Improving the Performance of IP Filtering using a Hybrid Approach to ACLs

John N. Davies, Vic Grout and Rich Picking

Centre for Applied Internet Research (CAIR)
Glyndŵr University, Wrexham, UK
j.n.davies@glyndwr.ac.uk
v.grout@glyndwr.ac.uk
r.picking @glyndwr.ac.uk

Abstract: With the use of policy based security being implemented in Access Control Lists (ACLs) at the distribution layer and the increased speed of interfaces the delays introduced into networks by routers are becoming significant. This paper investigates the size of the problem that is encountered in a typical network installation. Additionally since specialized hardware is not always available a hybrid approach to optimizing the order of rules in an ACL is put forward. This approach is based on the off-line pre-processing of lists to enable them to be re-ordered dynamically based on the type of traffic being processed by the router.

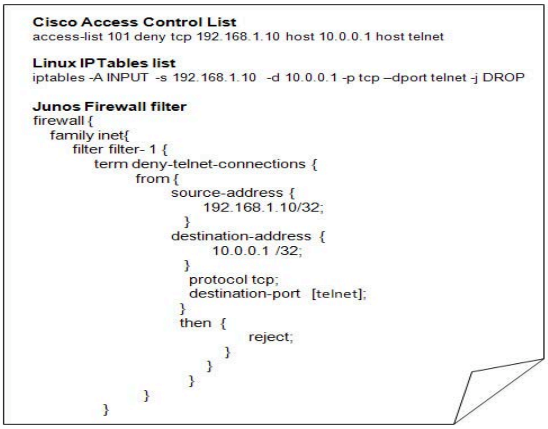
1 Introduction

Policy driven security is normally implemented at the network infrastructure level by the use of *Access Control Lists* (*ACLs*) in firewalls and routers [GM05]. This means that every packet that arrives or is transmitted by one of these devices has to be checked to see if the packet is acceptable or should be disgarded. Checking every packet provides the level of security that is required but there is a price to be paid in the overall performance of the network [St01, Mo99].

In general the more complex/detailed the policy the more detailed the ACL. Since the policy is defined by individual rules in a list, then the more detailed the requirement, the greater the number of rules that are necessary [GMD05]. This can have a number of effects, the possibility of an incorrect implementation resulting in leaving holes in the security and an increase in the delay to the packet being forwarded. These issues have been of concern for a number of years and attempts have been made to improve the situation either by providing management aids to re-write the lists or providing specialised hardware.

Access Control List rules are defined in a number of different formats depending on the hardware and operating system utilized e.g. Cisco IOS, Juniper Networks JUNOS (firewall filters) or Linux (Iptables). Figure 1 shows examples of Cisco IOS [Se01], JUNOS [MR08] and Linux [Pu04] to indicate the similarities.

Despite the differences in format the functionality is very similar; a packet is either forwarded or discarded based on the matching of fields in the packet with the specified rules. Rules are created from a permit or deny statement followed by variables that can include source address, destination address, protocol or port addressing within the protocol.



Cisco Access Control List
```
access-list 101 deny tcp 192.168.1.10 host 10.0.0.1 host telnet
```

Linux IPTables list
```
iptables -A INPUT  -s 192.168.1.10  -d 10.0.0.1 -p tcp –dport telnet -j DROP
```

Junos Firewall filter
```
firewall {
   family inet{
       filter filter- 1 {
           term deny-telnet-connections {
                   from {
                           source-address {
                                 192.168.1.10/32;
                              }
                           destination-address {
                                   10.0.0.1 /32;
                              }
                               protocol tcp;
                               destination-port  [telnet];
                             }
                             then  {
                                           reject;
                                       }
                                   }
                               }
                     }
           }
```

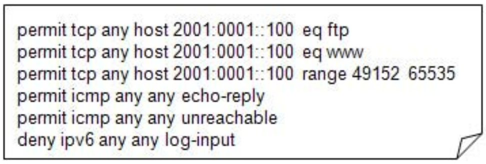
Figure 1: Comparison of rules used in different Operating Systems

Figure 1 applies to packets from one IP address, one protocol and application number. To implement a meaningful security policy then there have to be many such rules. Building a complete list for every conceivable host and protocol would result in an enormous number of rules $=232 \times 232 \times 255 \times 65,535$; however, Cisco, for example limit the operational size to 250,000.

