

# Implementing SACLs to Defend and Detect Attacks

Michael J. Martin

This month's article continues to discuss intrusion detection and defense. We'll talk about creating IP static access control lists (SACLs) for filtering unwanted traffic, specifically RFC-1918 addresses; implementing RFC-2827 recommendations (IP spoofing protection, filtering inbound traffic with a source IP address that matches the local prefix addresses); non-allocated IP prefixes; and some other filtering recommendations you should consider.

## **I have a firewall and intrusion detection system (IDS). Why do I need SACLs on my router?**

First, if you do not have a firewall and/or network IDS and have publicly accessible hosts, then you should at a minimum be using inbound and outbound SACLs, so just skip to the next section. If you do have a firewall or host or network IDS (HISD/NIDS), there are some very good reasons to use SACLs for both security and performance.

In short, you can discard traffic you simply do not want, i.e., bogus packets with spoofed or RFC-1918 addresses, denial of service (DOS) and distributed DoS attacks, and any network service you do not support (Internet gaming, instant messaging, etc.). In terms of efficiency, this just makes good sense. There is simply no need to process traffic that is not legitimate. Depending on the size of your Internet connection and the visibility of your hosts, network administrators can expect any from 2% to 10% of their utilized inbound network bandwidth to be consumed by "noise."

Firewalls and IDS work by processing the entire (inbound/outbound) traffic flow. This processing entails inspecting the entire packet contents by comparing the packet to a rule base (firewall) or signature database (IDS) and then determining a handling action (forward/drop/alert/alarm). In the case of firewalls that operate using Network Address Translation (NAT), there is the added NAT processing as well. All of these processes consume resources. In low to moderate throughput environments, the processing of an extra 2% to 15% of traffic does not translate into perceivable performance degradation.

In high throughput environments, however, the processing of noise traffic does impact performance. For example, an Internet access point with two DS3s from different ISPs has a maximum throughput of 90 Mbps. If we install a Nokia 500 running Check Point FW1 with NAT, which has total forwarding capability of 90+ Mbps with one internal and one external interface, we are able to utilize all of our available bandwidth. If we assume the average for noise, just one of our unfiltered Internet connections at wire rate will process 4.25 Mbps of noise. That makes a total of 8+ Mbps of traffic that is not at all relevant to our production requirements that our firewall needs to process. On the security side, implementing SACLs provides another layer of filtering in addition to the firewall. It also provides protection against potential firewall policy misconfiguration and illegal hosts that circumvent the firewall entirely.

When using NIDS, SACLs help to address performance problems. Like firewalls, NIDS have a throughput-processing ceiling. Most (not all) NIDS have problems processing consistent traffic rates above 15 to 20 Mbps, resulting in dropped (unprocessed) packets and, in some cases, system crashes. In our example environment, if we intended to have Internet accessible hosts, we would need to add additional firewall interfaces (decreasing our firewall's throughput capacity) so that we could structure the traffic throughput (by distributing the hosts across multiple DMZ interfaces) to allow the NIDS to be effective. This approach, however, has both scaling problems and performance side effects.

As if the performance issues were not enough, there is also the performance impact of event alarms. Implementing NIDS on publicly accessible network segments has some questionable value, because the majority of detectable events are unresolvable from a litigation standpoint, consisting mostly of vulnerability scans and "script kiddie" attacks using spoofed IP addresses. Defense -- keeping current

# Implementing SACLs to Defend and Detect Attacks

Michael J. Martin

with system patches and using SACLs -- is much more advantageous than detection. The latest Network World IDS shootout found that the major drawback to most NIDS is system failure due to inability to process excessive alarms or database lockups. Installing NIDS on an unfiltered network will only result in NIDS crashes and tons of false alarms. Implementing SACLs allows the network administrator to filter out script kiddie scans, known network-based attacks, Trojan horses and other non-supported network application if desired. Dropping the noise traffic before it reaches the NIDS defends the network from potential attacks and minimizes the generation of both false and unresolvable condition alarms. This leaves the NIDS throughput available to monitor for legitimate attacks.

