

Configuring CBAC Policy Configuration and CBAC Implementation

Michael J. Martin

The August installment of the "Router is the firewall" series provided an overview of the three-step CBAC configuration process, along with the first step, "Traffic Qualification." This month we will focus on step two: policy configuration and CBAC implementation on IOS FFS capable routers.

Policy Configuration

CBAC can be utilized as an edge (public/private transit point) or embedded (private/private transit point) firewall. In the August article, we covered strategies for determining the active services we need to allow for in the access policy. As we move on to step two, we are going to create a CBAC policy and configure a Cisco router to serve as an edge firewall. Implementing CBAC as an edge or embedded device follows the same three-step process. However, when implementing CBAC as an edge firewall to serve as a public/private transit device, there is the possibility that CBAC may need to be deployed with Network/Port Address Translation (N/PAT). This is required to utilize RFC1918 for private network addressing. CBAC is fully interpretable with PAT and NAT and can be overlaid over existing implementations, provided that accommodations in the access policy account for the address translation rules.

Creating Service Flow Diagrams

In the last installment of the CBAC series, we focused on network traffic qualification. If you try to implement CBAC -- or any other firewall, for that matter without qualifying the network traffic, you will most likely have some really upset users. The importance of traffic qualification cannot be over-emphasized. That said, once you have collected your service utilization data, the first step of policy creation is to create inbound and outbound service flow diagrams.

Why? CBAC opens the ports needed to host-to-host conversations to take place when they are first initiated. CBAC inspection, however, only operates after the connection has been completely established. But, in order for CBAC to operate at all, the conversation needs to be permitted. The SACL component of CBAC provides the facility to define the conversations that can be established. The traffic qualification process identifies the services you need to permit. The flow diagrams provide the "map" so you can visualize the data flows that the hosts need to function. Along with providing network visualization, the diagrams also provide:

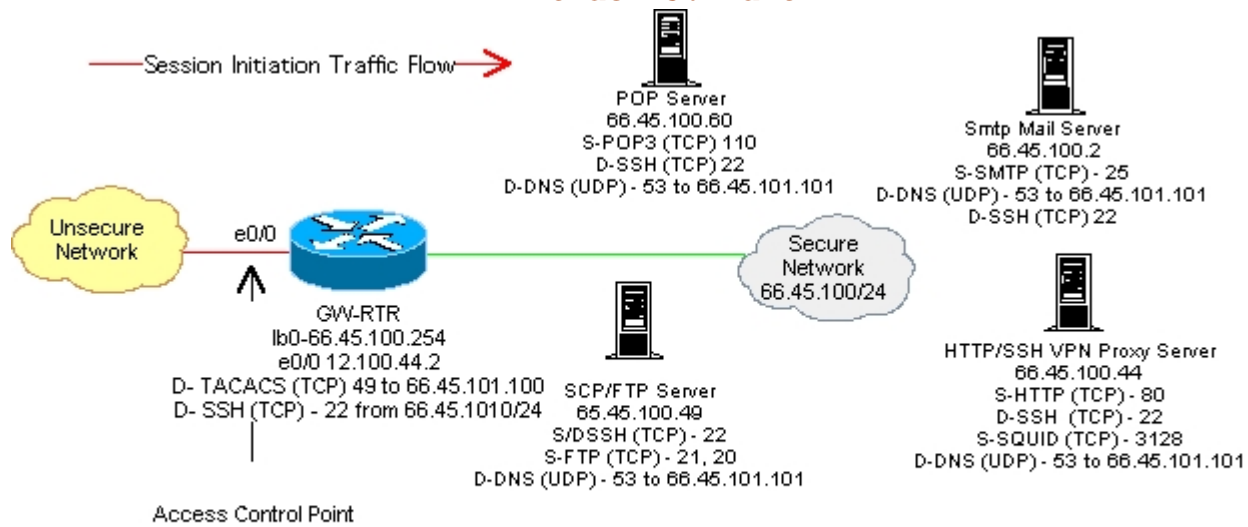
- Information: The diagrams serve as result report of the traffic qualification process and the information resource for generating the access policy rules.
- Documentation: For troubleshooting access-rule problems and perhaps more importantly they provide others the documentation to understand what you were thinking when you created the rule set.
- Ethnicity results: During the testing and tuning stage the maps detail what services should actually be visible to unsecured users.

