Internet Service Providers: The Little Man's Firewall.
A Case Study in ISP Port Blocking

Author: Luke Dudney

GSEC Practical Assignment v1.4b Option 2
09 December 2003

Abstract

There has recently been call for Internet Service Providers to begin filtering traffic related to the spread of malicious data traffic such as viruses, worms and open proxy abuse to and from their end-users. This case study outlines the planning, implementation, and results phase of such an endeavour by a medium sized national Australian ISP. It illustrates that a significant improvement in the security of the ISP network, end-user connections and indeed the Internet as a whole may be achieved by filtering access to ten TCP/IP ports extensively targeted by this malicious data traffic. By providing an "opt-out" mechanism for those end-users that do not wish to have such filtering applied, this heightened security is possible without negatively impacting connectivity or functionality.

This document is intended to be a high level case study in order to have relevance beyond the scope of the specific organisation, but at the same time provide enough detail to serve as a good illustration and reference document. While the implementation phase details this process as it applies to Cisco access server hardware, the general concepts explored are still valid across other platforms.

Existing Issues

Internet Worms

One of the major issues facing security professionals in the contemporary Internet is the proliferation of malicious code such as Internet worms and computer viruses. Such code regularly causes major problems on the global Internet by consuming network and system resources thus jeopardizing the availability, confidentiality and integrity of the data of connected computer systems. Predominantly, platforms targeted by these worms and viruses are the Microsoft (MS) Windows line of operating systems. Historically, these products are released to consumers with multiple security vulnerabilities on ports exposed to the Internet by default. Additionally, these MS operating systems make up the vast majority of Internet connected desktop systems which prepossess them an attractiveness to virus and worm authors. While patches for these security vulnerabilities are typically released soon after they are disclosed by the vendor, the fact remains that a very large number of Internet users are either lax or ignorant of the need to keep their operating systems up to date to protect against the latest threats. Recent efforts by Microsoft to enhance the security of their products such as their "Trustworthy Computing" initiative have been promising however, in the short term, have done little to stem the tide of malicious data traffic exploiting security vulnerabilities in their products.

One of the most devastating worms discovered in the wild in recent times has been the W32/Blaster worm, also known as the W32/Lovsan worm, which exploits a flaw in Microsoft's RPC implementation discussed in Microsoft Security Bulletin MS03-026 *Buffer Overrun In RPC Interface Could Allow Code Execution (823980)*. The worm attacks vulnerable services providing RPC over TCP. In MS03-026 Microsoft state that "RPC over UDP or TCP is not intended to be used in hostile environments such as the Internet"[1] and that they should be blocked by the firewall. On this issue the US Department of Homeland Security "...recommends filtering all network traffic that is not required for normal operation"[2] and CERT similarly recommends "that system administrators filter the ports listed above for both incoming and outgoing traffic"[3]. The exploited vulnerability in this

[1] http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp
[2] http://www.nipc.gov/warnings/advisories/2003/Advisory9102003.htm
[3] http://www.cert.org/advisories/CA-2003-23.html

instance was of a service that had no practical application on the public Internet and should therefore have ideally been blocked at the perimeter by the end-user. While the vulnerability and associated patch were released on July 16 2003, the fact that the worm was not released and discovered until August 11 2003 and the large number of end-users who became infected is indicative of the fact that timely patch application and the unnecessary exposure of vulnerable operating system services are recurrent major security problems among consumers. End-users are not heeding this important security advice and are continuing to deploy vulnerable systems or are not sufficiently updating and protecting existing systems. The ISP network resources consumed by this worm and the lost productivity for end-users due to infected and crashing systems was extensive, with some estimates putting the damage done by viruses in August (the month Blaster was released) at $US38 billion [4].

The official policy of the technical support centre at the ISP is that they are not able to offer assistance to end-users who are experiencing virus related issues such as crashing computers due to infection. Even with this policy in place, massive numbers of end-users still contacted technical support only to be informed that they were not able to receive any assistance. Security issues aside, the customer relations problem of such a policy during an outbreak was great and resulted in a lot of needless ill-will towards the ISP.

Open Proxy Hijacking

Another serious issue facing security specialists is the proliferation of spam email on the Internet. Spam, loosely defined as unsolicited bulk email, involves the mass emailing of advertising material to recipients who in most cases neither requested the material nor wish to receive it. Due to the architecture of the Internet it is the recipient that bears the vast majority of the cost of receiving the message rather than the sender. The cost is distributed among bandwidth, server resources lost time and productivity. This “recipient pays” system makes the sending of spam an extremely attractive opportunity for some of the less scrupulous Internet marketers -- indeed, so much so that spam is at such epidemic levels that it now “threatens to undermine the utility of email itself” [5].

While spam was in its infancy in the mid 1990's the spam messages were generally sent from dedicated hosting servers but it was not too long before most of these hosting servers were being blocked by the recipients – in essence the spammers were sitting ducks. After this the war against spam turned into an arms race with spammers becoming increasingly sophisticated in their delivery methods while anti-spammers and spam recipients responded with increasingly sophisticated blocking methods. Spammers started using open mail relay servers to send their mail in an effort to hide their tracks and this method is still frequently used today. While an open mail relay initially hides the real sender's location, that information is generally still available in the headers of the spam message making it easy to direct complaints to the owners of the offending IP address. More recently, however, spammers have begun making use of open proxy servers.

