The following paper was originally published in the
Proceedings of the USENIX Symposium on Internet Technologies and Systems
Monterey, California, December 1997

Secure Public Internet Access Handler (SPINACH)

Elliot Poger, Mary G. Baker
Computer Science Department, Stanford University

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org/

Secure Public Internet Access Handler (SPINACH)

Elliot Poger, Mary G. Baker
Computer Science Department, Stanford University
{elliot,mgbaker}@mosquitonet.stanford.edu

Abstract: *This paper describes a system that controls access to computer networks through publicly accessible LANs, enabling network administrators to authorize users either on a permanent or occasional basis. The system has been designed with minimal assumptions about the software and hardware required of users, and requires very little specialized equipment within the network infrastructure. We enumerate the requirements for such a system, describe the design and implementation of the system, and note tradeoffs between security and efficiency.*

1. Motivation

In early 1996, Stanford University completed a new building to house its Computer Science Department. The new building includes Ethernet ports in every office, as well as in various public spaces: meeting rooms, lobbies, and lounges. Unfortunately, 18 months after the building opened, concerns about unauthorized users tapping into the department network have prevented the activation of network connections in publicly accessible areas ("public ports"). Similar problems plague many other buildings, especially on college campuses, where the desire for mobile connectivity is high but physical security is lax. Even though building designers had the foresight to include network connections in many parts of these buildings, political and security considerations have led to a frustrating waste of potential network connectivity. Those who desire network connectivity in public parts of the building are forced to use wireless network connections, which are often slow and expensive.

There are several reasons Stanford University, and the Computer Science Department in particular, do not want to allow unknown users access to the building network. Most importantly, we do not want to allow rogue users to attack other computers connected to the building network in offices and labs. Although hackers can already attack department computers over the Internet, we do not want to make these attacks, as well as eavesdropping on network traffic, any easier by allowing them access within our network. Also, some network services outside our department use the source IP address of transmissions to grant access. For example, some Internet services have been licensed for use at Stanford University and are made available to any host with a Stanford IP address, and we are obligated to prevent abuse of these licenses. In general, we want to minimize the chances that someone will misuse the Internet from a Stanford IP address, and if this misuse does occur, we want to identify the perpetrator so that we can hold him accountable. Perhaps less of a concern is that of bandwidth—we don't want to allow unauthorized users to degrade everyone else's service in the building by using network bandwidth to which they are not entitled. Since physical security in the building is minimal, as it is in many universities, libraries, and public institutions, we need a mechanism for restricting access through public network ports if these ports are to be activated.

Once we have an access control mechanism in place, we can allow specifically authorized users to connect to the high-bandwidth wired network in the building from public ports without compromising network security. To provide this access control, we have constructed the *Secure Public Internet Access Handler* (SPINACH). In SPINACH, a self-configuring router controls per-user access from a public subnet to a private one, using Kerberos or a similar mechanism to authenticate users and provide an audit path before users are granted access. With the exception of one custom software component on the router, SPINACH uses only standard protocols and software and requires only minimal software (telnet or web clients) on users' machines.

The SPINACH system establishes a "prisonwall," controlling the flow of packets between those hosts connected to public ports and the rest of the building network. As opposed to a firewall, which protects machines *inside* a particular network from malicious users *outside* the network [2][4], the prisonwall protects machines *outside* one portion of a network by refusing to forward packets that come from unauthorized hosts within. As users within the prisonwall authenticate themselves and thus activate network access for their hosts, SPINACH maintains an audit trail so that the

users can be held accountable for traffic they generate on the network.

SPINACH has been designed with minimal assumptions regarding the network hardware available as well as the software installed on users' machines, so that it can be installed in a wide variety of institutions and require little ongoing oversight from network administrators. As such, it does not provide as high a level of security as some access control systems; however, it provides a useful level of security without requiring expensive network equipment or custom client software, and thus may be the most appropriate method of access control for some networks.

In this paper, we describe the design and implementation of the SPINACH system. Section 2 outlines the system requirements and policies. In Section 3, we describe the interfaces through which network users and administrators interact with SPINACH. Section 4 discloses the details of how we implement these policies and interfaces. The remainder of the paper describes the security tradeoffs in SPINACH, other systems with aims similar to ours, some possible future improvements to SPINACH, and conclusions we have drawn through this research.

2. System Requirements, Policies, and Definitions

The SPINACH system has two major functions: it controls the passage of network communications between public ports and the rest of the building network, and it provides a mechanism for unknown users to prove themselves as authorized so that they can have full network access. Both functions are implemented on the same network host, the *SPINACH router.* This section describes the requirements that the SPINACH router must fulfill, and the facilities that must be present within the network infrastructure and on hosts connected to public ports in order to implement both functions. SPINACH has been designed to require no special software on computers that users connect to public ports, and to require as little as possible of the network infrastructure, so that it can be deployed in any network installation with minimal expenditure of time and money.