Example Flow Diagrams

Our example network is (of course) simple. The secure network is a single Class C subnet containing user hosts and six services hosts. Four of the server hosts provide services that are accessible from outside of the secure network. One (AAA server) is only accessible locally and the other (DNS/DHCP) has a secondary dependency to external servers. Here is the inbound service flow diagram for our example network:

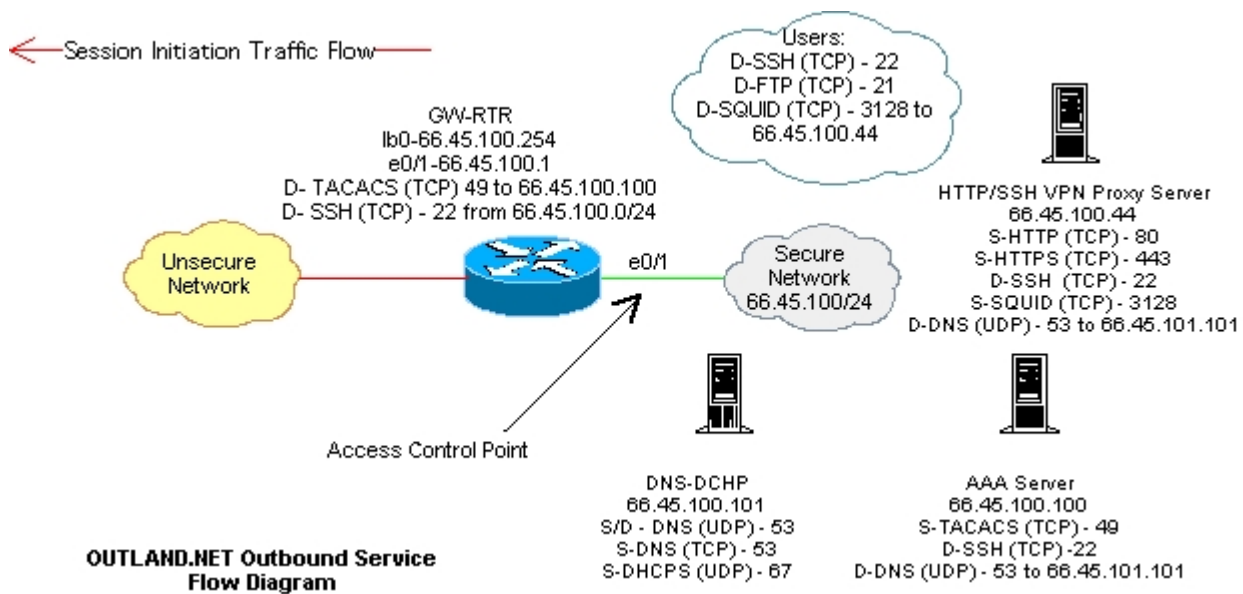
Configuring CBAC Policy Configuration and CBAC Implementation