An open proxy server is defined as “a computer that accepts connections from anyone, anywhere, and forwards the traffic from those connections as if it had originated locally from that host” [6]. The use of these proxy servers makes the job of tracking the spammer even more difficult as there is no way for the recipient to discover the identity of the original sender without access to the proxy server's log files which are typically not available to the recipient. Viewing the headers of the spam message, it appears as if the message originated from the open proxy server.

[4] http://australianit.news.com.au/articles/0,7204,78262 16%5E16681%5E%5Enbv%5E,00.html

[5] http://www.iia.net.au/nospam/

[6] http://darkwing.uoregon.edu/~joe/proxies/

In an end-user network a proxy service is generally only required to be exposed to the local area network (LAN) and not the public Internet to provide Internet access for the LAN. End-users often install proxy software on their PCs without even realising what they are doing or configure that software as an Internet facing open proxy server. It is these open proxy and open relay servers that cause the majority of headaches not only to the ISP's abuse handling department but to the performance and availability of the ISP's network services.

Spam runs bounced off end-user proxies to ISP mail servers can overwhelm the ISP's servers with spam and may also result in these servers being listed on public blacklists for some time. In one incident multiple end-user open proxies were exploited at the same time resulting in delays of up to eight hours for customer mail as well as addition to the DNSBL.ORG real-time blacklist resulting in further denial of service.

To gauge the severity of the open proxy problem and the time it would take for an open proxy to be discovered and exploited, an open proxy honeypot was deployed. After some research it seemed that the best software for the job was Bubblegum Proxypot available at http://world.std.com/~pacman/proxypot.html . This software emulates an open proxy server speaking the HTTP CONNECT, HTTP POST, SOCKS v4 and SOCKS v5 protocols, offers very granular control on the workings of the proxy such as resources limits and the logging of the “catches”. It doesn't actually pass any email message on to the intended recipient. Configuration options include bandwidth shaping, limiting the number of connections to individual hosts, /24, /16 and /8 address ranges, and delivery to a local Maildir directory.

The honeypot was installed at 5pm one afternoon. By 1:00am the next morning -- just over nine hours later -- the proxy had been discovered and was being actively exploited by spammers. The below table lists the statistics gained from the honeypot log files over the first 25 hours of operation.

Hour	***Messages Received***	***Recipients***
0-1	0	0
...		
8-9	0	0
9-10	97	1,441
10-11	654	9,810
11-12	726	10,874
12-13	660	9,888
13-14	575	8,643
14-15	733	10,989
15-16	798	11,974
16-17	877	13,154
17-18	873	13,219
18-19	6,031	91,716

Hour	*Messages Received*	*Recipients*
19-20	7,186	107,911
20-21	5,466	81,962
21-22	5,044	75,620
22-23	8,307	124,537
23-24	7,230	108,404
24-25	7,006	105,377
TOTAL	52,263	785,519

Table 1 -- Open Proxy Honeypot Statistics

When the honeypot was shut down sixty-six hours later the numbers had blown out to 229,468 individual messages sent to 3,360,181 recipients. These statistics paint an alarming security picture of the danger facing administrators installing proxy software and the ISPs through which they get their connectivity. As evidenced by the above activity, open proxy abuse is a huge security problem facing both end-users and ISPs. Combine this with the problems faced by open mail relay servers, and the spam scourge looks unbeatable.

Open Mail Relay Hijacking

Open mail relay servers are SMTP server programs that permit the relaying of mail “that is neither for nor from a local user,” making it “possible for an unscrupulous sender to route large volumes of spam.” [7]. Many end-users, typically businesses, local government, or educational organisations, decide that the scale of their network requires that they host their own mail server on their own premises at the end of their Internet connection. While this decision may in many cases be justified there are often insufficient IT security skills and experience on-site to maintain a publicly accessible mail server. Often the server is configured and appears to be functioning correctly as it accepts mail as expected where in fact the administrator has configured the server to permit open relaying. There have been incidents where these servers were configured to forward all mail through the ISP's mail servers and exploitation of these open relays has resulted in spam being relayed through the ISP's servers and their subsequent addition to RBLs resulting in further denial of service to end-users.

Figure 1 below is a simplified illustration of the network topology involved. Multiple Cisco LNS (L2TP Network Server) servers provide the endpoints for Layer-2 services purchased from a third party wholesale PSTN dial up modem provider and a third party wholesale xDSL provider. End-user connections are physically terminated by these third party providers who then bring up L2TP tunnels to the Cisco LNS access servers which provide the logical endpoint to the tunnel. As far as the end-user is concerned they are connected directly to the LNS. There are multiple points of presence within the ISP's network and the topology in each is similar enough to be represented by this single figure.

