User-Friendly Access Control for Public Network Ports

Guido Appenzeller Mema Roussopoulos Mary Baker
{appenz,mema,mgbaker}@cs.stanford.edu
Department of Computer Science, Stanford University
http://mosquitonet.stanford.edu

Abstract—**We are facing a growing user demand for ubiquitous Internet access. As a result, network ports and wireless LANs are becoming common in public spaces inside buildings such as lounges, conference rooms and lecture halls. This introduces the problem of protecting networks accessible through these public ports from unauthorized use. In this paper, we study the problem of access control through public network ports. We view this problem as a special case of the more general problem of access control for a service on a network. We present an access control model on which we base our solution. This model has three components: authentication, authorization, and access verification. We describe the design and implementation of a system that allows secure network access through public network ports and wireless LANs. Our design requires no special hardware or custom client software, resulting in minimal deployment cost and maintenance overhead. Our system has a user-friendly, web-based interface, offers good security, and scales to a campus-sized community.**

I. Introduction

The Stanford Computer Science Department is housed in a building that includes Ethernet ports in every office as well as in public spaces such as lounges, conference rooms, and lobbies. These public ports are connected to the department network. Until recently, the public ports had not been activated because of concerns that unauthorized users could gain access to the departmental network. Once inside the network they can cause damage by using it to launch attacks on computers in the department and use services and software that restrict access according to IP address. The department would additionally be exposed to potential liabilities for activities against other systems on the Internet (e.g. spamming, hacking) originating from these ports.

Current solutions to the problem of secure network access through public ports have several drawbacks: they are expensive; they are not user-friendly; they require constant maintenance; or they require special hardware or custom client software. Our goal is to provide a system that is low-cost, user-friendly, and scalable, and that requires little effort to maintain and no special hardware or custom client software.

We have built a system called the Secure Public Internet Access Handler (SPINACH). The prototype of SPINACH, described in [15], was low-cost and low-maintenance. While it did not require custom client software, regular users who did not own Kerberos-aware telnet clients had to request administrative assistance each time they used the system. Also, the telnet-based interface used was not user-friendly. We have redesigned SPINACH so that it achieves all of our goals. This makes it a viable solution for use by a variety of public organizations, such as public schools, libraries, and universities, where reasonably secure network access is desired, but keeping cost and maintenance effort at a minimum is essential.

The SPINACH system establishes a "prisonwall" which controls the flow of traffic between hosts connected to the public ports and hosts on the departmental network. Unlike a firewall, which protects machines *inside* a particular network from malicious users *outside* the network, the prisonwall protects hosts *outside* one portion of a network by refusing to forward packets that come from unauthorized hosts within that network. [15]

In this paper, we describe the design and implementation of the second version of the SPINACH system. In Section II, we discuss the overall design principles that guided us in our study of secure public port access control and the model on which we base our solution. This model has three components: authentication, authorization, and access verification. In Section III, we discuss the design features that are necessary to make the system as usable and widely applicable as possible. In Section IV, we describe the architecture of our system, and show how it maps to our model. In Section V, we evaluate the system using our design goals. In Section VI, we describe our preliminary experience with the system. In Section VII, we describe related work. Finally, in Section VIII, we describe future work, and in Section IX we state our conclusions.

II. A Model for Performing Network Service Access Control

Network service access control is the problem of determining whether a user should be allowed to access a particular service within a network. The problem of secure network access through public ports is a special case of network service access control, where the service offered is an open path to a subnet or the Internet.

We base our solution to the public port access problem on a model composed of three modules. This model, pictured in Figure 1, is inspired by the Kerberos [16] security model.

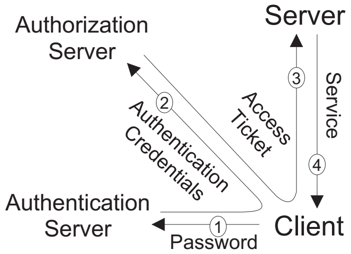

Fig. 1. Our access model: We separate the access control process into three steps: authentication, authorization and access verification. Each step may be handled by a separate server on the network.

A. Modules of the Access Control Mechanism

• **Authentication.** Authentication is the process of identifying a user by associating him with a known and trusted organization. In the real world, this could be done by showing an identification card that shows the user's name and that he is resident of a certain state. On a computer, the user might be asked to enter a username and password. If these appear in the password database, the computer will assume the user is the person described in the database.

- **Authorization.** Given the identity of a user, authorization is the process of deciding whether the user can have access to a particular service. Often the policy is to give access to all successfully identified users, but in many cases additional mechanisms are used to differentiate among users requesting a particular service.
- **Access Verification.** The access verification module enforces the decision made by the authorization module. Unless it receives proof that a user has been authorized to use a service, the access verification module denies access to the service. Note that the access verification module does not need to know the user's identity to perform its function.

An important characteristic of our model is the clean separation of authentication, authorization, and access verification. To make the separation of these three modules more clear, consider the following real-world example. A college student decides to meet her friends at a bar. The bar's bouncer requests that the student hand him her driver's license. If the picture on the license matches the student's face, the bouncer is satisfied. This is authentication. The bouncer then checks the student's date of birth. If the student is at least twenty-one years old, the bouncer places a band around her wrist. This is authorization. The student joins her friends and orders a drink. The bartender checks that she is wearing a valid wristband, and hands her a drink. He does not care what her name is nor what her age is; he only cares that she has a wristband. This is access verification.

