Popeye - Using Fine-grained Network Access Control to Support Mobile Users and Protect Intranet Hosts

Mike Chen
mikechen@cs.berkeley.edu

Barbara Hohlt
hohltb@cs.berkeley.edu

Tal Lavian
tlavian@cs.berkeley.edu

11 December, 2000

Abstract

We are facing a trend towards ubiquitous connectivity where users demand access at anytime, anywhere. This has lead to the deployment of public network ports and wireless networks. Current solutions to network access control are inflexible and only provide all-or-nothing access.

It is also increasing important to protect Intranet hosts from other mobile and static hosts on the same Intranet, in order to contain damages in the case that a host gets compromised.

We present an architecture that addresses these issues by using a programmable router to provide dynamic fine-grained network access control. The Java-enabled router dynamically generates and enforces access control rules using policies and user profiles as input, reducing administrative overhead. Our modular design integrates well with existing authentication and directory servers, further reducing admininstrative costs. Our prototype is implemented using Nortel's Accelar router and moves users to VLANs with the appropriate access privilege.

1 Introduction

Many organizations, such as UC Berkeley, would like to provide ubiquitous connectivity to its users as well as visitors. In order to deploy such connectivity, access control is required so that only authorized users have access to network resources. If unauthorized users gain access to the network, they could cause damage by abusing resources or attacking machines on the Intranet, they can also launch attacks against systems outside the organization. Current solutions to secure network access provide only all-or-nothing access, and often have high administrative costs.

Network access control is more than just protecting against unauthorized mobile hosts. Even static hosts on an Intranet may pose as a security threat. Take the recent Microsoft break-in [**?**] as an example: a host on the Microsoft Intranet was compromised through an email virus, then it was used to attack other hosts on the Intranet. This attack exploits the fact that most Intranet are fully reachable by all Intranet hosts[1]. The damage of this type of break-in would have been greatly reduced if a network access control system were in place on every port within the organisation to limit the networks that a host can reach.

A fine-grained and flexible network access control system with low administrative costs is clearly desirable, but today's solutions fall short in one or more areas.

The following are important properties that a network access control system should have:

Flexible and fine-grained access control It should be able to prevent source spoofing and support destination filtering. It should also support per-user and per-application policies.

Modular design It is important to be able to integrate well with existing services such as RADIUS and Kerberos for authentication and LDAP for user profiles.

Easy to manage Administrative cost is a major concern in any organization. The access control policy must be easy to specify and get right. Administrators should be able to specify system-wide, group-wide, and user-specific policies using some high level language, and have the system configure the components and enfore the policies automatically. To further lower administrative cost, different modules should be able to be managed by different adminis-

[1] Microsoft's Intranet was manually partitioned so that highly sensitive data such as source code was on a network unreachable from the compromised host. This type of manual partitioning is inflexible and does not support mobile hosts.

trators.

Support for mobile user It should dynamically configure network ports to give the right network access to the user.

High performance The access control system must be able to scale up to support many users with little performance impact.

Easy to use The system must be easy to use to be adopted by users. It should present familiar UI to the user and require little or no special software.

In this paper, we describe the design and implementation of Popeye, a network access control system. In Section 2, we describe related work. In Section 3, we discuss the design principles and architecture components. In Section 4, we discuss our prototype implementation. In Section 5, we evaluate our system using our design goals. In Section 6, we discuss our initial results and future works. In Section 7, we state our conclusions.

2 Related Work

SPINACH [8] provides access control on public network ports to allow only authorized users onto the network. Users authenticate themselves using kerberos-enabled telnet. The SPINACH router filters everything except DHCP traffic (so all users can get an IP address), SPINACH server traffic (so all users can authenticate), and authorized user traffic. SPINACH II [5] extends the work to use web-based authentication. It is deployed in the CS department of Stanford, and has been integrated with an existing authentication server.

SPINACH II's strengths are low cost, easy to use, and modular design. However, it only providies all-or-nothing network access control. A SPINACH user, including visitors, either gets full access to the network or no access at all. Second, SPINACH II filters packets based on MAC address which is prone to MAC address spoofing attacks, especially on ethernet networks where ARP reveals MAC addresses of other hosts. As a result, SPINACH II is useful in settings where network access control is necessary but not critical. Third, SPINACH II is software based and does not scale as well as hardware solutions.

Carnegie Melon's AuthNet [9] is similar to SPINACH in using MAC address filtering to allow only authenticated user traffic onto the campus network. A user registers his MAC address with a centralized server, and the Cisco VLAN switch filters everything but the packets with the registered MAC addresses. All "authenticated" users appear on the same VLAN even though they may be connected to different VLAN switches. It also offers all-or-nothing access to the network and suffers from the same MAC spoofing attacks.