2.1 Network Arrangement

The SPINACH system consists of a collection of public network ports on one or more LANs. These LANs are connected to the surrounding network infrastructure

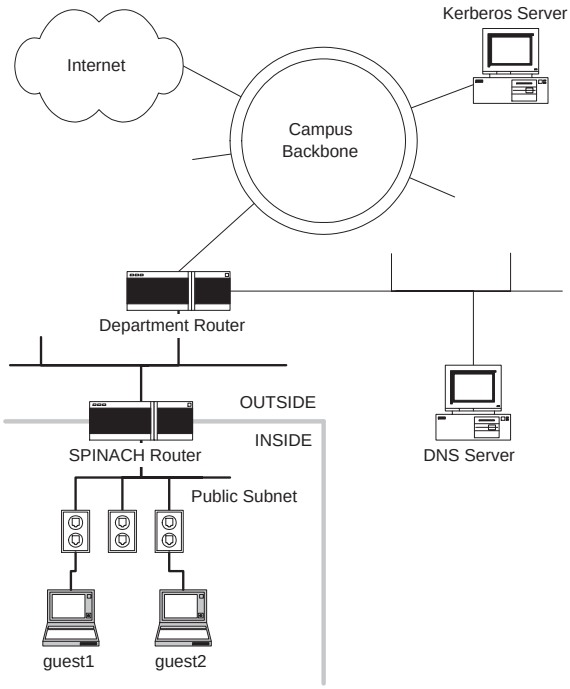

FIGURE 1. Network and security arrangement of the SPINACH system. The gray line running through the SPINACH router illustrates the prisonwall boundary, which separates the public subnet (inside) from the network as a whole (outside).

through a SPINACH router. The SPINACH router, an IP-routing Unix host (fully described in Section 4.1), forwards data packets between hosts on these public LANs and the outside networks. For routing purposes, hosts connected to the public ports are grouped into one or more IP subnets.

In our deployed SPINACH prototype (see Figure 1), the public ports are Ethernet ports located in publicly accessible areas of our building. These Ethernet ports are connected by a VLAN switch, so that data flows between them as if they were on the same LAN segment. Hosts connected to the public ports (labeled as "guest1" and "guest2" in Figure 1) are assigned addresses from one subnet, which we refer to as the "public subnet." The SPINACH router is connected to the same VLAN so it can route packets between the public subnet and the rest of the building network. In other SPINACH installations, some type of LAN other than Ethernet could be used, more than one LAN could be used to connect the public ports, and hosts could be arranged into more than one IP subnet, but for the purposes of this paper we assume the arrangement of our prototype system. Changing these parameters would require slight modifications to the routing and filtering

software on the SPINACH router, but the system would function in basically the same way. For example, even a wireless LAN such as WaveLAN could be used for the public subnet, so long as the SPINACH system software were modified to accept WaveLAN, rather than Ethernet, link-layer addresses.

Figure 1 also shows the department Domain Name Service (DNS) server and campus Kerberos server. The Kerberos server provides authentication services for users affiliated with the University. Some other authentication service could work as well, with modifications to the user-authorization software on the SPINACH router; in this paper, we assume the use of Kerberos. The DNS server is needed for hosts on the public subnet to find the IP address of the campus Kerberos server.

Because all packets that travel between hosts on the public network ports ("inside the prisonwall") and hosts elsewhere ("outside the prisonwall") must be forwarded through the SPINACH router, the SPINACH router can filter out all packets that are deemed dangerous. The SPINACH router creates a security boundary between the public Ethernet ports and all other networks.

2.2 Security Policy

Being a research institution, we do not want to squelch the development or use of new network applications by instituting overly specific rules regarding exactly what traffic is allowed on the public subnet [5]. Thus, rather than taking the typical firewall approach by allowing only the use of certain prescribed protocols through proxies running at the security boundary, we filter traffic on a per-*user* basis. We restrict use of the network through public ports to those people whom we can hold accountable for their actions. The SPINACH router allows these trusted users unrestricted access to the network and prevents untrusted users from accessing the network at all.

Traffic to and from hosts within the public subnet can be divided into three types. *Outgoing* traffic travels from within the public subnet to hosts outside. *Incoming* traffic comes from hosts outside the public subnet and is destined for hosts within. *Internal* traffic moves between two hosts on the public subnet. The SPINACH router uses different packet-filtering policies for incoming and outgoing traffic, following a particular set of rules to determine whether a given packet will be forwarded towards its destination or dropped. Internal traffic is not affected by the SPINACH router at all.

