

The Router Is The Firewall Part 4

Implementing IOS Authentication Proxy

Michael J. Martin

In Aug 2003, we discussed the concept of Active Network Access Control (ANAC). The Cisco IOS has two mechanisms for implementing ANAC: Lock & Key ACLs and Authentication Proxy. In part one we covered the more generic of the two, Lock & Key (L&K). This month we will focus on Authentication Proxy (AP).

L&K and AP Differences

While both L&K and AP are similar functionally, there are three key differences that may make one preferable to your environment. For starters, L&K is available on all IOS versions, while AP is only available on the IOS firewall feature set. The IOS firewall feature set costs more than the standard "IP Plus" IOS. So if cost an issue, L&K is probably a more attractive option.

The second factor is granularity of control. With L&K, the access policy is applied universally and is limited to a single access rule. So if you just want to authenticate access, then L&K is appropriate. With AP, each user or group can be configured to use a custom access-control policy allowing administrators the ability to control access to the host, network and service/protocol level. This makes AP well suited for controlling access to multi-service/multi-subnet networks. AP affords the fine control needed to implement a customized access policy at a centralized access point. Implementing L&K in this kind of environment would only afford user authentication -- without any access control.

The third factor is the trigger mechanism. L&K utilizes SSH or Telnet to provide a directed authentication interface. In order to access the secure network, a user must connect to the router and authenticate to facilitate access. AP, on the other hand, requires an HTTP request to be forwarded through the router. This has the advantage of operating passively, but requires that the user generate "appropriate" traffic to trigger authentication. If you are implementing AP as a means of outbound access control, this requirement should not present a problem (for most environments), as HTTP requests often make up the majority of "off-network" requests. AP works best with Java-enabled browsers (Cisco tests for compatibility with IE and Netscape only) but there is also support for non Java-enabled browsers. With Java disabled, the user needs to force a "reload" of the Web page after a successful authentication. This is a minor consequence in comparison to potential security issues when JavaScript is enabled. But it's there, and users need to be made aware.

A common issue with implementing AP is DNS failures that hang-up the HTTP trigger request. Before any HTTP request can be made, a DNS query is also required. If you host DNS locally (and have installed access rules to allow the server to make external queries), this should not be a problem. In the case that your hosts use externally hosted DNS then you need to make accommodations for DNS. There are two ways to handle this. The first is to allow external DNS queries without authentication. (Make sure you implement CBAC, otherwise you will need leave to UDP wide open. That's not a good idea if you're trying to be secure.) The other method is to utilize static NAT. Create an inside-to-outside static translation. Hosts can then send queries to the locally accessible translated address and the router will proxy the requests.

No Security Solution Is Perfect, There Is Always A Downside

Just to get started on the right foot, both methods have a core flaw you must be aware of. Both L&K and AP are vulnerable to IP spoofing attacks. There are two types of IP spoofing attacks you need to account for. The first one, I am sad to say, you cannot really do anything about. Any IP address-based security policy can be subverted if the attacker can masquerade as an IP address that is permitted or hijack an already established session. Both attacks require the attacker to have direct access to the wire, which makes them difficult, but not impossible, to pull off once access to the connection path is available. The valid host can be disabled utilizing a DOS attack or system-specific exploit, and the attacker's host can assume its IP identity. This is a risk inherent in IP-based security, and you need to make a risk-versus-reward call on this before implementing L&K or AP.

The Router Is The Firewall Part 4 Implementing IOS Authentication Proxy

Michael J. Martin

Once you have decided to implement L&K or AP, there is another type of IP spoofing "problem" you need to address. This one you can do something about. Once a "hole" is open, it is open for some period of time, and there is no further authentication required during this time span. Since access is based on IP, if a user opens a hole from a multi-user system, anyone on that system will have the same level of access afforded to the legitimate user. There is also the potential in DHCP environments for a legitimate host to "leave" the network and have a new host be granted a pre-authorized IP. Once again, these are exploits that are hard to pull off, but they are risks nonetheless. However, with this class of problem there are protective measures that can be taken.

