Global Information Assurance Certification Paper

Egress Filtering: More Than Just a Good Neighbor Policy

GIAC Security Essentials
Certification (GSEC)
Practical Assignment
Version 1.4c

Option 1 - Research on Topics
in Information Security

Submitted by: John M. Spitler
March 27, 2005

Table of Contents

Abstract/Summary

Egress filtering has been lauded for years as a good practice to minimize unnecessary Internet traffic. In today's environment, it is now a key practice in firewall and router configuration to establish a true default deny stance.

This paper will document the benefits that egress filtering has on the Internet community and how egress filtering would have reduced the damaging effects of well known worms such as Code Red and SQL Slammer. Internal benefits are also presented, including the mitigation of other malicious code such as IRC bot trojans. Also discussed is a real life example of how the use of egress filtering as a layer of defense would prevent exploitation of a Windows password-cracking exploit. The main goal of this paper is to educate why egress filtering is not only a "good neighbor policy" but is also highly effective in minimizing risk for companies that implement a conservative rulebase for outbound connections.

In addition to examples where egress filtering will assist risk mitigation, issues regarding the implementation and potential caveats that may be encountered are addressed.

What is Egress Filtering?

Using firewalls and Access Control Lists (ACLs) on routers to determine the allowance of inbound traffic is a widely known and extensively followed practice in network security. Establishing inbound or ingress rules is a cornerstone in preventing unauthorized access to devices that reside in a company's internal or DMZ networks.

The practice of filtering traffic generated from a company's internal network(s) traveling outbound to less trusted networks is called egress filtering. Egress filtering is not as widely adopted due to the rationale that firewalls and other filtering devices exist to "keep the bad guys out", not to protect less trusted networks from company's internal trusted users and devices. When egress filtering is adopted, it is more likely using a specific deny approach to prevent users from accessing services such as Instant Messaging and Peer to Peer file sharing services instead of establishing a default deny stance for outbound packets.

Egress Filtering as a Good Neighbor Practice

Source-based Egress Filtering

In January 1998, RFC 2267 (which was later superseded by RFC 2827 in May 2000) introduced the concept of filtering based on source IP address to prevent the SYN flood Denial of Service (DOS) attack.[1,2] The reasoning is very straightforward: no Internet Service Provider should allow traffic with a source address that is in reserved private address space, also known as RFC 1918 address space, to route to the Internet. Traffic that falls within the following networks will not route back to the source device as these address are not in the routing tables of Internet routers.[3]

Reserved Private Address Space as Defined by RFC 1918
10.0.0.0/8
192.168.0.0/16
172.16.0.0/12

SYN floods affect the availability of the target machine by forcing it to use its resources to send SYN-ACK packets to an enormous number of spoofed SYN packets that are sent to the device. The recommendation of RFC 2267 to block these packets at the source would prevent a company's network from acting as an unknowing participant in an attack on another IP address and with no potential drawbacks. Even if packets with private addresses as the source IP are benign in nature, the destination machine cannot route reply packets back to the source.[4]

The RFC also addresses the case where the spoofed source address is a publicly routable address. This creates a need to implement a rule that will deny any traffic to the Internet which has a source IP address that is not allocated by the ISP.[1,2]

The principle set forth in RFC 2267 can be applied locally to Internet-facing routers that are managed by companies instead of their ISPs. There should be an anti-spoofing rule on the Access Control List (ACL) of the router that denies any traffic that has a source address which is not within the scope of the internal network range. Without this rule in place, detection of spoofed traffic originating from internal networks is an arduous process.

Setting up Source IP address-based egress filters which prevent packets with a source which falls outside of the network ranges of the internal networks is a practice that will have no negative effects upon implementation.[4] Companies can implement more granular source-based egress filtering rules as needed.