To alleviate this problem there are operators that can be utilised to reduce the number of rules, for example a wild card mask, the lt (less than), gt (greater than), neq(not equal) and range operators can be used to increase the range that this rule applies to. This reduces the number of rules in the list; however, it increases the complexity of each rule. Additionally, at the end of the list is an implicit Deny everything [Co02].

Due to the manual implementation of lists, incorrect implementation of ACLs is clearly of great concern and this is normally addressed by exhaustive testing both on and off line and products exist to aid this process [Ag04]. Monitoring the network for intrusions has become best practice. The main concern of this paper however is the effect that ACLs have on network performance.

When considering data traffic there are 2 levels of addressing that are required; the MAC address at the data link level and the IP address at the network level. At level 3 of the OSI model, the network layer, the current implementation is IP version 4 (IPv4) and so this paper discusses the issues associated with this protocol. In the future the network layer is likely to change to IP version 6 (IPv6) or ISO addressing. Even though the application of ACLs to these protocols is very similar, (Figure 2) [Ci08], the analysis has been left for future work since they only exasperate the current problems.



```
permit tcp any host 2001:0001::100  eq ftp
permit tcp any host 2001:0001::100  eq www
permit tcp any host 2001:0001::100  range 49152  65535
permit icmp any any echo-reply
permit icmp any any unreachable
deny ipv6 any any log-input
```

Figure 2: A typical Ipv6 Access Control List

2 Network Design

When designing large networks, if the Cisco recommended philosophy is adopted (Figure 3), then the core of the network is built using very powerful routers containing optimized hardware intended for moving packets from one port to another very quickly. It is recommended that these routers are not used for implementing security since this would slow down the switching speed of the core network. *Border routers* are employed as gateways into the *autonomous networks* [Ci08a].

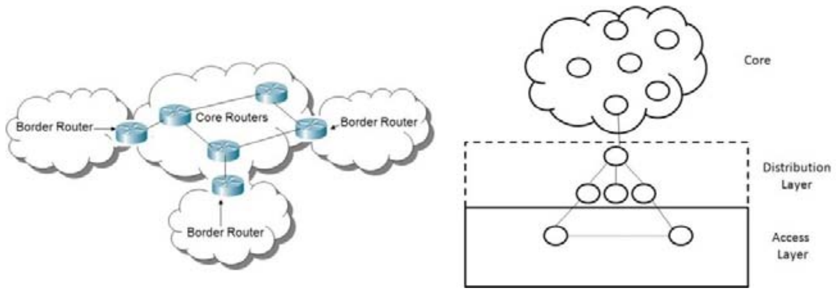


Figure 3: Network design philosophy

A security policy would be defined for each Autonomous network and part of this policy would be implemented at the distribution layer of the network within the border and associated internal routers. This means that the hardware used for this security policy is often not optimised for this purpose. Investigations associated with these problems and possible improvements are the subject of this paper.

2.1 IP Routing

IPv4 addresses are 32 bits, but dependent on whether classful or classless addressing is used, they are augmented by up to a 32 bit network mask. Routing of packets from one network to another is performed by comparing the destination address with an entry in a routing table in each node. As far as routing is concerned, then for every destination address in the packet, there will exist either an interface number on the device on which to forward the packet or no entry when the packet will be dropped or sent down a specified interface. Entries into the routing table are made either by manual configuration or by the Routing Protocol employed in the network.

Implementation of this look up is normally carried out by using specialised hardware in a similar way to that used in switches. Level 2 switches filter on MAC addresses which are 48 bits. An exact match for the 48 bits has to be found and so a Content Addressable Memory (CAM) is used. This memory based device performs parallel content comparisons to find a valid match. Dependant on the size of the application and the technology used for 48 bits this can be completed in ~15nsecs [Xi08].