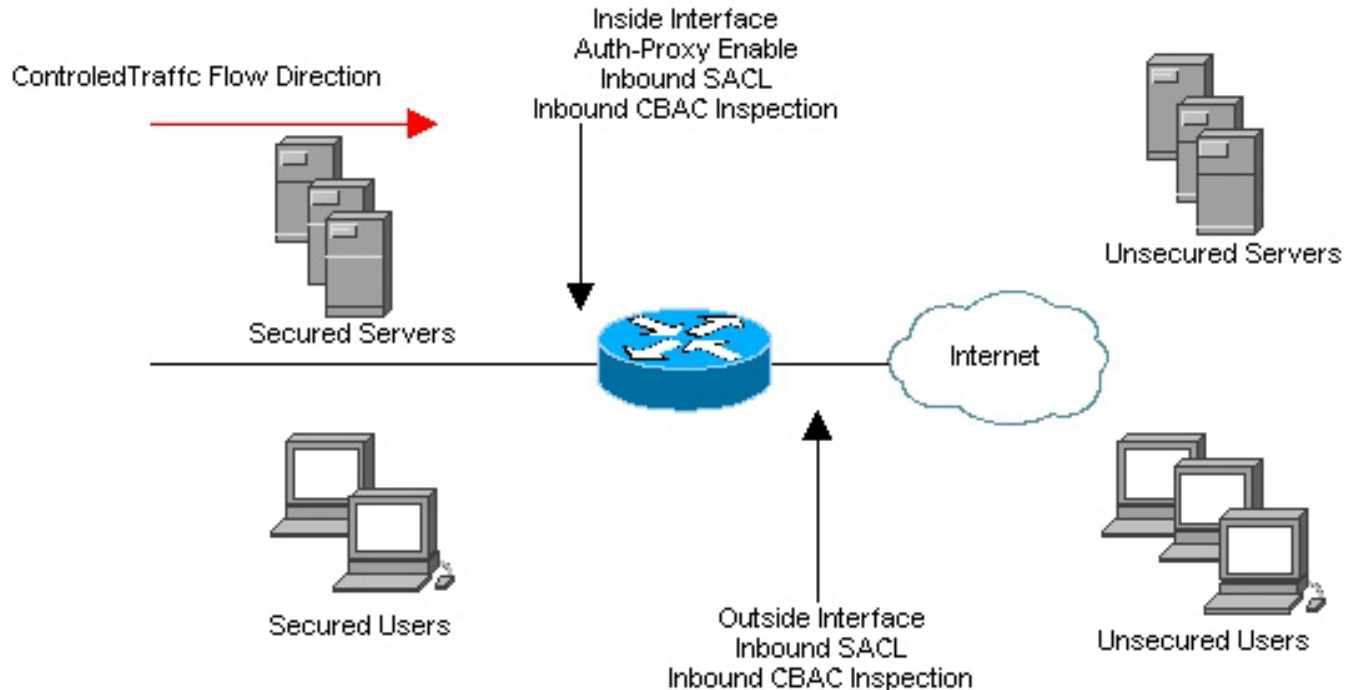
For starters, set session timeouts for short access intervals. It may be a pain to have to re-authenticate, but it minimizes the risk. In DHCP environments, utilize "static address assignments." This is a little complicated and more administrative and user effort is required, but it's a more secure approach to utilizing DHCP. (Security does not come without a price.)

AP Implementation Models

There are two Authentication Proxy ANAC implementation models: Outbound Proxy and Inbound Proxy.

The Outbound Proxy access-control model provides control for users on the secured network accessing unsecured systems.