[7] http://whatis.techtarget.com/definition/0,,sid9_gci782509,00.html

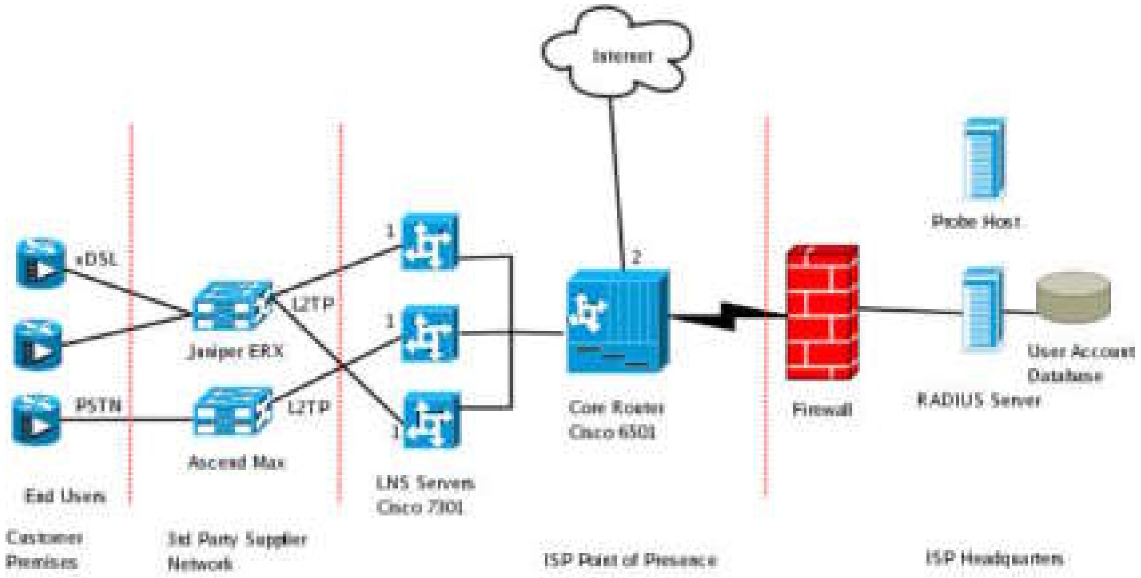

Figure 1 – Simplified ISP Network Diagram

1. *Virtual-Access Interfaces*
2. *Upstream Internet Connection Interfaces*

Implementation

The problems described in the preceding section would seem at first glance to be extremely difficult to eliminate. One of the key concepts learnt in the SANS Security Essentials track was that of "defence in depth" which argues that multiple layers of defence combine to provide heightened security. Most end-users have no layer of defence whatsoever. Johannes Ulrich's paper "Internet Service Providers: The Little Man's Firewall?"[8] argues that ISPs are in a unique position to provide one of those layers of defence, but to what extent can this protection be provided and when does it start to interfere with legitimate use?

Investigations into the problems described above reveals a similar trait between open proxy abuse and the exploitation of the RPC DCOM vulnerabilities by the Blaster worm – they both utilise IP service ports that see application on LANs or secured WANs but little if any legitimate use on the public Internet. Implementing some sort of packet filtering on these ports would greatly improve the security situation, and Johannes Ulrich's paper provided a firm basis from which to investigate this course of action further. After discussions among all relevant departments at the ISP including management, network operations, technical support, and network abuse handling, the following were set out as resolute requirements for such a course of action:

1. The filtering would need to be applied to all end-users by default;
2. Only ports that see little or no legitimate use on the public Internet and are regularly targeted by malicious activity should be filtered;
3. Any port filtering done would need to be done with an absolute minimum of negative end-user impact;
4. The method employed should not have any significant impact on network hardware resources such as routers and network access servers;
5. Any port filtering done would need to be able to facilitate disablement by the end-user at their own convenience.

[8] http://www.sans.org/rr/special/isp_blocking.php

The reasoning behind requirement 1 is that if the service were "opt-in," only end-users who are technically astute enough to understand what the service achieves would enable it and presumably they would already have some sort of defence in place. Non technically astute users would be unlikely to enable the service, defeating the point of the exercise. Points 2&3 recognise that while the ISP understands that significant security improvements are possible with this system, they may also severely impact customer connectivity and, subsequently, increase the call load on the ISP's technical support department. Requirement 4 concedes that the service is provided gratis thus it is not practical to purchase faster hardware to cater to this endeavour. Point 5 recognises that some end-users oppose any sort of filtering by the ISP on principal, while others may have need for these ports to be open and so makes available to them the functionality they require.

Data related to the activity reported to the abuse handling department were analysed and, as a result, the following ports were singled out to be filtered:

Port (protocol)	***Direction***	***Service Description***
135(tcp/udp)	Incoming/outgoing	Windows Networking
137 (udp)	Incoming/outgoing	Windows Networking
138 (udp)	Incoming/outgoing	Windows Networking
139 (tcp)	Incoming/outgoing	Windows Networking
445 (tcp/udp)	Incoming/outgoing	Windows Networking
593 (tcp)	Incoming/outgoing	Windows Networking
1080 (tcp)	Incoming/outgoing	SOCKS proxy
3128 (tcp)	Incoming	HTTP Proxy
8080 (tcp)	Incoming	HTTP Proxy
6588 (tcp)	Incoming	HTTP Proxy

Table 2 – Intended Access Controls

This set of services represents a balance between heightened security and other business considerations such as technical support call load. The HTTP proxy ports are only blocked in the incoming direction as there is a legitimate use for the traffic to flow in the outgoing direction. For example, an end-user might be required to access a corporate or ISP proxy server on these ports but it is unlikely that they would be hosting a proxy server on their connection for this reason due to bandwidth limitations. If for some reason they are using their connection in this way, requirement 5 above facilitates a way for the end-user to enable this.

