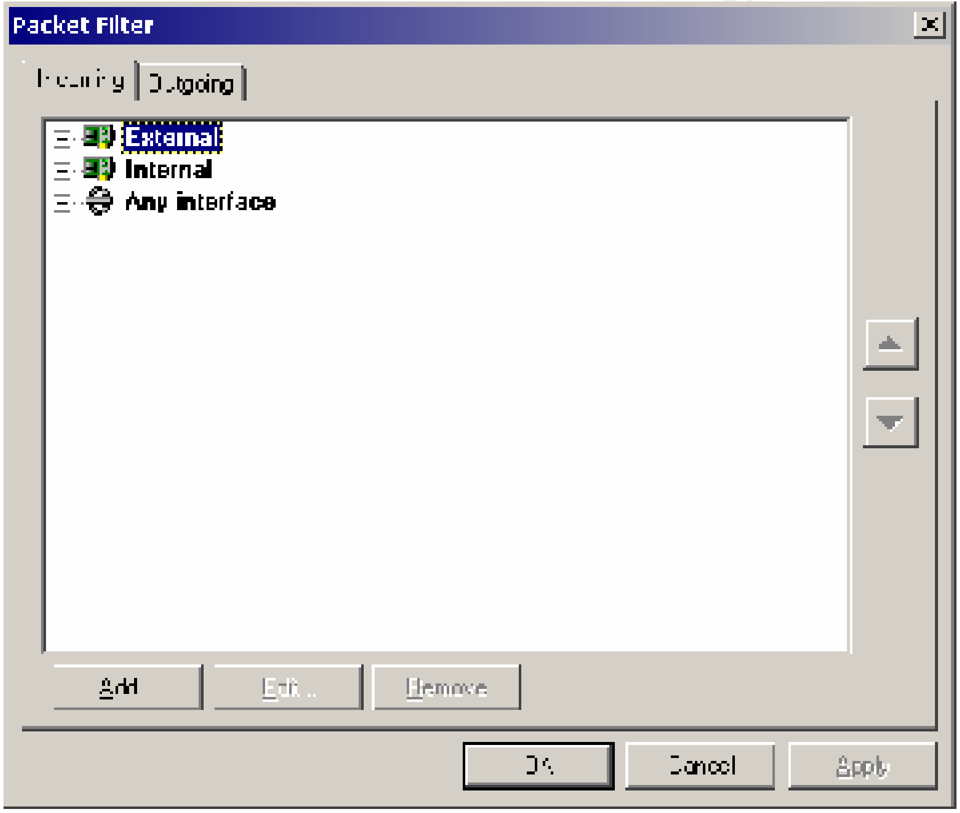
Ronald Ross
SANS DC 2000
GIAC Firewall and Perimeter Protection Practical
August 15, 2000

I. Introduction

The purpose of this tutorial is to illustrate how to implement a perimeter defense solution based on the security policy set forth in the *SANS Top Ten Vulnerability List*.

The perimeter defense solution I will be using is WinRoute Professional 4.1 build 22 (http://www.tinysoftware.com), running on Windows NT 4 Server Service Pack 6a (Workstation may also be used), the OS has been hardened. WinRoute Professional is an ICSA certified firewall/routing software that offers a more approachable interface. The reason behind using WinRoute is to illustrate to small companies that a good perimeter defense solution can be implemented without spending a large sum of money and without having to immerse yourself in Linux (also, I believed there would be a large amount of practicals using Linux – don't want overkill).

Here is an example of WinRoute's Packet Filter screen:



As you can see in the above example, there is a listing for Incoming and Outgoing, with listing for External interface, Internal interface and Any interface. The rules are listed in this screen.

Ronald Ross
SANS DC 2000
GIAC Firewall and Perimeter Protection Practical
August 15, 2000

Below is our security policy, taken from the *SANS Top Ten Vulnerability List*. It details those services needing filtering or blocking. The example screened network, seen in Figure 1, behind the perimeter defense solution will contain a mail relay server that will send and receive mail. I have not included the Internal network in this drawing, as it is beyond the scope of this document. Also, it may be a good idea to have another defense solution in front of the Internal network.