Secure to Unsecure Active Access Control



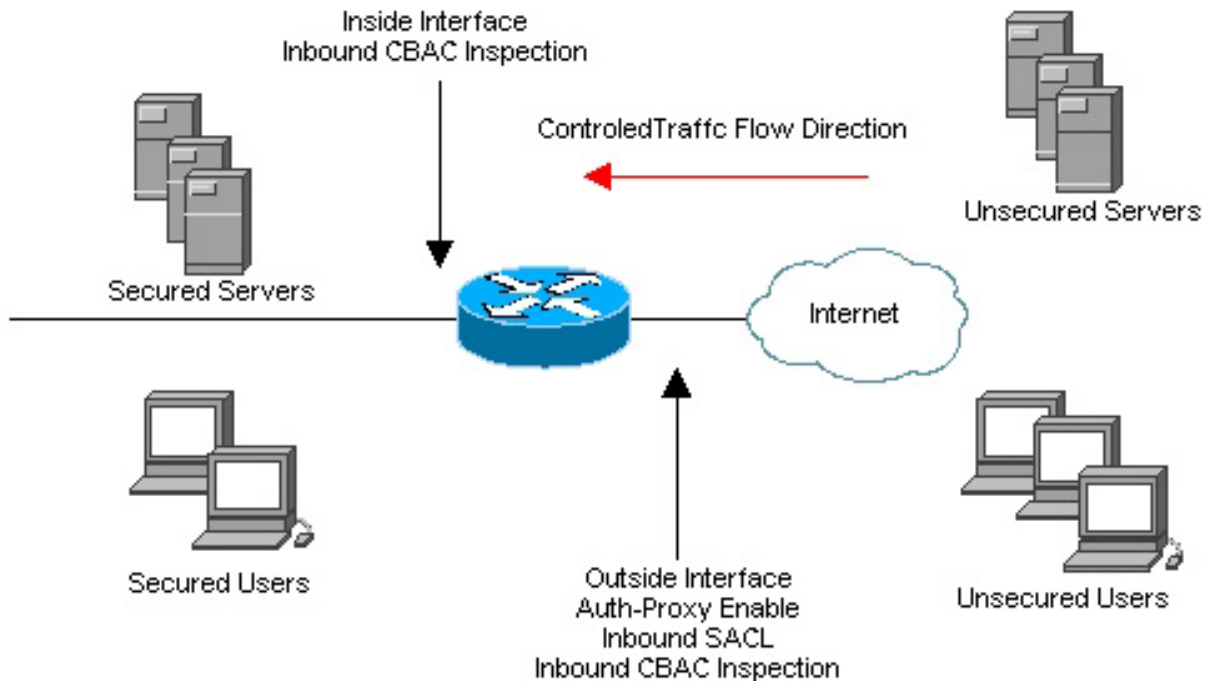
This application of AP is ideal for environments where there is an additional need to "tighten up" an outbound-only firewall policy. Remote office and secure-segment firewall applications are often designed to allow full outbound access for "secured users" and to simply deny inbound access. By adding AP to this scenario, outbound access can be tailored to meet individual user needs, furthering the security posture of the segment by controlling the service activities of "secured" users.

The Router Is The Firewall Part 4 Implementing IOS Authentication Proxy

Michael J. Martin

The Inbound Proxy access-control model provides for unsecured users to access secured systems.

Unsecure to Secure Access Control



This model is by far the more "traditional" application of AP. Users need access to one or more "protected" services. From an administrative perspective, a good strategy for implementing this kind of control is to create "groups" that define different levels of access. Then assign users to the group that provides the access they require. For example, a company provides access to e-mail, intranet Web site and HTTP proxy server over the Internet. All employees get access to mail and the HTTP proxy, but only the sales staff gets access to the intranet site. Why provide HTTP proxy server access? Because the company does not want its users' Web activity tracked when they're on the road, and the cache speeds up access to commonly accessed sites when using dial-up. On top of the user access, administrators in the company will want access to the routers and servers. To support these different access requirements, we would create different TACACS groups. (We will get into the TACACS definitions in the next section.)