## Does the use of SACLs impact the routers performance?

Routers and firewalls extract the IP payload from Layer 2 (L2) frames from an incoming interface, process the IP packet, then encapsulate the IP packet in the L2 frame type of the outgoing interface. The use of inbound SACLs on Cisco routers imposes little performance impact, since traffic matching is performed before the packet is processed for forwarding. The IP packet is extracted, filtered, and then processed for filtering. Outbound SACLs are applied post-process, so their use imposes additional packet processing and consumes slightly more router resources. For this reason, the use of inbound access control lists is preferred for packet filtering operations.

## IOS Access Lists

Before an SACL becomes an instrument for filtering IP traffic, it starts off as a plain access list. IOS provides a means for creating ordered arrays for classifying L2 and L3 traffic. These arrays are then used by various IOS operations for identifying traffic that should be processed by the said operation. Some examples of these operations are distribution lists (used for route filtering), route maps (used with policy routing), crypto maps (used for applying IPsec policies) and access groups (used for implementing SACLs on router interfaces). In addition to IP, the IOS supports access lists for AppleTalk, IPX, and Ethernet MAC. Each access list type has an associated integer range. When an administrator creates an access list, it must be assigned a unique numeric ID from the range associated with each access list type.

Note: For simplicity's sake, the terms SACL and access list will be synonymous for the remainder of this article. The following guidelines relating to the creation of SACLs apply to all applications of access lists.

## Implementing IOS SACLs

IP, AppleTalk, L2 MAC, IPX -- if the IOS can route the traffic, the IOS can filter the traffic. Our discussion will deal only with TCP/IP SACLs of which there are two flavors, "standard" and "extended." Before we discuss the differences between them, there are three quick caveats: First, as I inferred above, SACLs are applied to interfaces as inbound (traffic flowing into the router) or outbound (traffic flowing out of the router) filters. SACLs are assigned to an interface using the interface configuration command `<ip access-group {number/name} {in/out}>`. If no filter point is defined, the SACL will be applied as an inbound filter (only on IOS versions prior to 12.0).

Second, both standard and extended SACLs work from the precept of "implicit deny all," meaning that unless traffic is explicitly permitted, the traffic will be discarded (by default, this is appended to the end of each SACL). Third, SACLs are processed from the top down. However, the IOS will reorder SACL entries using a low-to-high numbering scheme when the SACL is loaded into memory relative to its placement. This should not affect the operation of the SACL except from the perspective of rule processing. One of the guidelines to keep in mind with SACLs is that you should place the rules that will be most often applied at the head of the SACL. Depending on the type of entry, particularly filtering rules against hosts, some reordering may occur regardless of the order the entries are made.

# Implementing SACLs to Defend and Detect Attacks

Michael J. Martin

## Standard SACLs

Standard SACLs (S-SACLs) provide network protocol (OSI-RM L3) match filtering on a packet's source IP address. Matching is processed from the top of the list; the first match determines the packet handling action. Cisco uses numbers to identify different types of SACLs. IP S-SACLs use the number ranges 1 through 99 and 1,300 to 1,999. Two packet-handling actions are supported: permit (forward) or deny (drop into the bit bucket of despair). The source address definition supports traffic filter qualification using a combination of classful network/mask, VLSM or subnet/mask, host address or just plain "any." When creating network address matching statements, subnet masks are entered using Cisco's reverse mask "wildcard" notation. For example, a class "C" network mask is written as 0.0.0.255 instead of 255.255.255.0. When using classful addressing, the wildcard method is quite straightforward, but with VLSM it becomes more of a pain. To calculate the wildcard mask, subtract the subnet address mask from 255.255.255.255. The result is the wildcard mask. Here is the standard S-SACL configuration syntax:

```
<access-list {id} {permit/deny} {host/src addr/any}{wildcard mask} {log}>
```