The same memory device can be configured to act as a Ternary Content Addressable Memory (TCAM) which allows the network mask to be utilised providing the ability to programme 1,0 & X ('don't care' bits). With a single access, the value returned can be configured to the lowest address, which in terms of networking is defined as the network address for that area. These devices are optimised for matching values since this has to be done for each packet handled and so has an effect on the overall performance of the network. This however does affect the initial loading of the memory that can take a relatively long time.

2.2 Effect of Access Control List on packets delay

Access Control Lists are placed on the interfaces of the routers and are configured to apply to either incoming or outgoing packets on that interface. So, following the routing decision to select the interface to be used for forwarding the packet, then the next step is to decide whether the packet should be permitted or denied, in line with the list.

By configuring a network and measuring the time delay experienced by packets traversing a router it is possible to get a feel for the size of the problem [DG05]. Figure 4 shows a comparison of the time differences of packets being passed through a router subjected to initially no ACL, then a Standard ACL with 100 rules and then an Extended ACL with 100 rules. It can be seen that implementing a Standard ACL can increase the overall delay through a router by around 10%, and for an Extended ACL 85%. There is therefore a large time difference and so an area for improvement.

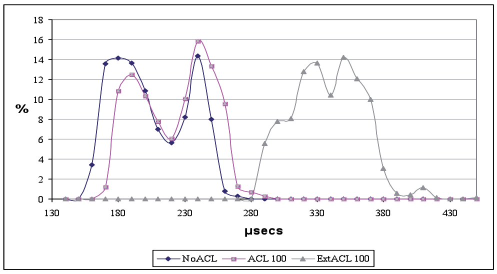


Figure 4: Distribution of delay times (µsecs) when implementing ACLs

3 Scope of the Problem

When analysing the packet it is normal to look at the number of bytes, which can then be converted to times, by considering the technology involved. Considering a typical TCP/IP packet the following applies:

$$B_{pkt} = B_{mac} + B_{ip} + B_{tcp} + B_{data} + B_{crc} \qquad (1)$$

Where $Bpkt$ the total number of bytes in the packet, B_{mac} the bytes in the MAC layer, B_{ip} the bytes in the IP layer, B_{tcp} the bytes in the TCP layer, B_{data} the bytes of data or padding and B_{crc} the checksum. There is also an inter-frame gap that governs the minimum time at which packets could arrive at an interface for a given direction B_{ig}.

3.2 Decision Time for Routing

Packets found on the Internet vary greatly, but analyzing a typical TCP/IP packet shows that the size can vary from 64 to 1518 bytes of data. This is made up of 3 layers of network overhead plus data. From the point of view of network performance the worse case conditions are for minimum size packets.

In an ideal world the routing decision for a packet would be made before the arrival of the next packet (*line speed*). This would give the best network performance and would reduce the size of the input buffers required. So for a TCP/IP packet with $B_{pkt}=64$ it would only be possible to start matching the destination IP address after $B_{mac}+B_{ip}$ i.e. 34 byte times.

In reality this does not happen since the normal implementation of the interfaces, e.g. Ethernet, is that since data is moved using direct memory access (DMA) to reduce the load on the processor, the processing can only start when the complete packet has been received. This leaves only the inter-frame gap B_{ig} that is 12 bytes before the next packet could start to arrive. Even so, as long as the input buffer is sufficiently large to store the new packet, then the previous packet does not have to be handled for a minimum of $B_{pkt}=64$ bytes.

To convert this to a time, then it would be $B_{pkt}+B_{ig}\times 8$ = 608 bit times i.e. 6.08 msecs for a 100 Mbps network. Clearly the input buffer size will influence the window size being used by TCP.

3.3 Speed of access

Ethernet connections to the desktop run at 100Mbps and 10Gbps is relatively common for backbone connections. In 2002 the IEEE ratified the 10 Gigabit Ethernet 802.3ae specifications and at present the IEEE Ethernet Task Force are investigating the use of 40Gbps and 100Gbps, the latest draft of the 803.2ba standard was released in November 2009 [IE09].

Bandwidths of 100 Mbps, 10Gbps, 40 Gbps and 100Gbps relate to a device having to clock data into a buffer at 10nsecs, 0.1nsecs, 0.025nsec and 0.01nsecs respectively. So 608 bit times relates to 6msecs, 60nsecs, 15nsecs and 6nsecs. This indicates that consideration needs to be given to the computer network theory, which implies the line speeds are the bottlenecks, whereas with these line speeds the decision process is clearly a significant delay.