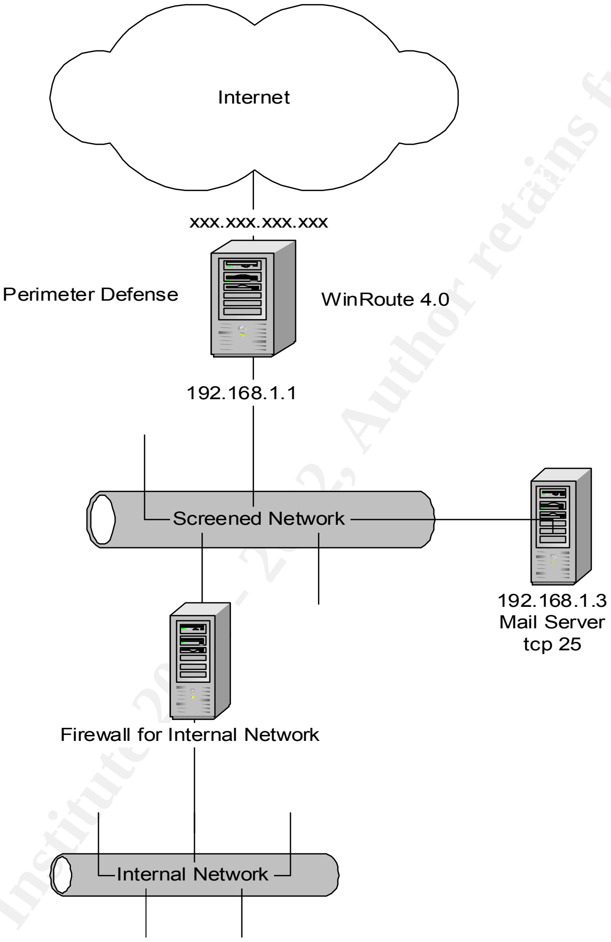


Figure 1 – Illustration of example network

Ronald Ross
SANS DC 2000
GIAC Firewall and Perimeter Protection Practical
August 15, 2000
Below is the Security Policy we will follow, from the *SANS Top Ten Vulnerability List*[1].

Security Policy

In this section, we list ports that are commonly probed and attacked. Blocking these ports is a minimum requirement for perimeter security, not a comprehensive firewall specification list. A far better rule is to block all unused ports. And even if you believe these ports are blocked, you should still actively monitor them to detect intrusion attempts. A warning is also in order. Blocking some of the ports in the following list may disable needed services. Please consider the potential effects of these recommendations before implementing them.

1. Block "spoofed" addresses-- packets coming from outside your company sourced from internal addresses or private (RFC1918 and network 127) addresses. Also block source routed packets.
2. Login services-- telnet (23/tcp), SSH (22/tcp), FTP (21/tcp), NetBIOS (139/tcp), rlogin et al (512/tcp through 514/tcp)
3. RPC and NFS-- Portmap/rpcbind (111/tcp and 111/udp), NFS (2049/tcp and 2049/udp), lockd (4045/tcp and 4045/udp)
4. NetBIOS in Windows NT -- 135 (tcp and udp), 137 (udp), 138 (udp), 139 (tcp). Windows 2000 - earlier ports plus 445(tcp and udp)
5. X Windows -- 6000/tcp through 6255/tcp
6. Naming services-- DNS (53/udp) to all machines which are not DNS servers, DNS zone transfers (53/tcp) except from external secondaries, LDAP (389/tcp and 389/udp)
7. Mail-- SMTP (25/tcp) to all machines, which are not external mail relays, POP (109/tcp and 110/tcp), IMAP (143/tcp)
8. Web-- HTTP (80/tcp) and SSL (443/tcp) except to external Web servers, may also want to block common high-order HTTP port choices (8000/tcp, 8080/tcp, 8888/tcp, etc.)
9. "Small Services"-- ports below 20/tcp and 20/udp, time (37/tcp and 37/udp)
10. Miscellaneous-- TFTP (69/udp), finger (79/tcp), NNTP (119/tcp), NTP (123/tcp), LPD (515/tcp), syslog (514/udp), SNMP (161/tcp and 161/udp, 162/tcp and 162/udp), BGP (179/tcp), SOCKS (1080/tcp)
11. ICMP-- block incoming echo request (ping and Windows traceroute), block outgoing echo replies, time exceeded, and unreachable messages

Ronald Ross
SANS DC 2000
GIAC Firewall and Perimeter Protection Practical
August 15, 2000

II. Description

In this section, I will describe the services we will block and give examples of how to implement our policy using WinRoute. At the end of the document I will list the order for the rules, along with an explanation. Due to WinRoute's interface, I am not going to go through every filter in the next section, as this would be cumbersome to you, the reader (You also can't tell if it's incoming or outgoing).