The SPINACH router forwards all *outgoing* traffic from those hosts on the public subnet which a user has authorized using the procedure described in Section 3.2. All outgoing packets from unauthorized hosts are dropped, except packets addressed to the trusted DNS or Kerberos server; this traffic is necessary for hosts within the public subnet to authorize themselves. Once a user has authorized a host on the public subnet, the SPINACH router forwards all outgoing traffic from that particular host. An audit trail which records the identity of the user who authorized this host enables network administrators to hold the user accountable for any malicious traffic that originates from this host.

The SPINACH router forwards all *incoming* traffic, because we are solely concerned with hosts inside the prisonwall wreaking havoc upon the rest of the network, rather than the reverse. Information coming into the prisonwall from outside is not considered a security threat, because it is assumed that any hosts inside the prisonwall that are trying to extract secret information from outside machines would have to initiate such transactions from within the prisonwall, and unauthorized hosts are not allowed to send outgoing traffic in the first place.

The SPINACH router exerts no control whatsoever over *internal* traffic; these packets are carried directly from one public port to another through the LAN which connects them. Thus, any hosts that are connected inside the prisonwall must tolerate a hostile network environment.

In addition to policies regarding the awarding of network access to users, there must be policies regarding the removal of network access. At present, the SPINACH router authorizes network access for four hours at a time; the length of this timeout is a parameter we plan to experiment with, as described in Section 7. If a user wants to remain connected to the network for longer than this period, he must re-authorize his connection using the procedure described in Section 3.2.

2.3 Types of Users

In many SPINACH installations, it will be appropriate to group users according to the access permissions that should be granted to them, as well as the resources that are available to authenticate them. In our prototype installation here in Stanford's Computer Science Department, we have identified three such types of users: "Department Users," "University Users," and "Guests."

Department Users already have access to the building network in private offices and labs, but desire to connect temporarily in another part of the building, for example, to check e-mail while sitting in a conference room or lounge. Since they already have access to the building network, but simply want to connect in a different physical location for convenience, we should have no security concerns about allowing them to connect through public ports. Also, Department Users already have authentication records in the campuswide SUID (Stanford University Identification) database.

University Users already have access to Stanford's computer network in the public computer labs, and perhaps in the residence halls, but do not presently have the ability to connect to the network within the Computer Science building. System administrators within the CS Department are rightfully concerned about allowing them unrestricted access to networks within our building that they have not been able to use in the past. Like Department Users, University Users already have entries in the SUID database.

Guests are not in the SUID database and thus do not currently have the ability to access Stanford's network at all. Typically this group contains visitors from industry and other universities who are in the CS Department to meet with professors and students or attend symposia. Quite often these visitors bring their own laptop computers and would like to connect to their home networks through the Internet to access their e-mail or retrieve files. Before the implementation of the SPINACH system, there was no established mechanism for allowing these short-term visitors network resources, so guests have been forced to use low-bandwidth, high-cost wireless connections or informally borrow the use of a desktop machine in some willing person's office. Because relationships with these outsiders are important to Stanford, we should provide a mechanism for them to utilize our network resources in some reasonable way while they are visiting.

In general, different types of users may be extended different access rights on the network, at the discretion of the network administrator. In our case, due to the concerns of department network administrators, University Users are currently denied network access; Department and authorized Guest users are allowed unrestricted network access.

2.4 Hardware and Software Requirements of the Client

Especially because we have the various classes of users described above, it is important that we support many different configurations of hosts with minimal assumptions about the software present on these machines. Even University and Department users have a variety of platforms: DOS, Windows 3.1, Windows 95, Macintosh, and various flavors of Unix. We cannot foresee all platforms visitors from off-campus will use. Thus, writing and maintaining special network access software for such a large and growing number of platforms would be a burden on our network administrators. Also, visiting users would need to install this custom software on their computers to use our system, and that could be a hassle for them. We would thus like to rely solely on client software that most users will already have installed on their networked computers, or can easily obtain from other sources.

We *can* assume that the user's computer has some basic network software on it, since the user presumably has been using it to connect to some other network. Almost all networked computers will have either a telnet client or a web browser; if a visitor's computer has neither of these, they can most likely obtain one easily from a number of sources. (Our prototype system requires users to run a telnet client; an alternative web interface is currently under construction.) In addition, an increasing number of networked computers have Dynamic Host Configuration Protocol [3] (DHCP) and/or Kerberos [8] clients—for example, the widely-used Windows 95 operating system includes DHCP client software. In the design of our access restriction system, we require only a telnet client on the visitor's computer; if a DHCP or Kerberos client is present, we use it to simplify the configuration and authorization processes.

2.5 Requirements of the Network Infrastructure

Although it is less of a concern than the minimal software requirements on the client end, we also want to minimize the amount of maintenance overhead on the SPINACH router and elsewhere in the network. The less of a burden we place on network administrators, the less resistance we will encounter in deploying our system both within our department and in other institutions.

