Drop-in Security for Distributed and Portable Computing Elements

Vassilis Prevelakis
vp@drexel.edu
Drexel University

Angelos Keromytis
angelos@cs.columbia.edu
Columbia University

Abstract

The widespread use of mobile computing and telecommuting has increased the need for effective protection of computing platforms. Traditional schemes that involve strengthening the security of individual systems, or the use of firewalls at network entry points have difficulty accommodating the special requirements of remote and mobile users. We propose the use of a special purpose drop-in firewall/VPN gateway called *Sieve*, that can be inserted between the mobile workstation and the network to provide individualized security services for that particular station. *Sieve* is meant to be used like an external modem: the user only needs to plug it in. Its existence is transparent to the user, requiring no modification to the workstation configuration. To function in this role, *Sieve* has been designed to be compact, low-cost, requiring little administration or maintenance. In this paper, we discuss the features and advantages of our system. We demonstrate how *Sieve* was used in various application areas (home, university environment, etc.) and describe our future plans.

Keywords: VPN, Firewall, IPsec, embedded systems, OpenBSD

1. Introduction

The recent advances in networking have created a situation where computers are practically always connected to the Internet. Cable modems and DSL lines for the SOHO environment and wireless networking for laptop and handheld computers ensure that the resources of the Internet are always accessible. Against the convenience of "always on" connections we must balance increased exposure to attacks.

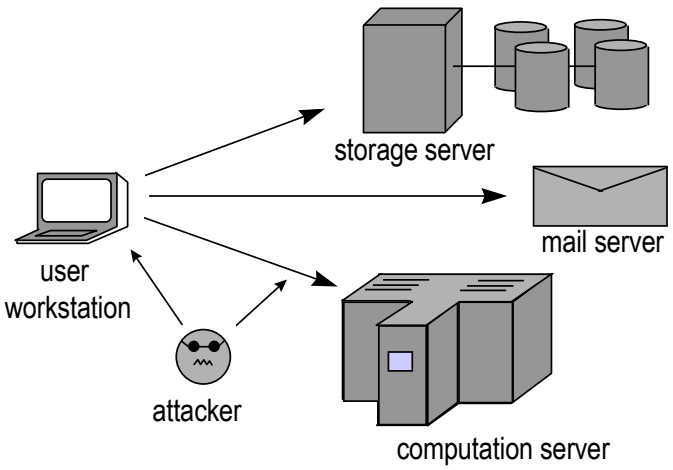

Figure 1: By accessing network resources, the user workstation runs the risk of having its communications intercepted, or being attacked by malicious third parties.

This work was supported by DARPA and NSF under Contracts F39502-99-1-0512-MOD P0001 and ANI-0133537.

Traditional approaches for countering threats involve the use of firewalls (Cheswick and Bellovin 1994). Unfortunately firewalls are better suited to organizations that can afford to pay for the configuration and maintenance of such systems. Moreover, firewalls provide a hard shell protecting the soft core of the internal networks. This architecture does not protect against internal threats, nor does it protect mobile users, telecommuters, or users of wireless connections (see Figure 1).

Ioannidis et al (Ioannidis, Keromytis, *et al*, 2000) propose the use of a "distributed firewall," i.e. the integration of firewall functionality in every machine in the network. However, this approach requires that all computers in the network run appropriately modified operating systems and applications which, for the time being, is not feasible on a large scale basis. On the other hand, existing systems such as Windows NT (and its derivatives) and most Unix and Unix-like systems already provide security features that can be used to implement firewall functionality on every machine. The difficulty of securing general purpose operating systems has impeded the widespread use of this approach. Moreover, it is difficult to ensure that a secured system remains secure after the user has had the opportunity to install software and perform reconfigurations and upgrades.

Recognizing the futility of attempting to secure the user machines themselves, we propose the use of a portable "shrink-wrapped" firewall (which we have called *Sieve*). This is a separate machine running an embedded system that includes firewall capabilities and is normally placed between the general purpose computer and the network (Figure 2). The problem of securing the firewall becomes much simpler as the platform is a special-purpose one with a highly controlled architecture.