1. Block "spoofed" addresses-- packets coming from outside your company sourced from internal addresses or private (RFC1918 and network 127) addresses. Also block source routed packets.

Imagine if I were to call your significant other when you were not at home, pretending to be you. I then explain to your significant other that I have another significant other. The key to this is that I can imitate your voice perfectly. What would happen? You might be in for a whole lot of trouble when you get home. This is somewhat analogous to IP spoofing. I send a packet into your network destined for a host with the source address of one of your internal systems. This host will send a packet back to clueless system. Now if I send a whole bunch of icmp echo requests to multiple internal hosts with the source address of one internal system. That could be some bad news.

Source routed packets are packets that contain a specific route for the return packet to follow. If an attacker inserts his/her own system into the route list, then the packet is swallowed by that system. Depending on the content of the packet, this can be dangerous.

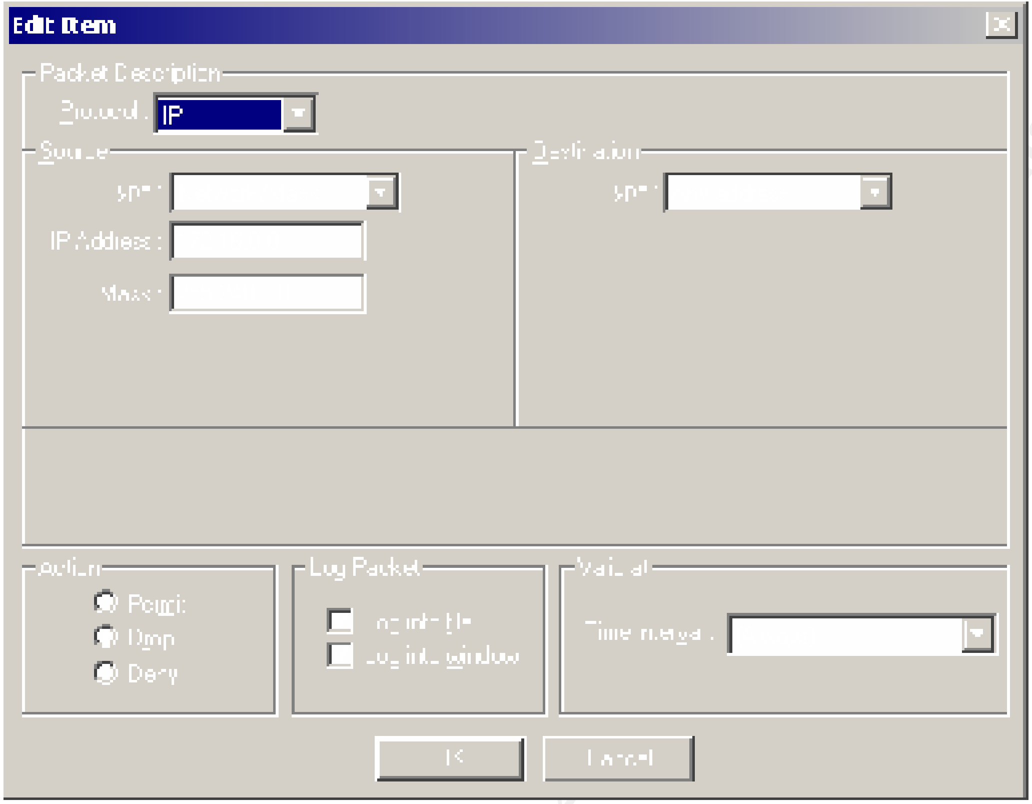


Figure 2 – Spoofed Address example

In Figure 2, we use the IP facility of WinRoute's packet filter. We have a Source and a Destination, along with an Action (Permit – allow the packet in; Drop – destroy the packet; Deny – destroy the packet and send a message back to the originating source). We also have the option of Logging the packet – into a file, the console window, or both.

As can be seen in Figure 2, any packet matching a source address of 172.16.0.0 mask 255.240.0.0 will be dropped. We also log the packet.

I did not, however find a way using WinRoute to block source routed packets. I will send a message to Tiny Software to see if there is a way to do so.

