The Shrink-Wrapped VPN Node

Technical Report MS-CIS-00-21

Digital Systems Laboratory
Department of Computer and Information Science
University of Pennsylvania

Vassilis Prevelakis
vp@aegis.cis.upenn.edu

Angelos Keromytis
angelos@dsl.cis.upenn.edu

Abstract

The wide availability of public domain IPsec implementations allows the creation of VPNs based on low-cost platforms. However, setting up a VPN node involves a lot of work such as the creation of IPsec Security Associations and associated tunnels, including the necessary management of keys. Moreover, routing and firewall facilities must be provided to ensure the isolation of the members of the VPN from the public Internet. In this paper we present a drop-in VPN node that is compact, low-cost and requires little administration or maintenance. We discuss the features and advantages of our system. Next, we demonstrate how this system was used for the creation of a VPN linking networks within the University campus with others located in outside locations (e.g. other companies, home networks etc). Finally, we present our evaluation of the work performed and describe our future plans.

1. Introduction

The predominance of the Internet has resulted in unprecedented connectivity. Especially in developed countries, this means that, regardless of location, there is always a nearby connection to the Internet. In turn, this changes the economics behind the provisioning of closed networks, whether they are within a given company or organization, or linking different corporate entities. The cost of connecting to public networks decreases more rapidly than that of private networks (i.e., networks consisting of leased lines or circuits) thus making financially more tempting going over the Internet rather than using a private network. Moreover, as consumers get higher speed connections to the Internet (via cable modems, high speed DSL lines, etc.) there is more pressure to use these links to telecommute. In both cases the use of VPNs is obligatory to safeguard the participants from outside interference.

In the first case, cost is not the dominant factor, but in the second one, where we are dealing with VPNs consisting of large numbers of nodes, cost considerations play a significant role. In this paper we will examine ways whereby both small and large scale VPNs can be deployed cheaply and efficiently.

A typical VPN configuration is shown in figure 1 where a number of LANs are linked to the public Internet. The intention is that the hosts in each LAN may communicate with hosts in all the other LANs but communication with the rest of the Internet should be restricted. This is achieved by providing each network with a node that implements the security policy of the VPN. This usually means that the node maintains secure tunnels (i.e. encrypted) with one or more remote networks and also acts as a firewall restricting connections with hosts outside the VPN. In the next section we discuss the requirements for such VPN nodes. Next, we describe our drop-in VPN node that is a hardware/software combination constructed by integrating existing open-source components. We then present a prototype VPN and discuss how our system copes with the requirements we have set. Finally we present comments on the design

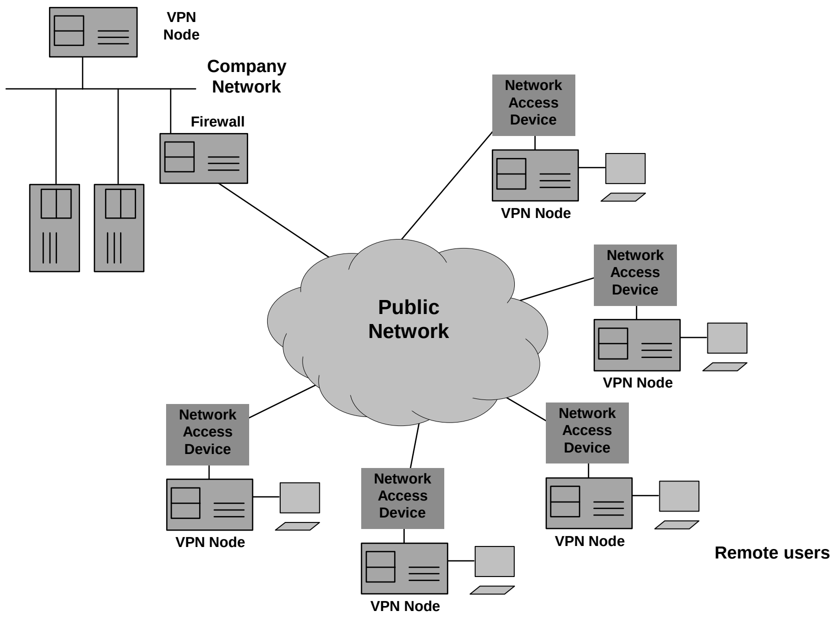

Figure 1. The VPN nodes are located between the network access device and the internal network

and implementation of our system and our plans for future enhancements.