- GenAccess provides access to pop3 (port 110), smtp (port 25) and http-proxy (port 8080)
- SaleAccess provides access to pop3 (port 110), smtp (port 25) and http-proxy (port 8080) and https (443)
- AdminAccess provides access to pop3 (port 110), smtp (port 25) and http-proxy (port 8080), https (port 443), ssh (port 22) and Windows Terminal Service (ports 3389 & 1493)

The Router Is The Firewall Part 4

Implementing IOS Authentication Proxy

Michael J. Martin

Once the TACACS groups are configured, we can just place the user in the correct group. Here is a TAC_PLUS user definition example:

```
user = zach {  
    login = cleartext "apassword"  
    member = admin  
}
```

- Looking at the service model diagrams, it becomes clear from an implementation perspective that the differences between the two AP implementation models has to do with the placement of the following:
- SACL(s) defining (both static and dynamic via AP) the services/ports permitted to be established through (i.e., in) the interface.

CBAC inspection policy that permits the return traffic, allowed by the SACL and the dynamic AP rules. The Authentication Proxy listener service, which inspects for trigger traffic to prompt the user for authentication.

Of course, along with the configuration there is an assumption that you have a functioning and accessible TACACS+ server -- it is an essential, but sometimes overlooked component of AP. Though it seems obvious, remember to allow communication between the router and the TACACS+ server when creating your base filtering SACLS. If the router and TACACS server cannot communicate, AP will not work. This fact also needs to be kept in mind in terms of service. If there is no TACACS server, users will not be able to access services. If you currently have only a single TACACS server in production, prior to implementing AP you should consider the virtues of server redundancy.

Configuring Authentication Proxy

Regardless of the implementation model, there are five configuration steps:

- Configuring AAA new model authentication and authorization
- Configuring the TACACS server users and group profiles
- Configuring SACLS/IP access groups for the router interfaces
- Configuring the CBAC inspection policy(s)
- Configuring Authentication Proxy on the appropriate trigger interface.

Configuring AAA New-Model Authentication And Authorization

AAA is an operational requirement for AP to work. The dynamic access control list (DACL) statements are downloaded from the TACACS (or RADIUS if you wish) server. No AAA server, no AP. Hence the earlier comment on the need for TACACS server redundancy. The AP AAA configuration involves the following:

```
Enable AAA New Model  
outland-fw(config)# aaa new-model
```

Enable the DEFAULT named list authentication method. A note of caution here: The DEFAULT list applies to all VTY/TTY interfaces that do not have an authentication method explicitly defined, because AAA does not work without TACACS support. Adding local authentication will provide you a backdoor entry if the TACACS server fails, but it will not provide backup for AP. Rather than adding local as a secondary method, create an ADMIN named authentication list that utilizes local authentication (you will also need to create a local user). Install it on the console, AUX ports and a

The Router Is The Firewall Part 4

Implementing IOS Authentication Proxy

Michael J. Martin

VTY rotary group (Telnet to the router through port 3001 to access the VTY) so you can access the router in the event that the TACACS server is down or unreachable.

```
outland-fw(config)# aaa authentication login default group tacacs+
outland-fw(config)# aaa authentication login ADMIN local
outland-fw(config)#line console 0
outland-fw(config-line)#login authentication ADMIN
outland-fw(config)#line aux 0
outland-fw(config-line)#login authentication ADMIN
outland-fw(config-line)#line vty 5
outland-fw(config-line)#rotary 1
outland-fw(config-line)#login authentication ADMIN
Enable exec and auth-proxy authorization.
```

```
outland-fw(config)# aaa authorization auth-proxy default group tacacs+
outland-fw(config)# aaa authorization exec default group tacacs+
```

Define the TACACS server, server key and message source interface (remember to have a primary and secondary server).

```
outland-fw(config)# tacacs-server host 172.30.1.69
outland-fw(config)# tacacs-server host 172.30.1.169
outland-fw(config)# ip tacacs source-interface Ethernet1/0
outland-fw(config)# tacacs-server key secretkey
```