We take advantage of the existing campuswide Kerberos authentication service, as well as the departmental DNS server, to simplify some users' connection process as

described in Section 3.2. No modifications to these servers are required. The only modification required of the network infrastructure beyond the SPINACH router itself is that the Department Router (see Figure 1) must be configured to forward all packets destined for the public subnet through the SPINACH router.

3. User Interfaces

Most users only see the client interface to SPINACH, through which they enable network access across the SPINACH router. Both this interface, and the interface used by network administrators to maintain the SPINACH system, are described in this section.

3.1 Authentication Mechanisms

To limit use of the public ports to those who are authorized, SPINACH must provide mechanisms for users to authenticate themselves when connecting their hosts to the public subnet. Once the user's identity has been proven, the SPINACH router can enforce the security policies listed in Section 2. Until the user authenticates himself, he must be treated as an unauthorized user.

Department and University users have permanent entries in the campuswide Kerberos authentication database. By installing Kerberos client software on their laptops, these users may authenticate themselves with the SPINACH router by presenting a Kerberos ticket that has been obtained from the campus server. Department and University users who do not install Kerberos software on their laptops may obtain one or more guest passwords and connect using the same method as Guests (described below).

To become an authorized Guest user, a visitor must obtain clearance to use our network facilities. This is provided in the form of a *guest (userID, password) pair* which is generated at the request of some authorized person such as a faculty member or network administrator (see Section 3.3). The userID and password are both human-readable strings which can be given to the Guest and which the Guest enters using a telnet or HTTP connection to the SPINACH router. Since this transmission goes across the public subnet in cleartext, it must be a single-use password so that replay attacks are fruitless [6]. The userID need not be unique; it simply makes the system more robust in the face of password-guessing attacks, by increasing the number of combinations that must be attempted by miscreants.

3.2 Connection Procedures

A user who is connecting his laptop to a public network port within a SPINACH installation follows these steps:

1. The user connects his laptop to one of the public Ethernet ports.
2. If the user's laptop has DHCP client software, it automatically retrieves network configuration information from a DHCP server running on the SPINACH router and configures the laptop accordingly. Since this exchange occurs between the laptop and the SPINACH router—entirely *within* the public subnet, rather than *through* the prisonwall—packets from the as-yet unauthorized laptop are not blocked. If the user's laptop does *not* have DHCP client software, the user must configure the laptop's network software manually, entering the IP address and IP routing information marked on the Ethernet port so that packets are properly routed through the SPINACH router.
3. If the user is permanently authorized—that is, a Department or University User—and has Kerberos client software on his laptop, he enters his personal password into his Kerberos client software to obtain a *ticket* from the trusted campus Kerberos server (see Figure 1). A special IP packet filter rule on the SPINACH router allows unauthorized machines to communicate with only the trusted campus Kerberos server and the department's DNS server through the prisonwall. If a permanently authorized user wants to access the network from a laptop *without* Kerberos client software, he must obtain a one-time guest password and log in in the same way visitors do (described in the following step).
4. The user initiates a telnet connection to the SPINACH router using the telnet client on his laptop. As in step 2, since this communication is *to* rather than *through* the prisonwall, it is not blocked. If the telnet server sees that the user has obtained an appropriate ticket from the trusted campus Kerberos server, the IP address and hardware (Ethernet) address of the laptop are recorded and the laptop is authorized to use the network facilities. Otherwise, the modified telnet server on the SPINACH router prompts the user to enter a userID and single-use guest password. If the user enters a valid (userID, password) pair, network access is granted.

Once the SPINACH router's modified telnet server has granted network access, a filtering rule (as described in Section 4.3) is added that allows all traffic coming from this host to be forwarded out of the prisonwall as neces-

sary for a certain length of time. The user's telnet client displays a message to this effect and then is automatically disconnected. The user then has unrestricted network access for a certain length of time (currently four hours).

3.3 Generating Guest Passwords

We must provide some mechanism for generating guest passwords for visitors. These guest passwords are generated on the SPINACH router itself; this avoids transmitting them in cleartext across the network until they are actually used. A small number of users, chosen by the network administrators who install the SPINACH system, are given user accounts on the SPINACH router. When these users initiate a Kerberos-authenticated and encrypted telnet session with the SPINACH router, they are allowed to log in and obtain a shell process on the router (which is running Unix). Then they can run a special password-generating program that creates human-readable one-time passwords. The (userID, password) pairs are entered into a database on the router for future comparison and displayed on the user's telnet client. Since the telnet session is known to be encrypted, there is no danger of new passwords being snooped by other hosts on the network. It is up to administrators at sites where SPINACH is deployed to develop a mechanism for distributing the one-time passwords to visiting users.