Source IP address-based filtering is an effective defense in preventing spoofed packets from reaching the Internet, but not all malicious packets feature a spoofed source IP address. Examples of these cases will be addressed in the next section as the benefits of egress filtering extend well beyond merely preventing devices within your internal networks from being used as a source for attacks on others. Egress filtering is essential for establishing a sound security posture.

Mitigating Effects of Malicious Code

Malicious code in the form of viruses and worms may require the ability to make outbound connections to spread or to inflict further damage to the victimized host. This section will show some examples where port-based egress filtering would have prevented the spread of two of the most infamous worms. In addition, the prevention of arbitrary code execution on infected devices and the benefits of early detection of infected internal devices will be presented.

Code Red

Computer Economics estimated that the Code Red worm (and its many variants) had a worldwide economic impact of 2.62 billion US dollars in 2001.[5] This worm exploited a known Microsoft vulnerability which was announced by Microsoft on June 18, 2001 in the Microsoft Security Bulletin MS01-033.[6] The vulnerability was a buffer overflow of a Microsoft Indexing Service. The worm which exploited the vulnerability was released and began infecting servers on July 19, 2001.[7]

Egress filtering on the networks of particular devices running the IIS server would not have prevented infection of those systems. Applying the Microsoft patch as recommended by MS01-033 would have prevented the infection of a device, but unfortunately, the patch did not achieve widespread implementation.

Where egress filtering would have helped lessen the devastating effects of Code Red and its variants was the prevention of the spread of the worm by devices infected with the malicious code.

Code Red spread over TCP port 80 (HTTP). The infected device attempted to send a crafted GET request to a random IP address with the intent of finding an unpatched IIS server and exploiting the buffer overflow, thus creating a new victim and proliferator of the Code Red worm.[7]

TCP port 80 is the port on which the vast majority of HTTP traffic travels on and is a port that is expected to be open and visible to the Internet. This makes it a very desirable port for worms to utilize as ingress filtering will not prevent a web server from accepting traffic on a port it must listen on to function. This is where egress filtering should have been implemented as a layer of defense based on a very basic concept: **Clients request web pages, servers serve pages.**

Devices that serve web pages should not be used to surf the web (or initiate any outbound connections to the Internet). Their reason for existence and role is to serve HTTP pages, not request them. There is no reason that the vast majority of the servers which were infected by (and subsequently spread) the Code Red worm needed to initiate HTTP connections to the Internet.

If a strong egress filtering stance had been implemented on the routers and/or firewalls that served as boundary protection for these devices, the spread of the worm would have been considerably throttled. A filter that prevented outbound traffic on port 80 from these servers would have prevented a device, which was already a victim, from becoming another attacker.

SQL Slammer

One of the main contributors to the large propagation and impact that Code Red had was the fact that the devices were not current on patches. The patch for the buffer overflow vulnerability was available a month before it was exploited via the Code Red worm. In January 2003 it was evident that many did not learn from that mistake. Vulnerabilities which were discovered in Microsoft's SQL server were documented in a CERT Advisory on July 29, 2002.[8] Almost exactly half a year passed from the time of that advisory recommending applying patches which mitigated the vulnerability to the time where the worm known as SQL Slammer or Sapphire infected unpatched and Internet visible Microsoft SQL Servers and MSDE 2000 (Microsoft SQL Server Desktop Engine).[9]

The SQL Slammer worm traveled on UDP 1434 and hosts infected with Slammer would attempt to spread the worm to randomly chosen IP addresses. There was no other malicious payload or side effects other than the DOS caused by the tremendous amount of network traffic generated by the worm. According to a report published by the Cooperative Association for Internet Data Analysis during the first minute of the spread of the Sapphire/Slammer worm, the infected population doubled in size every 8.5 (±1) seconds.[10]

Egress filtering was recommended as one part of a multi-faceted defense strategy by the CERT Coordination Center in their advisory in response to the exploitation of the vulnerability (CA-2003-04) and by Cisco in their SAFE SQL Slammer Worm Attack Mitigation white paper.[11] The recommended egress filtering of UDP 1434 as well as the filtering of ingress UDP 1434 and the patching of the vulnerable devices are all part of a layered security approach and all should have been performed previous to the introduction of the worm.