B. Module Requirements

In many systems these three steps are performed in one module. In the example above, the bouncer performed both the authentication and the authorization of the student. We believe that in very large systems, however, it is beneficial to keep these three steps separate for a number of reasons:

- **Differing Levels of Security.** The level of security required differs significantly from step to step. Compromising the access verification module gives an adversary access to the service. However an intrusion into the authentication module is more severe as an adversary can now impersonate the user for this as well as any other service. In a network, the authentication module would therefore be protected behind a firewall with data entry capabilities restricted to a small group of people. Depending on the service, the authorization and access verification modules may not need as high a level of security, and should not be required to adhere to the same policy as the authentication module. Also, compromising one module should not put the other modules at risk.
- **Modules Controlled by Different Parties.** We want to allow different parties to have control over each of the modules. Authentication might be assigned to a large central authority such as a company or a university. The authorization policy for a service might be set by the entity or department maintaining that service. Access verification needs to be performed on every machine rendering the service.
- **Flexibility and Scalability.** Modular design allows us to reconfigure one module without affecting the others. In our design, we take advantage of the existing campus Kerberos infrastructure, but could easily replace it with a different authentication scheme. Also, several instances of one module can exist and be used in parallel. For example a single authorization module can be configured to accept authentication credentials from several authentication modules, both in local and remote networks. This allows us to scale access control for services across multiple administrative domains.

III. Usability Issues

We want to make the SPINACH service as "usable" as possible. That is, we want to enable a variety of organizations such as public schools, libraries, and universities to offer and use the SPINACH service. To do so, we must achieve four goals: place minimal software and hardware requirements on the client, make the service as user-friendly as possible, require a minimal amount of overall system administration, and make the service scalable. We address each of these goals below.

Minimal client requirements. The SPINACH service should work with any common operating system without additional software. Among the most common operating systems are Windows 95, Mac OS and Unix-flavored systems. SPINACH should easily support all of these operating systems. Therefore, the only software we *require* of the client is a web browser that supports the Secure Socket Layer (SSL) [8]. This requirement is minimal since today's popular web browsers support SSL. We *recommend* that the client support Dynamic Host Configuration Protocol (DHCP) software [7]. This is solely for user convenience, so that users need not configure IP addresses manually. On the hardware side, we want to support as many network interfaces as possible. Currently, we support any interface that uses IP and has the notion of a hardware address (e.g., Ethernet, Wireless Ethernet).

Minimal network hardware requirements. The SPINACH service should not require any special hardware in the network. Current solutions require special brands of VLAN switches that enable and disable ports as hosts are authenticated. Many existing Ethernet installations do not use special VLAN switches, and retrofitting a building or college campus to use such VLAN switches would be time-consuming and costly. Furthermore, imposing special hardware requirements precludes using a network access service on other kinds of networks such as wireless LANs and WANs. Currently, SPINACH does not require any special network hardware and runs on both Ethernet and wireless LANs such as Lucent's WaveLan [3].

User-friendly interface. Users should have an appealing, easy-to-use interface that offers comprehensive instructions for each step of the SPINACH process. To achieve this goal, we have developed a web-based interface to the SPINACH service.

Minimal administrative effort. The SPINACH service should require minimal maintenance effort from system administrators. Changes in user software platforms should have no effect on maintenance. This means that handing out special software to clients is not an option, because maintaining up-to-date versions of the software for every possible client platform is extremely tedious. The goal of minimal administrative effort goes hand in hand with the goal of minimal client requirements.

Scalability. The SPINACH service should support a large number of users without degraded performance or substantial increase in administrative effort. Currently more than 10,000 people on the Stanford campus can use the SPINACH system. SPINACH can easily be scaled up to serve users belonging to remote organizations. Plans are currently under consideration to give access to users who have been successfully authenticated by authentication modules outside Stanford.

IV. System Architecture

The overall architecture of the SPINACH system is displayed in Figure 2. We have set up the SPINACH service on both the public Ethernet ports and on a Lucent WaveLan [3] network installed in the Computer Science Building. Note that Figure 2 shows only the public Ethernet ports. The WaveLan network and all of the public Ethernet ports are logically placed on one subnet labeled as the public subnet in Figure 2.

This public subnet is connected to the departmental network (and thus the Internet) through the SPINACH router. The SPINACH router thus has three interfaces: an Ethernet interface on the departmental network, and an Ethernet and WaveLan interface on the public subnet.

The main purpose of the SPINACH service is twofold: 1) to protect hosts outside the public subnet from unauthorized hosts within the public subnet, and 2) to provide an access control mechanism that allows users running hosts on the public subnet to authorize themselves and gain full network access.

