

Authentication Gateway

Steve Brewer
Biology Computer Resource Center

Goals

- Provide “real” security
- Minimize client configuration
- Minimize administrative overhead

- Use a simple enough system that we can understand it in detail.

These were our goals for a wireless deployment. Basically we want something simple, that doesn't require a lot of effort on either the part of the administrators or users.

the problem

- wireless is coming
- allows anonymous access (potentially from outside building)
- WEP is broken
- WEP doesn't provide by-user, loggable, authentication anyway

These are the problems as we see them.

Gateway Overview

- Gateway with 2 interfaces
 - firewall between interfaces
 - running dhcpd and named
- Client fires up wireless interface
 - interface configured via DHCP
- Client authenticates as an individual
 - gateway adds rule(s) to route client packets
- Client surfs the web (or whatever)

This is basically what we envisioned going to the process.

NoCatAuth -- overview

- Two hosts (or one): gateway and authserver
- gateway runs perl script as root -- manipulates iptables ruleset
- authserv runs httpd with real certificate and complex mod-perl cgi.
- GPG used for secure communication between gateway and authserv.

We looked at two systems. NoCatAuth and the Authentication Gateway. I describe them a bit and then contrast them.

NoCatAuth -- user experience

- User fires up interface
 - gateway configures interface via DHCP
- User uses web-browser to connect to any host
- Gateway redirects connection to authserv URL
- User authenticates
- Browser opens pop-up window that will periodically refresh itself, preventing connection from timing out.

The user experience is good because no configuration or specialized software is needed on the client. The client can just fire up a web-browser, they are captured and authenticated, and then directed to where they started going. Very slick.

Authentication Gateway -- overview

- One host (or two): gateway
- gateway listens for ssh connection
- small PAM module adds rule to firewall
- PAM can authenticate against any PAM-friendly authentication source
 - local password file
 - LDAP
 - NIS
 - NIS+

Authentication Gateway -- user experience

- User fires up interface
 - gateway configures interface via DHCP
- User connects to gateway via ssh
- User authenticates
- Connection active until ssh session closed.

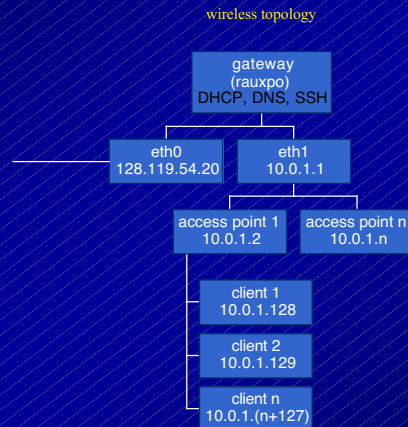
The Authentication Gateway requires ssh on the client and requires the client to know the hostname to connect to.

Comparison

- NoCatAuth
 - Pros
 - no client configuration
 - no specialized software needed
 - filtering on hw address
 - Cons
 - needs gateway to run as root
 - needs real https certificate
 - complex mod-perl
 - one-off authentication database
 - designed for 'open' networks -- must be adapted to our purpose
- Authentication Gateway
 - Pros
 - very simple
 - no root process needed
 - no certificate needed
 - uses standard authentication
 - Cons
 - filters on IP
 - requires client to know hostname
 - requires client to have ssh client
 - requires gateway to offer shells to connecting hosts

Basically, we fell on the side of the Authentication Gateway: We liked the extremely simplicity of the system. It does require a bit more configuration on the client side, but that may not be entirely bad. Not much is required and we want to have that much oversight over the system anyway (that people have to talk to us at least once, anyway).

our environment



This is sketch of the environment we envision constructing. Basically, we're talking about building a parallel physical network that will go into one interface of the gateway. Then we have one place where wireless users get IP addresses and are authenticated. We probably have the infrastructure to do this in our facility.

configuration of access points

- no encryption
- no access lists
- bridging on
 - ip
 - appletalk

The access points are set up with minimal configurations. We still don't quite have appletalk working though, darn it.

initial iptables rules

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -A INPUT -i eth0 -m state --state NEW, INVALID -j DROP
iptables -A FORWARD -i eth0 -m state --state NEW, INVALID -j DROP
iptables -I FORWARD -o eth0 -j DROP
iptables -I FORWARD -s 10.0.1.0/24 -d 10.0.1.1 -j ACCEPT
```

** Note: I had to separate the lines containing NEW, INVALID to two lines each. Go figure.

These are the initial iptables rules as described in the Authentication Gateway HOWTO. I had to make a few adjustments to make it work.

pam_iptables.c and /etc/pam.d/sshd

```
static int insert_rule(char *rhost)
{
    char *args[] = { "iptables", "-I", "FORWARD", "-s", rhost, "-o", interface, "-j", "ACCEPT", NULL };
    return exec_iptables(args);
}

#%PAM-1.0
auth    required    /lib/security/pam_stack.so service=system-auth
auth    required    /lib/security/pam_nologin.so
account required    /lib/security/pam_stack.so service=system-auth
password required    /lib/security/pam_stack.so service=system-auth
session required    /lib/security/pam_stack.so service=system-auth
#this line is added for firewall rule insertion upon login
session required    /lib/security/pam_iptables.so debug
session optional    /lib/security/pam_console.so
```

I pulled out this snippet from the PAM module code to show what the rule looks like that gets inserted when one authenticates. Its pretty simple stuff.

The line in the PAM configuration code invokes the pam_iptables module and writes the rule into the firewall upon authentication.

URLS

http://www.itlab.musc.edu/~nathan/authentication_gateway/
<http://nocat.net/>

Here are a couple of useful URLs.

thoughts

- added entries for DHCP hosts in /etc/hosts to speed up ssh connections
- should probably use restricted shells for connections
- should probably run snort on the gateway to keep a handle on what's happening

These are a few implementation details that I've been thinking about. Other than the first, I haven't done them yet. They all seem like good ideas to me, though.

Questions?