Hidden Benefits

The two examples above detail how egress filtering could have prevented a company's infected host(s) from spreading malicious code to other hosts on the Internet. In both cases random IP addresses were used as the targets so the most obvious benefits of the egress filtering would be reaped by the "other guy"

and not by the company who implemented the egress filters as the filtering did nothing to prevent the infection of the device.

There are two very significant benefits that the company with an infected host can obtain if egress filters prevented the attempted propagation.

There are countless worms in existence, all of which by definition, are attempting to self-propagate. Code Red and Sapphire/Slammer are two very infamous and damaging worms and many organizations were guilty of not having the proper ingress filters and patch implementations in place to prevent infection. In many recent cases the malicious code was introduced via laptop computers used in networks other than the corporation's (hotels, wireless hotspots) and are then re-introduced back into the corporate network. The infected laptop may attempt to spread itself to random addresses on the Internet. If egress filtering is in place and the filtering device is logging the blocked attempts, then the infected device can be identified quickly through log inspection (or even quicker if alerts are generated in the event of multiple failed connections) and taken off of the network and inspected and eventually patched.

In addition to that benefit, potential damage to a company's reputation is avoided because the fact that one of your company's devices was infected with a worm has now been kept in-house. This is more desirable than being alerted by another company that they are witnessing suspect connection attempts to their IP address space from one of your devices.

IRC Worms/Trojans

Motives for egress filtering are not limited to benefits gained due to the prevention of worm propagation. There are instances of malcode which count on the ability of an infected host to initiate an outbound connection so full compromise of the device can be achieved.

Over the years there have been numerous worms created with the intent of device compromise. Gaining complete access to a device is a grand achievement for a malicious hacker. Many of the worms that are written for this purpose cause infected devices to connect to specified IRC channels over TCP port 6667 and listen for commands from the attacker.[12]

One example is W32.Spybot.DNB. This worm was discovered on September 13, 2004. One of its damaging effects is creating a backdoor by connecting to an IRC channel and listening for commands from the attacker which include, but are not limited to, downloading and execution of files, launching of DOS attacks, stopping and starting processes and transmission of system information to the attacker.[13]

Blocking outbound TCP 6667 will prevent the IRC channel from being contacted, and will prevent the remote access to the infected host. It must be noted that

there is still a critical need to identify devices which are infected as there are other damaging results of infection. As stated before, if alerts are triggered when attempts to the IRC port are blocked, the removal of the infected device from the network and the rest of the incident response plan can take place quickly and lessen the effect that the device or devices can have on network resources.

This is one example of many existing worms that attempt to connect to IRC channels. In the future, variants of existing worms as well as new worms will attempt to compromise a device in this manner. Keeping anti-virus signatures up to date should prevent most of this malicious code from infecting a system, but the egress filter provides an extra layer of defense.

Zero-day Exploits

Unlike Code Red and Slammer, zero day exploits will take advantage of vulnerabilities on the same day that the vulnerability is made public.[15] In these cases, patches and anti-virus signatures are not yet available. Defenses to these exploits must obviously be proactive in nature and should include removal of any unnecessary services on devices which are potential targets. In the likely event that the exploit travels over allowed ports, egress filtering is the most cost-effective defense to prevent the spread of the exploit.

Defense in Depth

Mitigation of risk involving malicious code is not the only reason to institute egress filtering. Establishing an outbound default deny stance is one element in a layered security approach.

