

The Router Is The Firewall Part 3

Configuring CBAC

Michael J. Martin

In this installment of the "Router is the firewall" series, we'll look at the Context Based Access Control (CBAC) configuration process and some real-world implementations. We'll provide an overview of the CBAC configuration process, and then focus on the initial part of the process -- traffic qualification. For those of you new to the series, CBAC is an enhancement to IOS's access control filtering providing stateful session tracking and dynamic access control list (ACL) rule management. CBAC, in conjunction with the additional security enhancements provided through the IOS firewall feature set, offers a very flexible and cost effective alternative to traditional hardware and software-based firewalls. In light of current events (see Tools for surviving the Cisco IOS flaw), some administrators may consider CBAC as a further enhancement to their current static access control (SACL) implementations.

Configuration Overview

Configuring CBAC is a three-phase process. I use the term "phase" here to emphasize the need for some consideration to be paid before actually configuring CBAC on the router. The three phases are:

1. Traffic qualification
2. Policy configuration and implementation
3. Policy testing and tuning.

Phase one is the most vital and involved part of configuring CBAC. In order for CBAC (or any firewall implementation) to be effective, a full understanding of the network protocols in use between internal (behind the firewall) and external (in front of the firewall) systems and independencies between systems on the network is essential.

Phase two is the actual router configuration of CBAC. This involves the following:

- Securing the router!
- Determine the interfaces and filtering points needed to accommodate the network's operation
- Create the static access control lists that will be used by the interface access groups
- Create the CBAC inspection policy
- Install access-groups on the required router interfaces
- Install inspection policies on the required router interfaces.

Phase three is implementation of the access and inspection policies, which consists of:

- Monitoring of SACL log output
- Implementation and monitoring of CBAC auditing
- Tuning of inspection and conversation timers.

A complete CBAC configuration, in terms of time and effort, depends largely on the size and type of environment. A remote office implementation requiring outbound access only can easily be completed in a few hours. Alternatively, a large-scale enterprise with a number of services with system interdependencies could require a great deal of effort during phases one and three. Inevitably, it all comes down to how well you (or your server administrators) know your server environment, and how well you understand the network requirements needed to support access to those services.

Traffic Qualification

Routers function as inter-exchange points on a network in a fashion similar to a highway rotary. Network traffic passes through the device on its way to a destination point, and the router interfaces function as on and off ramps (again with the rotary analogy). Their role in the network makes them an oddity in some sense. The majority of traffic router processes are destined for other IP hosts while a

The Router Is The Firewall Part 3

Configuring CBAC

Michael J. Martin

minority of traffic is intended for the actual router. When functioning as an IP transit device, a properly configured router needs to accommodate for this dichotomy of traffic when implementing traffic control access groups to protect against infrastructure attacks. At a minimum, traffic filters on ingress router interfaces (those facing networks outside of your administrative control) should accommodate the following:

- Block RFC-1918 addresses- provided you not using RFC-1918 addressing on your network.
- Block bogon addresses - non-routable IP prefixes that are not utilized on Internet and special use address space as defined in RFC 3330.
- Drop packets containing the local network prefixes as source addresses in IP packets coming from outside your network. This should never happen. Filtering network prefixes under your administrative control protects you against spoofed IP attacks.
- Drop non-initial fragmented packets. Fragmentation attacks to both the router and downstream hosts should be defended against. Use of the IOS <fragment> ACL keyword can be used to drop non-initial fragmented packets.
- Explicitly permit routing protocol data from qualified hosts.
- Explicitly permit ICMP message types.
- Block VTY access over ingress interfaces.

On egress interfaces:

- Local network prefixes as source address should be explicitly permitted to traverse the egress interface. Packets with a source address outside of the local prefixes (unless you are serving as transit for another network out of your administrative control) should be discarded.
- Non-initial fragmented packets should be discarded. Fragmentation attacks to both the router and upstream hosts should be defended against. Use of the IOS ACL keyword can be used to drop non-initial fragmented packets.
- Explicitly defined access rules for routing protocol data exchange.
- Explicitly permit ICMP message types.
- Explicitly permit VTY access from specific hosts or network prefixes.

When changing the disposition of the router from a transit gateway to a firewall gateway you must have an understanding of the ingress and egress traffic flow in order to develop the needed enhancements to the ingress and egress SACLs and to construct the CBAC inspection policy. Specifically, your analysis should provide you with answers to the following data points:

- The service ports utilized
- Any system interdependencies (i.e., upstream AAA servers, DNS servers, etc.)
- The number or volume of sessions per minute.