Here is a sample standard SACL for filtering RFC-1918 prefixes:

```
access-list 10 remark Network and Loopback Addresses
access-list 10 deny host 0.0.0.0 log
access-list 10 deny 0.0.0.0 0.255.255.255 log
access-list 10 deny 127.0.0.0 0.255.255.255 log
access-list 10 remark RFC-1918 Addresses
access-list 10 deny 10.0.0.0 0.255.255.255 log
access-list 10 deny 172.16.0.0 0.15.255.255 log
access-list 10 deny 192.168.0.0 0.0.255.255 log
access-list 10 remark Multicast Class "D" address space
access-list 10 deny 224.0.0.0 31.255.255.255 log
access-list 10 Generic permit all statement
access-list 10 permit any
```

## Extended SACLs

Extended SACLs (E-SACL) provide network protocol (OSI-RM L3 and OSI-RM L3) match filtering on a packet's protocol, source and destination IP address and port number (TCP/UDP only). As is the case with S-SACLs, the IOS associates specific ID number ranges with extended SACLs. ID range 100 to 199 is available in all IOS implementations that support extended SACLs, and range 2,000 to 2,699 is available on IOS version 12.x and later. In addition to source and destination matching support, E-SACLs can match on a number of extended options. Here is E-SACL configuration syntax:

```
<access-list {id} {permit/deny} {protocol} {host/src addr/any} {wildcard
mask} {src port} {operators [qualifiers]} {host/dest addr/any} {wildcard
mask} {dest port} {operators [qualifiers]}>
```

While most of the E-SACL command syntax is self-explanatory, the protocol definition and the optional ACL operators require some further discussion.

With E-SACLs, the protocol entry is key. It defines the packet type that is being filtered. The entry can be either an integer (1 to 255) or IP protocol keyword. A list of some of the more popular keywords is below, with their IANA (Internet Assigned Numbers Authority) protocol ID integer. The value in filtering by protocol type is the degree of granularity it allows the administrator to control the flow of traffic to host and subnets. Keep in mind that when filtering by protocol, the IP keyword overrides any equivalent protocol-specific (i.e., TCP or UDP) entry. When using protocol-specific entries be sure to

# Implementing SACLs to Defend and Detect Attacks

Michael J. Martin

enter them from most specific (i.e., TCP, ICMP, etc) to least specific (i.e., IP). Here are some of the more commonly filtered protocols:

- IP** Any protocol in an IP packet
- TCP** Only TCP packets (Protocol ID 6)
- UDP** Only UDP packets (Protocol ID 17)
- ICMP** Only ICMP packets (Protocol ID 1)
- OSPF** Only OSPF packets (Protocol ID 89)
- GRE** Only GRE packets (Protocol ID 47)
- AH** Only IPsec/Authentication Header packets (Protocol ID 51)
- ESF** Only IPsec/Encapsulation Payload packets (Protocol ID 50)

E-SACL operators vary depending on the protocol definition. Below is a short list of the commonly used operators, their function and availability by protocol:

Operator	Function	Protocol availability
<b>eq</b>	Match on L4 port number	TCP/UDP
<b>gt</b>	Match on any L4 port number higher than one defined	TCP/UDP
<b>lt</b>	Match on any L4 port number less than one defined	TCP/UDP
<b>range</b>	Match on any L4 port number within defined range (low to high)	TCP/UDP
<b>established</b>	Drop any packet with only a SYN flag set. Packets with ACK or RST will be accepted	TCP
<b>log</b>	Log matches to this entry. Messages are sent to "notice" logging facility. The log statement can be used alone or post-pended after another operator statement.	All protocols
<b>log-input</b>	Functions as the log operator but includes the input interface in report.	All protocols

There are a number of other operators for IGMP and ICMP that are not listed above. To find out more, use the IOS's command line help. Here is a sample E-SACL for filtering RFC-1918 prefixes:

```
access-list 101 remark Network and Loopback Addresses
access-list 101 deny ip host 0.0.0.0 any log
access-list 101 deny ip 0.0.0.0 0.255.255.255 any log
access-list 101 deny ip 127.0.0.0 0.255.255.255 any log
access-list 101 remark RFC-1918 Addresses
access-list 101 deny ip 10.0.0.0 0.255.255.255 any log
access-list 101 deny ip 172.16.0.0 0.15.255.255 any log
access-list 101 deny ip 192.168.0.0 0.0.255.255 any log
access-list 101 remark Multicast Class "D" address space
access-list 101 deny ip 224.0.0.0 31.255.255.255 any log
access-list 101 remark Generic permit all statement
access-list 101 permit ip any any
```