When an ACL is used then further fields are required, usually the protocol and port number associated with the protocol. The protocol part of the IP packet appears 2 bytes before the start of the IP addresses and the port number is in the 4 bytes following the addresses. Applying an ACL to an interface there is a requirement for the processor in the router to take action and there is an overhead associated with this.

4 Implications on Access Control Lists

4.1 Analysis of ACL problem

Depending on the hardware and operating systems deployed, there are restrictions governing the quantity and placement of ACLs, e.g. each router will have a number of interfaces i and each interface will have two directions, in(Rx) and out(Tx). A packet has a source address (Sa) and a destination address (Da), a protocol (Pr) and another condition (Pn) e.g. port number. For any packet on a particular interface, for a given direction, there is a possibility of 2 decisions, pass or discard. As a result of performing the routing function then values for i and either Rx or Tx are obtained. There is generally a restriction that only allows 1 ACL to be applied to each of these.

4.2 Designing, Writing and Editing ACLs

ACLs have to be designed to ensure that the declared security policy is met. This is normally done manually by network personnel with the aid of simple text editing or GUI based tools [Ev10]. Attempts have been made by manufacturers to optimize the lists e.g. CiscoWorks [Ci04], which helps increase packet-forwarding speeds by removing redundant entries and appropriately merging and consolidating rules. However due to the limitations of the optimization and the original context being lost, it was not very successful and has now been discontinued.

4.3 Compiling lists at Run time

A feature available on a very limited number of high-end switches and routers is Turbo ACLs [GS02]. This enables an ACL created normally to be compiled at run time into a series of tables which are used as a look up, resulting in a matching being made in 5 steps irrelevant of the number of rules [RBM01].

4.4 Switching Hardware

TCAMs are used in routing packets and so they are an obvious choice when looking for a hardware solution to improve the performance of ACLS. Unfortunately, there are a number of problems that have to be addressed associated with their use, particularly the complexity of supporting logic, memory size (4096 words) and cost [Xi08]. They are programmable devices so this extends their applicability; however, this cannot really compensate for the possible memory size requirement of each protocol on every directional interface. Despite this it is anticipated that this will be used in the future implementations.

4.5 ACL handling with Route Lookup

Certain switches have the ability to utilize line-cards that contain *Application-Specific Integrated Circuits* (*ASICs*) [Ci07]. When these line-cards are fitted it can process input ACLs alongside route lookup without using the processor. This results in increased performance when compared to Linear ACL and Turbo ACL processing. Since this is restricted to input packets and has card memory size limitations, it can only handle around 128 rules, so this is not a general purpose solution.

5 Linear ACLs

The hardware implementations discussed above are limited at present, it is therefore important to look at possible steps that can be taken to improve the performance with existing equipment. Many existing implementations rely on the ACL being compared in a linear fashion because this is the way the hardware has been designed. This paper presents a hybrid approach by optimizing the rules off-line before loading them into a real time system.

5.1 Incremental Rule Approach

The simplest implementation of an Access Control list is to compare the packet with each rule in turn until there is a match for the packet parameters. A match for each packet must exist since the final rule of a list can be assumed to be an implicit deny of all packets.

Since the number of rules compared before a match can be found, has an effect on the performance of the network, the ordering of rules in the list is significant. Carrying out similar measurements to those described in 2.2 [Da05] but with the matching being forced to occur after 10, 50 and 100 rules it can be seen that the delay is increased by 10%, 45%, and 85% respectively (Figure 5).

Network performance will depend on the order in which the rules appear in the list. If most packets match the first rule in the list then the performance would be optimised.