Carnegie Melon's NetBar system [7] uses a Cisco Catalyst VLAN switch to isolate all the public ports on a "un-authenticated" VLAN with limited connectivity to the authentication server and the DHCP server. Once the user obtains an IP address through DHCP, the user authenticate themselves using their Kerberos password. Once authenticated, the server sends an SNMP message to the VLAN switch to move the port to a VLAN with full connectivity. When the client disconnects from a port, the VLAN switch detects the drop in link status, and moves the port back to the "un-authenticated" VLAN. NetBar prevents MAC address spoofing attacks but it also only provides all-or-nothing network access control.

InSite's [6] design requires the use of NetBar VLANs and custom software on the client. To support all possible client platforms, administrators would need to support multiple versions of the client software. The administrative cost and support overhead of the design makes it unrealistic for deployment.

UC Berkeley has proposed a design [10] that requires special DHCP software on the clients, intelligent hubs that can turn ports on and off, and a modified DHCP server that only hands out configuration to authenticated clients. The custom software requirement makes it vulnerable to the same kinds of deployment problem as InSite.

All five solutions discussed above offer all-or-nothing access control, and they don't offer protection between hosts on the Intranet. They may be suitable for university campuses where secure network access is not critical and the requirement is to reduce unauthorized access to the network. For organizations that need stronger access control, we prefer Popeye to the Stanford, CMU, UCB, and U of Michigan designs.

In the above related work the additional functionality was done by software base routers. We can differentiate our work by using the capabilities of hardware base routing. This allows us have wire speed, fine grain filtering, and scalability. Figure 1 shows the separation of the control plane and forwarding plane in hardware based routers. This is the difference of combining forwarding and control in software based

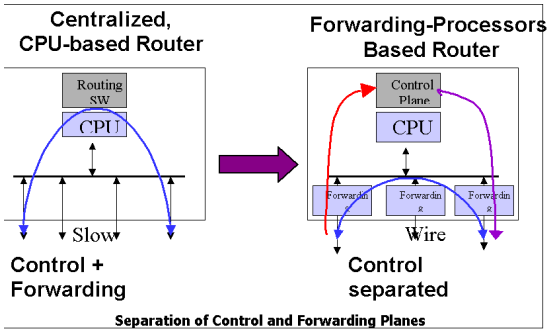

Figure 1: Software-Based and Hardware-Based Routing

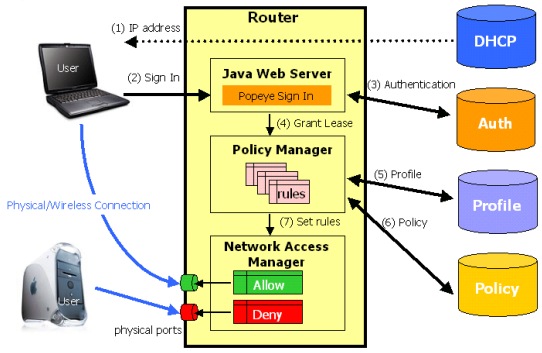

Figure 2: Popeye Architecture

routing. In software based routing all packets are processed by the main cpu. This puts some load on the performance and the capabilities of previously related work. Our approach results in enhancing the forwarding capabilities in about two orders of magnitude and the number of ports in about one order of magnitude. In our approach the filtering is done by the hardware without additional load on the cpu. The cpu can be idle while the router is doing the above work in wire speed. This allows scalability and fine grain filtering that was limited in previous work.

3 Design

This section describes the design of Popeye, a network access control system. Popeye has a modular design comprised of five components which are a mix of our own Popeye services and existing services. Figure 2 illustrates the architecture with the five components; Java Web Server, Policy Manager, Network Access Manager, external Security services, and an external DHCP server.

The overall design approach is to run our custom Popeye services on a programmable router, and to run the existing services on host machines. We choose a programmable router solution for its high performance and the ability to do flexible, fine-grained access control. This includes access control at the IP, protocol and applications layers. In our design, we assume all ports are directly connected to a programmable router.

For our design to be effective, it must be easy to use. We want to allow a visitor network access, but at the same time we want to protect the organisation's Intranet. Below we present a scenario which illustrate our expectations from a user's perspective.

