

Cisco Access Control List Optimization (March 2007)

Gareth Owen Evans FdSc, NEWI, University of Wales.

Abstract—Access Control Lists (ACLs) are used to prevent possible risky traffic from reaching parts of your network that it shouldn't be reaching - (filtering) similar to a firewall, and to selecting types of traffic for different traffic policies. Considering that matching packets to ACL takes time and due to security ACL lists are getting longer and longer, now I'm sure there's a better and easier way to use them efficiently.

Index Terms—Access Control List, ACL, Cisco, Network Security and Routing.

I. INTRODUCTION

Access Control Lists (ACLs) are an absolutely huge can of worms. If used correctly and efficiently they can mean a smooth network for the end user and a headache-free environment and hassle-free network for the Network Manager, but used incorrectly they could mean that there's a gaping huge hole in the network defences against hackers, viruses and worms which is open to abuse once it's found – which these days is only days or even hours away!

Seeing as the Computing and Computer Network industries are always looking at being able to do things faster and how corporate networks are getting larger by the day – sometimes one thing holds them back and that's the security side of things.

II. ACCESS CONTROL LIST BACKGROUND

Most corporate networks use Cisco networking equipment which runs the Internetwork Operating System (IOS). One part of the IOS is Access Control Lists (ACLs) which are used to filter traffic depending on the *data packet's profile*. ACLs are an ordered list of rules which the router steps through when a packet is received and looks to match the *data packet's profile* to a single ACL rule before permitting or denying the packet depending on the matched rule. The ACL rules can have characteristics of their own; permit or deny, port number, source subnet, IP address or network, destination subnet, IP address or network and direction which the packet

G. O. Evans is a third year BSc (Hons) Computer Networks degree student in the University of Wales, North East Wales Institute of Higher Education (NEWI), Wrexham, North Wales, United Kingdom. President and one of the founders of NEWI Computing Society. (e-mail: S0402222@newi.ac.uk)

traffic is moving in. With the possibility of logging a match or not for each ACL rule. [1]

ACLs also come in two flavors, Standard and Extended. Extended ACLs have a sub-flavor for TCP, UDP, ICMP and IP packet traffic; I will only list the IP and TCP sub-flavors here as these examples are the only ones used in this paper.

A. Standard ACL Syntax

```
access-list access-list-number {permit/deny}
{host/source source-wildcard/any}
```

Fig 1

B. Extended ACL Syntax - IP

```
access-list access-list-number [dynamic dynamic-name
[timeout minutes]]
{deny | permit} protocol source source-wildcard
destination destination-wildcard [precedence precedence]
[tos tos] [log | log-input] [time-range time-range-name]
```

Fig 2

C. Extended ACL Syntax - TCP

```
access-list access-list-number [dynamic dynamic-name
[timeout minutes]]
{deny | permit} tcp source source-wildcard [operator
[port]]
destination destination-wildcard [operator [port]]
[established]
[precedence precedence] [tos tos] [log | log-input]
[time-range time-range-name]
```

Fig 3

III. ACCESS CONTROL LIST EXAMPLES

A. Internal – External Network Security

For example, a Network Manager may use a Cisco router as a *border gateway* between the internal corporate network and external networks, most likely to be the Internet, where all network traffic trying to enter the internal corporate network should be denied access. So an example ACL for external to

internal access would be:

```
access-list 102 permit tcp any any established
access-list 102 deny any any
```

Fig 1

The ACL above (Fig 1) allows any TCP connections which are established from inside to continue until complete and the bottom line denies any other connections access. But only if things were this simple! As somebody somewhere will always want a port, a collection of ports or access past the firewall opened up.

B. Internal Network Security

ACLs can also be used to secure networks or subnets on the internal network too. Let's assume, we have the Workstation network using the Class A IP Addressing, starting *10.0.0.1*. The Server network uses a Class B IP Addressing, starting *172.16.0.1* and the Accounting network using a Class C IP Addressing, starting *192.168.0.1*.

Now for security reasons, you wouldn't want employees being able to access the Accounting network from any other network but you would still like a block of 30 IP addresses within the Workstation network to be able to access it for network management reasons. So you would use the example ACL below:

```
access-list 11 permit 10.0.0.0 0.0.0.31
access-list 11 deny 10.0.0.0 0.0.0.0
access-list 11 deny any
```

Fig 4

The first line in the ACL (Fig 4) will allow all connections from the IP addresses in the range of *10.0.0.1* to *10.0.0.30* (The IP Addresses *10.0.0.0* and *10.0.0.31* are Network Address and Broadcast Address respectively and aren't normally included in an IP Address Range).