The Router Is The Firewall Part 3

Configuring CBAC

Michael J. Martin

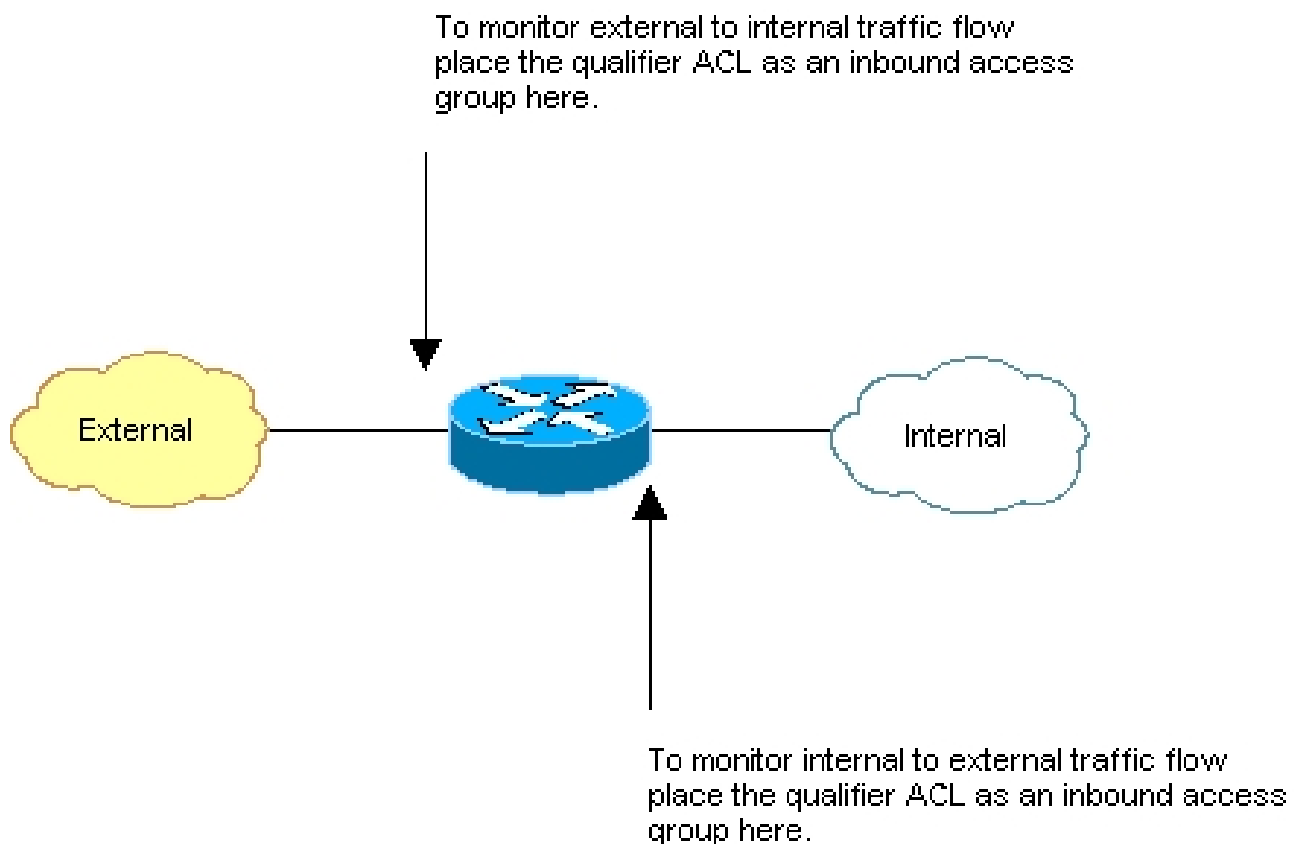
The goal is to construct a server/service map of the network detailing (1) ingress and egress traffic flow by service, and (2) an estimate baseline for sessions (establishment, open, close) per minute.

Analysis Options

This article is not about traffic analysis. There are a number of excellent books and countless Internet resources on this subject. That said, here two quick approaches to collecting the baseline data you need to get started with CBAC.

Using IOS SACL's to Qualify Traffic

If you are converting an existing network gateway router over to CBAC or are installing additional defense points, SACL can be deployed to "qualify" the protocols and services in use on the network.