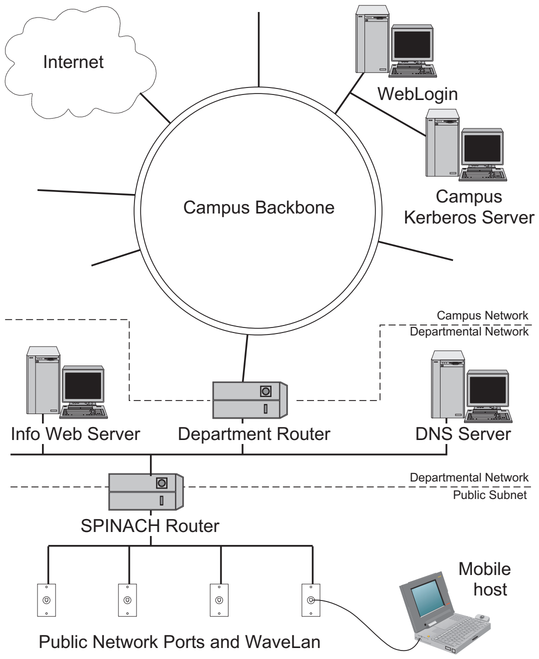

Fig. 2. SPINACH Network Layout

To protect hosts outside the public subnet, the SPINACH router establishes a “prisonwall.” This prisonwall controls the flow of traffic between hosts inside the public subnet and hosts outside the public subnet. Unlike a firewall, which protects machines *inside* a particular network from malicious users *outside* the network, the prisonwall protects hosts *outside* one portion of a network by refusing to forward packets that come from unauthorized hosts within. [15]

The prisonwall gives limited network access to all hosts, by forwarding all packets destined for the WebLogin and departmental DNS servers (pictured in Figure 2), regardless of whether these packets come from authorized hosts or unauthorized hosts. This is because these servers are needed in the access control procedure. The prisonwall gives *full* network access only to hosts used by persons who have successfully completed the access control procedure.

We use five pieces of software to provide a mechanism that allows users to gain full network access through the public ports. These are: the WebLogin Authentication Server, the SPINACH Authorization Server, the Access Verification Server, the User-Interface Web Server, and the DHCP Server. The WebLogin server runs on a machine connected directly to the campus backbone (see Figure 2). The other four pieces of software run on the SPINACH router. These are pictured in Figure 3. We describe each of the pieces of software below.

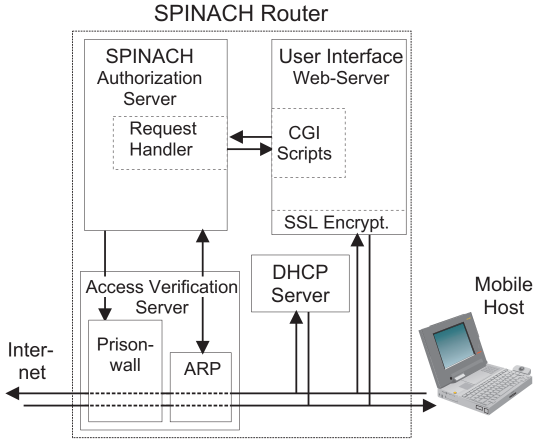

Fig. 3. SPINACH Server Architecture

A. *WebLogin Authentication Server*

The SPINACH service uses WebLogin to perform user authentication. WebLogin is a service implemented by the Stanford Distributed Computing Consultants [17]. This service offers a web-based interface to the campus Kerberos authentication service. Any server that wants to use WebLogin for authentication redirects the browser of the user to the site `weblogin.stanford.edu`. The user enters his Kerberos login name and password into a form provided, and the WebLogin server uses this information to obtain a Kerberos ticket encrypted using the secret DES key of the server from which the user was directed. This ticket is encapsulated into a web-cookie and passed back to the user’s browser. The user is then redirected back to the server. The user’s browser passes the cookie to the server, which then verifies the encapsulated Kerberos ticket. All transfers are encrypted using the Secure Socket Layer (SSL) [8], to protect the transmission of the encapsulated Kerberos tickets.

WebLogin allows us to take advantage of the existing campus Kerberos infrastructure, without requiring users to run Kerberos client software on their laptops. This is very helpful, because

Kerberos software is difficult to install and use. We estimate that Kerberos clients are installed on less than three percent of the laptops in the Stanford Computer Science Department. As a result, the inclusion of WebLogin in our design greatly enhances the usability of the SPINACH service.

Another important feature of the SPINACH design is that the user and the organization that is running the authentication database both do not have to trust the SPINACH Administrators. If SPINACH is set up by an untrusted party (e.g., a student residence), this party cannot read the user's authentication information.

We could easily plug a different authentication scheme into our design, should the need arise. This is possible because of the clean separation of authentication from authorization and access verification. We considered three other alternatives:

- One-time Password Schemes: Setting up a one-time password scheme such as S/KEY [9] for the entire campus community requires assigning passwords to each potential user. This would have involved too much administrative effort on our part.
- Certified Public Keys: Using public keys that have been certified by some trusted authority is an excellent authentication option. Currently, most users do not have public keys so managing keys for all users would have been necessary. Additionally, the support for personal public keys in current web browsers is still new, and users have little experience in using public keys.
- Tamper-proof Hardware: This option involves storing the user's password or key on portable and tamper-proof hardware such as smart cards and Java rings. The user's password (or key) never leaves the hardware and cannot be extracted by normal means. Although this technology is highly secure, it is also not yet commonly used.