Instead of listing each IP address in the range a *wildcard mask* is used. The second line will deny any other connection from the Class A IP Addressing range *10.0.0.0* to *10.255.255.255* and the third and final line being a *catch-all-else* deny statement – denying any other connection from any other subnet or network. So if a connection came from the IP Address *10.0.0.5*, its data packet would match the first line in the ACL and it would be permitted entry to that network, if a data packet from the IP Address *10.0.0.93* its connection would be denied because it would match the second line in the ACL denying it access.

IV. THE GROWING PROBLEM

Let's assume the corporate internal network is using the Class

B IP Addressing Range, starting *172.16.0.0*. On the internal network we have a Web Server using the IP Address *172.16.1.1* and Mail Server using the IP Address *172.16.1.2*.

So we now update the ACLs on the *border gateway* router.
Fig 1

```
access-list 102 permit tcp any any established
access-list 102 permit tcp any host 172.16.1.1 eq 80
access-list 102 permit tcp any host 172.16.1.2 eq 25
access-list 102 deny any any
```

Fig 5

Say we need to add port 443 for Secure Socket Layer (SSL) access to the Web Server, port 143 for Internet Message Access Protocol (IMAP) access to the Mail Server and port 110 for Post Office Protocol version 3 (POP3) access to the Mail Server but block ports 3306, 1433 and 81.

```
access-list 102 permit tcp any any established
access-list 102 permit tcp any host 172.16.1.1 eq 80
access-list 102 permit tcp any host 172.16.1.1 eq 443
access-list 102 deny tcp any host 172.16.1.1 eq 81
access-list 102 deny tcp any host 172.16.1.1 eq 3306
access-list 102 deny tcp any host 172.16.1.1 eq 1433
access-list 102 permit tcp any host 172.16.1.1 eq 25
access-list 102 permit tcp any host 172.16.1.1 eq 143
access-list 102 permit tcp any host 172.16.1.1 eq 110
access-list 102 deny any any
```

Fig 6

The ACL grows to accommodate these changes easily but how is the border gateway router coping with all this? Every time a packet reaches the external interface the router must run each packet through the list to see if it matches any ACL rule and then decides what to do with it. For small ACLs this isn't a problem but what if the ACL was around 100 lines long or larger and the router started logging each time a packet was denied or permitted access? [2]

You would quite possibly notice a high latency and delay timings on data passing through the router and high CPU and memory usage due to these factors.

V. ACL OPTIMIZATION

While ACLs grow in length, this is when things get complicated with ACLs and serious mistakes are made with security and it makes it very hard to debug and troubleshoot problems. One ACL denying access may overlap what another ACL is permitting. As it's the first ACL that the router comes across first (top downwards) that matches the packet profile that's the rule in the ACL that the router uses to deny or permit the packet onward travel.

Each time the router has to go through the ACL from top to

bottom its using processing power, memory and causing data delay and increasing latency timings meaning the end-user has to wait that little bit longer for a connection to be made.

A. Wildcard Masks

There are basic ACL Optimization techniques which can be used to reduce ACL length by a few lines. Using *Wildcard Masks* is one.

The following part-ACL can be reduced from seven lines....

```
access-list 10 permit 205.178.18.8 0.0.0.0
access-list 10 permit 205.178.18.9 0.0.0.0
access-list 10 permit 205.178.18.10 0.0.0.0
access-list 10 permit 205.178.18.11 0.0.0.0
access-list 10 permit 205.178.18.12 0.0.0.0
access-list 10 permit 205.178.18.13 0.0.0.0
access-list 10 permit 205.178.18.14 0.0.0.0
access-list 10 permit 205.178.18.15 0.0.0.0
```

Fig 6

...to a single line!

```
access-list 10 permit 205.178.18.8 0.0.0.7
```

Fig 7

Both (Fig 6) and (Fig 7) are equal in the packet matching sense but with the obvious line count of (Fig 6) being larger then (Fig 7).