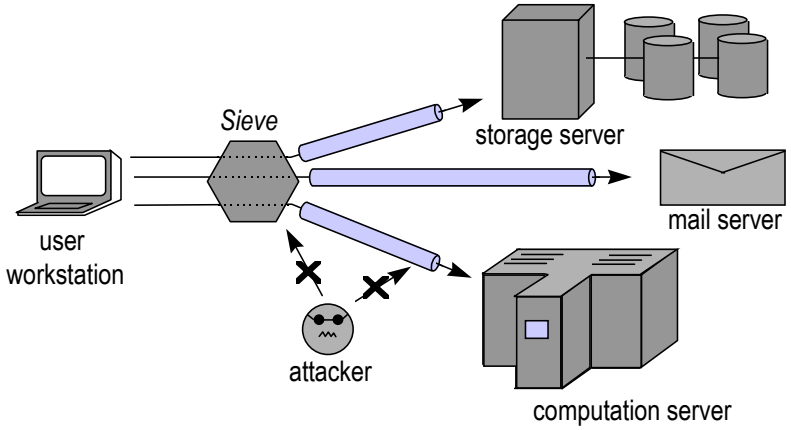

Figure 2: *Sieve* provides firewall services and creates secure links to other servers in the network, establishing a secure overlay network that is inaccessible by third parties.

To avoid the need for labor intensive reconfigurations and to provide flexibility, we allow the security policy of this embedded platform to be downloaded from the network.

For a "shrink-wrapped" firewall to be effective, the following prerequisites must be satisfied:

- Low cost in terms of hardware, software, configuration and maintenance.
- While it may restrict some services, it must be totally transparent to authorized services.
- Offer secure connections to servers and other network assets, thus protecting the communications between the protected system and other resources.
- Be flexible: it should be able to accommodate various security policies and different types of network attachments (serial, Ethernet, wireless).
- Be resistant to tampering. Furthermore, in cases where there are indications that a station has been compromised, it must be easy to restore its original configuration.
- To simplify centralized management and troubleshooting, it must offer a standard platform for the execution of common network management and monitoring tools. Workstation users should not have access to the management information.
- Finally, regardless of the profile of the end user, the "shrink-wrapped" firewall must be able to be deployed with minimal overhead.

Existing commercial solutions do not offer the right mix of open standards and low price. In fact many 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 net-

works 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. Another benefit of using Open Source software is that, unlike proprietary solutions, the code can be freely audited thus providing security through openness.

3. System Architecture

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 project's budget constraints.

The prime considerations in the design of *Sieve* have 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 components of the *Sieve* platform and discuss the various design decisions.

3.1 Simplicity - Reliability

There are several good reasons to maintain low platform complexity:

- 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.
- A non-standard platform such as *Sieve* will have to be easy to master, otherwise new staff will not be able to support it.
- *Sieve* is intended for production use, thus the administrators must have confidence in the platform.

3.2 Operating System

From the very beginning, we 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 2.9 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(1)` 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 are available through the *ports* system.
- Good security. The designers of OpenBSD have paid a lot of attention to the security profile of the system, creating a robust environment.

3.3 IPsec

IPsec is a suite of protocols (Akinson, 1995) that provide encryption, authentication and integrity checking at the network layer. *Sieve* employs IPsec in tunnel mode with encryption (ESP) (Figure 3). Tunneling consists of encrypting and encapsulating a normal IP packet within an IPsec packet. Since both the header and payload of the original packet are encrypted, the internal structure of the private network is concealed from intruders (Shah and Holzbaur, 1997).

The use of tunnel mode also allows us to use the *Sieve* nodes as routers sending packets from the remote home LANs to the main corporate network (Scott, Wolfe, *et al*, 1998). Under this scenario, addresses from the internal corporate network may be allocated to workstations at the employees' homes.

Another configuration (layer 2 VPN) utilizes bridging (Keromytis and Wright, 2000) rather than routing so that the remote node appears to be directly connected to the corporate network. This approach also allows the use of non-IP protocols such as LAN-Manager and Novel IPX/SPX.

Two sets of IPsec connections are maintained for each *Sieve* node. One carries the VPN data while the other is used for the management of the *Sieve* 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.