Shell Shoveling

Shell shoveling is a technique which allows an attacker to gain a command line shell on a target host. In the following example, the netcat utility would be listening on TCP ports 80 and 25:

nc <IP address of attacker> 80 | cmd.exe | <IP address of attacker> 25

This will succeed for the attacker if he/she was able to get the netcat utility uploaded to the target and listening on the specified ports, and if the firewalls permit inbound traffic on port 80 and outbound traffic on port 25.[14]

If successful, the attacker now has a shell on the victim device. The netcat utility could be uploaded to a poorly configured device, or installed by a malicious insider. Based on the concept explained earlier, clients initiate connections, servers respond. Assuming this is a web server which should be listening on port 80, any outbound connections initiated from this device should be denied.

Application Vulnerabilities

The following example will illustrate how egress filtering as one layer can prevent the disclosure of sensitive information when other security practices are not followed in earnest:

Company A has a web application which will take a user's input and will take the information provided and pass it to an internal application server. Dynamic content is then served back to the user based on his or her input. When the web server connects to the application server, the web server authenticates over the SMB port (TCP 445).

The user john doe inputs the text "red" the URL that is created is:

www.foo.com/foo?user=john%20doe&input=red&path=bar

While not obvious, a hacker was able to figure out that 'bar' was a hostname for the application server that received the information provided by the user. This malicious user performs some URL manipulation and enters the following string into his location bar:

www.foo.com/foo?user=badguy&input=test&path=<IP address designated by hacker>

The hacker has replaced the hostname of the internal server with the IP address of his own machine. The user and input parameters in this case are irrelevant. Company A has no egress filtering and the web server attempts to connect to the hacker's machine on port 445 and provides its NTLM hash.

The malicious hacker is running an exploit called Smbrelay. This is an executable freely available from phreak.org and its function is to collect NTLM authentication hashes and convert them to a file which can be used by L0phtcrack to crack the password.[16]

There are many problems present in this example. The hacker should not have been able to change the URL parameter, and even more important, the parameter that included the hostname of the application server should not exist in the URL parameters.

The situation would be less bleak if outbound SMB traffic is blocked at the firewall. The benefit again would be two-fold, the hacker will not receive the NTLM hash, and the chances are higher that the traffic will be flagged and the improper URL parameter will be discovered and remediated.

Implementation

Egress filtering is a security practice that must be followed to achieve a solid security stance. There is no "one size fits all" approach and the ease and extent of implementation is dependent on many variables, including the complexity of the existing network and the corporate culture.

There are many companies that will be resistant to this change as it carries the stigma that their employees are not trustworthy and are being treated as the "bad guys". Companies that have clear policies that dictate acceptable use of network resources will have an easier time of implementing practices that restrict the ability of their users to access specific ports on Internet hosts. Small companies or newer companies might not have these policies in place and may meet resistance in the implementation of egress filters. The communication to the senior management must be clear that the purpose of the outbound filters are not solely to prevent internal users from accessing services on the Internet and are essential in the protection of internal devices. Senior management must understand the need for the implementation of the changes and provide their support. If there is not a documented policy which addresses the need to achieve a default deny outbound stance, one should be created, approved, and implemented.

There are still many potential negative issues above and beyond the user perception. Any changes that restrict traffic traversing networks have the potential to affect traffic that is necessary for business operations. If this was an entirely new network that is being architected, then the stance taken for egress filters can be the same as ingress filters: *Block everything and add rules to allow traffic as needed*.

The changes surrounding setting up the filters are much harder to do in a mature or complex network as there may be justified reasons to have particular ports open to the Internet. In this case, a documented plan to implement egress filters without disruption to justified network traffic will need to be created.

Again, every company will differ in their plans, but some general guidelines can be followed to ensure the result of the implementation will be the most conservative outbound filtering stance, without interruption of existing business services.

The most important course of action that must occur for each change will be a change control process to identify what devices are affected, and the owners of the devices must be involved as they will have the knowledge of what services the device needs to access in order to perform its role in the network.