3.1 A Usage Scenario

A visitor comes to Berkely and wants to connect to the network. He connects his machine to a physical port or uses a wireless NIC card and gets an IP address through DHCP. He then points his web browser to a specific web site and is asked to enter a username and password. Whereupon, the system gives him the right permission to use the network. The type of permission can be no access, access to the internet, or access to certain security domains within the organisation.

3.2 Web Server

Because it is a simple way to access our system, we decided to use a web interface for the mobile clients. However, as we will explain later, we need to protect the visitor's IP address. The solution is to have the web server run directly on the Accelar as an HttpServlet. The Web Server validates the user with the Authentication Server, records the physcial port and MAC address of the user, and grants a lease to the user. The lease must be periodically renewed by the user.

3.3 Policy Manager

The Policy Manager simplifies the task of authorization. It presents a high level language which makes it easy for administrators to specify policies and expresses these policies as rules to be manipulated by the low level Network Access Manager.

The Policy Manager takes a user profile from the Profile Server and a policy from the Policy Server as input, and generates the security rules for a particular user. The rules are then passed to the Network Access Manager for enforcement.

3.4 Network Access Manager

The Network Access Manager dynamically enforces the security rules it receives from the Policy Manager. This is a low level service which is able to configure the router, set packet filters, and set QoS parameters on demand.

One of the features we implemented, discussed in Section 4, is dynamic control of virtual lans, vlans. Even though, many vlans may be on the same router, no packets are allowed to cross the vlan boundaries. This is supported directly by the hardware. This can be viewed as several networks that are not attached, even though they are on the same router.

IP filtering, another feature discussed in Section 4, can also be dynamically controlled by the Network Access Manager. With dynamic IP filtering we can modify a user's access on the fly. For example, we can block the access of a specific source, destination, or application.

3.5 Security Services

Popeye uses existing services to manage authorization, user profiles, and security policies. This allows for a modular design and separation of privilege[sec], as the different security services can be managed by different security domains.

3.5.1 Authentication Server

The Authentication Server is an external service which validates users. The Web Server queries the Authentication Server to validate users requesting network access. RADIUS or Kerberos could be used for this service.

3.5.2 Profile Server

The Profile Server is an external service which stores user profiles. The Policy Manager queries the Profile Server for instances of user profiles. LDAP could be used for this service.

3.5.3 The Policy Server

The Policy Server is an external service that stores the policies of security domains. The Policy Manager queries the Policy Server for policies which match user profiles.

3.6 DHCP Server

The DHCP Server is used to give a visitor an IP address in order to use the network. It can be configured to issue a particular range of IP addresses on specific subnets. In Section 4 we discuss our use of DHCP in more detail.

4 Implementation

This section describes a prototype implementation of Popeye. The key technology we use is the Oplet Runtime Environment (ORE) from Nortel Networks OpenetLab [2].

Our prototype is implemented using a Nortel Networks Accelar 1100-B router, a Linux server, and the research software from Nortel Networks OpenetLabs. ORE is a Java-based software platform for deploying services on network elements such as routers, switches and hosts. ORE runs on the network element as a Java virtual machine and enables services, called oplets, to be dynamically downloaded to to the network element. These oplets can provide new functionality, such as traffic monitoring and intrusion detection. Since ORE can run on a host, oplets are developed on a host machine and then downloaded to the desired network element.

For our project, Popeye, ORE runs on the Accelar and oplets are developed on a Linux server. The Accelar is booted from an image on the Linux server using tftp. Oplets are installed from an http server running on the Linux server. The Linux server has two network interfaces: one to the Berkeley Intranet and one to the Accelar.

Our goal as described above is to enforce per-user access control to support mobile users and protect the Intranet from outsiders. We focus our efforts on differentiating ourselves from previous research: mainly the dynamic hardware configuration. We bypass the authentication step. For policy management we assume a single policy, as a proof of concept, which is to grant all visitors access to the Internet only. We

also assume an environment where mobile clients are directly connected to a physical port on the Accelar.

4.1 Web Server

The web server is implemented as a Java HttpServlet [3] oplet running on ORE. The advantage of running the Web Server on the Accelar is that we can get the visitor IP address directly from the client and keep it secret. As we will show later, the IP address is used to identify what physical port a visitor is connected to. The Popeye servlet serves a web page which requests a user's name and password. It then calls the Network Access Manager.