2. Requirements

A major decision in the design of a VPN node is whether it should be embedded in the access device that links the LAN to the public Internet or whether it should a standalone device that sits between the network access device and the LAN. The multitude of access technologies that are available today and the low cost of the popular access devices (router, cable modem, wireless bridge etc.) makes it very difficult to justify the embedded approach. While the addition of yet another box has some obvious disadvantages it does keep down the complexity of the VPN node and also allows us to provision a single device that is independent of the access technology used. Moreover, given the low cost of Ethernet adapters, our approach does not impose a significant cost penalty.

The other requirements for the VPN node were as follows:

- Low cost, probably a single board computer (SBC) with two Ethernet ports.
- Minimal administrative overhead. This implies easy configuration and no administrator intervention after installation. Moreover, the bulk of the work for the construction of the software distribution for the network monitoring station should be devoted to integration of existing tools and packages, rather than the development of new code that would have to be maintained.
- Offer secure (encrypted) network connections with the other members of the VPN.
- Protect the VPN members from interference from the public Internet.
- Be resistant to tampering; in cases where there are indications that a station has been hacked, it must be easy to restore its original configuration.
- Offer a standard platform for the execution of common network management and monitoring tools. VPN users should not have access to the management information.
- Finally, regardless of the profile of the end user, the node must be able to be deployed with minimal overhead. The complexity of the creation of a VPN

station configuration should not be greater than the preparation of a company identification badge. In many cases they would probably be produced at the same time.

Existing commercial solutions do not offer the right mix of open standards and low price. In fact many VPN solutions have a per node pricing model that is based on the assumption that remote locations are company branches. Thus, they have pricing structures that deal with tens or hundreds of nodes. Scaling them to networks with thousands of nodes produces outrageous prices. Thus, we decided to investigate an open source solution. The advantage of this approach is that it offers enormous potential for customization coupled with a low cost per node since there are no software licensing costs.

3. The design of the VPN node

Creating a system in-house has many pitfalls, mainly related to the fact that the platform design, implementation and support all have hidden costs that must be brought out into the open and accounted for. Just because a piece of software is free does not mean that its deployment in a production environment is without cost.

A lot of attention has to be given to the integration, large scale production and maintenance of the nodes, in order that a usable system be achieved within the budget constraints of the project.

The prime considerations in the design of the VPN node has been simplicity and security. In this section we will elaborate on these two issues and examine their impact on the design of the operating environment. We will also present the major elements that constitute the VPN station and discuss the various design decisions we had to make.

3.1 Simplicity - Reliability

We attempted to keep the complexity of the platform as low as possible for the following reasons:

- A complex design is difficult to verify and control. This implies that maintaining the security posture of the platform after its original roll-out will be difficult.
- Network administrators come and go. A non-standard platform such as the VPN node will have to be easy to master, otherwise new staff will not be able to support it.
- The VPN node is intended for production use. The administrators must have confidence in the platform, otherwise on every instance of a problem they will suspect the platform and thus waste time investigating the platform instead of investigating the problem.

3.2 Operating System

From the very beginning, the design team wanted a platform that could accommodate tools for remote monitoring and management. The requirement that the station should operate in residential environments, without a monitor, keyboard or mouse effectively disqualified all Windows platforms. From the available UNIX or UNIX-like systems we eventually chose OpenBSD 2.7 for the following reasons:

- Built-in support for the transport layer security protocols (IPsec) that offer secure communication channels between stations. Since these channels are created by the networking code in the kernel, the encryption is transparent to applications. Thus, programs such as rlogin that have no encryption facilities can take advantage of the built in security offered by IPsec without any modifications to the application code.
- Like other free UNIX clones, a large number of programs such as tcpdump, snmpd, ssh, etc. are either supported in the base release or can be easily ported.
- Good security. The designers of OpenBSD have paid a lot of attention to the security profile of the system, creating a robust environment that is resistant to security related attacks. In fact, on the OpenBSD web site (*http://www.openbsd.org/goals.html*) it is claimed that OpenBSD passes Ballista's (now called Cybercop Scanner by Network Associates, *http://www.nai.com/products/security/cybercop_scanner*) tests with flying colors.