3.4 Long-Term Maintenance

One of the goals of the SPINACH system is to minimize the maintenance required for continued operation. Besides generating guest passwords, there is usually no manual maintenance required. But, should a network administrator want to examine the audit trail maintained by the SPINACH router or debug a problem on the router, he can follow the same procedure as mentioned above to log in and execute arbitrary commands on the router. The number of users with accounts on the SPINACH router should be kept to a minimum so that there is less chance of malicious or inept activity on the router that opens security holes.

4. System Implementation

4.1 Software on the SPINACH Router

The SPINACH router is an Intel Pentium-based computer running a Linux 2.0.30 kernel modified to filter IP packets based on hardware address as well as IP address. Since the SPINACH router is connected to a network with many untrusted hosts, it is best to run as few network servers as possible on the router to reduce the possibility of break-ins [2][4]. However, there are a few pieces of software that must be running on the router to implement prisonwall functionality—that is, to forward network traffic into and out of the prisonwall (subject to the policies in Section 2.2), and to allow hosts within the prisonwall to move from "unauthorized" to "authorized" status as appropriate.

The following pieces of software must be running on the SPINACH router:

1. **packet filter**: routines within the IP forwarding code in the kernel that allow packets to be routed or dumped selectively, based on source and destination port numbers and IP addresses, as well as source hardware addresses.
2. **prisonguard**: a user-level process that is always running on the router, modifying the packet filter parameters in the kernel as necessary and maintaining databases of guest passwords and authorized University/Department users.
3. **modified telnet server**: modified so that when most users connect to the telnet server on the SPINACH router and are properly authenticated, network access is enabled but a login shell is not provided.
4. **authorization clients**: processes that communicate with the prisonguard process to enable and disable network access and generate guest passwords.
5. **DHCP server**: a standard DHCP server, with no authentication extensions, which allows for automatic configuration of IP and higher-layer protocol information on any host with a DHCP client.

4.2 Communication Between Authorization Clients and the Prisonguard

The *prisonguard* process is so named because it maintains control over all security features of the prisonwall (SPINACH) router. It keeps a record of all generated guest (userID, password) pairs as well as a list of permanently authorized users, validates entered guest passwords against the list of generated ones, and modifies the packet forwarding rules in the kernel as appropriate in various cases. While other pieces of the software, such as the modified telnet server and the guest password management program, are short-lived processes and exist only long enough to collect information from one particular host as it moves from authorized to unauthorized, the prisonguard process runs constantly and

maintains all the state necessary to perform appropriately. For example, when a host has been authorized for a certain amount of time, the prisonguard keeps track of how long the host has been operating and when its authorization should be revoked and is responsible for revoking the authorization at that time. Although the prisonguard process maintains this state in main memory for efficient operation, it also writes the information to disk periodically in case the SPINACH router crashes or shuts down.

The short-lived processes that communicate with the prisonguard and tell it to authorize and unauthorize hosts, as well as generate and check guest passwords, are called *authorization clients* (see Figure 2). They are started as a result of a new user attempting to authorize his own host, or a SPINACH administrator running code on the router itself. Only known authorization clients on the SPINACH router have a legitimate need to communicate with the prisonguard, which acts as the authorization server, so communication with the prisonguard takes place over Inter-Process Communication (IPC) [9] that is only accessible to these processes and the superuser. To implement this secure IPC, the prisonguard uses a Unix domain socket linked to a Unix file that is only accessible by the superuser and other members of the group *authclients*. The following functions are available for authorization clients to call for communication with the prisonguard:

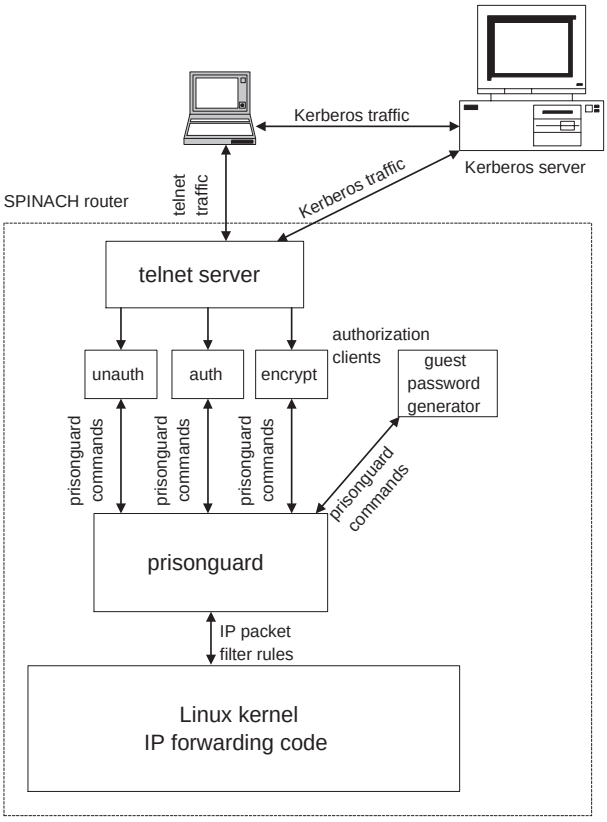

FIGURE 2. Inter-Process Communication between authorization clients and the prisonguard, and between the prisonguard and kernel IP forwarding code.