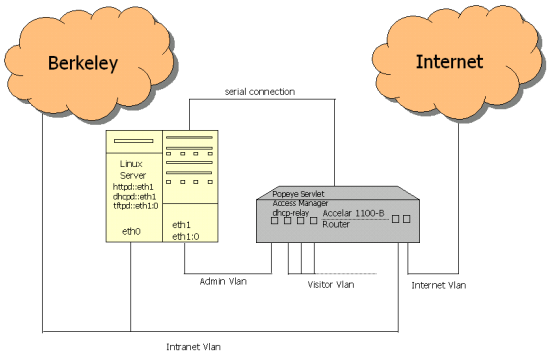

Figure 3: Popeye Setup

4.2 Network Access Manager

We considered two approaches for implementing the Network Access Manager; IP packet filtering and vlan access control.

4.2.1 IP Packet Filtering

The Java Forwarding API, JFwd, is a low-level ORE service, which allows customized oplets to alter routing and forwarding behaviors by accessing the hardware instrumentation. It includes a number of generic service mappings such as MAC address, ARP, IP routing, IP filters, IP Diffserv and VLAN (Virtual LAN). A typical use of the JFwd API is to instruct the forwarding engine to alter packet processing through the installation of IP filters.

Early in the project, we tried to use the ORE JFwd service to do packet filtering by application and protocol. This would allow us to distinguish the applications the user may have access to and provide a security mechanism on the type of applications we want to allow. For example, visitors might be prevented from using a Napster application , but allowed to use the http protocol. Unfortunately, we discovered that the JFwd service was not working properly in the latest version of the Accelar so, we abandoned that approach and pursued another approach.

Figure 4: MIB Tables

4.2.2 Vlan Approach

The Accelar supports virtual local area networks, vlans. Vlan is a way to separate a single physical lan into several virtual networks. We configured the

Accelar to have four vlans: Administration, Visitor, Intranet and Internet vlan. Except for the Administration vlan, routing is not allowed between vlans. In this way we partition the network into security domains. No packets, other than configuration and those setting vlans, are allowed by the hardware to move between vlans.

Figure 3 shows our configuration. The dhcpd, httpd, and tftpd servers are on the Administration vlan. One port of the Intranet vlan is connected to the Berkeley Intranet. One port of the Internet vlan is connected to the Internet. All the remaining ports are connected to the Visitor vlan.

When a user initially signs in to Popeye, they are physically connected to a port on the Visitor vlan. In this approach we want to move a guest user from the Visitor vlan to the Internet vlan. This is done by accessing the Management Information Base, MIB, on the Accelar. The MIB is a collection of managed objects that together form a virtual information store. It is made available through a set of generic APIs exported by ORE called JMIB.

Figure 4 shows some of the virtual tables available from the JMIB. There are two ways we know of to change the vlan of a port. One is to update the vlanIds attribute of the Port table and the other is to update the portMembers attribute of the Vlan table. However, to update either of these tables we need to discern which port our guest is connected to. The ifIndex attribute of the NetToMedia table stores the port in its two right most bytes. From the ip address we received from the http connection we can index into the NetToMedia table and get the port number.

4.3 DHCP Server

We configured a DHCP server on the Linux host to serve several networks from the separated vlans. We connected the dhcp server to the Administration vlan and we set a dhcp-relay between the Visitor, Intranet and Internet vlans, to the Administration vlan. This design allows us to assign different subnets to the different vlans. In the initiation of a new mobile user, the dhcp server assigns an ip address from the Visitor vlan subnet. In the second stage, after the visitor is authenticated and his port has been moved to the destination vlan (in our example the Internet vlan) he will get a new ip address. The DHCP lease is set really short on the Visitor subnet, so the client's ip address will automatically change.

5 Evaluation

5.1 Measurements

We measured the vlan setup latency with calls to java.lang.System.currentTimeMillis() [4]. The latency of the vlan setup is 2.0 seconds.

We measured bandwidth with the Iperf [1] tool, by sending udp packets between vlans. Actual packets are routed at wire speed, switched 100Mbps.

The Accelar supports up to 384 physical ports.

5.2 JMIB and JFWD

To access the low level instrumentation of the router we are using the JMIB, a Java API for the MIB variables. JMIB is based on the published MIB definition of the router. The advantage of doing so is that we can get easy access from the application running on the router to the device instrumentations. However, this is slow. In our measurement it takes two seconds to dynamically configure a physical port from one vlan to another vlan. This is on the slow path, and for our application, two seconds is OK. Using JMIB might not be acceptable for applications that need better response (e.g. reading a value 100 times a second). For fast access monitoring we will need to bypass the JMIB API and add low level optimizations which access low level C wrappers and JNI access to the router hardware.