There are a much larger number of ports exposing vulnerable services such as port 80 (www) or port 21 (ftp) and blocking these ports would improve the security of the customer connections and the ISP network however doing so would also cause considerable support issues as they do have widespread legitimate use on the Internet. The line has to be drawn somewhere lest the logical conclusion to the process be reached – an Internet connection with all ports blocked!

Consultation with the head network administrator revealed that there are multiple options available for implementing such port filtering for end-users. Some of the options considered are listed below, along with a discussion on the perceived benefits and drawbacks and the reasons behind these perceptions.

The implementation of the filtering is done via Cisco extended access lists. The following implement the required access controls.

“outgoing” access list:

1. permit tcp any any established
2. permit ip any host probehost.example.com
3. deny tcp any any eq 135
4. deny tcp any any eq 139
5. deny tcp any any eq 445
6. deny tcp any any eq 593
7. deny tcp any any eq 1080
8. deny tcp any any eq 3128
9. deny tcp any any eq 6588
10. deny tcp any any eq 8080
11. deny udp any any eq 135
12. deny udp any any eq 137
13. deny udp any any eq 138
14. deny udp any any eq 445
15. permit ip any any

“incoming” access list:

1. permit tcp any any established
2. permit ip host probehost.example.com any
3. deny tcp any any eq 135
4. deny tcp any any eq 139
5. deny tcp any any eq 445
6. deny tcp any any eq 593
7. deny tcp any any eq 1080
8. deny udp any any eq 135
9. deny udp any any eq 137
10. deny udp any any eq 138
11. deny udp any any eq 445
12. permit ip any any

There are points to note about the composition of the access lists. Rule 1, “permit tcp any any established,” is important as it bypasses the remainder of the access list for any established TCP connection which both improves performance but also permits established connections using these ports to continue. Any TCP session which has already been established (and thus must have passed the access list) is not required to be checked against each rule of the access list. For example, if an end-user initiates a connection to a remote web server on port 80(tcp) using a source port of 1080(tcp), without this entry the return packets would be dropped by incoming rule 7 and the web service would be unreachable. Note that this does not imply that the router is acting as a stateful firewall – it is basing this decision merely on the the flags in the TCP header of the packet.

Confirmation of this behaviour may be confirmed with th HPING2 tool available at http://www.hping.org/. This tool permits the user to send specifically crafted packets and is handy for firewall rule set testing. In the case of testing this access list the following packets were sent to a server which was running the tcpdump tool available at http://www.tcpdump.org/ to see the traffic that made it through the access lists.

Firstly, we'll try a TCP packet with only the SYN flag sent but no filtering enabled. This would be typical of the first packet in the 3-way TCP handshake.

```
[15:13:36]luke@usaji:~$ sudo hping -S -p 135 targethost
HPING targethost (qe0 targethost): S set, 40 headers + 0 data bytes

[15:18:55]luke@targethost:~$ sudo tcpdump -n src host usaji
tcpdump: listening on bge0
15:18:58.510902 usaji.65239 > targethost.135: S 252565185:252565185(0) win 512
15:18:59.521902 usaji.50573 > targethost.135: S 1371519572:1371519572(0) win 512
```

The SYN (S) packets are arriving at targethost.

With the filtering enabled the same test gives the following results.

```
[00:30:13]luke@usaji:~$ sudo hping -S -p 135 targethost
HPING targethost (qe0 targethost): S set, 40 headers + 0 data bytes
[00:29:56]luke@targethost:~$ sudo tcpdump -n src host usaji
tcpdump: listening on bge0
```

No packets were received by the target host indicating that the access list was correctly blocking the packets. If, however, we imitate an established TCP session by combining the SYN and ACK (A) flags:

```
[02:19:31]luke@usaji:~$ sudo hping -SA -p 135 targethost
HPING targethost (qe0 targethost): SA set, 40 headers + 0 data bytes

[02:19:25]luke@targethost:~$ sudo tcpdump -n src host usaji
tcpdump: listening on bge0
02:26:58.815118 usaji.64928 > targethost.135: S 1698103930:1698103930(0) ack 1065970219 win 512
02:26:59.815067 usaji.56669 > targethost.135: S 1380745249:1380745249(0) ack 1411566662 win 512
02:27:00.825611 usaji.59660 > targethost.135: S 357563188:357563188(0) ack 1875901789 win 512
02:27:01.836357 usaji.55447 > targethost.135: S 493810595:493810595(0) ack 697013920 win 512
02:27:02.846429 usaji.64877 > targethost.135: S 1744379422:1744379422(0) ack 359868159 win 512
02:27:03.856017 usaji.63805 > targethost.135: S 1357544213:1357544213(0) ack 1821703722 win 512
```

The packets are received! While the implications of this demonstration do not diminish the effectiveness of the access controls as a TCP connection cannot be established in this manner, they do illustrate the limitations of this sophistication of packet filtering. This activity can not be called fire walling in the true sense; a more apt description would be “The Little Man's Firewall”[9].