Michael J. Martin



OUTLAND.NET Inbound Service Flow Diagram

Here is the outbound service flow diagram



OUTLAND.NET Outbound Service Flow Diagram

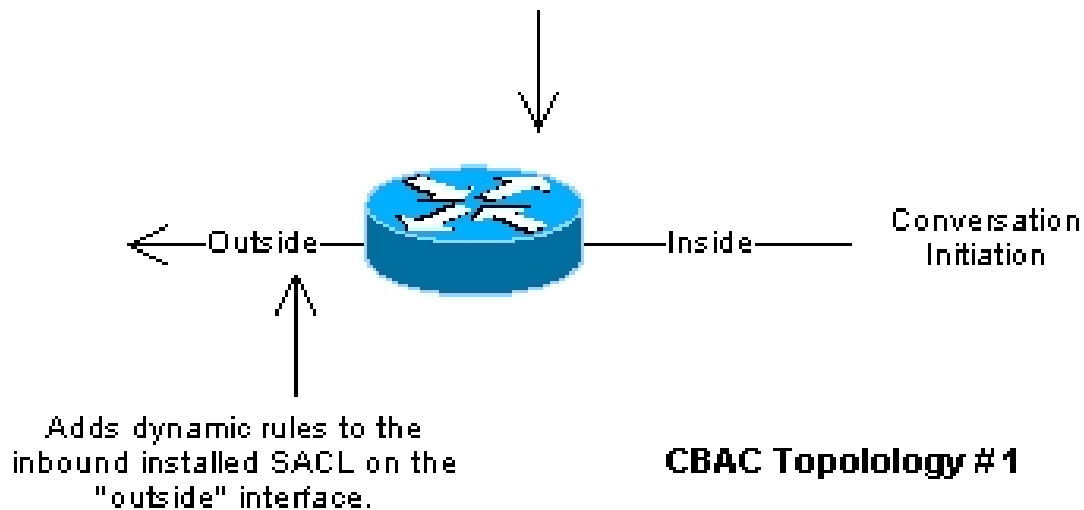
The diagrams contain the host IP address along with the host's service offerings (indicated by an S) and service primary and secondary dependencies (indicated by a D). The router's CBAC ip access-group rules will reflect those dependencies by allowing access to the hosts and service ports.

About "host dependencies" -- hosts connected to a network do not operate in a vacuum. Often, they have dependencies on other hosts in order to serve their function. These dependencies can be categorized into two classes: primary and secondary. A primary dependency is a communication exchange between two hosts that is essential in order for the host to perform its primary function. An example of this is a Web server that functions as a front-end to a DBMS server. If the Web server and DBMS server cannot exchange data, then there is no purpose in having the Web server. A secondary dependency is a communication exchange that, while important, its absence would not render the