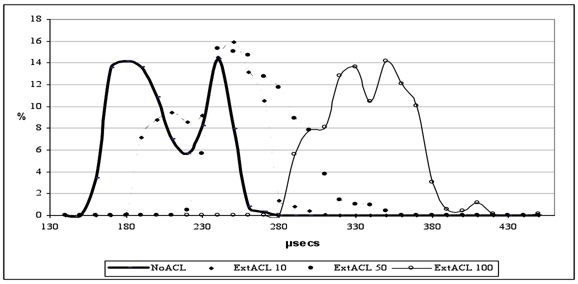
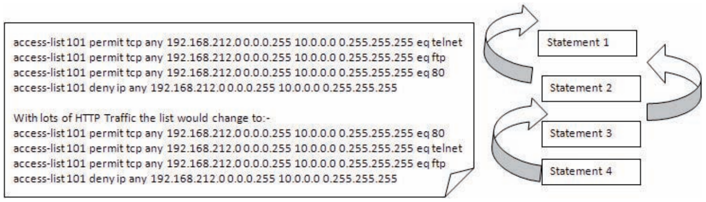


Figure 5: Effect of number of rules executed on delay

5.2 Promotion of Rules

If it were possible to change the order of the statements in an ACL depending on the type of traffic, then this would improve the overall performance of a network (Figure 6). Cisco considered this in their ACL manager by utilizing the ACL Hits Optimizer, it placed the most frequently hit rules ahead of the less frequently 'hit' rules based on network packet matches [Ci04]. But the reordering of the rule is only performed if the new order does not change the ACL semantics, which is dependant on the way in which the list has been designed [SQ03]. Clearly the overall effect of this is variable.



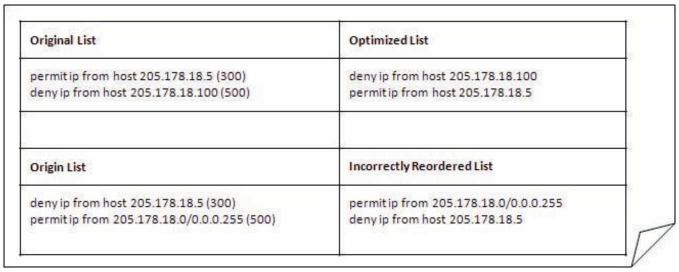
```
access-list 101 permit tcp any 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255 eq telnet
access-list 101 permit tcp any 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255 eq ftp
access-list 101 permit tcp any 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255 eq 80
access-list 101 deny ip any 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255

With lots of HTTP Traffic the list would change to:-
access-list 101 permit tcp any 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255 eq 80
access-list 101 permit tcp any 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255 eq telnet
access-list 101 permit tcp any 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255 eq ftp
access-list 101 deny ip any 192.168.212.0 0.0.0.255 10.0.0.0 0.255.255.255
```

Figure 6: Promotion of rules

5.3 Restructuring of Rules

Figure 7 shows examples where re-ordering of rules can and cannot be carried out. In the bottom case, even though the fit list shows that the rules should be re-ordered (300,500), the rules are dependant and so the original semantic of the list would be contravened. The proposal of this paper is to remove the grouping of the rules. This can be done by restructuring the list to produce either: all the rules as permit statements with a deny all rule at the end or the list as deny statements with a permit all rule preceding the implicit deny rule at the end. This will enable each rule (other than the last statement) to be promoted without limitation.



Original List	Optimized List
permit ip from host 205.178.18.5 (300)	deny ip from host 205.178.18.100
deny ip from host 205.178.18.100 (500)	permit ip from host 205.178.18.5
Origin List	**Incorrectly Reordered List**
deny ip from host 205.178.18.5 (300)	permit ip from 205.178.18.0/0.0.0.255
permit ip from 205.178.18.0/0.0.0.255 (500)	deny ip from host 205.178.18.5

Figure 7: Incorrect re-ordering of ACL

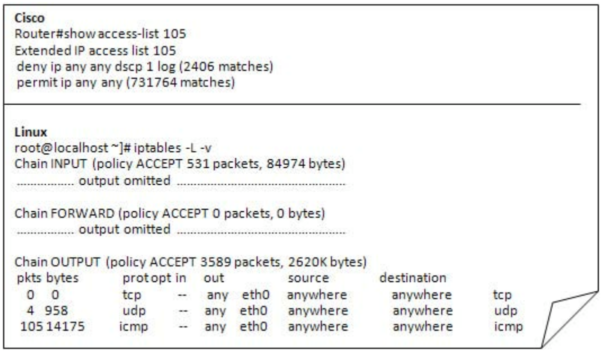
To meet this requirement it is possible that the design of the list could be changed; however it is more likely that an off-line restructuring system would be used to re-write the list. The system would create a permit/deny matrix for every source/destination and protocol that clearly would be very large. Populating the matrix could either be carried out by the use of a GUI or from existing lists. A new list would be created by considering the original list in reverse order i.e. starting with the implicit deny all and working through each rule in turn until the top is reached. As the list is considered, elements of the matrix may be written to more than once, but since it is carried out in the reverse order then the original semantics are maintained. This computed list can then be loaded as the ACL into the router in the usual way.