Configuring The TACACS Server Users And Group Profiles

Once the AAA configuration is complete, we need to create the TACACS user or group access policies. An access policy is defined using the root attribute/value pair {service = auth-proxy}. There are two parameter A/V pairs associated with the auth-proxy root pair, {priv-lvl = <0-15>} and {proxyacl#n=<permit acl statement>}. (The n value represents the statement number, i.e., 1,2,3, etc.) Working with the Inbound Proxy model we discussed above, here is what the tac_plus configuration looks like:

```
# Shared Key
key = secretkey
# Accounting File Location
accounting file = /var/tmp/acctfile

##### Users

user = admin-user {
    login = cleartext "password"
    member = AdminAccess
}

user = sales-user {
    login = cleartext "password"
    member = SaleAccess
}
```

The Router Is The Firewall Part 4 Implementing IOS Authentication Proxy

Michael J. Martin

```
user = gen-user {
    login = cleartext "password"
    member = GenAccess
}

##### Groups

group = AdminAccess {

service = exec {
    priv-lvl=15
}
# service=exec AV value defines the privilege level the users logs in as
# for EXEC TTY or
# the http server

service = auth-proxy
{
priv-lvl=15
proxyacl#1="permit tcp any any eq 110"
proxyacl#2="permit tcp any any eq 25"
proxyacl#3="permit tcp any any eq 443"
proxyacl#4="permit tcp any any eq 8080"
proxyacl#5="permit tcp any any eq 22"
proxyacl#6="permit tcp any any eq 3389"
proxyacl#7="permit tcp any any eq 1493"
proxyacl#8="permit udp any any eq 53"
}
}

group = SaleAccess {

service = auth-proxy
{
priv-lvl=15
proxyacl#1="permit tcp any any eq 110"
proxyacl#2="permit tcp any any eq 25"
proxyacl#3="permit tcp any any eq 443"
proxyacl#4="permit tcp any any eq 8080"
proxyacl#5="permit udp any any eq 53"
}
}

group = GenAccess {

service = auth-proxy
{
priv-lvl=15
proxyacl#1="permit tcp any any eq 110"
proxyacl#2="permit tcp any any eq 25"
proxyacl#3="permit tcp any any eq 8080"
```

The Router Is The Firewall Part 4

Implementing IOS Authentication Proxy

Michael J. Martin

```
proxyacl#4="permit udp any any eq 53"  
}  
}
```

Alternatively, if we were configuring an Outbound Proxy configuration, the `tac_plus` configuration could look like this:

```
# Shared Key  
key = secretkey  
# Accounting File Location  
accounting file = /var/tmp/acctfile  
  
##### Users  
  
user = gen-user {  
    login = cleartext "password"  
    member = GenAccess  
}  
  
group = GenAccess {  
  
service = auth-proxy  
{  
priv-lvl=15  
proxyacl#1="permit tcp any any eq 22"  
proxyacl#1="permit tcp any any eq 80"  
proxyacl#1="permit tcp any any eq 443"  
proxyacl#2="permit udp any any eq 53"  
proxyacl#3="permit icmp any any eq echo"  
}  
}
```

This simple Outbound Proxy example allows any user assigned to the GenAccess group to utilize SSH to connect to external hosts, query external DNS servers, surf the Web and ping Internet hosts. It's simple, but you get the point.

Configuring Sacs And IP Access Groups For The Router Interfaces

Both the Inbound and Outbound Proxy models rely on dynamic ACL entries regulating the conversations that pass through the router. There are two sources for these dynamic ACL rules. The primary source, which dictates what conversations can be initiated through the router, are the ACL entries stored on the AAA server. Counter to these rules are the "return path" statements added by CBAC. Although the majority of access filtering decisions are handled dynamically, SACLs installed as IP access groups must be installed on appropriate router interfaces in order to provide placeholders to prefix the dynamic ACL rules.