Rule 2, “permit ip host probehost.example.com any” permits the probing of end-user connections for open proxies even while all other access to these ports is filtered. More on this later.

The “permit ip any any” is important as a cisco extended access list has “an 'implied deny' for traffic that is not permitted, which blocks all traffic”[10]. This line lets unfiltered traffic pass normally.

There were four realistic options available to implement the access controls:

1. Filter the ports at the ISP's network edge on the border routers.

Method:

[9] http://www.sans.org/rr/special/isp_blocking.php

[10] http://www.cisco.com/warp/public/707/confaccesslists.html

Create an access list defining the rule set required, and apply it to restrict traffic on the interfaces of the upstream connections (point 2 from Figure 1). For example, given that the upstream connection interfaces are GigabitEthernet0/0.2 and ATM4/0.40 the access list would be configured and applied in the following way:

First, create an access list to implement the filtering requirements:

```
router# configure terminal
router(config)# ip access-list extended defaultfilter-in
router(config-ext-nacl)# permit tcp any any established
router(config-ext-nacl)# permit ip host probehost.example.com any
router(config-ext-nacl)# deny   tcp any any eq 135
router(config-ext-nacl)# deny   tcp any any eq 139
router(config-ext-nacl)# deny   tcp any any eq 445
router(config-ext-nacl)# deny   tcp any any eq 593
router(config-ext-nacl)# deny   tcp any any eq 1080
router(config-ext-nacl)# deny   tcp any any eq 3128
router(config-ext-nacl)# deny   tcp any any eq 6588
router(config-ext-nacl)# deny   tcp any any eq 8080
router(config-ext-nacl)# deny   udp any any eq 135
router(config-ext-nacl)# deny   udp any any eq 137
router(config-ext-nacl)# deny   udp any any eq 138
router(config-ext-nacl)# deny   udp any any eq 445
router(config-ext-nacl)# permit ip any any
router(config-ext-nacl)# exit
```

Then create an access list to filter the outgoing connections:

```
router# configure terminal
router(config)# ip access-list extended defaultfilter-out
router(config-ext-nacl)# permit tcp any any established
router(config-ext-nacl)# permit ip any host   probehost.example.com
router(config-ext-nacl)# deny   tcp any any eq 135
router(config-ext-nacl)# deny   tcp any any eq 139
router(config-ext-nacl)# deny   tcp any any eq 445
router(config-ext-nacl)# deny   tcp any any eq 593
router(config-ext-nacl)# deny   tcp any any eq 1080
router(config-ext-nacl)# deny   udp any any eq 135
router(config-ext-nacl)# deny   udp any any eq 137
router(config-ext-nacl)# deny   udp any any eq 138
router(config-ext-nacl)# deny   udp any any eq 445
router(config-ext-nacl)# permit ip any any
router(config-ext-nacl)# exit
```

Next, apply that access list to each upstream interface.

```
router(config)# interface GigabitEthernet0/0.2
router(config-if)# ip access-group defaultfilter-in in
router(config-if)# ip access-group defaultfilter-out out
router(config-if)# interface ATM4/0.40
router(config-if)# ip access-group defaultfilter-in in
router(config-if)# ip access-group defaultfilter-out out
router(config-if)# ^Z
router# write
```

Discussion:

This option is beneficial in that it is extremely easy to implement and requires a one-off router configuration change for network administration staff. It is applied to all current end-user connections instantaneously and to all future connections. However, if even a single end-user becomes infected with a virus by other means they have an exposed vector by which to infect every other vulnerable end-user by bypassing the access list. It does not scale well if multiple end-users wish to be exempted from the access control – a network administrator would have to change the access list manually for each exemption and the end-user would require a static IP address or IP range.

2. **Define an access list on each LNS server, and reference this access list in the default virtual interface template on those LNS servers to be assigned to every new connection.**

Method:
Create the same access list as in Option 1 above, however, the “incoming” and “outgoing” directions have now been switched, as we are dealing with an end-user facing interface rather than an Internet facing interface.
Next, apply this to the Virtual Template Interface that is used by the LNS to create Virtual-Access interfaces (point 1 from Figure 1) for customer connections.

```
router# configure terminal
router(config)# interface Virtual-Template1
router(config-if)# ip access-group defaultfilter-out in
router(config-if)# ip access-group defaultfilter-in out
router(config-if)# ^Z
router# write
```

Note:
This access list will be applied to all current Virtual-Access interfaces that have been created from the Virtual-Template interface just modified. All new connections will have this access list applied to both incoming and outgoing traffic. WARNING: If you have hundreds or thousands of established tunnel interfaces it will take the router a lot of CPU time to rebuild each virtual interface with this new configuration. While it is not supposed to crash in this situation, the ISP's first experience with this showed that it is very possible!

Discussion:
This method is very easy to implement and protects against intra-network infection or exploitation. It is applied to all current end-user connections instantaneously and to all future connections as each existing Virtual-Access interface is updated with the new configuration. However, it is impossible to exclude specific end-users without modifying the access list which impacts performance each time it is done. It does not scale well if multiple end-users wish to be exempted from the access control – a network administrator would have to change the access list manually for each exemption and the end-user would require a static IP address or IP range.