Plans are currently under consideration to make the WebLogin code publicly available. This will enable the WebLogin service and consequently the SPINACH service to be used at other organizations. Until this happens, organizations wanting to port SPINACH to their systems will need to implement a web-based mechanism, similar to WebLogin, that uses the existing authentication infrastructure to identify users. Once this has been done, SPINACH can be ported easily.

B. SPINACH Authorization Server

The SPINACH Authorization Server acts as the driver of the SPINACH service. It generates the web-pages displayed by the Web Server, performs authorization, and invokes the Access Verification Server.

Currently, no specific authorization policy has been set. That is, the SPINACH server currently grants full network access for two hours to all successfully authenticated users. This includes all Stanford affiliates that are registered in the Stanford Kerberos database. No policy has been set because our building administrators have not yet decided to which Stanford affiliates the SPINACH service should be accessible. When they do, we will need a simple way of specifying high-level rules that authorize users according to certain criteria. For example, a rule might authorize "all graduate students and faculty in the Stanford Computer Science Department" or "all people who have an office in the Computer Science Building." Authorization schemes such as access control matrices are widely discussed in [6]. We extend this discussion in the Future Work Section.

The SPINACH server maintains a list of all current users and directs the Access Verification Server to terminate network access for hosts when their period of authorization expires. It also maintains an audit trail so that authorized users are held accountable for any malicious traffic they generate against hosts outside the public subnet. Thus, the department network is guarded from both malicious authorized and malicious unauthorized users.

C. Access Verification Server

The Access Verification Server contains two components: the prisonwall module and the ARP module. When invoked by the SPINACH server, the prisonwall module grants full network access to a host, by forwarding all packets from that host out to the departmental network, as described earlier. The ARP module monitors the contents of the ARP cache, and ensures that no packet leaving the public subnet has a source hardware and IP address pair that is inconsistent with the ARP cache. Although this guards against IP address spoofing, this does not guard against hardware address spoofing. We address this problem in Section V-C.

In many current systems, authorization and access verification are performed within the same module. We believe this is not a good idea. Authorization needs to be performed only once for each service type, whereas access verification must be performed on every attempt to access an instance of a service type.

Consider a user who wants to print a document. The user will want to use the print service. There may be several printers offering the print service, and each printer will make its service available to the user if it receives some "proof of authorization to print," often in the form of a ticket. The user needs to obtain authorization only once to print, and may present the same ticket to any number of printers, until the ticket expires. Since authorization is needed just once, there is no need to put this functionality in every printer. This would complicate the printer server implementation. Second, if the access verification module is compromised, then the worst that happens is that the service is used by an unauthorized user until the ticket expires; the authorization module is never compromised. Finally, separating the functionality of the authorization and access verification modules allows a change in authorization policy to be made without affecting the access verification module.

Note that in our implementation, the SPINACH Authorization Server and the Access Verification Server are both running on the SPINACH router. If a higher degree of security is needed, we can easily run the SPINACH Authorization Server on a different machine located behind a firewall. This would require the SPINACH Authorization Server to authenticate itself (possibly through SSL) to the Access Verification Server.

D. SPINACH User Interface

The user interface of SPINACH is a modified Apache Web Server, located at `http://spinach.stanford.edu`. It displays the initial welcome screen to the user and guides users through the whole access control process. When a user connects to the web server, he is redirected to the WebLogin server. The WebLogin server authenticates users and redirects them back to the web server. The web server decodes the cookie generated by the WebLogin server and runs a CGI script that passes authentication information to the SPINACH Authorization Server.

The web pages that the web server displays are either locally stored or generated by the SPINACH server. The web server also takes care of the encryption of the communication between the user and the SPINACH system. It uses the Secure Socket Layer for communication with the user's browser, and has a module that allows it to decrypt and decode cookies received from the WebLogin server.

E. DHCP Server

We use a standard Dynamic Host Configuration Protocol (DHCP) server [7] that hands out network configuration information in the form of a lease to hosts on the public subnet. Hosts use this information to configure themselves automatically. Hosts without DHCP support must be manually configured by users. DHCP client software is available on a wide variety of platforms, including the widely-used Windows 95 operating system, and many Unix-flavored operating systems. Therefore, we believe it is reasonable to assume that most users have hosts with support for DHCP.

F. System Usage Scenario

In a normal usage scenario, the user plugs his laptop into a public port. The laptop automatically obtains a DHCP lease, and configures itself accordingly. The user sets his browser to point to the SPINACH Web Server located at `http://spinach.stanford.edu` (This step might become automatic in the near future. A solution involving IP redirection is currently being developed). The user's browser is redirected to the WebLogin Authentication Server. The authentication information the user types into the WebLogin form is sent to WebLogin using SSL. The user can verify the identity of the WebLogin Server by using his browser to check its certificate. When the Weblogin Server has finished authenticating the user, it redirects him back to the SPINACH Web Server's page. At this point, the SPINACH Web Server grants access to a CGI script that connects to the SPINACH Authorization Server. The SPINACH Authorization Server verifies that the user is a Stanford affiliate by checking the authentication information it receives from the CGI script. It then invokes the Access Verification Module which manipulates the prisonwall so that packets from the newly authorized host flow through. A log entry is generated so that it may be used for auditing purposes should the newly authorized host misbehave.