- **guard_authorize_nonguest(ipaddr, username)**
  The Department or University user by this username (already authenticated via Kerberos) has requested use of the network. If this username is that of an approved user, allow the host with this IP address use of the network. Make note of the source Ethernet address of the last packet received from this IP address so that other hosts cannot mistakenly or maliciously assume the same IP address and pass as the same host.

- **guard_authorize_guest(ipaddr, userID, passwd)**
  A Guest has entered this userID and one-time guest password to request use of network facilities. If this (userID, password) pair is valid, allow the host using this IP address access to the network. Make note of the source Ethernet address of the last packet received from this IP address so that other hosts cannot mistakenly or maliciously assume the same IP address and pass as the same host.

- **guard_unauthorize(ipaddr)**
  Disallow the host using this IP address access to the network. As noted above, the prisonguard process takes responsibility for automatically unauthorizing a host after some amount of time, so this command is not commonly called. However, future security improvements to the SPINACH system may require this functionality. For example, we may want to provide a tool with which network administrators can remove users' network access privileges before they would normally expire.

- **guard_getpassword(keyinfo, passwd)**
  Asks the prisonguard to generate and return a new one-time guest password. The keyinfo field contains the guest userID, as well as information about the Guest user and/or the Department contact who has requested this password, for auditing and/or billing purposes.

As shown above, the authorization clients can demand that the prisonguard allow or disallow network access to a particular host; thus, it is very important that access to all processes running with a group id of *authclients* is closely guarded. It is for this reason that only a very few users—those who are maintaining the SPINACH

system or generating guest passwords—are allowed to log in and execute arbitrary code on the SPINACH router. The modified telnet server allows other users only to enable network access for their hosts.

4.3 Filtering Rules

Packets are filtered during the forwarding process within the Linux kernel [10] according to the security policies set out in Section 2. All packets leaving through the SPINACH router are assumed guilty until proven innocent; that is, they are dropped unless they meet at least one of the following criteria:

- They are destined for the trusted departmental DNS server (may be needed to find Kerberos server), *or*
- They are destined for the trusted Kerberos server (needed for authorization process), *or*
- They come from an authorized user.

A packet is judged to come from an authorized user if:

- Its source IP address is that of an authorized host, *and*
- Its source hardware (Ethernet) address matches the hardware address recorded when this host was authorized.

The second check is necessary to guard against unauthorized hosts within the prisonwall that spoof IP addresses of authorized hosts. When a host is newly authorized, the SPINACH router freezes its ARP entry that maps the host's IP address to its observed hardware address. As long as this host is authorized, the ARP entry will not be modified. Any packets that later purport to come from this host's IP address will be checked against the previously recorded hardware address. Unfortunately, hardware addresses are spoofable as well, although not as easily as IP addresses; we address this problem in the following section.

5. Security Considerations

Any network security system involves tradeoffs among the strength of security provided, the equipment required for implementation, and the inconvenience caused to its users. Network administrators must decide which factors are most important and choose a security system accordingly.

The SPINACH system is designed with minimal assumptions about the network hardware and client software. We have made these constraints so the system will be useful in many network installations, and to free network administrators from the burden of maintaining special software for an ever-changing array of client machines. The cost of these gains is that the SPINACH system does not provide the same level of security as some other schemes (two of which are described in the following section) that require more sophisticated network hardware or client software.

By filtering packets from within the prisonwall by hardware address as well as IP address, the SPINACH router prevents casual miscreants from using public network ports without authorization. It is fairly simple in most operating systems to modify the source IP address of transmissions manually; modifying the source hardware address is more difficult, making illicit use of the network that much less likely. The determined hacker, however, will be able to obtain unauthorized access to the network by observing traffic on the public subnet, capturing hardware address / IP address pairs from legitimate users, and then modifying his network interface parameters to imitate an authorized user. Depending on the particular situation, the legitimate user may or may not be aware that something is amiss.

In practical terms, there are two types of Ethernet media: switched and shared. In a shared medium, hardware address filtering is the best security we can provide without additional software requirements of the clients. On a switched Ethernet, as we have in our building, packets that are sent to or from a particular host are not seen at all network ports; the switch routes packets destined for a particular hardware address to its port only. Because of this, it is significantly harder in this type of system for a malicious user to snoop other users' hardware addresses and then impersonate them. It is still possible, however; broadcast packets are typically forwarded to all network ports, at which point the source hardware address can be discovered by a malicious user.