3. **Define the entire access list in each user's RADIUS attributes using Cisco:Avpairs.**

Method:
Create new RADIUS Cisco:Avpair attributes in each customer's attributes list in the manner described at http://www.cisco.com/warp/public/480/radius_ACL1.html :

```
 ip:inacl#1=permit tcp any any established
 ip:inacl#2=permit ip  host any probehost.example.com
 ip:inacl#3=deny   tcp any any eq 135
 ip:inacl#4=deny   tcp any any eq 139
```

```
ip:inacl#5=deny   tcp any any eq 445
ip:inacl#6=deny   tcp any any eq 593
ip:inacl#7=deny   tcp any any eq 1080
ip:inacl#8=deny   udp any any eq 135
ip:inacl#9=deny   udp any any eq 137
ip:inacl#10=deny   udp any any eq 138
ip:inacl#11=deny   udp any any eq 445
ip:inacl#12=permit ip any any

ip:outacl#1=permit tcp any any established
ip:outacl#2=permit ip host probehost.example.com any
ip:outacl#3=deny   tcp any any eq 135
ip:outacl#4=deny   tcp any any eq 139
ip:outacl#5=deny   tcp any any eq 445
ip:outacl#6=deny   tcp any any eq 593
ip:outacl#7=deny   tcp any any eq 1080
ip:outacl#8=deny   tcp any any eq 3128
ip:outacl#9=deny   tcp any any eq 6588
ip:outacl#10=deny  tcp any any eq 8080
ip:outacl#11=deny   udp any any eq 135
ip:outacl#12=deny   udp any any eq 137
ip:outacl#13=deny   udp any any eq 138
ip:outacl#14=deny   udp any any eq 445
ip:outacl#15=permit ip any any
```

Discussion:

This method provides an extremely configurable way to enable and disable the filtering on each individual end-user's connection. It protects against intra network infection or exploitation. However, it expands the size of the end-user RADIUS attributes table by many orders of magnitude severely impacting resource requirements of the RADIUS servers. The filtering is not applied to the user until the next time they connect which in the case of xDSL consumers could be weeks.

4. **Define the access list on the LNS servers, and reference that access list in each customer's RADIUS attributes using Filter-Ids.**

Method:
Create the same access list as in Option 1 above. Next, reference this access list in the RADIUS attributes of each customer, in the manner described at http://www.cisco.com/warp/public/480/radius_ACL1.html :
Filter-Id: defaultfilter-out.in
Filter-Id: defaultfilter-in.out

Discussion:
This method provides an easy way to enable and disable the filtering on each individual end-user's connection and adds only two entries to the user's RADIUS attributes. However, changes are not realised until the next time the user reconnects.

It was at this time, while considering the options for end-user access control, that Microsoft released Microsoft Security Bulletin MS03-039 which detailed three new vulnerabilities that are "similar to the one used by the Blaster worm, and the risk of a new worm outbreak is very real"[11]. Considering the delay between the release of MS03-026 and the Blaster worm, and the fact that "Blaster or one of its variants could be re purposed in short order to exploit one or more of the newly identified vulnerabilities" [12], there was the perception in the information security community of a imminent threat of an outbreak even more devastating than that of August 11. The realisations re learnt and strengthened during the Blaster outbreak – that not all end-users are keeping up to date with patches and are exposing services to the Internet unnecessarily – were still well in mind.

The initial reaction was that urgently there needed to be put in place protective measures to avoid the sort of chaos caused by the Blaster. The easiest and most comprehensive method to achieve this was to block the vulnerable ports as soon as possible, a course of action reinforced by Microsoft when they recommended that "ISPs should institute port filtering or other forms of fire walling at the network edge, between the customer link segments and the Internet itself" [13].

The priority in this situation was to protect the vulnerable end-user connections as soon as possible in order to avert another major security incident on the network. As a large percentage of the end-users connect using always-on technologies such as ADSL, it is not uncommon for a single connection to last for weeks at a time. This, combined with the fact that "the window of opportunity in which to take protective measures could be shorter than it was in the case of Blaster" [14], the only viable option that would effect immediate protection is option 2 listed above -- define an access list on the LNS and then reference that access list in the Virtual Template interfaces. At this stage there was only an increased threat to the Microsoft Windows networking ports and so only these needed immediate attention. The rest of the ports earmarked for filtering would be left until a later date when the enable/disable feature had been developed. The following access list was devised and applied to both the incoming and outgoing access-groups on the Virtual-Template interfaces of the LNS servers:

```
ip access-list extended defaultfilter
 permit tcp any any established
 permit ip host probehost.example.com any
 permit ip any host probehost.example.com
 deny   tcp any any eq 135
 deny   tcp any any eq 139
 deny   tcp any any eq 445
 deny   tcp any any eq 593
 deny   udp any any eq 135
 deny   udp any any eq 137
 deny   udp any any eq 138
 deny   udp any any eq 445
 permit ip any any
```

It is possible to define just the one access list and apply this to both the incoming and outgoing directions because the same restrictions were required on traffic traveling in both directions. This was applied and a post made to one of the user discussion forums. User response was generally positive with only two end-users requesting that the filtering be disabled. This response may be interpreted as further evidence that there is very little legitimate use of these ports on the public Internet.