The figure above illustrates the filter placement needed to collect inbound-> outbound or outbound -> inbound traffic flow data. Here is an example of a "collection" SACL:

```
access-list 144 permit icmp any any log-input
access-list 144 permit tcp any any gt 1 log-input
access-list 144 permit udp any any gt 1 log-input
access-list 144 permit udp any eq 500 any eq 500 log-input
access-list 144 permit udp any eq 520 any eq 520 log-input
access-list 144 permit esp any any log-input
access-list 144 permit ospf any any log-input
access-list 144 permit eigrp any any log-input
access-list 144 permit gre any any log-input
```

The Router Is The Firewall Part 3

Configuring CBAC

Michael J. Martin

The SACL contains specific permit statements for the commonly used IP network transport (UDP 500 is for ISAKMP) management and route announcement (UDP 520 is RIP) protocols. The installation of this SACL as an <ip access-group> in conjunction with the routers logging function provides the data needed to generate a service and traffic flow map. Each time a packet matching one of the listed protocols passes through the router interface the ACL will log the event. Providing data on the packets source address, source port, destination address and destination port. Here is a log output sample:

```
Jul 29 18:45:20 EDT: %SEC-6-IPACCESSLOGP: list 144 permitted tcp
172.30.71.4(49175) (FastEthernet0 0030.6525.6a43) -> 204.60.219.167(80), 1 packet
```

```
Jul 29 18:45:22 EDT: %SEC-6-IPACCESSLOGP: list 144 permitted tcp
172.30.71.4(49176) (FastEthernet0 0030.6525.6a43) -> 66.218.78.157(80), 1 packet
```

When using this collection approach keep in mind that the router will "log" every packet that passes through the interface. On a transit router with a large amount of traffic passing through, this could result in a very large amount of data if the qualification access-group is installed and just forgotten. A little caution on your part will avoid the collection of useless data, crashing the router by overwhelming the CPU or overloading of your logging system. If you're using buffered logging you will get very little data to work with. If you are offloading your log data to a syslog server, you could fill up your log file system pretty quick.

A good approach for using this collection option is to install the SACL for five minute intervals, twice an hour during peak utilization periods (i.e, AM between 9-10, 11-12 - PM between 1-2, 4-5). Use the collected data as a sample and construct your initial map after you have collected a weeks worth of data. The raw syslog data will need to be processed a little to make it more manageable. You are interested five data values:

- The protocols in use on the network
- Who is starting conversations (Src_Adr:Src_Port)
- Who is replying (Dst_Adr:Dst_Port).

To extract this data from the syslog datafile you can use AWK to format the data and GREP to qualify it. To print out the protocol, source, destination host and port information use the following command:

```
awk '{print $14"t" $15"040""040"$19}' $LOGFILE
```

Where \$LOGFILE is the absolute path to the syslog data file. To see the data by specific protocol PIPE the AWK output into GREP with the protocol name as the matching value. Here is a command syntax example for extracting the TCP transactions:

```
awk '{print $14"t" $15"040""040"$19}' /var/log/gw | grep "tcp"
```

To see the same data for UDP transactions change the GREP matching value to "udp"

```
awk '{print $14"t" $15"040""040"$19}' /var/log/gw | grep "udp"
```

Here is an example of the AWK formatted output:

```
udp      172.30.71.4(49187)  204.60.0.3(53),
udp      172.30.71.4(49187)  204.60.0.2(53),
```

The Router Is The Firewall Part 3

Configuring CBAC

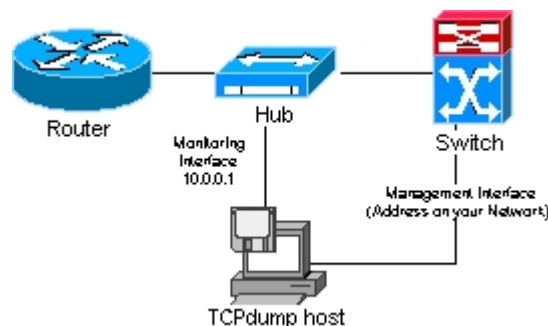
Michael J. Martin

```
udp    172.30.71.4(49187)  192.168.100.1(53),
udp    172.30.71.4(49187)  216.140.16.254(53),
tcp    172.30.71.4(49201)  4.17.168.6(80),
tcp    172.30.71.4(49)    172.30.71.1(11056),
tcp    172.30.71.4(49)    172.30.71.1(11057),
tcp    172.30.71.4(49)    172.30.71.1(11058),
tcp    172.30.71.4(49199)  4.17.168.6(80),
```

Once you have extracted the data, additional massaging may be needed to classify your servers and the services they are providing (regardless if they are supposed to) and the types of services your users are utilizing both internally and externally. I guarantee you will learn a thing or two about your network.