5.4 Packet Monitoring

The Cisco '*show ip access-list interface*' [Ci06] and the Linux '*iptables –L –v*' [Pu04] can be used to monitor the number of packets matching a rule (Figure 8). Based on this value a decision whether to promote a rule can be made. Hence the most used protocols migrate to the top resulting in less delay through the router. Since ACLs are updated rarely and the traffic passed through routers is highly variable the ideal situation would be for this promotion to take place dynamically. This can be achieved very efficiently [GDM07, GD10].

5.5 Dynamic Promotion of Rules

This philosophy has been implemented in a Linux environment to substantiate the approach [DG08]. An iptables set of rules was set up using a number of accepts for different protocols with a final deny. This structure was used since packets that are to be forwarded will get the better performance than packets that are going to be rejected. A program is run to continually monitor the number of packets that are handled by each rule. Based on the packet value a decision is made whether to promote that rule since the structure has been designed so that semantics are not compromised. Initially this decision moves the rule up the list by one place and the counters are reset ready for the next monitoring.



```
Cisco
Router#show access-list 105
Extended IP access list 105
 deny ip any any dscp 1 log (2406 matches)
 permit ip any any (731764 matches)

Linux
root@localhost ~]# iptables -L -v
Chain INPUT (policy ACCEPT 531 packets, 84974 bytes)
................ output omitted ...............................................

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
................ output omitted ...............................................

Chain OUTPUT (policy ACCEPT 3589 packets, 2620K bytes)
pkts bytes      prot opt in   out          source          destination
  0   0          tcp     --   any  eth0   anywhere         anywhere       tcp
  4  958         udp     --   any  eth0   anywhere         anywhere       udp
 105 14175       icmp    --   any  eth0   anywhere         anywhere       icmp
```

Figure 8: Monitoring Rules hits in a list

6 Conclusions

With the utilization of high speed network interfaces the delays introduced by router processing has become significant and calculations show that hardware implementations are necessary to maintain these speeds. Due to the security policy setup in autonomous systems being implemented in ACLs, packets are subjected to many tests and therefore a great deal of thought has to given to performance issues. Unfortunately most routers being used at present in the distribution layer are not equipped with this capability in hardware so an investigation of alternatives is valid. These alternatives require the optimization of the rules used that form the list.

Ideally, lists should be optimised based on the packets that are being handled by the router. Due to the mix of packets and protocols utilized in networks, the most advantageous way of carrying this out is dynamically, i.e. monitor the type of traffic and change the order of the rules in line with the protocols being handled. To ensure the integrity of the list, the order of the rules must be capable of being changed without compromising the original semantics. For flexibility it should apply to all rules. The hybrid approach taken in this paper is to optimize the rules off-line before loading them into the real time system.

There are some issues that need to be addressed to help improve the overall performance of such a system. Continual monitoring of the list can adversely affect the performance of the router since it is using processor time so some consideration to the repetitive rate needs to be given. Since the maximum size of ACLs can be of the order of 250,000 a rule that has a high hit rate could take a long time to reach the top. It is possible in future to reorganize the list so that rules are moved more than one position. The classic 'chatter' problem can occur where certain rules are continually changing places with each other. With a little thought, hysteresis or a back off value, can be built into the algorithm to avoid this problem. Future work in this area is proposed.

References

[Ag04] Agilent Technologies Inc: *N2X Packets and Protocols Testing Access Control Lists*, Application Note http://cp.literature.agilent.com/litweb/pdf/59891092EN.pdf: 2004.