Configuring CBAC Policy Configuration and CBAC Implementation

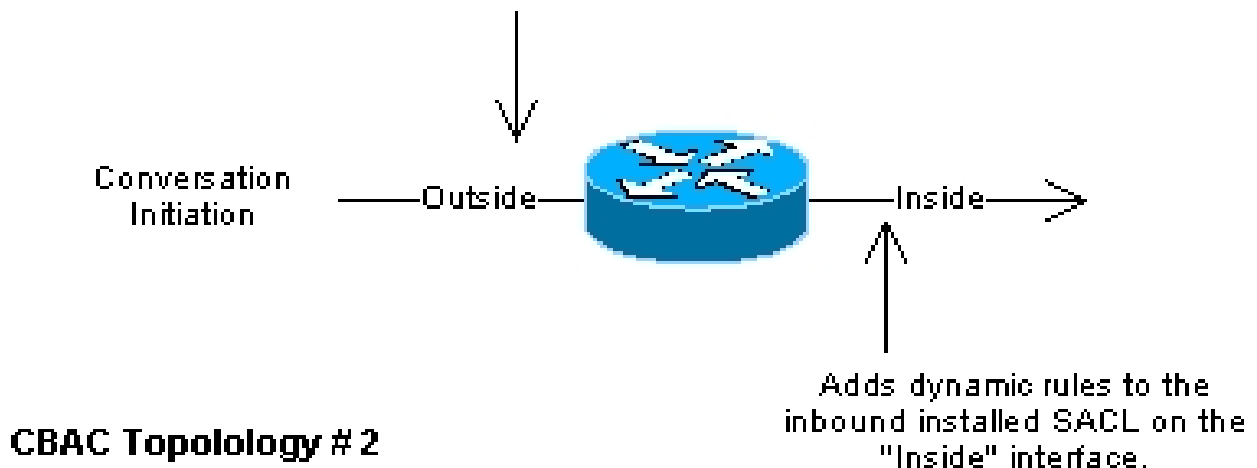
Michael J. Martin

CBAC Inspection Installed inbound on the "inside" Interface



CBAC inspection on outside interface with access-control on the inside interface

CBAC Inspection inbound Installed on the "Outside" Interface

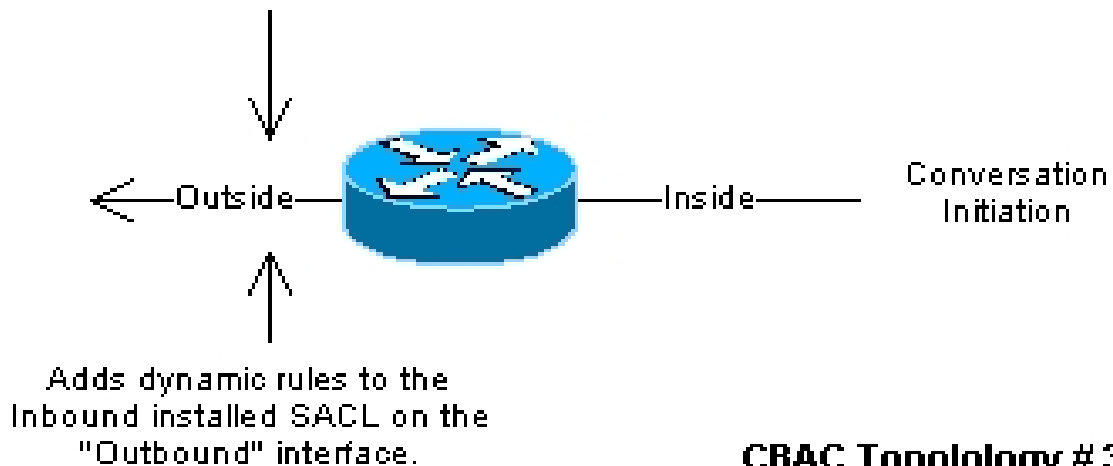


CBAC inspection and access control on the outside interface

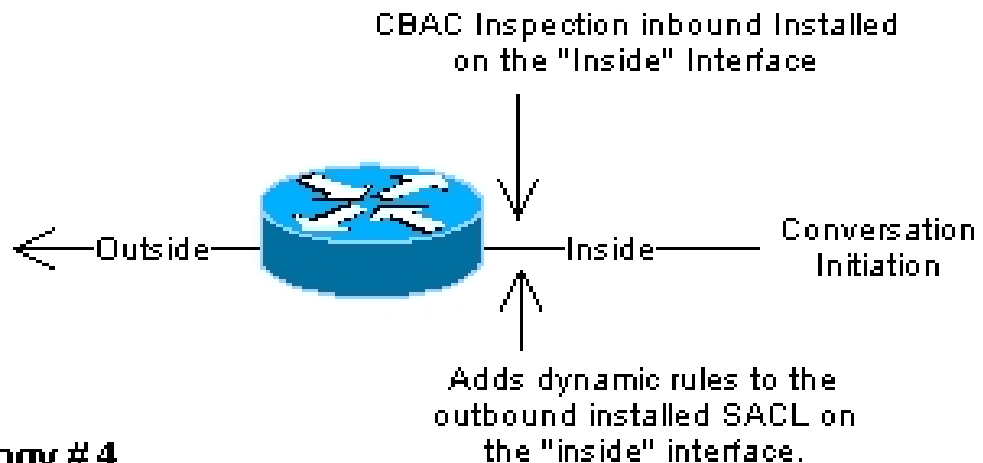
Configuring CBAC Policy Configuration and CBAC Implementation

Michael J. Martin

CBAC Inspection outbound
Installed on the "Outside" Interface



CBAC inspection and access control on the inside interface



Our example environment requires private-to-public and public-to-private conversations. So a combination of topologies one and two is a good mix. It supports the access model needed with limited complexity and will be easy to troubleshoot.