3.3 IPsec

IPsec is a suite of protocols [RFC1825] that provide encryption, authentication and integrity checking at the network layer. The VPN employs IPsec in tunnel mode with encryption (ESP) [RFC1827] (figure 2). Tunneling consists of encrypting and encapsulating a normal IP packet within a IPsec packet (see figure 3) . Since both the header and payload of the original packet are encrypted, the internal structure of the private network is concealed from intruders [Shah97].

The use of tunnel mode also allows us to use the VPN nodes as routers sending packets from the remote home LANs to the main corporate network [Scot98]. Under this scenario addresses from the internal corporate network may be allocated to workstations at the employees' homes.

Two sets of IPsec connections are maintained for each VPN node. One carries the VPN data while the other is used for the management of the VPN node itself. By using separate IPsec connections we ensure that users cannot access the management information or be in a position to contact other nodes through the VPN.

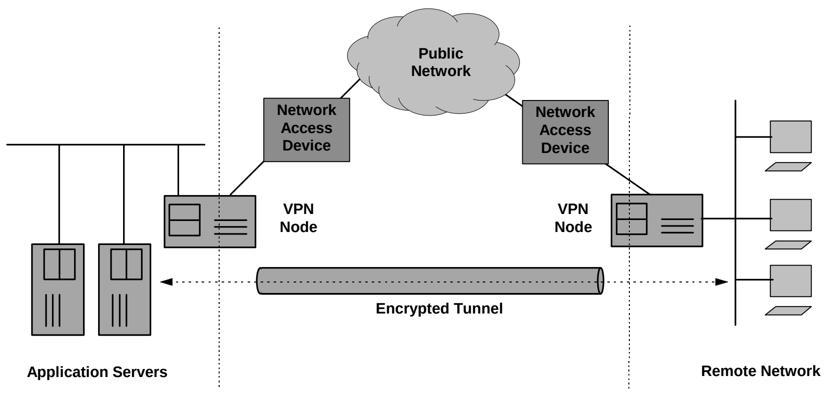

Figure 2. The IPSEC tunnel provides a secure connection between the two local area networks over the public Internet.

3.4 Key Management

Running IPsec with statically defined SAs as we did in [Prev99], is like running the Internet with static routing tables. The resulting VPN is extremely inflexible and keys are not changed as often as they should because of the effort and disruption to service. Moreover, since SAs contain source and destination IP addresses, they have to be changed each time the IP address of one of the endpoints changes. Users that connect to the Internet via dial-up connections or even permanently connected users that are assigned IP addresses via dhcp cannot use statically assigned SAs.

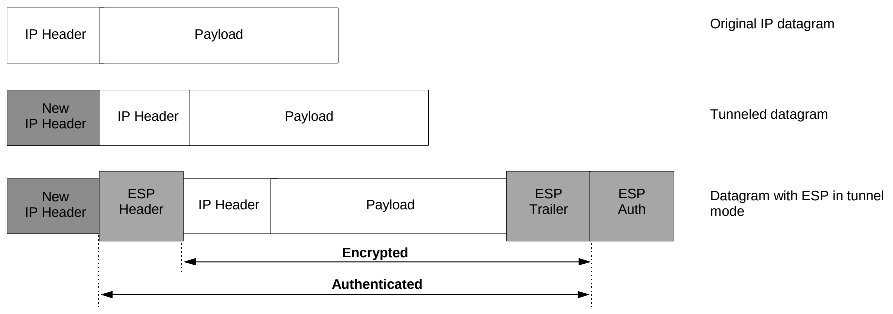

Figure 3. IPsec using ESP in tunnel mode

Workarounds to the above problems exist and are discussed in detail in [Prev99]. However, these solutions lack elegance and are not suitable for large scale VPNs.

Purchases via credit cards provide a good analogy to the problem of setting up flexible SAs. When purchasing an item, the customer presents the merchant a credit card. The merchant does not need to keep a record of all the people that have a VISA card in order to complete the transaction. Instead, the merchant contacts the credit card company and gets an authorization. In essence the credit card company vouches for the customer.

In the same way, one of the endpoints (A) of a VPN tunnel presents a certificate that is signed by a certification authority (CA) acceptable to the other side (B). There is no need for B to have previous knowledge of A since the certification authority vouches for the authenticity of (the certificate presented by) A.

There are two differences with the credit card example. The first is that there is no on-line communication with the CA during the negotiation. Endpoint B has the public key of the CA and can thus verify the certificate presented by A. The second difference is that A does not trust B and so B must also present a certificate to A. The second certificate must be signed by a CA acceptable to A. In out system all certificates are signed by the same CA but this need not be the case.