V. System Evaluation

We evaluate SPINACH using three criteria: ease of use from the user's perspective, scalability, and security robustness. SPINACH is ideal for novice users, and scales to a campus-sized community. Although SPINACH is not as secure as some other systems (see Section VII), it provides a good measure of security for the cost and administrative effort required to keep it running.

A. Usability and Administration

Currently, SPINACH is running and serves all public ports in the Stanford Computer Science Building. Users report that the SPINACH service is extremely easy to use. All users who have an entry in the campus Kerberos database have successfully been able to obtain network access.

Figure 4 shows an example of what the user actually sees on his screen when using SPINACH. The user first points his browser to the SPINACH Web Server at `http://spinach.stanford.edu` and sees the welcome screen (Figure 4a). The user selects the link entitled "Stanford Faculty, Staff or Students." This downloads the WebLogin page (Figure 4b). The user enters his login name and password. A web page appears indicating that the user has been authenticated (Figure 4c). The user clicks on the link provided that will send his browser back to the SPINACH service page. A final web page appears, indicating that the user has been granted full network access (Figure 4d). The user is now free to web surf, telnet into a remote machine, or perform any other network activity.

The administrative effort for SPINACH is minimal. Since all students, faculty and staff are registered in the campus Kerberos database when they first join Stanford, new users do not need to contact the SPINACH administrators before using the SPINACH service.

B. Scalability

In our design, there are two potential bottlenecks: WebLogin and the SPINACH router. We address both below, and show that SPINACH is scalable to a campus-sized community of over 10,000 users.

We depend on WebLogin for authentication. WebLogin is currently being used by the entire Stanford community including students, faculty, and staff. In total, WebLogin serves over 10,000 users. [2] From this, we can safely conclude that the authentication component of our design is not a bottleneck.

The SPINACH router by itself may not be able to support packets originating from 10,000 authorized hosts. However, this problem is solved by adding more machines to act as SPINACH routers. The public subnet can be partitioned into groups. Each router would be in charge of one group. This would make the routing task manageable. Recall that our modular design allows several instances of a module to coexist. Therefore, running several Authorization and Access Verification Servers in parallel can be done easily.

C. Security

The security of an access control service such as SPINACH does not depend solely on its ability to prevent access by unauthorized users. Additional issues are whether a malicious user can do damage to other users, and most importantly the severity of the consequences of a break-in into the SPINACH system.

Intrusion Tolerance. As SPINACH is publicly accessible and potentially can be operated by unreliable parties, it is important to consider the consequences of an adversary breaking into the SPINACH router. A user breaking into the SPINACH router is not able to steal authentication information about other users. Authentication is done directly with WebLogin and all traffic is encrypted using SSL. The only major damage caused by a break-in is compromising the DES key of the SPINACH server which then has to be reissued.

low) could however cause damage to the user by performing malicious activities in his name.

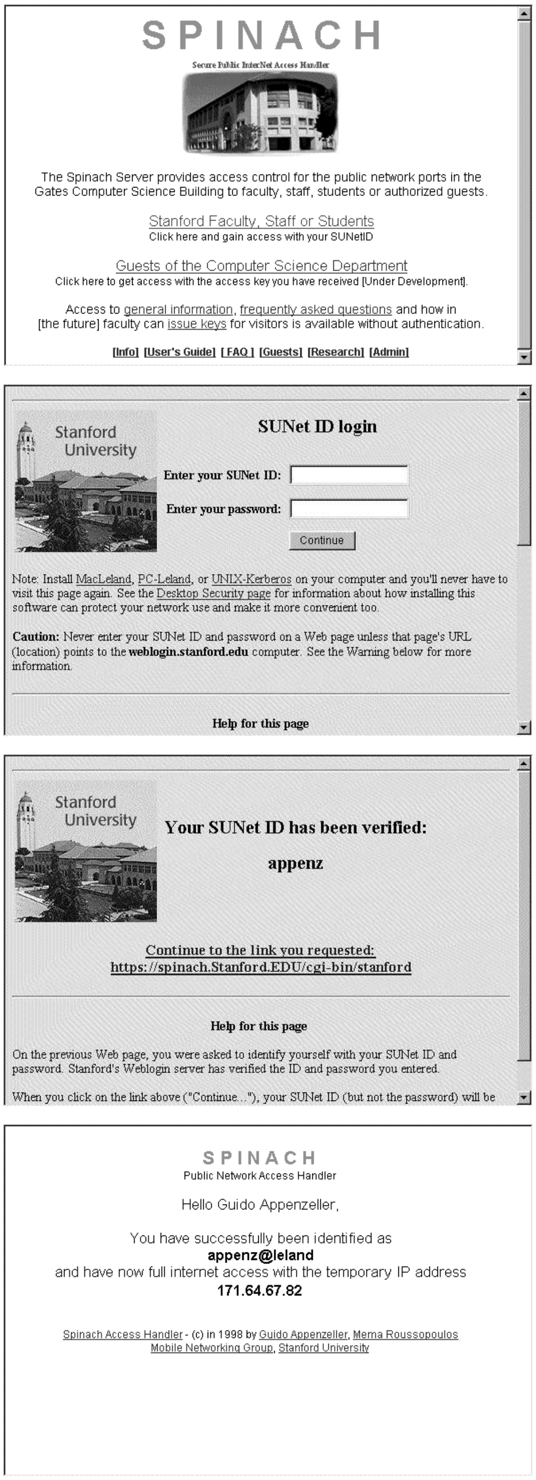

Figure 4: (a)-(d), User's view of SPINACH

Protection from other Users. SPINACH does not protect users from other hosts running on the public subnet. Any person connected to the public subnet can monitor other users' traffic or try to hack their systems. SPINACH, however, does track hosts once they are given network access so that they may be held accountable for any malicious traffic they generate against hosts outside the public subnet. An adversary that is able to spoof hardware addresses and thus impersonate another user (see below) could however cause damage to the user by performing malicious activities in his name.