SACL Creation

Using the flow diagrams as a guide, we need to create two SACLs to define the permitted outbound conversations and provide the placeholder for dynamic ACL statements that will allow the conversation "return" traffic. CBAC will work with both numeric and named extended access lists. While both perform equally well, using named access lists has some advantages. First, they act as a troubleshooting aid because the access-list name can be utilized as a reference. Second, named access lists are editable. So hosts and services can be added and removed without having to create a new list. Because it's a good idea to take every advantage we can, our examples will use named lists.

Configuring CBAC Policy Configuration and CBAC Implementation

Michael J. Martin

Outside Inbound List

Base the list on the findings detailed in the inbound service flow diagram. Users outside of the secure network need access to:

- POP3 running on host 66.45.100.66
- SMTP running on host 66.45.100.2
- SSH/FTP running on host 66.45.100.49
- HTTP, SSH, and Squid running on host 66.45.100.44

Along with these TCP based services, we also need to configure (if desired) ICMP. As of 12.2.15T, stateful inspection of certain ICMP message types has been added.

- Type 0 Echo reply
- Type 3 Destination unreachable
- Type 8 Echo request
- Type 11 Time exceeded
- Type 13 Timestamp request
- Type 14 Timestamp reply.

This enhancement was added to allow hosts connected to the secured network to generate ICMP messages to hosts on the unsecured side of the router. The option provides little effect when implemented to track ICMP messages originated from unsecured hosts. So if ICMP message support from external hosts to internal hosts is required, support should implement for specific hosts and for specific message types. Additionally, it is a good idea to implement ICMP rate limiting (see October 2002) to counteract any potential ICMP exploit that could be deployed against the secured hosts. Because these are "externally" accessed services, support for the ping command is a necessity for troubleshooting. So we will include support for ICMP echo and echo-reply for the four servers.

Here is the configuration syntax dialog for our external inbound SACL:

```
lab-router(config)#ip access-list extended external-services-SACL
lab-router(config-ext-nacl)#permit tcp any host 66.45.100.60 eq pop3
lab-router(config-ext-nacl)#permit tcp any host 66.45.100.2 eq smtp
lab-router(config-ext-nacl)#permit tcp any host 66.45.100.49 eq 22
lab-router(config-ext-nacl)#permit tcp any host 66.45.100.44 eq 22
lab-router(config-ext-nacl)#permit tcp any host 66.45.100.49 ftp
lab-router(config-ext-nacl)#permit tcp any host 66.45.100.49 eq ftp
lab-router(config-ext-nacl)#permit tcp any host 66.45.100.49 eq ftp-data
lab-router(config-ext-nacl)#permit tcp any host 66.45.100.44 eq 3128
lab-router(config-ext-nacl)#permit tcp any host 66.45.100.44 eq 80
lab-router(config-ext-nacl)#permit icmp any host 66.45.100.60 echo
lab-router(config-ext-nacl)#permit icmp any host 66.45.100.2 echo
lab-router(config-ext-nacl)#permit icmp any host 66.45.100.44 echo
lab-router(config-ext-nacl)#permit icmp any host 66.45.100.49 echo
lab-router(config-ext-nacl)#permit icmp any host 66.45.100.60 echo-reply
lab-router(config-ext-nacl)#permit icmp any host 66.45.100.2 echo-reply
lab-router(config-ext-nacl)#permit icmp any host 66.45.100.44 echo-reply
lab-router(config-ext-nacl)#permit icmp any host 66.45.100.49 echo-reply
```

Inside Inbound List

For our example, the inside filter list follows the same line as the outside filter; permitted services are explicitly defined. This approach is the most secure, but is also the most constrictive to the user. If the