The good news is that the most important devices in companies' networks rarely need to initiate any connections out to the Internet. As stated in the Code Red example, servers exist to serve pages, not request HTTP from other servers. If

web servers are configured securely, then they will perform only that task and require no need to initiate any outbound connections. An argument may be made that the servers acquire patch updates over HTTP, but even if this is absolutely necessary and cannot be accomplished by a dedicated server which will then push updates to the other devices, then egress filtering still can be established and specific allow rules can be implemented permitting connections to those trusted IP addresses that host the updates.

In addition to a restrictive filter that prevents connections initiated from web servers, a filter that restricts any outbound traffic that does not have a ***source*** port of TCP 80 should be implemented as well. The only traffic from an HTTP web server should have a source of 80 (and also 443 if it serves HTTPS pages). The only outbound packets that originate from the servers should be in response to client connections and therefore the source port will be the port on which the server established the connection. This rule can be applied to all servers, for example, only source port TCP 25 for a company's mail servers should be allowed.

All critical devices, beginning with the servers that are located within a company's DMZ network, should be evaluated to determine on what, if any, TCP and UDP ports they need to establish connections in order to function properly. When this process is completed for all critical devices, implicit allow rules can be established and paired with a default deny rule that prevents all other outbound connection attempts.

Filtering of outbound connections should extend to the desktop environment after the server population has been addressed. There may already be specific egress filters for the user workstations preventing access to Internet services such as peer to peer file sharing networks. The eventual goal should be that all outbound connections are utilizing specific allow rules. The work factor for this process will be dependant on the size of the desktop population and the number of different groups in the organization. The marketing department may need to access a focus group website on TCP port 8080, while the accounting department needs to use Secure FTP on port 22 to download tax forms.
A company can establish a designated subset of desktops in each department to apply the filters to and then monitor and ensure that no business critical services are affected. After an established testing period with no issues, the filters can be implemented across the entire department.

This level of granularity may take a great deal of effort to implement, especially if IP addresses for the user workstations are dynamically allocated. Static IP addresses may need to be implemented on the devices which require specific outbound connections. Egress filtering should not be deployed to the extent that the man hours required to implement and maintain them produce a higher cost to the company than the events they are designed to prevent. A more realistic approach for many companies using the above example would be a specific rule

to allow traffic from the Internal networks to the defined Internet servers on ports 22 and 8080. This would allow users from the marketing department to send packets on port 22, which is not needed, but the risk is low and may be acceptable.

A general implementation plan is summarized below:

Step 1. Identify all critical devices.

Step 2. Meet with the administrators of critical devices. The goal should be determining what, if any, ports need to remain open for outbound connections from the devices.

Step 3. Complete change request forms and submit to the proper groups for approval.

Step 4. Implement the egress filters on the servers and monitor for any negative effects.

Step 5. Determine what level of egress filtering will be established on the user workstations and define what outbound connections will be allowed.

Step 6. Complete change request forms and submit to the proper groups for approval.

Step 7. Implement the filters on select desktops and monitor for any interruption in business services.

Step 8. Establish egress filtering across the desktop ranges for the business units.

Step 9. Ensure that the firewall in which the filters are implemented upon is logging the failed connections to identify hosts infected with malicious code and to identify users who are attempting to access restricted services.[17]

Testing the Egress Filtering

After the implementation is complete, periodic testing should occur to ensure that unwanted outbound traffic is being blocked by the egress filters. Tools such as hping2 (http://www.hping.org) and firewalk (http://www.packetfactory.net/Projects/firewalk) can be used on UNIX based devices to send packets on specified ports to specified external IP addresses.

Nmap (http://www.insecurity.org) is a tool that will run on both UNIX-based and Windows devices to send packets to specified address and port ranges. Running these tools can verify that the egress filters have been implemented successfully and the failed attempts are appropriately logged.

Conclusions

Egress filtering provides benefits to the Internet community and also to the companies that implement the filters. The practice of setting up filters to block outbound traffic is an important element in the prevention of worm propagation and possible call-backs, such as connections to IRC channels by infected devices.