3.4 Key Management

Using IPsec with statically-defined Security Associations (SAs)[1] as we did in (Prevelakis, 1999), is equivalent to running the Internet with static routing tables. The resulting VPN is inflexible and keys are not changed as often as prudent cryptographic practice suggests, 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. Workarounds to these problems exist and are discussed in detail in (Prevelakis, 1999). 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 a credit card to the merchant. The merchant does not need to

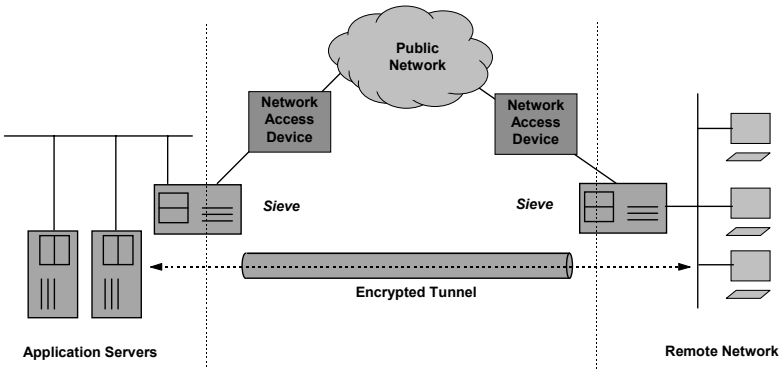

Figure 3: The IPsec tunnel provides a secure connection between the two local area networks over the public Internet.

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 receives 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 from 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 any 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 our system all certificates are signed by the same CA, but this need not always be the case.

A serious issue with certificates is revocation, i.e., what happens if, for example, a *Sieve* node is lost or stolen. There are two mechanisms that can prevent compromised nodes from linking to the VPN: (1) certificates have limited lifetimes and so, unless renewed, become worthless after a specified interval, (2) by changing the policy file we can prevent nodes from accepting connections from blacklisted nodes.

---

[1] Security Association is the set of parameters for one-way communication between two nodes (cryptographic keys, choice of algorithm, etc.).

3.5 Firewall

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

- Public Internet.
  This refers to packets coming in from the interface that is connected to the public network. These packets are generally blocked except IPsec packets, since IPsec has its own security mechanisms. Moreover, we also allow ICMP echo and reply messages for network troubleshooting, but we block other ICMP messages.
- Transit packets flowing through the VPN.
  Packets received from the interface connected to the local (protected) network) and destined for the remote end of the VPN connection fall within this category of restrictions. While allowing the packets to be routed through the VPN, we generally do not allow connections to the *Sieve* nodes themselves. Exceptions to this rule include services such as `dhcp` that are required for the operation of the node. We also allow certain types of ICMP packets for network troubleshooting.
- Traffic between the VPN nodes.
  In this category we have packets that are exchanged between the VPN nodes themselves. This kind of communication 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 *Sieve* from the user workstation. 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 *Sieve* nodes.

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

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 *Sieve* nodes. In both home and mobile environments 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 is related to the way that these machines are 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 *Sieve* 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 diskette, CDROM, or a solid state disk (e.g., Compact Flash). In our prototype system, we are using Compact Flash as the boot medium, so in the following paragraphs we use the term CF.

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.[1] 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 *crunchgen* utility (da Silva, 1998). 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 /bin directory where all the system commands are accessible as apparently distinct files.

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

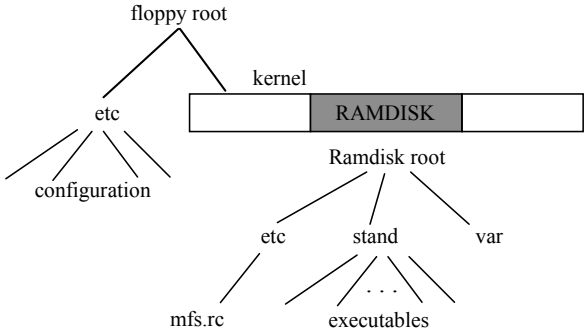

Figure 4. The organization of the *Sieve* distribution

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

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

4. Prototype

In this section, we describe the *Sieve* prototype and three examples of its use: office desktop, wireless, and home gateway.

4.1 Office Desktop

Contrary to popular belief, internal networks in most organizations are not safe. Although protected by firewalls, these networks are vulnerable to internal attacks (e.g., by coworkers, worms and viruses from other infected machines, etc.). Our approach is to install *Sieve* stations between each sensitive desktop and the internal network. As it has been noted in (Prevelakis, 1999), the ability to manage these stations via a centralized Network Management System creates a safe and predictable platform from which user network problems can be diagnosed remotely.

We use the *Sieve* in bridge mode so that the desktop appears to be directly connected to the local area network.