By requiring a switched Ethernet instead of the much cheaper (and often already installed) shared-medium variety, the SPINACH router could communicate with the LAN switch to associate a physical port identifier with each legitimate user and thus prevent a hacker on some other port from imitating the legitimate user. Another approach would be to require custom software on all clients so that they could be repeatedly authenticated without user intervention. The first approach might require an expensive retrofit of the existing network architecture; the latter would require special software to be installed on all clients. If the network administrators require absolute protection against unau-

thorized network use at any cost, these changes may be appropriate; however, if they are constrained by time and money, they can use SPINACH to provide the best security possible for their network.

SPINACH is also vulnerable to a denial-of-service attack mounted by a malicious user against the Kerberos or DNS servers. Such an attack could block access to those services even to people outside the public network. A seemingly simple solution to this problem is to filter out Kerberos and DNS requests in addition to other packets from sources that seem to generate an extraordinary number of requests. With more experience about normal request loads, we will be able to judge whether this is a reasonable approach.

Another security concern in the SPINACH system is the vulnerability of hosts connected *within* the public subnet. Since all filtering by the SPINACH router occurs on the boundary of the public subnet, rather than within it, there are no restraints placed on the interaction between hosts on the public subnet. This means that even unauthorized users connected to the public subnet will be able to send packets to and receive packets from authorized users on the same subnet. Users of the public subnet should be made aware of this possibility; any user of the public subnet uses the network at his own risk, and it is his responsibility to encrypt his transmissions or fortify his host against attack.

Many network administrators may find that installing a system such as SPINACH will keep unauthorized network use to a tolerable level. SPINACH does not preclude the use of traditional firewalls; installing firewalls around the most sensitive parts of a network will protect them from abuse from public network ports and elsewhere.

6. Related Work

We are aware of two other proposals to deal with the problem we attempt to solve with the SPINACH system: Carnegie Mellon's NetBar proposal, and UC Berkeley's position paper, "Authenticating Aperiodic Connections to the Campus Network." Both of these systems, rather than using hardware address information to filter packets, use expensive, proprietary solutions at the link layer to isolate unauthorized hosts. Also, they both require more in terms of software that must be installed on every client.

In the NetBar system [7], a Cisco Catalyst VLAN switch is used to isolate unauthorized hosts. When a new client attaches to an available network port, it receives an IP address via DHCP, and the port is connected to a VLAN with limited connectivity suitable only for completing the authorization process. The user then uses Kerberos to authenticate himself, at which point the NetBar server sends an SNMP message to the VLAN switch, connecting this port to an "attached" network with full connectivity. Using the VLAN switch to connect and disconnect hosts in this manner provides more security than SPINACH, because it is not vulnerable to hardware address spoofing. However, it also uses proprietary signaling methods and thus relies on the presence of a specific brand of VLAN switch. (Many existing Ethernet installations do not use a VLAN switch at all; the NetBar's designers admit that the need for switched Ethernet ports can often constitute a financial barrier, particularly on a college campus.) In addition, the NetBar proposal requires all hosts to run DHCP and Kerberos client software, and only allows users with entries in the Kerberos database to connect to the network. The NetBar could be easily extended, though, to allow the use of guest passwords as we have done in SPINACH.

UC Berkeley's proposal [11] is even more demanding in terms of client software and network infrastructure. Hosts that are connecting to the campus network must contain DHCP client software that has been enhanced to exchange authentication information, rather than using the widely implemented Kerberos protocol or allowing the entry of guest passwords over telnet. This makes it extremely difficult for short-term visitors to make use of the campus network; they must install new DHCP client software on their machines, and network administrators must maintain such DHCP clients for all platforms that visitors are allowed to use. Also, their scheme for enabling and disabling communications on particular network ports, while more secure than hardware address filtering, requires the use of a LAN hub that has been specially modified by the manufacturer. This will make the use of this system impossible in many cases.

7. Future Work

As we receive feedback from users and administrators of the SPINACH system in our building, we will modify the policies and interfaces to fit their suggestions. We anticipate adding a more user-friendly HTTP interface for Guest users to authenticate themselves. Also, we may implement the S/IDENT protocol as an alternative to Kerberized login for authentication of Department and University users that would require no manual login procedure.

In the interest of improving security without requiring more sophisticated network hardware and client software, we will also tune the security policies to minimize a hacker's "window of opportunity" obtained through hardware address spoofing. First, we will adjust the authorization timeout period to reflect the observed duration of network usage, so that it is long enough to inconvenience few users but limits the amount of time a hacker can access the network with a captured hardware address after the legitimate user has stopped using the network. Also, we will look into disabling network access for a host after some period of inactivity, the goal being to mark an IP address / hardware address pair as unauthorized before a hacker realizes that the legitimate host is no longer on the network.