Configuring CBAC Policy Configuration and CBAC Implementation

Michael J. Martin

external service does not utilize a permitted protocol and service port, it will not work. Base this list on the findings detailed in the outbound service flow diagram. Hosts on the secure network need to access the following protocols on external hosts:

- SSH/FTP for the general user population
- HTTP/HTTPS for host 66.45.100.44
- DNS (TCP) for host 66.45.100.101
- DNS (UDP) for host 66.45.100.101
- TACACS for 66.45.100.254

In most environments, HTTP is permitted. Unfortunately, most of the applications administrators do not want users to use will operate over or tunnel through HTTP (80) or HTTPS (443). To counteract this, an HTTP/HTTPS Squid proxy server has been implemented, and users are not permitted to originate HTTP/HTTPS connections outbound. The implementation of Squid provides some big advantages. First, it eliminates the user's ability to exploit the use of port 80 and 443 for unauthorized applications. Second, it manages the available bandwidth more effectively, because users share a common Web cache. Third, (this may be a disadvantage) it allows the company to monitor HTTP activity, specifically inappropriate activity so it can protect itself. Finally, it can serve as a Web anonymizer for external hosts (which is why external hosts are permitted access) who do not wish to have their IP addresses logged on Web server access logs. Squid supports several authentication schemes, so access can be controlled.

Here is the configuration syntax dialog for our internal inbound SACL:

```
lab-router(config-ext-nacl)#ip access-list extended internal-perm-outb-serv
lab-router(config-ext-nacl)#permit tcp host 66.45.101.100 host 65.45.101.254 eq tacacs
lab-router(config-ext-nacl)#permit tcp host 66.45.101.100 host 65.45.101.254 gt 1024
lab-router(config-ext-nacl)#permit tcp 66.45.101.0 0.0.0.255 host 65.45.101.1 eq 22
lab-router(config-ext-nacl)#permit icmp 66.45.101.0 0.0.0.255 host 65.45.101.1 echo
lab-router(config-ext-nacl)#permit tcp host 66.45.100.44 any eq 80
lab-router(config-ext-nacl)#permit tcp host 66.45.100.44 any eq 443
lab-router(config-ext-nacl)#permit tcp host 66.45.100.101 any eq domain
lab-router(config-ext-nacl)#permit udp host 66.45.100.101 any eq domain
lab-router(config-ext-nacl)#permit tcp 66.45.101.0 0.0.0.255 any eq ftp
lab-router(config-ext-nacl)#permit tcp 66.45.101.0 0.0.0.255 any eq ftp-data
lab-router(config-ext-nacl)#permit icmp 66.45.101.0 0.0.0.255 any echo
lab-router(config-ext-nacl)#permit icmp 66.45.101.0 0.0.0.255 any echo-reply
```

Inspection Rule Creation

Cisco recommends that you tune the stateful inspection rule variables prior to creating the CBAC inspection rules. However, to properly tune many of these variables you really need data on things such as session volume (per service), session set up, session duration, etc. (We will get into how to collect this data next time.) Of course, this is data most folks don't have when they're setting up their firewalls. So how exactly are you supposed to tune CBAC correctly? After all, the value of tuning is to tighten up the configuration to meet your site's requirements. You will find that once the firewall is implemented, this will become an ongoing effort. So when you first implement CBAC, is best to run with the defaults, collect data, and then tune.