4.2 Wireless network

Wireless networks, even with encryption activated (Stubblefield, Ioannidis, *et al*, 2002) are particularly vulnerable to intrusion attacks. A popular hacker pastime is to cruise around town with a laptop trying to connect to wireless networks. Wireless networks should be considered unsafe and thus always linked to the internal network via a firewall.

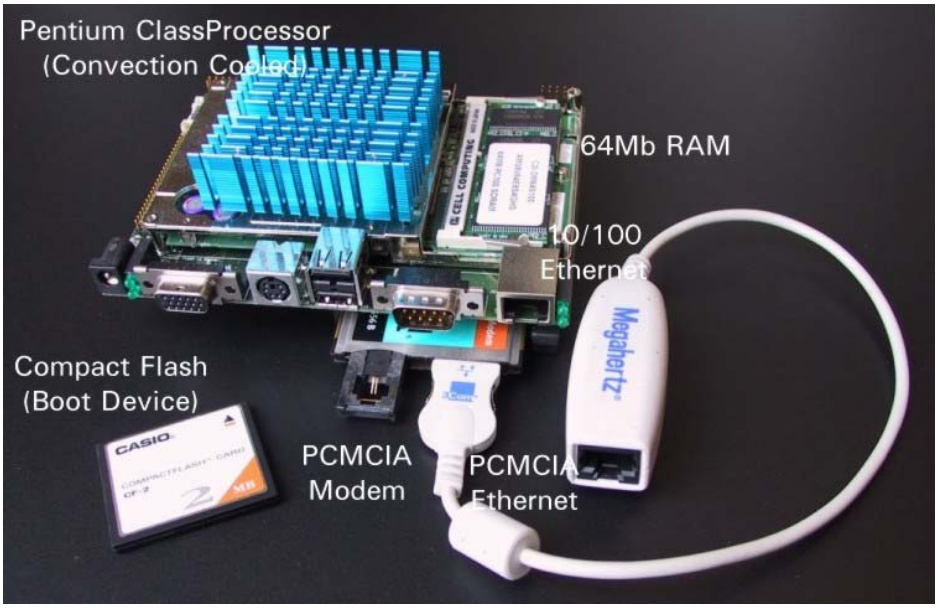

Figure 5. Wireless configuration.

In (Boscia and Shaw, 2000) the authors describe the difficulties in managing a wireless network (handing out addresses, opening connections through the firewall, etc.). They also include a number of solutions, but they admit that these solutions have weaknesses and, thus, they use them to provide access only to non-critical hosts in their network.

Normally the system is used in bridging mode, although there were cases where the system administrators want to know which machines are on the wireless segment. The main difference with the desktop configuration is that in the wireless configuration all communication is carried over secure links. We use IPsec in tunnel mode (Figure 3) to link the workstation to the internal network. In this case, the public network of Figure 3 is the wireless network.

4.3 Home Network Gateway

Internet connections at home are seldom used by only one person or for only one task (e.g. work). By placing a *Sieve* between the home network and the ISP connection, we have the option of forcing everybody to go through 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, the *Sieve* can be configured to allow only certain PCs or network segments to be part of the VPN, while the rest work as if they were directly connected to the ISP. In this scenario, the *Sieve* can also perform the task of a NAT router, allowing the user to share a single ISP-provided IP address among multiple workstations.

4.4 Hardware Platform

The VPN software runs on standard PC hardware. While most of the development was carried out on decommissioned PCs, the size, 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. Compared to ordinary hard drives, this offers improved reliability.

After examining a number of products we chose the NetCARD system (see Figure 5). 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 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. A wireless Ethernet card may be attached to the system, eliminating the need for fixed wiring.

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 `dhcp` 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 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 (Prevelakis, 1999). In the previous project, a number of VPN stations were deployed within the University of Piraeus, in Greece, 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 three years.

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 IPsec configuration, 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 6).

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 (Denker, Bellovin, *et al*, 1999). 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.

As part of our future plans, we intend to improve the automatic configuration of our system. The goal is a system that explores the network it is connected to (discovering default routers, dhcp servers and so on) and configures itself accordingly.