The number and placement of SACLs and the rules they should contain depends on the AP model you're implementing, what core services you need to allow to support AP operation (i.e., TACACS & DNS), and what you wish to allow without requiring any authentication.

At a minimum, the IP access-group SACL must have a deny ip any any statement. However, if you are permitting services such as SSH, DNS or TACACS, you can use explicit permit statements for these services, and rely on the ACL's implicit deny action to block everything else. This approach

The Router Is The Firewall Part 4 Implementing IOS Authentication Proxy

Michael J. Martin

works because the static and dynamic ACL rules are interpreted using a specific hierarchy. The IOS attempts to optimize static ACL statements when they are entered. So in some cases the way you enter ACL statements is not the way the router interprets them when performing matches. (To see the ACL ordering that the router is processing matches against, use the exec command <show ip access-list>.) When AP and CBAC are inactive on the router, the ACL rule hierarchy followed is (from top to bottom): AP Entries, CBAC entries, Static Entries. Here is a <show ip access-list> command output example from an Inbound Proxy AP configuration:

```
outland-fw# sh ip access-lists
Extended IP access list public-access
permit icmp host 172.30.71.5 any
permit udp host 172.30.71.5 any (30 matches)
permit tcp host 172.30.71.5 any (239 matches)
permit tcp host 207.44.192.2 eq www host 172.30.1.69 eq 2777 (1 match)
permit tcp host 207.44.192.2 eq www host 172.30.1.69 eq 2773 (1 match)
permit tcp host 64.5.48.152 eq www host 172.30.1.69 eq 2775 (1 match)
permit tcp host 64.5.48.152 eq www host 172.30.1.69 eq 2774 (1 match)
permit tcp host 64.5.48.152 eq www host 172.30.1.69 eq 2772 (1 match)
permit tcp host 64.5.48.152 eq www host 172.30.1.69 eq 2771 (1 match)
permit tcp host 64.5.48.152 eq www host 172.30.1.69 eq 2769 (1 match)
permit tcp any host 172.30.71.99 eq 22
outland-fw#
```

In the above example, the IP extended access list has a single permit statement:

```
ip access-list extended public-access
permit tcp any host 172.30.71.99 eq 22
```

This allows the router to be accessed without AP authentication (a good idea). The AP rules are appended to the top of the ACL and they will always appear at the top of the list. This is done so they can override any static or CBAC statement that may conflict with them. (They are noted in red in the output example.) The CBAC rules are prefixed to the AP rules or appended to the static rules, depending on how you want to look at it. This allows the CBAC rules to overrule any static statement conflicts. (The CBAC statements are in green in the output example.) The static ACL rules are at the end (in black in the example).

As mentioned above, the number of SACL and IP access groups needed depends on the AP model. The Outbound Proxy model requires an inbound SACL on the inside (secure) and outside (insecure) interfaces. In order for AP to work, the router needs to be able to access the TACACS server and the hosts need to access DNS. Here is an example SACL for the inside interface in an Outbound Proxy implementation:

```
ip access-list extended ip-inside-inbound
permit udp any host 192.168.100.1 eq domain
permit tcp any host 172.30.71.1 eq 3001
permit tcp any host 172.30.71.1 eq 22
permit tcp host 172.30.71.20 eq tacacs host 172.30.71.1
permit tcp host 172.30.71.21 eq tacacs host 172.30.71.1
```

This ACL allows SSH and Telnet via rotary 1 (3001) administrative access to the router along with TACACS and DNS queries to an external DNS server. All other traffic is discarded. The outside

The Router Is The Firewall Part 4

Implementing IOS Authentication Proxy

Michael J. Martin

interface SACL options are even simpler. If administrative access from unsecured hosts is required, then the SACL needs to permit VTY access ports, like the private SACL example above, in which we need to permit SSH and Telnet to rotary 1.

```
ip access-list extended ip-public-inbound
permit tcp any any eq 22
permit tcp any any eq 3001
```