Fake SPINACH Routers. A computer that is connected to the public network could pose as a fake SPINACH router, give out DHCP leases, and accept authentication cookies. It could then use these cookies to gain access to the Internet. The user can detect this attack, since the fake SPINACH Router lacks the certificate of SPINACH and either cannot use SSL for the connection or has to use a different hostname. In reality not all users pay attention to the site or whether encryption is used, and hardly any users verify the certificate of a site. For this reason we are considering additional security mechanisms (e.g. listening to DHCP offers on the network). Authentication cookies obtained by a fake SPINACH server are only valid for a short time and cannot be used to impersonate the user at other services.

Denial of Service Attacks. SPINACH is vulnerable to denial of service attacks launched on both the DHCP and SPINACH Web servers running on the SPINACH router. A possible solution is for the router to do additional IP-level filtering of DHCP and web requests from sources generating a high number of requests. The same denial of service attack can be launched against WebLogin or the department DNS server.

Hardware Address Spoofing. One way an adversary can try to get access to the network is by spoofing IP addresses of hosts that have been given access. Simple replication of the IP address is not enough as SPINACH verifies the hardware address of the sender. If the host however is additionally able to spoof the hardware address it can gain unauthorized access.

One way to avoid hardware spoofing attacks is to use strong authentication at the level at which the filtering is done. As we filter at the IP level this would mean using IP SEC [1] to authenticate on a per-packet basis. This would involve special client software.

A second way to avoid hardware spoofing attacks would be to use special network hardware such as certain VLAN switches that enable and disable ports as hosts that are connected to them are authenticated. This solution, however, is expensive and precludes the use of SPINACH on a wireless LAN. We discuss this further in the Related Work Section.

VI. Preliminary Experience

The SPINACH system has been up and running since September of 1998. We have set up the SPINACH service on both the public Ethernet ports and on a Lucent WaveLan [3] network installed in the Computer Science Building.

The SPINACH system maintains a log of all users of the system. This log is useful not only for security purposes, but also serves as an indicator of when and how often the system is being used. In the log we record the IP address of the user's host, the time network access is initiated, the time network access is revoked, etc. Note that we warn users of the collection of this information on the SPINACH web page.

Figure 4 shows the usage of the SPINACH system from September 20 through November of 1998. Until now, SPINACH has had a total of 935 user sessions or a median of 12 user sessions per day. As one might expect, usage of the SPINACH system is heavier during weekdays than weekends. The SPINACH system is used more heavily on the WaveLan network than

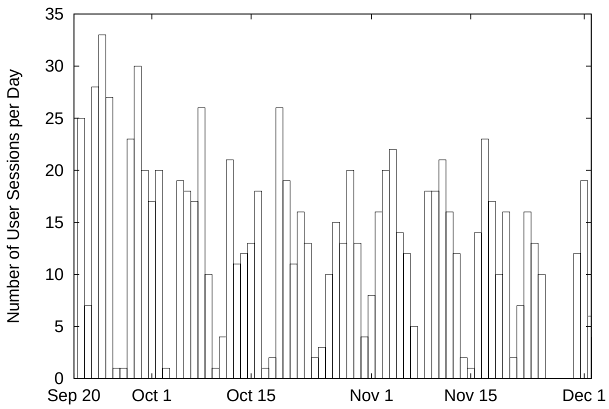

Fig. 4. Usage Statistics for SPINACH

through the Ethernet ports. This is also to be expected since the WaveLan network gives users the freedom to move about the building while they work.

We are pleased to find that the administrative effort involved in setting up and maintaining SPINACH on both network types is indeed minimal. After launching the service the system has averaged only one to two cases where users needed help per month.

VII. Related Work

We have benefited greatly from work on the previous version of SPINACH [15]. The first version of SPINACH had the same general goal as the current version, that of securely granting network access to authenticated users through the public ports in our building. It enabled both on-campus users as well as guest users to obtain network access. On-campus users are people who have entries within the campus Kerberos database. Guest users are visitors to the Computer Science Department who need access to the Internet. Without the SPINACH service, visitors desiring to check their e-mail or perform other network-dependent activities would informally borrow the offices and desktops of building residents. Users used a telnet-based interface to connect to the SPINACH router. Stanford-affiliated users would enter their Kerberos password which would be checked against the campus Kerberos database. Guest users would enter a one-time password obtained from a system administrator which would be checked against a database stored locally on the SPINACH router.