JFwd is defined as an API to access the forwarding engine. JFWD performs mapping for underline hardware functionality like MAC addresses, ARP, IP routing, IP filters, IP DiffServ, and VLANs. The forwarding engine performs at wire speed with no latency. Some monitoring needs to be done in wire speed, or at least sampling in high frequencies. As a result, some control features need to be optimized. JMIB as an underline API for JFWD might not be acceptable to some applications. In these cases we will need to optimize JFWD to have direct access to the forwarding engine and bypass the JMIB.

6 Discussion and Future Work

Popeye also supports MAC address based access control for wireless clients. Since wireless LANs are essentially a shared ethernet, MAC address spoofing

may be a problem. For organizations such as universities that desire some access control but don't require strong security, Popeye is perfect for the task. For organizations that require strong security, one approach may be to put the wireless LAN on an outside network and require users to use Virtual Private Network (VPN), such as IPSec, to get back into the Intranet. This approach has its own problems in that the hosts on the WLAN may still be compromised through the wireless interface while connected to the VPN, allowing an attacker to hop to other machines on the Intranet. Popeye can compliment such an approach by containing the damage.

Another approach is to have base stations perform user authentication and prevent MAC spoofing. This can be achieved through the use of per-user keys rather than network keys and is supported by 802.11 vendors such as Lucent and Ericsson. When coupled with Popeye, the approach provides as good as security as wired networks.

For future work, we believe the policy specification deserves more research. We plan to design (or use an existing) policy language and build tools that simplify policy specification. Perhaps conflict resolution is important as conflicting policies may need to be specified. For example, a policy may specify that no customer service representitives, other than the manager of customer service, may have access to the product developement hosts. Our intuition is that the more specific policy overrides more general ones. We plan to look into packet filtering policy languages and tools and reuse as much as possible. We also plan to implement the Policy Manager that can interpret policies and generate rules.

As we discovered during the Popeye implementation, the ORE service, JFwd, which implements IP filtering, does not work properly on the latest version of the Accelar router. We have reported this problem to Nortel and hope to incorporate IP filtering into our prototype once it becomes available. We also hope to add support for RADIUS as the authentication server and LDAP as the profile server.

Additionally, we hope to deploy it on the UC Berkeley campus-wide wireless network. The deployment will allow us to validate the modular design and study the performance of our system.

Last, maybe Nortel will turn this into a commercial product one day.

7 Conclusion

Many of today's organisations use firewalls to protect their Intranets from outside attacks. Within the Intranet, although assets are protected through authentication and permissions, attackers still have access to the local area network and can try to attack. Frequently, visitors are not able to have Internet access with their mobile computers for company security reasons.

We propose a mechanism which would provide protection within the organization. It would block attackers within from attempting attacks by inhibitting their ability to send packets to the network. We would also would like to allow visitors to safely access (from the organisation's perspective) the network and Internet.

This will be done by dynamic IP filtering and dynamic vlan configuration. We can add fine-grain QoS to applications, users, and destinations. Dynamic network access control allows for more flexibility and innovation on the type of services that can be provided. The fact that new features and services can be added to the router on the fly, allows for flexibility, customization, and innovation.

In essence we want to provide not only an external firewall, but an internal firewall for each subnet on the network. Programmable routers will provide filtering mechanisms to enable this type of security. For instance, the hardware can filter by source, destination, port, and protocol. An access matrix can be built for any combination of accesses between subnets.

Currently organisations protect themselves from the outside world by firewalls on the edge of their organisations. However, they are not protected from the inside. The Popeye prototype demonstrates a way to provide partial firewall functionality on every port inside an organisation.

References

[1] Iperf. http://dast.nlanr.net/Projects/Iperf/index.html.

[2] Nortel networks openetlab. http://www.openetlab.org .

[3] Sun httpservlet. http://java.sun.com/products/servlet/.

[4] Sun java.lang.system.currenttimemillis(). http://java.sun.com/products/jdk/1.1/docs/api/java.lang.Syster

[5] M. Roussopoulos G. Appenzeller and Mary Baker. User-friendly access control for public network ports. In *Proceedings of IEEE INFOCOM*, 1999.

[6] Peter Honeyman. Workstation authorization. http://www.citi.umich.edu/u/honey/talks/insite/, 1997.

[7] Erikas Aras Napjus. Netbar - carnegie mellon's solution to authenticated access for mobile machines. http://www.net.cmu.edu/docs/arch/netbar.html, 1997.

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

[9] Ryan Troll. http://www.citi.umich.edu/u/honey/talks/insite/. http://www.citi.umich.edu/u/honey/talks/insite/, May 1998.

[10] D. L. Wasley. Authenticating aperiodic connections to the campus network, 1996.