[11] http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp
[12] http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp
[13] http://www.microsoft.com/serviceproviders/security/isp_039.asp
[14] http://www.microsoft.com/serviceproviders/security/isp_039.asp

Efforts were then directed at making the service optional and also integrating the remaining ports into the filter. The best option to achieve this outcome is option 4 -- define the access list on the LNS servers, and reference that access list in each customer's RADIUS attributes using Filter-Ids. While the ISP wanted end-users to be able to disable the port blocking at their own convenience the ISP did not wish to be in a situation where end-users were exposing open proxies to the Internet by doing so. An automated open proxy probe to check for incorrectly configured systems was required.

The tool chosen for this job was Michael Tokarev's proxycheck, freely available at http://www.corpit.ru/mjt/proxycheck.html . This program connects to various proxy services, requests a connection to a specified host on a specified port and expects a certain banner to be displayed indicating that the proxy connection has been successful and that the host is therefore an open proxy. A secured Postfix SMTP service was configured on the probe host to provide such a banner for testing. To enable automation proxycheck was placed into the following simple wrapper script to provide a response either indicating that no open proxy was found or that an open proxy was found and the protocol it was using:

```
#!/bin/bash
if [ -z "$1" ]; then
  echo "Usage: test.sh <IP address> [verbose]"
  echo "adding verbose will produce the test results"
else
  init_date= `gawk 'BEGIN{print systime()}'`

  # if the person on the command line wants some verbose output
  if [ "$2" == "verbose" ]; then
    /usr/local/bin/proxycheck -aaaaa -n -vvvvv -d10.10.10.10:25 \
    -c chat:"220 probehost.example.com ESMTP Postfix":" \r\n" $1 2>&1
  else

    # open proxy test
    PROXYTEST=`/usr/local/bin/proxycheck -aaaaa -n -vvvvv -d10.10.10.10:25 \
    -c chat:"220 probehost.example.com ESMTP Postfix":" \r\n" $1 2>&1 | grep -e "success" -e "granted" \
     | awk -F: '{print $3"("$2") "}'`
    # Check the output of the Open Proxy test
    if [ -z "$PROXYTEST" ]; then
      echo "OPEN PROXY = NO"
    else
      echo "OPEN PROXY = YES"
      PROXYTEST=`echo $PROXYTEST | sed -e s/\n//g`
      echo "PROXY PORTS = $PROXYTEST"
    fi
  fi
  date=`gawk 'BEGIN{print systime()}'`
  let scantime="$date - $init_date"
  echo "Scan took $scantime second(s)."
fi
```

When an end-user requests that the port blocking be disabled on their account this script is run against their current IP address. If the probe is successful (ie. it successfully connects to probehost.example.com and is presented with the Postfix SMTP banner), an open proxy is confirmed and their request to disable the access control is denied. They are then directed to a web page indicating what the problem is and the way in which they can resolve the issue. As these proxy ports did not pose as imminent and clear a threat as those left vulnerable by MS03-039, the single perceived drawback of Option 4 -- the fact that it is not applicable until the next time the end-user reconnects, is no longer a problem. Users may disconnect and reconnect at their own convenience and have the new filtering applied then.

The updated access controls were then implemented with the following access list.

```
ip access-list extended defaultfilter-in
 permit tcp any any established
 permit ip host probehost.example.com any
 permit ip any host probehost.example.com
 deny   tcp any any eq 135
 deny   tcp any any eq 139
 deny   tcp any any eq 445
 deny   tcp any any eq 593
 deny   udp any any eq 135
 deny   udp any any eq 137
 deny   udp any any eq 138
 deny   udp any any eq 445
 permit ip any any

ip access-list extended defaultfilter-out
 permit tcp any any established
 permit ip host probehost.example.com any
 permit ip any host probehost.example.com
 deny   tcp any any eq 135
 deny   tcp any any eq 139
 deny   tcp any any eq 445
 deny   tcp any any eq 593
 deny   tcp any any eq 1080
 deny   tcp any any eq 3128
 deny   tcp any any eq 6588
 deny   udp any  any eq 135
 deny   udp any any eq 137
 deny   udp any any eq 138
 deny   udp any any eq 445
 permit ip any any
```

The IP address of the open proxy honeypot was then changed to a "clean" IP with the connection protected by the above access lists. The below table lists the statistics gained from the honeypot log files over the first 25 hours of operation.

Hour	***Messages Received***	***Recipients***
0-1	0	0
...		
24-25	0	0
TOTAL	0	0

Table 3Open proxy honeypot statistics with access controls enabled

The filtering has been 100% effective in preventing open proxy exploitation in this case.

Results

While the solution described above is far from perfect and leaves "plenty of room for other vulnerabilities" [15], an additional protective layer in the "defence in depth" strategy has been added to end-user connections. A straightforward and uncomplicated way for end-users to protect themselves from attackers and malicious code has been developed, one that to some extent mitigates the risk posed by two of the most pressing security issues to consumers – timely patch application and the exposure of unnecessary services to the Internet. As the vast majority of end-users will not require these ports to be open to the Internet there has been minimal disruption to Internet services and in the small number of cases where it is required an easy mechanism for disabling the filtering has been provided.