[Ci04] Cisco Systems, Inc.: *CiscoWorks Access Control List Manager 1.6*: http://www.cisco.com/en/US/products/sw/cscowork/ps402/products_data_sheet09186a080092440.html: Feb 2004: Accessed February 2010.

[Ci06] Cisco Systems, Inc.: *Using Network-Based Application Recognition and ACLs for Blocking the 'Code Red' Worm* Aug 2006, Doc ID: 27842: Accessed February 2010.

[Ci07] Cisco Systems, Inc.: *Implementing Access Lists on Cisco 12000 Series Internet Routers*: Feb 19, 2007, Document ID: 40742: Accessed February 2010.

[Ci08] Cisco Systems, Inc.: *Implementing Traffic Filters and Firewalls for IPv6 Security*: http://www.cisco.com/en/US/docs/ios/ipv6/configuration/guide/ip6sec_trfltr_fw.html: 2008: Accessed February 2010.

[Ci08a] Cisco Systems, Inc.: *Protecting Your Core: Infrastructure Protection Access Control Lists*: Oct 21, 2008 Document ID: 43920: Accessed February 2010.

[Co02] Colton, A.: *Cisco IOS for IP Routing*, Rocket Science Press Inc., 2002.

[DG05] Davies, J.N.; Grout, V.: Network Monitoring and Measurement, *First International Conference on Internet Technologies and Applications (ITA 05)*, Wrexham, 2005.

[DG08] Davies, J.N.; Grout, V.: Restructuring Rules in ACLs to Allow Dynamic Optimisation, *9th INFORMS Telecommunications Conference*, University of Maryland, USA, March 2008.

[Ev10] Evans, G.: Cisco *ACL Editor and Simulator*: http://www.garethevans.info/products/acleditor: Accessed February 2010.

[GD10] Grout, V.; Davies, J.N.: A Simplified Method for Optimising Sequentially Processed Access Control Lists, *Proceedings of the Sixth Advanced International Conference on Telecommunications (AICT 2010)*, May, 2010, Barcelona, Spain.

[GS02] Greene, B.R.; Smith, P.: *Cisco ISP Essentials*: Cisco Press Networking Technology Series, 2002.

[GMD05] Grout, V.; McGinn, J.; Davies, J.: Real-Time Optimisation of Access Control Lists for Efficient Internet Packet Filtering, *Journal of Heuristics*, Vol. 13, No. 5, October 2007.

[GM05] Grout, V.; McGinn, J.: Optimisation of Policy-Based Internet Routing using Access Control Lists, *9th IFIP/IEEE Symposium on Integrated Network Management* May 2005 Nice, France.

[GDM07] Grout, V.; Davies, J.; McGinn, J.: An Argument for Simple Embedded ACL Optimisation, *Computer Communications*, Vol. 30, No. 2, January 2007.

[IE09] IEEE P802.3ba 40Gb/s and 100Gb/s Ethernet Task Force http://www.ieee802.org/3/ba/; July 2009

[MR08] Marschke, D.; Reynolds, H.: *JUNOS Enterprise Routing*, O'Reilly Media, 2008

[Mo99] Morrissey, P.: The Cost of Security on Cisco Routers. Network Computing, February 1999.

[Pu04] Purdy, G.: Linux Iptables, O'Reilly Media Inc., 2004

[RBM01] Ruiz-Sánchez, M.; Biersack, E.W.; Miguel, Á.: Survey and Taxonomy of IP Address Lookup Algorithms, INRIA, Sophia Antipolis, France January 15, 2001

[Se01] Sedayao, J.: Cisco IOS Access Lists, O'Reilly Media Inc., 2001

[SQ03] Shih, C-S.; Qian, J.: Security Policy Derivation, in CS497: Cryptography and Computer Security, University of Illinois at Urbana Champaign, 2003 http://www.sal.cs.uiuc.edu/~steng/cs497_01/qian.ppt.

[St01] Stoica, I.: Route Lookup and Packet Classification, CS 268, Department of Electrical Engineering and Computer Science, University of California, Berkeley USA. Feb 2001.

[Xi08] Xilinx, Inc.: Content-Addressable Memory v6.1 Product Specification DS253 2008