A serious issue with certificates is revocation, i.e. what happens if, for example, a VPN node is lost or stolen. Our system does not currently support certificate revocation but there are two mechanisms that can prevent compromised nodes from linking to the VPN. The first is that certificates have limited lifetimes and so, unless renewed, become worthless after a specified interval. The second is that, by changing the policy file, we can prevent nodes from accepting connections from blacklisted nodes.

3.5 Firewall

VPN nodes must be able to allow traffic from the interior network to flow through the VPN to the other internal networks, while at the same time they should allow only a very restricted set of incoming connections. On the other hand, connections from other VPN nodes must be accepted.

The VPN may be viewed as a transit network located between the end-user workstation and the internal network.

In the VPN node design we have used the packet filtering functionality of the OpenBSD kernel with a configuration that imposed three classes of restrictions:

- public Internet (“outside” interface)

  All packets are blocked except IPsec packets, since IPsec has its own security mechanisms, and ICMP echo and reply messages but excluding the other ICMP control messages.

- Transit packets flowing through the VPN (“inside” interface).

  While allowing the packets to be routed though the VPN we generally do not allow connections *to* the VPN nodes themselves. Exceptions to this rule include services such as dhcp that are required for the operation of the node. We also allow through certain types of ICMP packets for network trouble-shooting.

- Traffic between the VPN nodes (“inside” interface).

  This is mainly for management and node administration. Generally no restrictions are placed to this type of traffic.

Given that we are enforcing no access restrictions within the VPN, we were extremely concerned about allowing access to the VPN nodes from workstations located at the users premises. When considering security mechanisms there is always a need to strike a balance between security and convenience. Making life difficult for the end users would only mean that they would avoid using the VPN or find ways to disable or bypass various security mechanisms, thus compromising the security posture of the entire network. At the same time we did not wish to allow unsophisticated users access to the VPN nodes.

In the end we decided that users may access their local VPN node only from their own “inside” network. In this way only users suitably authorized would be able to access the configuration of their own VPN stations.

3.6 RAM-based system

In order to produce a simple and reliable system we decided to dispense with the hard disk. The reason behind this decision was twofold: reliability and support. Although disk drives tend to be reliable, they would have to operate continuously throughout the life of the VPN nodes. In a residential environment equipment tend to be subject to all kinds of abuse (knocked about, powered down without shutting down the system, relocated while in operation, etc.). Hard disks produce a fair amount of heat and noise and are also more prone to failure in these conditions.

The second and more important reason was related to the way that these machines were intended to be used. For our purposes, hard disks are already huge in terms of capacity and are getting bigger all the time. This free space can cause all kinds of trouble; for example, it may be tempting to fill it with data that should not be stored in the VPN node in the first place. This means that stations can no longer be redeployed easily because this information must be backed up, or processed. Secondly, if a station is compromised, the intruders will be able to use this space as a bridgehead, transferring and installing tools that will enable them to attack other network assets.

On the other hand, diskless machines bring with them a whole collection of problems and administrative headaches. They are also basically incompatible with our objective of using standalone machines with encrypted tunnels for all communications over the public Internet.

Instead, we use a RAM-based system where the software is loaded once during boot and then the system runs entirely on the system main memory (RAM). The boot medium may be floppy, CDROM, or a solid state disk (e.g. Compact Flash). In the following paragraphs we use the term floppy but the other media can be used just as well. In fact, in our prototype system, we are using Compact Flash as the boot medium.

In order to produce a RAM-based system, we adopted the techniques used by the PICOBSD project which is a collection of FreeBSD configurations that can be accommodated within a single boot floppy (*http://www.freebsd.org/~picobsd*). The PICOBSD project provides configurations for a dial-up router, dial-in router (ISP access server), general purpose router and firewall. The PICOBSD technique links the code of all the executables that we wish to be available at runtime in a single executable using the *cruchgen* utility [Silv98]. The single executable alters its behavior depending on the name under which it is run (`argv[0]`). By linking this executable to the names of the individual utilities we can create a fully functional /stand directory.