Without the burden of tuning, creating CBAC rules is quite simple. The number of protocols for which CBAC can provide inspection varies depending on the IOS version. The configuration command syntax, however, does not. For most "protocols" the command format is <ip inspect name {rule set name} {protocol} {alert [on|off]} {audit-trail [on|off]} {timeout [in sec]}>. The alert switch enables/disables violation notices. The audit-trail option turns on detailed session tracking (src/dest

Configuring CBAC Policy Configuration and CBAC Implementation

Michael J. Martin

address and port numbers and bytes transferred). If you enable RPC inspection, the configuration syntax is

```
<ip inspect {name [rule set name]} rpc program-number {prog #} wait-time {in min} {alert [on|off]} {audit-trail [on|off]} {timeout [in sec]} >.
```

Cisco again recommends the creation of a single rule set for most implementations. If, however, CBAC inspection is being implemented on multiple interfaces and/or different directions, it is advisable (according to Cisco) to create multiple rule sets. After a lot of experimentation, it seems that the only reason to configure more than a single CBAC rule set is when the inspection requirements for the interfaces are different. And even then, you can create one meta-list that covers the requirements of both.

For our example network, we need to define four inspection rules for the external interface and five rules for the internal. Here is the configuration syntax dialog for the external interface CBAC rule set:

```
lab-router(config)#ip inspect name external-inspection-inbound tcp
lab-router(config)#ip inspect name external-inspection-inbound udp
lab-router(config)#ip inspect name external-inspection-inbound ftp
lab-router(config)#ip inspect name external-inspection-inbound smtp
```

Here is the configuration syntax dialog for the internal interface CBAC rule set:

```
lab-router(config)#ip inspect name internal-inspection-outbound tcp
lab-router(config)#ip inspect name internal-inspection-outbound udp
lab-router(config)#ip inspect name internal-inspection-outbound ftp
lab-router(config)#ip inspect name internal-inspection-outbound smtp
lab-router(config)#ip inspect name internal-inspection-outbound icmp
```

These two rule sets hit the mark for most environments (notice that audit-trail logging has not been enabled; we will get into why next month). The TCP and UDP rules provide inspection for any single-session application. The FTP and SMTP inspection rules monitor for protocol specific violations. As for ICMP inspection, it works only for the supported protocols and only tracks state based on the source of the message. So while it allows administrators to provide tasteful ICMP support, its usefulness is questionable in contrast to using tools like rate limiting to buffer potentially harmful ICMP use. The best bet is to utilize both CBAC ICMP inspection and rate limiting.

Interface Configuration

It would seem that CBAC was developed as a "hard on the outside, soft on the inside" security technology (at least that how the documentation portrays it). The Cisco guideline on how to install CBAC on router interfaces is slanted toward this usage model. After a little experimentation, however, it can be quickly determined that CBAC is capable of more than just protecting a SOHO from the Internet. That's why we've discussed all of the CBAC topology variations covered above. Hopefully it will provide a fuller picture to you than the Cisco documentation.

Our example implements CBAC inspection from externally and internally spawned connections. For the sake of simplicity it uses topologies one and two in combination. Interface configuration involves two steps. First, you must apply the SACL as an IP access group on the interface:

```
lab-router(config)#interface ethernet 0/0
lab-router(config-if)#ip access-group external-services-SACL in
```

Configuring CBAC Policy Configuration and CBAC Implementation

Michael J. Martin

Then, you will apply the CBAC inspection rule that will modify the IP access-group SACL. In this case, it will be installed on the inside interface. The configuration syntax is <ip inspect {rule set name} {in|out}>.

```
lab-router(config)#interface ethernet 0/1
lab-router(config-if)#ip inspect internal-inspection-outbound in
```

Conversely, the CBAC rule set installed on the external interface will modify the SACL associated with the IP access group installed on the inside interface.

```
lab-router(config)#interface ethernet 0/1
lab-router(config-if)#ip access-group internal-perm-outb-serv in
lab-router(config-if)#exit
lab-router(config)#interface ethernet 0/0
lab-router(config-if)#ip inspect internal-inspection-outbound in
```

That covers the basics for implementing CBAC on IOS FFS routers. But don't despair, we still have to cover tuning, auditing, troubleshooting, debugging and logging. I'll save that for next month. As always, hope you found this useful and feedback is always welcome. See you next month (perhaps at the Networking Decisions Conference in Atlanta) same bat time, same bat channel.