Using Tcpdump To Qualify Traffic

An alternative to using SACL's to qualify traffic is to use TCPdump. TCPdump is one of those staple tools that Network and System administrators alike reach for whenever they need to take a look at the actual network packets. It was written way back in the day and runs on UNIX and Windows and is consistently maintained by its author, Van Jacobson. It's not quite a packet sniffer but close enough for government work. For our auditing purposes we are interested not in the whole packet but just the IP protocol information. So we will need to clean up the data a little to make it more manageable, just as we did for the SACL logs. The advantage to using TCPdump over SACL's is that TCPdump provides access to additional traffic attributes beyond host and port information. Just some of these are: details on TCP flags, ICMP message types, markers for specific services utilizing TCP or UDP for transport like RIP and ISAKMP. Overall a far greater amount of detail then can be captured with SACL logging. There is one downside (again, with the downsides UGH, but there is always a downside) you need to run TCPdump on a UNIX or Windows server attached to a hub with your router in order to collect the data.



While this does add additional complexity to your network, it's actually not a bad idea to have your routers interconnect to your switches through shared-segment hubs so you can do this kind of monitoring. While most switches provide support for port mirroring, many implementations can only mirror one direction and all implementations will add additional load to the switches CPU. 10/100 hubs are inexpensive and you should not be running your routers in Full-Duplex mode anyway. As long as the router and switch are the only devices passing traffic on the hub you can run them both in FDX. The monitoring interface will need an IP address for TCPdump work, but it will not transmit any traffic (you can snip the TX pair on the cable if you want to make sure). A management interface connected to your network will also be needed if you want to access the TCPdump server over the network.

The Router Is The Firewall Part 3

Configuring CBAC

Michael J. Martin

Creating The Traffic Collection Statement

Lets create a TCPdump statement to collect the same information that our IOS collection SACL does. To collect packets with TCPdump by IP protocol type the command keyword is <proto {protocol ID #}>. Our TCPdump CLI syntax looks like this:

```
tcpdump -i en0 -nN proto 6 or proto 17 or proto 1 or proto 47 or proto 50  
or proto 51 or proto 89 or proto 88
```

The TCPdump IP protocol ID's, correspond to the IANA protocol ID assignments:

IANA Information				TCPdump Value
1	ICMP	Internet Control Message	[RFC792]	= proto 1
6	TCP	Transmission Control	[RFC793]	= proto 6
17	UDP	User Datagram	[RFC768,JBP]	= proto 17
47	GRE	General Routing Encapsulation	[Tony Li]	= proto 47
50	ESP	Encap Security Payload for IPv6	[RFC2406]	= proto 50
88	EIGRP	EIGRP	[CISCO,GXS]	= proto 88
89	OSPF	OSPF	[RFC1583,JTM4]	= proto 89

The above TCPDump capture will yield a lot more data then the qualifier ACL, here is just a sample of what the raw capture looks like:

```
17:46:42.480746 172.30.71.11.500 > 172.30.71.1.500: isakmp: phase 1 I agg: [|sa|  
17:46:46.180293 172.30.71.1 > 224.0.0.5: OSPFv2-hello 44: backbone dr 172.30.71.1  
[tos 0xc0] [ttl 1]  
17:46:53.111653 172.30.71.5.49370 > 172.30.71.1.23: P 498445595:498445596(1) ack  
806466151 win 33580 (DF) [tos 0x10]  
17:46:53.115286 172.30.71.1.23 > 172.30.71.5.49370: P 1:2(1) ack 1 win 3974 [tos  
0xc0]  
17:46:53.127508 172.30.71.5.49370 > 172.30.71.1.23: . ack 2 win 33580 (DF) [tos  
0x10]  
17:46:53.140400 172.30.71.5.49370 > 172.30.71.1.23: P 1:2(1) ack 2 win 33580 (DF)  
[tos 0x10]
```

As I mentioned earlier, we do not need this level of detailed information to determine the hosts and protocols active on the network. So to make it a little more readable we can use AWK and GREP to format and match the specific protocols we are looking for. Here is TCP command syntax to see traffic data for each of the collected protocol types (using UNIX):