If no AP unauthenticated administrative access is permitted, all that is needed is:

```
ip access-list extended ip-public-inbound
deny ip any any
```

The Inbound Proxy model only requires an SACL be installed on the public interface. Its configuration matches that of the Outbound Proxy public SACL.

```
ip access-list extended op-public-inbound
permit tcp any any eq 22
permit tcp any any eq 3001
or
ip access-list extended op-public-inbound
deny ip any any
```

Configuring The CBAC Inspection Policy

The CBAC inspection rules are determined by the services permitted by the AP profiles and unauthenticated services defined in the IP access group SACLs. Let's take a quick look at the one of our AP policy examples:

```
service = auth-proxy
{
priv-lvl=15
proxyacl#1="permit tcp any any eq 110"
proxyacl#2="permit tcp any any eq 25"
proxyacl#3="permit tcp any any eq 443"
proxyacl#4="permit tcp any any eq 8080"
proxyacl#5="permit tcp any any eq 22"
proxyacl#6="permit tcp any any eq 3389"
proxyacl#7="permit tcp any any eq 1493"
proxyacl#8="permit udp any any eq 53"
}
}
```

The AdminAccess policy from our Inbound Proxy model example supports access to the following: POP3, SMTP, SSH, Windows Terminal Services, HTTP, HTTPS, DNS lookups, and the HTTP proxy server. All of these services utilize a single TCP or UDP conversation session. So, at a minimum, we need to enable basic TCP and UDP inspection.

```
outland-fw(config)#ip inspect name AP-Policy tcp timeout 300
outland-fw(config)#ip inspect name AP-Policy udp timeout 120
```

The Router Is The Firewall Part 4 Implementing IOS Authentication Proxy

Michael J. Martin

Of course, since we are also allowing SMTP, it's a good idea to enable CBAC's application-specific inspection for additional protection:

```
outland-fw(config)#ip inspect name AP-Policy smtp timeout 300
```

Once the CBAC policy is created, inbound traffic inspection needs to be configured on both the inside and outside interfaces. This is required for both of the AP models discussed here.

```
outland-fw(config)#int FastEthernet 0/0
outland-fw(config-if)#ip inspect AP-Policy in
```

This ensures that the appropriate "return path" rules are added to the correct SACL when conversations are established. Since CBAC inspection is required on both interfaces, you could consider creating a CBAC policy for the inside and outside interfaces. This, however, is not a requirement. A single CBAC policy can be used, even if the two interfaces have different service access requirements, as long as the CBAC policy meets the inspection requirements of both interfaces. There are some slight advantages to using two policies in terms of debugging and tuning. It is also more secure to use two policies -- service access requirements are different between the two (this applies for the Outbound Proxy model only).

Configuring Authentication Proxy On The Appropriate Trigger Interface

Once the IP access groups have been installed and CBAC inspection has been enabled, the final step is to configure and enable AP. The first task is to set the AP global inactivity timer. If no user activity (traffic) is detected before the timer expires, the user's policy is revoked and corresponding dynamic ACL statements are deleted. This is set with the global configuration command <ip auth-proxy auth-cache-time {1-35791 min}>:

```
outland-fw(config)#ip auth-proxy auth-cache-time 10
```

To avoid hang-ups with dynamic ACL statements applied by CBAC, the AP global inactivity timer should be higher than the CBAC inspect-rule inactivity timers. This will insure that CBAC clears any open channel paths before the AP policy is revoked.

The next step (this is optional) is to set the banner message that will be displayed on the AP login page when the user is prompted to authenticate. This is a great place to tell your users and potential intruders your access policy and the potential penalties they may face. The auth-proxy banner is set using the global configuration command <ip auth-proxy auth-proxy-banner {Fin Char}>:

```
outland-fw(config)#ip auth-proxy auth-proxy-banner ~
Enter TEXT message. End with the character '~'.
!!!!!!!!!!!!!!!!!!!!Private Computer System!!!!!!!!!!!!!!!!!!!!
```

```
You have accessed a Private Computer System. This
site is intended to be used by authorized personnel
for viewing and retrieving information. Unauthorized
attempts to upload information or change information
on this system is strictly prohibited and may be
Punishable under the Computer Fraud and Abuse Act
of 1986.
```

```
All access is monitored and usage of this system
```

The Router Is The Firewall Part 4 Implementing IOS Authentication Proxy

Michael J. Martin

is audited. All persons are hereby notified that use of this system constitutes consent to monitoring and auditing.