The aggregation of the system executables in a single file and the compression of the entire kernel allows a large number of facilities to be made available despite the small size of the boot medium. For example in the VPN node distribution we have decided to include the following commands:

Category	Commands
Shell Commands (Korn Shell)	`cat, chgrp, chmod, chown, cp, echo, kill, ln, ls, mkdir, more, pwd, rm, stty, telnet, test, w`
Administration	`date, dmesg, hostname, passwd, ps, reboot, update, vmstat`
System Configuration	`dev _mkdb, mknod, pwd_mkdb, swapctl, swapon, sysctl`
Daemons	`getty, inetd, init, login, snmpd, syslogd, telnetd, dhcpd`
Networking	`ifconfig, ipf, ipnat, ipsecadm, netstat, ping, route, traceroute, isakmpd, wicontrol, dhclient`
Filesystem	`mount, (cd9660, fdesc, ffs, kernfs, mfs, msdos, nfs, procfs), df, newfs, umount`

The root of the runtime file system, together with the executable and associated links, are placed in a ramdisk that is stored within the kernel binary. The kernel is then compressed (using *gzip*) and placed on a bootable floppy.

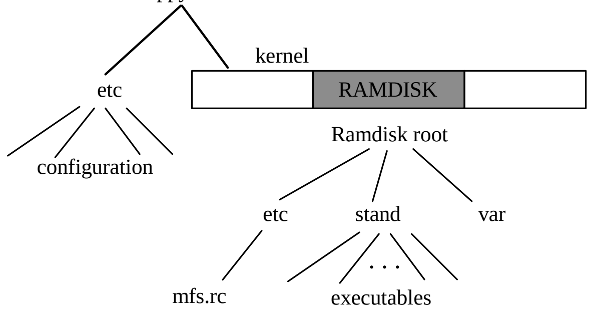

Figure 4. The organization of the floppy-based distribution

This floppy also contains the /etc directory of the running system in uncompressed form to allow easy configuration of the runtime parameters (Figure 4).

At boot time, the kernel is copied from the floppy disk to main memory, uncompressed and executed. The file system root is then located in the ramdisk. The floppy disk is mounted and the /etc directory copied to the ramdisk. At this point the floppy is unmounted and may be removed. The system is running entirely off the ramdisk and goes "multi-user" running the /etc/rc* scripts. Once the boot process is complete, user logins from the console or the network may occur. The floppy is usually write-protected so changes in the system configuration do not survive reboots. If, however, the floppy disk is not write protected, there exists a utility that can copy the contents of the ramdisk /etc directory to the floppy, thus making the running configuration, permanent.

This organization places the files that are unlikely to change between VPN nodes in the kernel where they are compressed, while leaving the configuration files in the /etc directory on the floppy. Thus, these files can be easily accessed and modified. Moreover, a single diskette image may be produced and the configuration of each station applied to it just before it is copied to the floppy.

4. Prototype network

The VPN nodes described in this paper have been used to create the network shown in Figure 5 linking research teams from various locations to the internal network of the University of Pennsylvania.

Even in academic environments where the security restrictions imposed by firewalls are relatively lax, there is a need for the provision of full connectivity between collaborating researchers. When the VPN is running, workstations in the three workgroups are able to communicate using the full suite of IP protocols (such as X11, NFS, telnet, rsh, rlogin).

As it can be seen in Figure 5, each of the three locations is linked to the other two via IPsec tunnels. This means that there is no need for additional routing tables since all the networks in the VPN are just one (virtual) hop from each other.

By write-protecting the distribution medium we ensure that its configuration cannot be altered. Of course the *running* configuration can be altered since all the files exist on the ramdisk (for example, we can create new accounts, add IPsec SAs etc.) but when the machine is rebooted, it will revert to the original configuration on the boot medium.

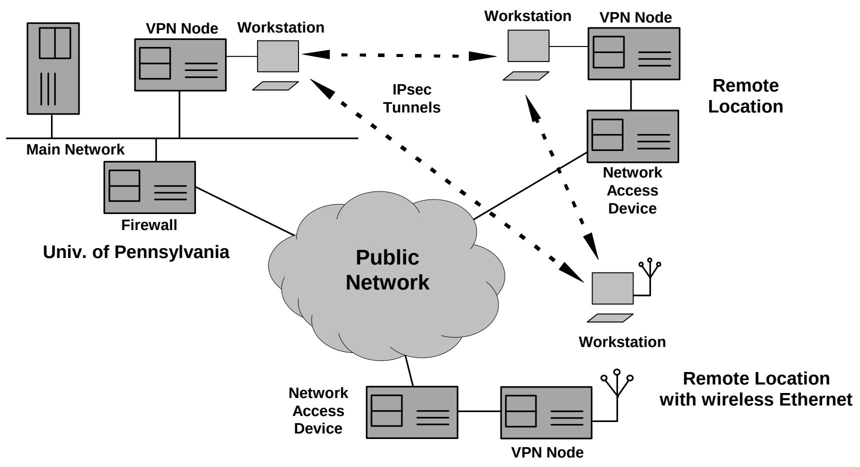

Figure 5. VPN nodes link the LANs of the three participating institutions.

