of the network, in the case of contractors. For kiosks or public access computers, they can be restricted to block all traffic except for port 80 (web traffic) that is heading outside of the private LAN. Remember that it doesn't have to be just port 80, though, an administrator can choose to block or allow any TCP/IP traffic seen fit. Another good example along these same lines might be for businesses that offer wireless Internet access, as some coffee shops and bookstores are starting to do. A bridging firewall could fit in between the wireless devices and the wired network, to prevent them from getting to anything that might be on the wired network.

Building further on the point of this firewall being able to fit nicely pretty much anywhere on a network, it doesn't have to be used on it's own. As a matter of fact, it would work very well in conjunction with another firewall

installed at the outer perimeter of a private network. Because it can't do NAT and is extremely difficult attacked directly, it can be put directly outside of a firewall that is doing network address translation. Since a routing firewall doing NAT is always going to be exposed to attack, even though it can protect itself to an extent with it's own rules it's not a bad idea to have another form of protection there. Depending on how sensitive the network being protected is, or how likely a target it might be (a high profile company or government agency), it could be a very good idea to have some added defense to help protect against any attacks that might be directed at the routing firewall.

While setting up a firewall is definitely a step in the right direction towards network security, it is just one tool of many that can be deployed. There are many other tools available, such as vulnerability scanners, strong security policies, and encryption that also help to protect from break-ins and information theft of corruption. Another good tool is an intrusion detection system, or IDS. An IDS is used to tell if an intruder has gotten past the defenses that the administrators have set up. Intrusion detection systems can be set up on individual computers, to watch certain important files that might be changed if an attacker were able to get access, and it is very common to see this used on firewalls to alert the administrators to any possible compromise on the firewall itself. This leads us to firewall, and can another possible use for the bridging firewall. It can be put inside of another be used as an invisible network intrusion detection system. A network intrusion detection system, or NIDS, monitors the traffic on a network looking for signs that an attacker may have gained access.

"The most widely deployed Intrusion Detection worldwide", according to Sourcefire, is Snort.[10] If choosing to use Snort, it can be installed rather easily from the OpenBSD ports collection. While having an invisible network intrusion detection system would be a nice thing to have, we are back again at the chicken and egg problem touched upon earlier. The problem is this: The lack of communication is a two way street. Not only are all the other computers unable to talk to it, but it is unable to talk to any other computer. How is the NIDS going to alert an administrator of a possible intrusion if it has no way to communicate with any other computer on the network?

There are a few ways to solve this problem, all of which compromise to some extent the advantages of having an invisible device. The easiest, least expensive, and by far least secure option is to assign an IP address to the inside interface of the firewall. This option makes the firewall still completely invisible to anyone outside the network, but allows communications with any computer inside. Depending on the number of computers operating in the private LAN it could still be a needle in a haystack if an intruder were able to get access to the private network. Even then it should be well defended, but this is still quite a step up from the security provided not having an IP address at all. A slightly better option would be to use switches to create a separate network for the bridging firewalls. This separate network could be blocked off from all other network traffic, and could be used to have another computer in a more convenient location communicate with the firewall(s). Virtual LANs

(VLANs) as they are called were not meant to be set up for security purposes, though. It is possible for an attacker to attack the switch itself to get access to the previously inaccessible network.[11] The safest but probably most expensive option, is to set up a third network card in the bridging firewall. This card could be given an IP address, and connected to a separate physical network. This could be the best option for a network with multiple bridging firewalls set up. If every bridging firewall had a third network card set up with an IP address on this private network it would make it much easier to check logs and receive alerts from the NIDS. It's important to point out, however, that the private network that these machine share should not be connected to anything else. If this network is kept physically separate from any other network, then the advantages gained from having IP-less devices is only slightly lost. With this type of setup, anyone who could get physical access to one of the bridging firewalls might be able to attack any of them, but this still hinges on the attacker being able to get physical access to machines which should be locked down and having enough time to find and exploit a hole.

CONCLUSION

When making choices such as these, it's necessary to weigh the advantages of something not only against its disadvantages, but also against the advantages of other available options. One of the best things about a bridging firewall is that this choice does not have to be an exclusive one. Even in situations where another type of firewall is the best alternative, the bridging firewall could still be considered as an addition.