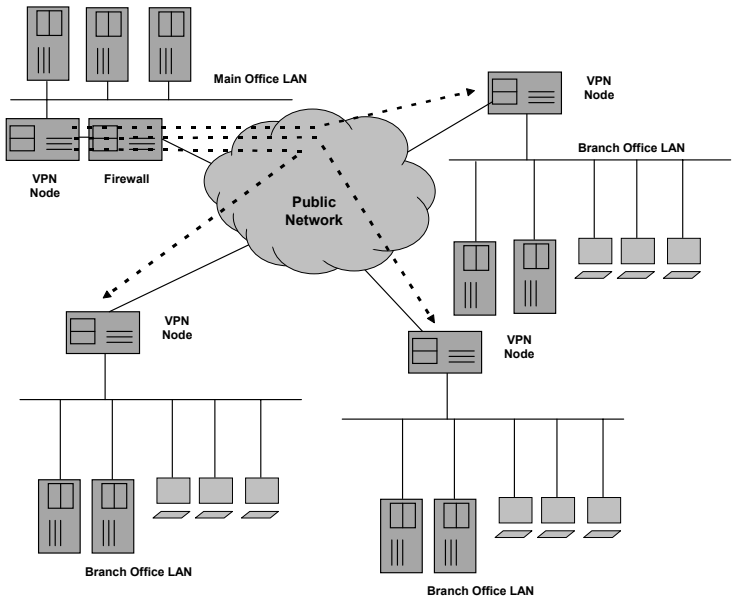

Figure 6. The VPN nodes link remote locations to the central office over a public network.

References

Boscia, N. K. and Shaw D.G. (2000) Wireless Firewall Gateway White Paper, NASA Advanced Supercomputing Division, Moffett Field, CA 94035

Cheswick, W. and Bellovin, S.M. (1994) Firewalls and Internet Security, Repelling the Wily Hacker, Addison-Wesley Professional Computing Series.

Denker, J.S., Bellovin, S.M., Daniel, H., Mintz, N.L., Killian, T. and Plotnick, M.A. (1999) "Moat: A Virtual Private Network Appliance and Services Platform," LISA'99: 13th Systems Administration Conference, Washington.

Ioannidis S., Keromytis, A.D., Bellovin, S.M., and Smith, J.M. (2000) "Implementing a Distributed Firewall," Proceedings of Computer and Communications Security (CCS), pp. 190-199.

Keromytis, A.D, Wright J.L. (2000) "Transparent Network Security Policy Enforcement," USENIX Annual 2000 Technical Conference - Freenix Refereed Track, San Diego, California.

Prevelakis, V. (1999) “A Secure Station for Network Monitoring and Control,” The 8th USENIX Security Symposium, Washington, D.C., USA.

Atkinson, R. (1995) “RFC-1825: Security Architecture for the Internet Protocol,” Internet Engineering Task Force.

Scott, C., Wolfe, P. and Erwin M. (1998) Virtual Private Networks, O’Reilly & Associates, Inc.

Shah D. and Holzbaur, H. (1997) “Virtual Private Networks: Security With an Uncommon Touch,” Data Communications.

da Silva J. (1998) “Cruchgen,” OpenBSD User Manual.

Stubblefield, A., Ioannidis, J. and Rubin, A.D. (2002) “Using the Fluhrer, Mantin, and Shamir Attack to Break WEP,” Proceedings of the 2002 Network and Distributed System Security Symposium, San Diego, California.

[1] http://www.freebsd.org/ ~picobsd

AUTHOR BIOS

Vassilis Prevelakis, Computer Science Department, Drexel University, Philadelphia.

Vassilis Prevelakis is an assistant professor in the Computer Science department, at Drexel University in Philadelphia, USA. Dr. Prevelakis received his Ph.D. from the University of Geneva in 1996. He has been actively involved in the area of secure systems and networks both as a researcher and as a consultant for the IT industry. He was a recipient of the NSF/CAREER award in the year 2001. His recent research interests include Automation and Security Applications for Home and Industrial Automation Networks, Embedded Systems, and the Design and Implementation of Robust Software.

Angelos Keromytis, Computer Science Department, Columbia University, NY.

Angelos D. Keromytis is an assistant professor in the Computer Science department, at Columbia University. He has been involved in the development of the IP Security standards since 1995 and has been the author and co-author of a number of implementations. His research interests include Trust Management (he is one of the designers of the KeyNote system), cryptographic protocol design, security management mechanisms for large networks, and distributed firewalls. Currently, he is working on protecting end services from distributed denial of service through use of overlay services and trust management techniques for access control. He is a Security Advisor at the Internet Engineering Task Force (IETF).