Merging workgroups from different networks implies dealing with such issues as whether the guests should be allowed access to the rest of the internal network as if they were local users, or whether the workgroup networks should be located outside the firewall (maybe in a special DMZ). Another similar issue is whether local researchers should be able to link their usual workstations to the VPN or whether they should be required to use dedicated workstations.

In situations where the VPN is linking researchers from, e.g., different companies, these issues would need to be seriously considered and the effect on the corporate security policy investigated before establishing the VPN [Ches94]. In our case convenience was considered more important than security and the VPN workstations were placed in the internal network. However, since the routes via the IPsec tunnels were not advertised internally, only the workstations in the VPN could communicate via the IPsec tunnels.

power requirements and, most importantly, the noise from the power supply fan, make such machines totally unsuitable for deployment in the field.

Single board computers (SBCs) allow the creation of small-factor convection-cooled systems. These designs are mostly compatible with the PC motherboards and cards so that there is no need for software porting. Moreover, solid state storage in the form of Flash RAM may be added thus improving the reliability of the system.

After looking at a number of products we chose the NetCARD system by Cell Computing (see figure 6). This single board computer is about 14cm by 10cm and combines on board Ethernet interface, Compact Flash and 2 PC-CARD slots along with the usual PC-style interfaces (floppy, IDE disk, etc.). We used this

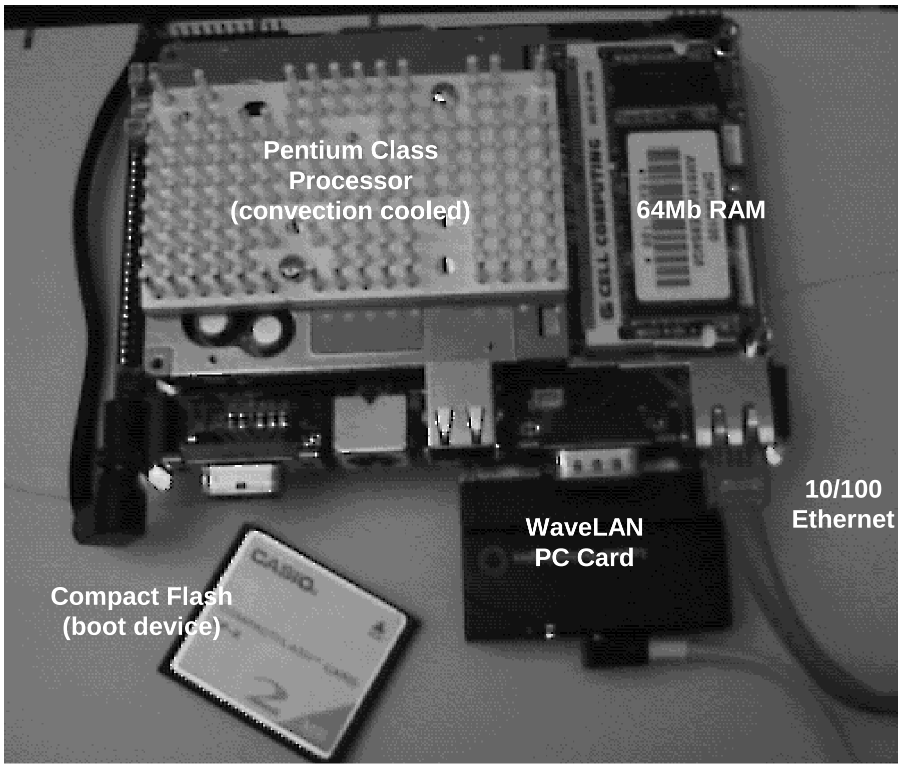
Pentium Class Processor (convection cooled)
64Mb RAM
10/100 Ethernet
WaveLAN PC Card
Compact Flash (boot device)

Figure 6. The VPN node hardware.