Egress filtering alone is not a solution, but is part of a defense-in-depth security strategy. Steps must be taken to prevent infection of company's devices by malicious code. Even if a company is diligent in ensuring that anti-virus signatures are up to date, it is still possible that a device can be introduced into the network that has already been infected, or a zero-day exploit takes advantage of a vulnerability before any anti-virus signatures have been made available.

In the event of infection, egress filters can help administrators discover the infection and identify the infected hosts. Egress filtering also prevents those devices from sending worm-initiated packets out to other hosts on the Internet, alerting others to the fact that your company is experiencing a malicious code incident.

Filtering and monitoring egress traffic is not only to combat malicious code. Application vulnerabilities and exploits such as shell shoveling can be defended using a conservative approach to allowance of outbound traffic.

Implementation of egress filtering is not without potential negative effects. An implementation plan needs to be created and backed up by security policies to prevent denial of business related services and communicate the purpose for the restrictive changes. All changes need to be approved through a change control process and tested before widespread implementation.

References

[1] Ferguson, Paul, Senie, Daniel, "Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing", RFC 2267, January 1998.

[2] Ferguson, Paul, Senie, Daniel, "Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing", RFC 2827, May 2000.

[3] Rekhter, Y., Moskowitz, R., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", RFC 1918, February 1996.

[4] SANS Global Incident Analysis Center, "Egress Filtering v 0.2" 29 February 2002 <http://www.sans.org/y2k/egress.html>.

[5] Computer Economics, "Malicious Code Attacks Had $13.2 Billion Economic Impact in 2001" 4 January 2002 <http://www.computereconomics.com/article.cfm?id=133>

[6] Microsoft Security Bulletin MS01-033, "Unchecked Buffer in Index Server ISAPI Extension Could Enable Web Server Compromise" 18 June 2001, updated 4 November 2003 <http://www.microsoft.com/technet/security/bulletin/MS01-033>.

[7] CERT Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow in IIS Indexing Service DLL, 19 July 2001 <http://www.cert.org/advisories/CA-2001-19.html>.

[8] CERT Advisory CA-2002-22 Multiple Vulnerability in Multiple Vulnerabilities in Microsoft SQL Server, 29 July 2002 <http://www.cert.org/advisories/CA-2002-22.html>.

[9] CERT Advisory CA-2003-04 MS-SQL Server Worm, 25 January 2003 <http://www.cert.org/advisories/CA-2003-04.html>.

[10] Moore, David et al. "The Spread of the Sapphire/Slammer Worm" 2003 <http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html>.

[11] Cisco SAFE Blueprint, "SAFE SQL Slammer Worm Attack Mitigation" 2003 <http://www.cisco.com/warp/public/cc/so/neso/sqso/worm_wp.htm>.

[12] "Known Trojans & Exploits to port:6667", <http://www.doshelp.com/Ports/6667.html>

[13] Symantec Security Response W32.Spybot.DNB, 13 September 2004, <http://securityresponse.symantec.com/avcenter/venc/data/w32.spybot.dnb.html >

[14] Symantec Security Response W32.Spybot.DNB, 13 September 2004, <http://securityresponse.symantec.com/avcenter/venc/data/w32.spybot.dnb.html

[15] McClure, Stuart and Scambray, Joel. " 'Inside-out' security pays attention to your revealing, vulnerable traffic" 20 March 2000, <http://www.infoworld.com/articles/op/xml/00/03/20/000320opswatch.html>

[16] Windows Networking.com, "SMB Relay Captures NTLM Hashes" 20 April 2004 <www.windowsnetworking.com/kbase/WindowsTips/WindowsXP/AdminTips/Network/SmbRelaycapturesNTLMhashes.html>

[17] Buff, Nathan. "Firewall Best Practices – Egress Traffic Filtering" ISSA Journal February 2003 < http://hhi.corecom.com/egresstrafficfiltering.htm>