The method implemented provides a practical balance between security and usability. The system probes an end-user's connection for open proxies when they request the filtering be removed but makes no attempt and enforcing continuing compliance. An end-user may install unsecured proxy software or may mistakenly reconfigure their existing software as an open proxy after they have requested – and passed – their removal from the service. The only real technical solution to this contingency is regular re-probing of end-user connections at reasonable intervals.

The data gathered from the open proxy honeypot listed in Table 1 would suggest that such probing is required to occur at intervals of no greater than eight hours as any longer would likely see spammers hijacking the server before it could be retested. This frequency of probing would probably be too invasive for many end-users. Some sort of network intrusion detection could be implemented but on a Gigabit speed network there are significant performance and cost issues to be considered.

The system as it stands is not extremely configurable – it's either enabled or disabled for all filtered ports leaving no room for customisation. With the groundwork done so far it should be relatively simple to extend the concept and provide a completely configurable access control service for end-users. Using option 3, it is possible to define explicit access lists within a customer's RADIUS attributes. This could be offered as a service to end-users wishing either to employ rudimentary custom access controls or to strengthen their existing defences as part of the "defence in depth" strategy. The downside to this option – the expansion of the size of the RADIUS attribute SQL table, could be offset by charging a modest fee for the service.

The open SMTP relay problem briefly discussed has not been mitigated. Implementing a blanket filter over all end-users, even with the option to disable it, would entail huge support issues as a large number of end-users legitimately run their own SMTP servers to provide mail services. However, further consideration of this course of action is certainly justified as eliminating the end-user open relay problem would greatly increase the security of the network and reduce the workload on abuse handling staff. Far more detailed and careful planning would be required for such a course of action to adhere to the four requirements laid down, specifically requirement 3 "any port filtering done would need to be done with an absolute minimum of negative end-user impact."

If an attack vector is discovered which exposes vulnerability on a commonly used port such as 80(tcp) www it will be non-viable to implement the same sort of filtering to protect

[15] http://www.sans.org/rr/special/isp_blocking.php

these services as the impact to end-users would be great.

These shortcomings aside, the benefits provided by the service implemented in this case study have improved both the security environment and the customer relations of the ISP. While it is difficult to obtain relevant quantitative data illustrating these benefits, I believe this response from senior technical support (a "frontline" employee) eloquently summarises the end result.

> Our efforts to curtail malicious & virulent network activity amongst our own users has had a major knock on effect in terms of our relationship with our customers, our interaction with customers, and in potentially thousands of individual cases, the avoidance of very unpleasant Internet experience. Although the potential to affect everyone for the better is not there (some will always protect themselves, others will just be lucky) but when you break it down, the net effect is a real improvement in the quality of Internet experience. People may not be aware of it, but aside from quality of connection, we are currently providing one of the cleanest corners of the Internet there is.

REFERENCES

Australian Internet Industry Association, *"IIA Anti Spam Initiative",* 19 September 2003, URL: http://www.iia.net.au/nospam/ (07 November 2003)

CERT, *"CERT® Advisory CA-2003-23 RPCSS Vulnerabilities in Microsoft Windows"*, 12 September 2003, URL: http://www.cert.org/advisories/CA-2003-23.html (10 November 2003)

Cisco Systems, *"Cisco - Configuring IP Access Lists",* 19 August 2003, URL: **http://www.cisco.com/warp/public/707/confaccesslists.html** (19 November 2003)

Cisco Systems, *"Cisco - How to Apply Access Lists to Dial Interfaces with a RADIUS Server",* 12 December 2003, URL: http://www.cisco.com/warp/public/480/radius_ACL1.html (12 November 2003)

Jenkins, Chris, 2003, " *Australian IT - The year ahead for viruses" ,* 11 November 2003, URL: http://australianit.news.com.au/articles/0,7204,7826216%5E16681%5E%5Enbv%5E,00.html (17 November 2003)

Microsoft Corporation, " *Microsoft Security Bulletin MS03-026",* 10 September 2003, http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp (13 November 2003)

Microsoft Corporation, *"Microsoft Security Bulletin MS03-039",* 10 September 2003, URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-026.asp (13 November 2003)

Microsoft Corporation, *"Recommendations to ISPs Regarding MS03-039",* 08 October 2003, URL: http://www.microsoft.com/serviceproviders/security/isp_039.asp (12 November 2003)

St Sauver, Joe , *"The Open Proxy Problem"* , 06 April 2003, URL: http://darkwing.uoregon.edu/~joe/proxies/ (07 November 2003)

TechTarget, *"insecure relay - a whatis definition - see also: open relay, third-party relay",* 03 November 2003, URL: http://whatis.techtarget.com/definition/0,,sid9_gci782509,00.html (07 November 2003)

Ulrich, Johannes, *"Internet Service Providers: The Little Man's Firewall?"* 12 September 2003, URL: http://www.sans.org/rr/special/isp_blocking.php (19 November 2003)

US Department of Homeland Security, *"Potential For Significant Impact On Internet Operations Due To Vulnerability In Microsoft Operating Systems' Remote Procedure Call Server Service (RPCSS)",* 10 September 2003, URL: http://www.nipc.gov/warnings/advisories/2003/ Advisory9102003.htm (10 November 2003)