4.1 Hardware Platform

The VPN software runs on standard PC hardware. While most of the development was carried out on decommissioned PCs, the size,

system both with a floppy boot medium and with a Compact Flash. The on-board Ethernet interface was used to connect the VPN node to the network access device, while an Ethernet PC-CARD provided the inside-network connection.

We used the Lucent Orinoco PC-CARD that provides wireless Ethernet, so that there was no need for fixed wiring or hubs. We feel that deployment at home would make the use of such wireless devices a must.

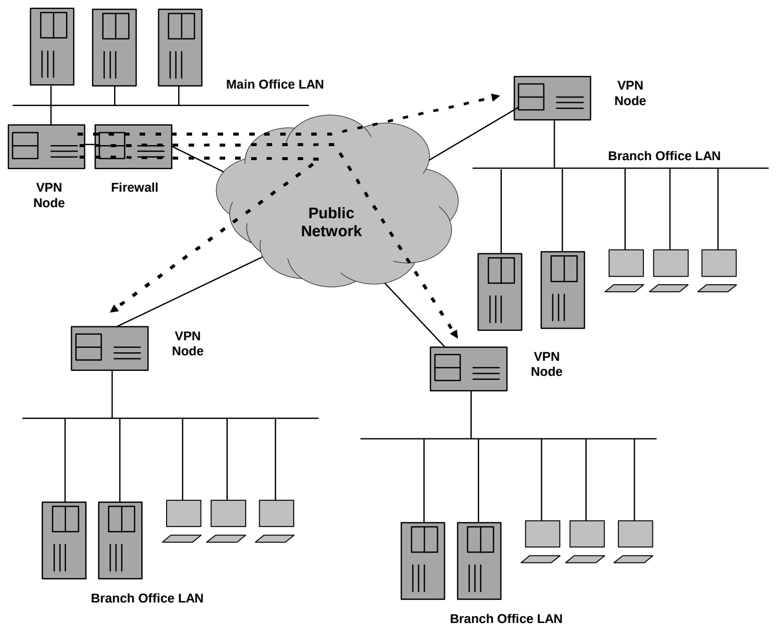

Figure 7. The VPN nodes link remote locations to the central office over a public network.

4.2 Operation

As soon as power is applied and the power-on tests are complete the PC BIOS loads the system from the boot medium and hands control over to the OpenBSD kernel. In order for the outside interface to be configured, the VPN node must find out the IP address provided by the ISP. If the IP address is always the same, then it can be included in the static configuration that is read off the boot medium. Otherwise the system uses `dhclient` to configure its interface. The inside Ethernet interface uses a pre-assigned address from the private internet range (RFC1918). The system also runs `dhcpd` on the inside interface so that workstations on the private network can be auto-configured. The system then runs the `isakmpd` daemon that creates the Security Associations and sets up the IPSEC tunnels. The packet filtering software ensures that the VPN is isolated from the outside world. The node may be powered down without the need for a shutdown procedure (e.g. `sync`).

5. Conclusions - Future Plans

The work presented in this paper is a continuation of the work described in [Prev99]. In the previous project a number of VPN stations were deployed within the University of Piraeus as part of a network of monitoring stations. The purpose of these Secure Network Stations (SNS) was to allow the creation of a secure network that allows administrators to manage and troubleshoot network elements such as routers, hubs, and switches deployed throughout the University campus. The SNS system has been in operation for more than a year.

The SNS nodes have different configurations from the VPN nodes discussed in this paper because they serve different roles. For example, SNS nodes need to forward SNMP traffic from the network elements and allow connections from inside the secure network to reach network elements located outside the SNS perimeter. Moreover, the system uses static SAs with manual keying which necessitates the production and distribution of updated configurations on a regular basis.

Nevertheless, the experience gained from their use helped in refining the requirements for the systems described in this paper. One notable decision that was directly influenced by the previous design was to have a fully connected mesh of IPsec tunnels linking every node to all the others. Most VPN solutions tend to link a central office with a number of remote locations with the IPsec tunnels arranged in a star (see Figure 7).

The many-to-many links allow the VPN to be resilient to failures of individual nodes and in the case where there is significant traffic between the remote nodes there is better utilization of the VPN resources as all packets go through at most one IPsec tunnel to their destination.