Although the telnet-based version of SPINACH achieved the goal of secure network access through public ports, there were two aspects that needed to be improved. To use the SPINACH service as a Stanford affiliate, users were required to carry telnet clients with support for Kerberos. This meant that Stanford users who did not have Kerberos support on their laptops could only use the SPINACH service as guests. This was particularly inconvenient for users who wanted to use the public ports every day and who were already in the campus Kerberos database. This was also inconvenient for system administrators who had to generate one-time passwords each time a user desired access as a guest. Our current version of SPINACH requires only a web browser on the client. This makes the SPINACH service convenient to use, even for Stanford affiliates who do not have Kerberos support on their laptops. Also, the telnet-based interface used by the previous version was not user-friendly. We have solved this problem by implementing a web-based interface, with which virtually all computer users are familiar.

To our knowledge, there are three other designs that propose solutions to the problem of secure access control for public network ports. These are the NetBar system at Carnegie Mellon and two designs by UC Berkeley and University of Michigan.

Carnegie Mellon's Netbar system [14] uses a Cisco Catalyst VLAN switch to isolate all the public ports on a "non-connected" VLAN with limited connectivity that is used only during the authentication process. When clients attach to a public port, they receive an IP address from a DHCP server running on the "non-connected" VLAN. Users must then authenticate themselves to a server on the VLAN using their Kerberos password. Once authentication is complete, the server sends an SNMP message to the VLAN switch signaling it to move the port to an "attached" network with full connectivity. When clients disconnect from a port (link status drops), the VLAN switch moves the port back to the "non-connected" VLAN.

NetBar is a much more expensive solution than SPINACH, because it requires the use of a special brand of VLAN switch. Many existing Ethernet installations do not use VLAN switches, and retrofitting a building or college campus to use VLAN switches would be time-consuming and costly.

Furthermore, the NetBar solution guards against many but not all attempts at hardware address spoofing. A malicious user could place a hub between the public port and the VLAN switch. Users would unwittingly connect to the public port and authenticate themselves to gain network access. When they disconnect, the hub would prevent the VLAN from detecting the link status drop, thus enabling the malicious user to spoof the host's hardware address and gain connectivity. In our opinion, the substantial cost of the special VLAN hardware is not justified by the small amount of extra security it gains over SPINACH. With SPINACH, we want to achieve the most security possible at a low cost with minimal hardware and software requirements.

UC Berkeley has proposed a design [18] that requires both special software on the client and special hardware. In particular, this design requires the use of an enhanced DHCP client, an intelligent hub that supports disabling and enabling of individual ports, and a DHCP server that has been modified to hand out configuration information only to clients that have gone through an authentication process. To support all possible client platforms, administrators would need to support multiple versions of the enhanced DHCP client software. Given the ever-increasing workload of administrators, this is not a realistic suggestion. Disabling and enabling network ports is more secure than hardware address filtering, but requires manufacturers to design hubs that specifically handle this. Negotiating with manufacturers could be difficult if not impossible, which further makes the Berkeley design difficult to deploy in a variety of environments.

Peter Honeyman at the University of Michigan has also proposed a design called InSite [11] that addresses network access control through public ports. This design requires the use of NetBar VLANs and custom software on the client, thereby making it vulnerable to the same kinds of deployment problems that we described for the CMU and Berkeley designs.

We prefer SPINACH to the CMU, Berkeley, and University of Michigan designs, because it is user-friendly, does not require any expensive hardware or special client software, and provides a good measure of security for the cost and administrative effort required to keep it running. Moreover, unlike these other designs, SPINACH is modular, enabling us to plug in a different access verification scheme easily, should the need for a higher level of security arise. This enables SPINACH to satisfy an organization's security needs as they evolve over time. Finally, SPINACH is designed to run on a variety of networks, including wireless LANs and WANs as long as these have the notion of a hardware address. Wireless LANs that behave similarly to wired Ethernet (e.g. Lucent's WaveLan) can use SPINACH without any modification. The other designs are rigid in this respect, because they are tied to special hardware.

VIII. Future Work

A. Guest Users

We plan to add support to SPINACH for temporary, guest users. We envision faculty and staff members using a web-based interface to generate one-time passwords for their visitors. This is slightly different from the previous version of SPINACH which required a small number of system administrators to telnet into the SPINACH machine to generate one-time passwords for all guests. We want to minimize the administrative effort required to maintain and deliver the SPINACH service. We can achieve this by distributing the load and by placing building residents in charge of enabling the SPINACH service for their guests.

B. Authorization

We plan to extend the authorization module so that it differentiates between groups of users. For example, we may want to allow all authenticated Computer Science graduate students to have access, but to disallow all Humanities undergraduates from obtaining access. This introduces several challenges.

As the user base of SPINACH constantly changes, it is impossible for us to maintain a list of people who fulfill the specified criteria and should be granted access. Instead, the SPINACH Authorization server must retrieve the information about the user (e.g. what his major or status is) in real-time from the Stanford directory service. Once the information has been retrieved, a set of rules are applied to it and the decision to grant access is made.