If we conclude from network administrators' feedback that a defense against hardware address spoofing is necessary, we will probably incorporate dynamic VLAN switching similar to that used in the NetBar, described above, to authenticate traffic based on physical LAN port rather than hardware address. This solution would significantly increase the cost of the network infrastructure needed to support SPINACH, but by combining dynamic VLAN switching with the dynamic IP filtering possible with the SPINACH router, we could continue to support client hosts without any special network software.

8. Conclusions

The Secure Public Internet Access Handler strikes a good balance between allowing temporary network users the freedom to use whatever applications they like on whatever platforms they like, and limiting network access to authorized users. Because it does not limit users to using a prescribed set of applications and protocols, it does not prevent users with network access from causing trouble. Instead, it limits the pool of users to those who will be accountable for their actions, i.e., Stanford affiliates and identified guests. This is an appropriate solution for an academic research environment, and may also be the best solution for public organizations such as schools and libraries that require some control over who is using their networks without the large capital outlays needed for alternative solutions.

As hackers become more savvy and switched Ethernet hubs less expensive, more feature-rich, and, we hope, more standardized, it may well make sense to use a switched LAN architecture to isolate unauthorized network users. Until this day comes, however, SPINACH will provide a useful measure of access control even in shared-medium Ethernet installations, and even better (though not bulletproof) security in a switched Ethernet environment.

Source code and other information regarding SPINACH will be made freely available on the MosquitoNet Project web site,
http://mosquitonet.stanford.edu/mosquitonet.html.

9. Acknowledgements

We would like to thank fellow group members Stuart Cheshire, Kevin Lai, Petros Maniatis, Diane Tang, Akihiro Tominaga, and Xinhua Zhao for their help in improving both the design of SPINACH and its presentation in this paper. We particularly thank Mema Roussopoulos for her help in implementing the system as well.

In addition, we thank Dan Boneh and Jonathan Stone for their advice in the design of SPINACH, and Armando Fox, Darrell Long, Nick McKeown, Lorenzo O'Reilly, and the anonymous reviewers for their careful readings and help in improving this paper.

This work was supported in part by a Terman Fellowship and a grant from the Keio Research Institute at SFC, Keio University, and the Information Technology Promotion Agency, Japan.

10. References

1. Alexander, S. and R. Droms. "DHCP Options and BOOTP Vendor Extensions; RFC-2132." *Internet Request for Comments*, no. 2132, March 1997. <http://ds.internic.net/rfc/rfc2132.txt>

2. Cheswick, William R., and Steven M. Bellovin. *Firewalls and Internet Security.* Addison-Wesley, Reading, MA, 1994.

3. Droms, R. "Dynamic Host Configuration Protocol; RFC-2131." *Internet Request for Comments*, no. 2131, March 1997. <http://ds.internic.net/rfc/rfc2131.txt>

4. Garfinkel, Simson, and Gene Spafford. *Practical Unix Security.* O'Reilly & Associates, Sebastopol, CA, 1991.

5. Greenwald, Michael B., Sandeep K. Singhal, Jonathan R. Stone, and David R. Cheriton. "Designing an Academic Firewall: Policy, Practice, and Experience With SURF." *Proc. IEEE Symposium on Network and Distributed Systems Security*, 1996. <http://www-dsg.stanford.edu/papers/firewall/>

6. Haller, Neil M. “The S/Key One-Time Password System.” *Proc. Internet Society Symposium on Network and Distributed System Security,* pp 151-157. San Diego, CA, February 1994.

7. Napjus, Erikas. “NetBar: Carnegie Mellon’s Solution to Authenticated Access for Mobile Machines.” <http://www.net.cmu.edu/design/netbar.html>

8. Steiner, J. G., C. Neuman, and J. I. Schiller. “Kerberos: An Authentication Service for Open Network Systems.” *Proc. USENIX Winter Conference*, February 1988, pp 191-202.

9. Stevens, W. Richard. *Unix Network Programming*. PTR Prentice Hall, Englewood Cliffs, NJ, 1990.

10. Vos, Jos, and Willy Konijnenberg. “Linux Firewall Facilities for Kernel-Level Packet Screening.” *Proc. NLUUG Spring Conference 1996.* De Reehorst, Ede, The Netherlands, May 1996. <http://www.xos.nl/linux/ipfwadm/paper/>

11. Wasley, D. L. “Authenticating Aperiodic Connections to the Campus Network.” *ConneXions (*Interop August 1996), vol. 10, no. 8, pp 20-26. <http://www.ucop.edu/irc/wp/wp_Reports/wpr005/index.html>