Another system that offers similar functionality to the one we have presented here is the Moat from the AT&T Labs [Denk99]. Like our system, the Moat also utilizes small single board computers running a lightweight version of Linux and create VPNs allowing AT&T research personnel to telecommute. The Moat follows the one-to-many VPN layout probably because it is not envisaged that there will be significant traffic between employees working at home. Remote stations with floating IP addresses (as is the case of most dial-up Internet connections) are treated by dynamically rewriting the IPsec configuration files. This requires that a central cite is always operational so that the VPN nodes can get the information they need to create their configuration files. In our system, the use of certificates, allows any two stations to negotiate SAs and create IPSEC tunnels. Additionally, the use of built-in facilities such as the `isakmpd` daemon make the system easier to maintain and port across Operating System releases.

In the following paragraphs we will discuss other enhancements that we plan to integrate in the near future.

5.1 VPN and non-VPN PCs on the home network

Internet connections at home are seldom used by only one person or for only one task (e.g. work). By placing the VPN-node between the home network and the ISP connection we are effectively forcing everybody to go though the company network. This is not entirely without merit because it means that the home computers are shielded behind the company firewall. However, there may be cases where we wish to have access to sites or services that are blocked by the company firewall. In such cases it would be a good idea to be able to allow only certain PCs or network segments to belong to the VPN, while the rest would work as if they were directly connected to the ISP.

There are three possibilities that we are currently investigating. The first is to plug the non-VPN PCs directly to the ISP feed. This allows us to have the non-VPN PCs behave as if the VPN did not exist. However, in this case the non-VPN PCs are exposed to attacks emanating from the public Internet.

Moreover, if the ISP allows the user to have more then one IP address per connection this setup can be straightforward. If, however, we have only one address, then we need some piece of equipment to do network address translation (NAT). Since we already have a powerful router as part of our VPN-node we may as well use it for this task as well. To be able to perform NAT we would need another network interface. This may be either in the form a separate Ethernet card or as a logical interface on the outside Ethernet card. The extra card solution allows us to do some packet filtering as well, so it offers the possibility of increased security.

5.2 Routing versus Bridging

Currently VPNs are created by joining separate networks via tunnels. Packets destined for a non-local network in the VPN are routed through one or more IPsec tunnels. While this may seen adequate for most practical purposes, the existence of the tunnels is visible, since they exist in the IP layer. This implies that routing tables must be employed (and hence software like `routed` or `gated` to manage those tables) to determine how the packets will be shuttled from one location to the next. However, if the tunnels are placed below the IP layer, then we can create a *virtual* LAN linking all the stations in the VPN. In this case all the workstations will belong to same IP network and behave as if they were connected to same Ethernet segment.

This behavior can be accomplished by configuring the VPN nodes as bridges rather than routers. The integration of IPSEC with bridging is already present in OpenBSD [Kero00] and we are currently working in the integration of these mechanisms in the VPN-node framework.

References

[Ches94] Cheswick, William and Steven Bellovin, "Firewalls & Internet Security, Repelling the Wily Hacker," Addison-Wesley Professional Computing Series, 1994.

[Denk99] Denker, John S., Steven M. Bellovin, Hugh Daniel, Nancy L. Mintz, Tom Killian and Mark A. Plotnick, "Moat: A Virtual Private Network Appliance and Services Platform," LISA'99: 13th Systems Administration Conference, Washington, November 1999.

[Kero00] Keromytis, Angelos D, Jason L. Wright, "Transparent Network Security Policy Enforcement," USENIX Annual 2000 Technical Conference - Freenix Refereed Track, San Diego, California, June 18-23, 2000.

[Prev99] Prevelakis, Vassilis "A Secure Station for Network Monitoring and Control," The 8th USENIX Security Symposium, Washington, D.C., USA, August 1999.

[RFC1825] Atkinson, R. "Security Architecture for the Internet Protocol," Internet Engineering Task Force, August 1995.

[RFC1827] Atkinson, R. "IP Encapsulating Security Payload (ESP)," Internet Engineering Task Force, August 1995.

[Scot98] Scott, Charlie, Paul Wolfe and Mike Erwin, "Virtual Private Networks," O'Reilly & Associates, Inc. 1998.

[Shah97] Shah Deval and Helen Holzbaur, "*Virtual Private Networks: Security With an Uncommon Touch*," Data Communications, Sept. 97,

[Silv98] Silva James da, "Cruchgen," OpenBSD User Manual, 1998.