The problem with this scheme is that users will not want all of their private information to be publicly available, but might agree to allow the authorization module to obtain a subset of this information to make its decision. A user might allow a specific (but not any) SPINACH router to know he is living in a certain student residence, but might choose to keep this information hidden from the public. How to give the user fine grain control over such information is still an open research issue. The upcoming P3P standard [13] is a promising first step towards privacy control. The proposed standards for authenticated LDAP [10] or LDAP over SSL address how requests for this type of information can be handled in a secure way.

The rules to perform the authorization check can be performed in a number of ways. For most applications, simple pattern matching on the attributes returned from the organization's database server should be sufficient.

C. Authentication

We also envision adding more authentication policies. We will extend SPINACH so that it allows users to be authenticated with authentication modules of other institutions. For example, SPINACH could support users authenticated with the Carnegie Mellon security infrastructure, in addition to the Stanford Kerberos infrastructure.

IX. Conclusions

We are facing a trend towards ubiquitous connectivity. As part of the MosquitoNet project [4], [5], [12], we believe in enabling ubiquitous connectivity, that is connectivity that can be achieved "anytime, anywhere." Users will want to have network access regardless of whether they are in their offices, in public building spaces, or in open spaces outside. The problem of network access control through public ports will become more prevalent as institutions construct new buildings or retrofit existing ones to reflect this trend.

Current solutions to the problem of secure network access through public ports present a wide range of obstacles: they are expensive; they require constant administration; they require special hardware or custom client software; or they are not user-friendly. Although SPINACH is not as secure as some other systems, it is low-cost, low-maintenance, and scalable. It is user-friendly, providing a web-based interface, and it requires no special hardware or custom client software. These features make SPINACH a viable solution for use in a variety of environments, such as public schools, libraries, and universities, where reasonably secure network access is desired, but keeping cost and maintenance effort at a minimum is essential.

X. Acknowledgments

We would like to acknowledge the help of several people who made SPINACH possible. Dan Boneh gave valuable input on the security aspects of SPINACH. Elliot Poger and Stuart Cheshire provided the original SPINACH idea and Elliot designed and implemented the first prototype. Brian Roberts was of great help setting up the system in our building. Finally we would like to thank Jeff Lewis and Tim Torgenrud for their support in integrating SPINACH with WebLogin, and Jeff Hodges for his advice on the LDAP infrastructure and future directions.

This research has been supported by a gift from NTT Mobile Communications Network, Inc. (NTT DoCoMo), a grant from the Keio Research Institute at SFC, Keio University and the Information-technology Promotion Agency in Japan, as well as a Sloan Foundation Faculty Fellowship and a National Science Foundation Graduate Fellowship.

References

[1] Ip security protocol. http://www.ietf.org/html.charters/ipsec-charter.html.
[2] Stanford university web authentication. https://weblogin.stanford.edu/.
[3] Wavelan, by lucent technologies, inc. http://www.wavelan.com/.

[4] Mary G. Baker, Xinhua Zhao, Stuart Cheshire, and Jonathan Stone. Supporting mobility in mosquitonet. In *Proceedings of the 1996 USENIX Technical Conference*, January 1996.

[5] Stuart Cheshire and Mary Baker. Internet mobility 4x4. In *Proceedings of SIGCOMM'96*, August 1996.

[6] Dorothy Denning. *Cryptography and Data Security*. Addison-Wesley Publishing, Inc., 1982.

[7] Ralph Droms. Dynamic host configuration protocol - rfc2131, 1997. ftp://ftp.isi.edu/in-notes/rfc2131.txt.

[8] Taher Elgamal, Sean Cotter, and the Netscape Security Team. Netscape security: Open-standard solutions for the enterprise, 1998. http://developer.netscape.com/docs/manuals/security/scwp/.

[9] Neil M. Haller. The s/key one-time password system. In *The Internet Society's 4th Annual Networking Conference*, 1994. ftp://ftp.bellcore.com/pub/nmh/docs/ISOC.symp.ps.

[10] Jeff Hodges. Lightweight directory access protocol (v3), extension for transport layer security, 1997. Work in Progress, IETF: draft-ietf-asid-ldapv3-tls-02.tx.

[11] Peter Honeyman. Workstation authorization. USITS conference, Work in Progress Report, See http://www.citi.umich.edu/ u/ honey/ ppt/ insite/ index.htm, 1997.

[12] Kevin Lai, Mema Roussopoulos, Diane Tang, Xinhua Zhao, and Mary Baker. Experiences with a mobile testbed. In *Proceedings of the Second International Conference on Worldwide Computing and its Applications*, March 1998.

[13] Steve Lucas, Joseph Reagle, et al. Platform for privacy preferences, the p3p project, 1998. http://www.w3.org/P3P.

[14] Erikas Napjus. Netbar: Carnegie mellon access for mobile machines. http://www.net.cmu.edu/design/netbar.html.

[15] Elliot Poger and Mary Baker. Secure public internet access handler (spinach). In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, 1997.

[16] J.G. Steiner, C. Neuman, and J.I. Schiller. Kerberos: An authentication service for open network systems. In *USENIX Winter Conference Proceedings*, 1988.

[17] Dwayne Virnau. Stanford distributed computing consulting, 1998. http://dcc.stanford.edu/.

[18] D. L. Wasley. Authenticating aperiodic connections to the campus network. *ConneXions, Volume 10, No. 8, pp 20-26*, 1996.