To report a security incident involving this system please call the Network Operations Center at 800-CALlNOC or send e-mail to noc@outland.net

!!

~

```
outland-fw(config)#
```

The banner message precedes the Username and Password dialog. It also precedes the challenge response window, which requires the user acknowledge the success of failure of the authentication attempt. So a long message (like this one) may require the user to scroll down the page to reach the prompt fields (which can annoy some users). This is a user interface call, but one that should be kept in mind.

Authentication Proxy rules are configured with an administrator-defined rule name. The named rule is then installed in accordance to the proxy model that is being imposed, which is to say the source of the traffic flow that is inspected, which in turn triggers the authentication challenge. AP rules are defined using the global configuration command `<ip auth-proxy name <rule name> http {auth-cache-time 1-35791 min} {list ACL 1-199}>`. AP lets you create two types of rules: those that will be applied globally, and those that will be applied specifically. Let's take a look at two examples, starting with the global rule:

```
outland-fw(config)#ip auth-proxy name global-ap http
```

This AP rule, as the name implies, will require all users who attempt to open an HTTP session to authenticate. Here is an example of a specific rule:

```
outland-fw(config)# ip auth-proxy name ap-applied http auth-cache-time 60 list 20
```

With this rule, only hosts listed in the standard access list 20 will be prompted for authentication (the auth-cache-time allows you to set a rule-specific inactivity timer). All other users will be restricted to only the services defined in the SACL that governs the IP access group associated with the AP trigger interface. This option affords administrators a little more control, allowing the creation of a "general access" policy defined by an SACL (with CBAC opening the return conversation path). Then AP can be utilized to permit specific services to individuals or groups.

Finally, with the AP rule defined, all that is left is to enable AP on a router interface. This brings back up the whole source of the traffic flow point again. When implementing the Inbound Proxy model -- AP is enabled on the inside interface -- regulating the flow of traffic generated by users on the "secure" network segment(s). Using the Outbound Proxy model, inspection is enabled on the outside interface, restricting the access "unsecured" users have to access hosts on the secure network. In both cases, the interface configuration command is `<ip auth-proxy {name}>`:

```
outland-fw(config-if)#ip auth-proxy global-ap
```

The Router Is The Firewall Part 4

Implementing IOS Authentication Proxy

Michael J. Martin

Troubleshooting And Monitoring AP

So you are done. You have implemented AP. You feel more secure. Your boss will feel more secure. The users, well, they are more secure. There are a few commands you will need from time to time to "clean up" issues.

- `<show ip access-lists>` - We covered it above, but it's worth mentioning again. This command will allow you to see the ACL that the router is matching against.
- `<show ip auth-proxy cache>` - This command will list the hosts/users that have (or are establishing) policies. The cache data includes the source IP, source port over which the authentication was established, and the idle-timeout for the rule.
- `<show ip inspect sessions>` - This command will list all of the CBAC sessions being tracked by the router and their state. The data provided includes the src host/port dst host/port, protocol and connection state.
- `<clear ip auth-proxy cache {* | A.B.C.D}>` - This command purges either a specific user (A.B.C.D) or the entire cache (*).

If you really get stuck, there are also debug commands for `<ip auth-proxy>` and `<ip inspect>`. Use these with caution on production routers. You will be flooded with data if the router is at all active. If it comes time to debug, duplicate your production implementation in the lab (the configuration at least) and test. You will find it far more manageable and productive to actually look at the data than be drowned in it.