So, could a bridging firewall be a good option for you? Taking into account the relatively small investment it takes to establish one, the cost of possible data theft/corruption, and its ability to be put to use almost anywhere to reduce that possibility, it is at least an option that should warrant heavy consideration.

REFERENCES

[1] Holland, Nick et al. “OpenBSD 3.3 Installation Guide.” 16 June 2003. URL: http://www.openbsd.org/faq/faq4.html#install . (19 June 2003)

[2] Holland, Nick et al. “Old PF FAQ.” 5 May 2003. URL: http://www.openbsd.org/faq/faq6pf.html . (19 June 2003)

[3] Underwood, Nate. “HOWTO: Transparent Packet Filtering with OpenBSD.” July 2002. URL: http://ezine.daemonnews.org/200207/transpfobsd.html . (4 May 2003)

[4] Baker, F. “Requirements for IP Version 4 Routers.” RFC 1812. June 1995. URL: http://rfc-1812.rfc-list.org/rfc-1812-85.htm. (4 May 2003)

[5] Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Addison-Wesley, 2001. 97-98

[6] Blanchard, Eugene. “Introduction to the ISO – OSI Model.” Introduction to Networking. URL: https://secure.linuxports.com/howto/intro_to_networking/c4412.htm . (12 May 2003)

[7] Hogan, Doug and Hinton, Bryan. “OpenBSD bridge without IPs using IPF tutorial.” March 2001. URL: http://www.daemonnews.org/200103/ipf_bridge.html . (21 May 2003)

[8] Holland, Nick et al. “PF: Packet Filtering.” 15 June 2003. URL: http://www.openbsd.org/faq/pf/filter.html . (17 June 2003)

[9] TechTarget. “proxy – a whatis definition.” 10 February 2003. URL: http://whatis.techtarget.com/definition/0,,sid9_gci212840,00.html . (3 June 2003)

[10] Sourcefire. “Sourcefire – Network Security.” URL: http://www.sourcefire.com/aboutus/about_us.htm . (3 June 2003)

[11] von Leitner, Felix. “Switching and VLAN Security FAQ.” URL: http://www.fefe.de/switch/. (5 June 2003)

---

[1] Holland, Nick et al. “OpenBSD 3.3 Installation Guide.” 16 June 2003. URL: http://www.openbsd.org/faq/faq4.html#install . (19 June 2003)

[2] Holland, Nick et al. “Old PF FAQ.” 5 May 2003. URL: http://www.openbsd.org/faq/faq6pf.html . (19 June 2003)

[3] Underwood, Nate. “HOWTO: Transparent Packet Filtering with OpenBSD.” July 2002. URL: http://ezine.daemonnews.org/200207/transpfobsd.html . (4 May 2003)

[4] Baker, F. “Requirements for IP Version 4 Routers.” RFC 1812. June 1995. URL: http://rfc-1812.rfc-list.org/rfc-1812-85.htm. (4 May 2003)

[5] Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Addison-Wesley, 2001. 97-98

[6] Blanchard, Eugene. "Introduction to the ISO – OSI Model." Introduction to Networking. URL: https://secure.linuxports.com/howto/intro_to_networking/c4412.htm. (12 May 2003)

[7] Hogan, Doug and Hinton, Bryan. "OpenBSD bridge without IPs using IPF tutorial." March 2001. URL: http://www.daemonnews.org/200103/ipf_bridge.html. (21 May 2003)

[8] Holland, Nick et al. "PF: Packet Filterin g." 15 June 2003. URL: http://www.openbsd.org/faq/pf/filter.html. (17 June 2003)

[9] TechTarget. "prox y – a whatis definiti on." 10 February 2003. URL: http://whatis.techtarget.com/definition/0,,sid9_gci212840,00.html. (3 June 2003)

[10] Sourcefire. "Sourcefire – Network Security." URL: http://www.sourcefire.com/aboutus/about_us.htm. (3 June 2003)

[11] von Leitner, Felix. "Switchin g and VLAN Security FAQ." URL: http://www.fefe.de/switch/. (5 June 2003)