To extract the UDP Session (SRC:SRC_Port > DST:DST_Port):

```
more $DUMPFILe | awk '{print $2"040" $3"040"$4 $5}' | grep udp
```

To extract ICMP sessions (SRC DST MSG) :

```
more $DUMPFILe | awk '{print $2"040" $5"040"$6"040"$7"040"$8"040" $9  
"040" $10}' | grep icmp
```

To extract OSPF sessions (SRC OSPF MSG Type):

The Router Is The Firewall Part 3

Configuring CBAC

Michael J. Martin

```
more $DUMPFILe | awk '{print $2"t" $5}' | grep OSPF
```

To extract ESP sessions (SRC > DST CRYPTO)

```
more $DUMPFILe | | awk '{print $2"040" $3"040"$4"040" $5}' | grep ESP
```

To extract ISAKMP sessions (required for IPsec)

```
more $DUMPFILe | awk '{print $2"040" $3"040"$4 "040" $5}' | grep isakmp
```

Once you have paired down the log file to the specific protocol, additional filtering can be performed to identify on and off network specific hosts and the services utilized on them. You may have noticed the absence of TCP in the examples above. As the name implies TCPDump built around examining TCP packets. Here is what the start of a TCP session looks like:

```
17:50:18.522825 216.136.173.10.110 > 172.30.71.5.49373: S  
765476511:765476511(0) ack 929924997 win 65535 <mss 1460,nop,  
wscale 1,nop,nop,timestamp 5990486 3449340534> (DF)
```

This is the second packet in the setup of a POP3 session. Notice the "S" in the fifth field. The "S" is an indicator that the SYN flag is set. Then in seventh field there is an "ack," indicating that the acknowledgement bit is set. Another indication that this packet is part of a session setup negotiation is the MSS option announcement. In the cases for the protocols listed above, the transactions in most instances (minus IPsec) require only a single packet. So not only do the reports provide protocol data but they also give you an idea of how many conversations are taking place. By timing the captures and doing a little further digging you can make some basic estimates on how many transactions are transpiring during a given time frame. This data can come in handy when setting session timeouts during phase three since a TCP session can contain a large number of packets, and in some cases spawn child sessions. To get an idea of the number of sessions starting and stopping along with the protocol data is quite helpful. Here are some extraction commands using AWK and GREP to give some stats on TCP sessions.

To extract data on the TCP sessions started including SRC:SRC_Port > DST:DST_Port information use:

```
more $DUMPFILe | awk '{print $2"040"$3"040"$4"040" $5"040" $7}' | grep  
"ack" | grep S | grep -v sackOK | grep -v OSPF
```

To extract data on the number of TCP sessions started during the collection period:

```
more $DUMPFILe | awk '{print $2"040"$3"040"$4"040" $5"040" $7}' | grep  
"ack" | grep -c S | grep -v sackOK | grep -v OSPF
```

The sessions started value is calculated using a count of the number of SYN-ACK packets collected during the capture period. The match is performed using SYN-ACK, rather than SYN packets because a SYN-ACK represents the second phase of the establishment process, where both parties have at least agreed in part to a conversation. SYN alone only indicates that a host wishes to have a conversation, with no idea if the target host will even be active.

To extract data on the TCP sessions closed including SRC:SRC_Port > DST:DST_Port information use:

The Router Is The Firewall Part 3

Configuring CBAC

Michael J. Martin

```
more $DUMPFILE | awk '{print $2"040"$3"040"$4"040" $5"040" $7}' | grep  
"ack" | grep F
```

To extract data on the number of TCP sessions started during the collection period:

```
more $DUMPFILE | awk '{print $2"040"$3"040"$4"040" $5"040" $7}' | grep  
"ack" | grep -c F
```

The return count value needs to be divided by two in order to get an accurate count of session terminations. This is because two FIN-ACK packets are sent (one by each host) when a session closes. Please keep in mind these are estimates. If you want perfect data you can purchase a Stealth Watch probe from Lancope and profile your network with certainty. That however costs money, and since we're a low cost network we are going with the "estimate" method. With that said let's move on to Phase two policy configuration and implementation.

Tune in next time... Oh my! Look at the time -- our session is over for now. It's summertime and it's time for all of us pale computer people to get out of the computer room and head out to the beach. Next month we will finish up CBAC configuration and tuning. Use your sun block and don't forget to filter your network.