As shown in (Fig 6) and (Fig 7), this is probably the most basic method of ACL Optimization but the hardest in my opinion as the removal of one rule within an ACL can be disastrous if it has been done as a mistake. A simple wildcard mask may miss a single IP Address out from its range that you're trying to target for permitting or denying access – again may be troublesome to debug and troubleshoot.

B. ACL Reordering

What if we could dynamically re-order ACL rules depending on usage, for example say the following part-Extended ACL was protecting a Web Server from Internet intrusion.

```
access-list 102 deny tcp any host 172.16.1.1 eq 3306
access-list 102 permit tcp any host 172.16.1.1 eq 8080
access-list 102 deny tcp any host 172.16.1.1 eq 1433
access-list 102 permit tcp any host 172.16.1.1 eq 3389
access-list 102 permit tcp any host 172.16.1.1 eq 23
access-list 102 permit tcp any host 172.16.1.1 eq 80
access-list 102 deny any any
```

Fig 8

Port Number	Description	Usage (%)
3306	mySQL Database	1.1
8080	Secondary Web Server	4.0
1433	Microsoft SQL Server	1.4
3389	Remote Desktop – Management	1.3
23	Telnet – Management	0.2
80	Primary Web Server	92.0
Total :		100

Fig 9

Port Number	Description	Usage (%)
80	Primary Web Server	92.0
8080	Secondary Web Server	4.0
1433	Microsoft SQL Server	1.4
3389	Remote Desktop – Management	1.3
3306	mySQL Database	1.1
23	Telnet – Management	0.2
Total :		100

Fig 10

From Fig 9 you can see that ordered by port number the usage is scattered with port 80 which has the highest usage is at the bottom of the ACL rules list, which means the router must step through each rule that is listed above it before I gets to the last line of the ACL to match a packet received on port 80 to. Now to optimize Fig 8 for speed, lower latency and lower delay it the ACL should be reordered as follows, with the highest usage rule being topmost and ordered downwards from there.

```
access-list 102 permit tcp any host 172.16.1.1 eq 80
access-list 102 permit tcp any host 172.16.1.1 eq 8080
access-list 102 deny tcp any host 172.16.1.1 eq 1433
access-list 102 permit tcp any host 172.16.1.1 eq 3389
access-list 102 deny tcp any host 172.16.1.1 eq 3306
access-list 102 permit tcp any host 172.16.1.1 eq 23
access-list 102 deny any any
```

Fig 11

As this has been done manually it isn't a problem but what if this function was done by an application or the router it's self and automatically depending on the network traffic at the time? I suppose it would be easier for the Network Manager but there's one rule within the ACL that worries me;

```
access-list 102 deny any any
```

Fig 12

What if the ACL was reordered incorrectly by an application or the router automatically? Ending up like Fig 13.

```
access-list 102 deny any any
access-list 102 permit tcp any host 172.16.1.1 eq 80
access-list 102 permit tcp any host 172.16.1.1 eq 8080
access-list 102 deny tcp any host 172.16.1.1 eq 1433
access-list 102 permit tcp any host 172.16.1.1 eq 3389
```

```
access-list 102 deny tcp any host 172.16.1.1 eq 3306
access-list 102 permit tcp any host 172.16.1.1 eq 23
```

Fig 13

The first rule in Fig 13 would deny all access to *anything and everything* on that network interface, even denying the access for management of the router from the outside! A complete Networking Disaster – well until somebody notices and reports to the Network Manager!

VI. CONCLUSION

Briefly, both ACL Optimization methods have their advantages and disadvantages, but both are currently done by hand rather than dynamically by an application or router, as an application or router could never get something so complex correct without a fair bit of processing power, number crunching and quite possible a lot of real-time data caption, logging and data analysis. Even a hybrid of both ACL Optimization methods would be advantageous for a corporate network and quite interesting!

REFERENCES

- [1] Cisco IOS Access Control List Syntax [online], Available: <http://techie.devnull.cz/aclcheck/aclsyntax.html>
- [2] Large example of a Border Gateway Router ACL [online] Available: <http://www.rpatrick.com/tech/acl/>
- [3] Cisco ACL Optimization [online], Available: http://www.cisco.com/en/US/products/sw/cscowork/ps402/products_use_r_guide_chapter09186a008008123d.html
- [4] Cisco ACL Syntax [online], Available: http://www.cisco.com/en/US/products/sw/secursw/ps1018/products_tech_note09186a00800a5b9a.shtml