Before we move on to named SACLs, there is one more notable thing about S-SACLs and E-SACLs - you cannot edit them. You can add to them, but you cannot remove an entry. Once you add an entry, it is set in stone and post-pended to the end of the list. If you try to delete an entry the whole list will be deleted.

# Implementing SACLs to Defend and Detect Attacks

Michael J. Martin

What this means is that you should create SACLs as ASCII text files and load them using the <copy ftp running-configuration> command. You will find that this is the best way to load up any SACL that will need to be edited with any regularity or has more than a few entries. It is also a good idea to save copies of your old lists so that you can easily revert back if there is a problem.

## Named SACLs

The two biggest problems with S-SACLs and E-SACLs are identifying them and editing them. Named SACLs (N-SACL) address these two problems. Named SACLs were introduced in IOS version 11.2.

N-SACLs allow the use of a human name as a reference instead of a number. The name can be up to 100 ASCII characters in length. An N-SACL can be used in the IOS context that a numbered SACL can be used. When used to filter traffic on a router interface, the interface sub-command <ip access-group {in|out}> is used to apply an SACL to the interface. Functionally, N-SACLs are the same as S-SACLs and E-SACLs. Their configuration is different, using a nested structure similar to a routing policy statement. The configuration syntax to create a named standard or extended SACL is the following:

```
<ip access-list {standard|extended} [ASCII name]>
```

Once the list is created, you are dropped into a sub-command configuration context. The command syntax parallels those available for creating S-SACLs or E-SACLs, depending on the type of list being created standard or extended. Once the entries have been made, the <exit> sub-command returns you to the main configuration prompt. To make additional changes, use the same N-SACL configuration statement and you will be returned to the N-SACL sub-command interface.

The major advantage to using N-SACLs is that they can be edited. That means that filter rules can be removed leaving the list and ordering intact. But wait! All is not all is rosy in the SACL garden. Like its numerical brethren, selective additions are not supported by N-SACLs. All new entries are post-pended to the end of the list, requiring the list to be deleted and reloaded if changes or new entries in the matching order need to be made using TFTP. However, if the list is well thought out, changes can be scripted because selected statement removal is supported (we will take this up later in this series). Here is a sample N-SACL for filtering RFC-1918 prefixes:

```
ip access-list extended RFC1918-IN
remark Network and Loopback Addresses
deny ip host 0.0.0.0 any log
deny ip 0.0.0.0 0.255.255.255 any log
deny ip 127.0.0.0 0.255.255.255 any log
remark RFC-1918 Addresses
deny ip 10.0.0.0 0.255.255.255 any log
deny ip 172.16.0.0 0.15.255.255 any log
deny ip 192.168.0.0 0.0.255.255 any log
remark Multicast Class "D" address space
deny ip 224.0.0.0 31.255.255.255 any log
remark Generic permit all statement
permit ip any any
```

## Conclusion

In this article we have looked at implementing numeric and named SACLs. Next month, we will review some tips for creating SACLs and take a look at what you should actually be filtering at your Internet gateway. So stay tuned.

# Implementing SACLs to Defend and Detect Attacks

Michael J. Martin

## Postscript

While the common nomenclature "access control list" (ACL) has been used for many years to refer to router filters, evolutions in the IOS's capability and developments in operating systems and filtering technologies in general have created a need for more specific or descriptive definitions. The terms "static" and "dynamic" ACLs are used to discriminate between the methods for controlling access to the network. Static ACLs are "static" rules that filter traffic. Network and host statements that permit access are always open. Dynamic ACLs open qualified access only for a limited period and track the state of communications exchange. When discussing network access control, defining the method of control is critical in describing the security solution.