2. Login services-- telnet (23/tcp), SSH (22/tcp), FTP (21/tcp), NetBIOS (139/tcp), rlogin et al (512/tcp through 514/tcp)

Login services should not be permitted from external sources, unless absolutely needed. Login services provide a toehold into your network. Accounts can be brute forced through the login services. Login services can also give an attacker valuable information regarding your systems, for example when you FTP into a Unix server, you are presented with the following information:

Ronald Ross
SANS DC 2000
GIAC Firewall and Perimeter Protection Practical
August 15, 2000

```
220 hpuxhost FTP server (Version 1.7.212.3 Wed Jul 14 10:24:05 GMT
1999) ready

User (hpuxhost:(none)):
```

Now I know what version of FTP is running and using nmap (http://www.insecure.org) I have a pretty good chance of gaining the OS information. (If you can't tell from the prompt, it's an HP-UX host). These services should be blocked at the perimeter so there is absolutely no chance of any external access.

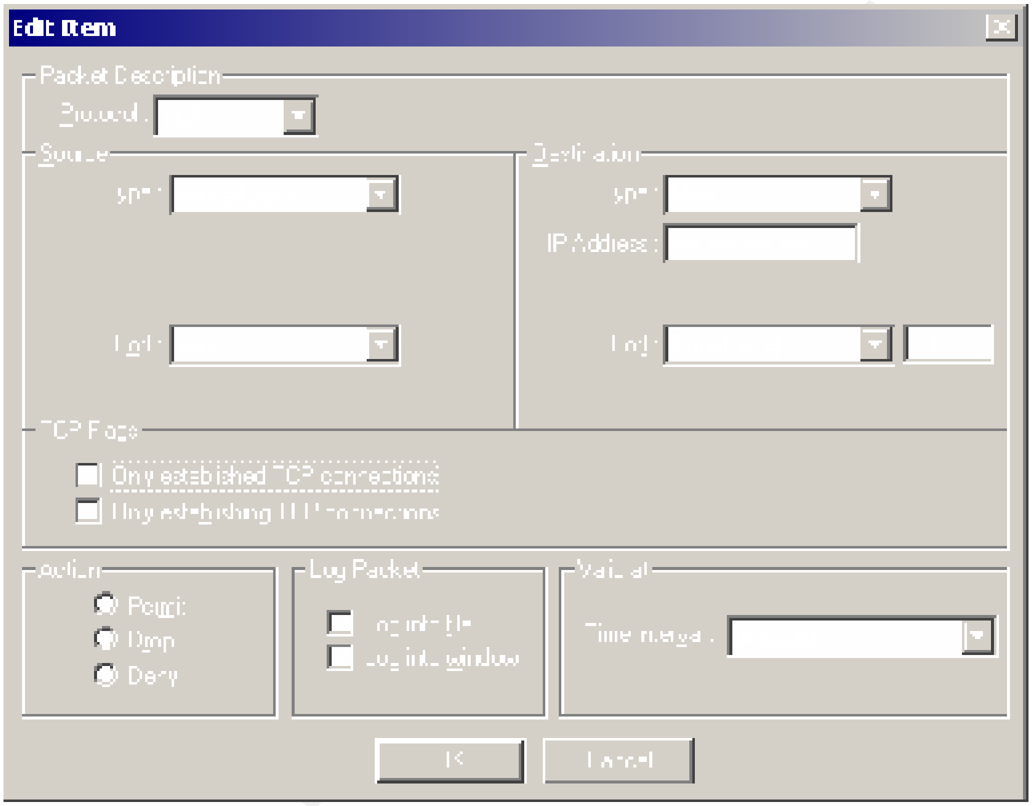


Figure 3 – Login service example: telnet

When we choose TCP as the protocol, we are presented with port options, as can be seen in Figure 3. Along with this, we are also presented with TCP flag options – Only established TCP connections (SYN bit is not the only bit set) and Only establishing TCP connections (only SYN bit is set).

In Figure 3, we block port 23/tcp (telnet), I'm not worried about the SYN bit being set (to initiate a connection), I'm just blocking all attempts to this port. I drop the packet and log.

3. RPC and NFS-- Portmap/rpcbind (111/tcp and 111/udp), NFS (2049/tcp and 2049/udp), lockd (4045/tcp and 4045/udp)

Remote Procedure Calls are program calls from one system across a network to execute on another system. NFS is an example of a file sharing daemon for Unix using RPCs. Another RPC based application is lockd which provides file- and record-locking services in NFS.

A Remote Procedure Call works in this way:

1. A client program calls RPC to request a service on a remote host.

2. Upon receipt of the request, the server dispatches a routine to perform the requested service on the remote host.

3. The server then sends back a reply and the procedure call returns to the client.

These services are not only open to misconfiguration, e.g. world-exportable NFS mounts, but also to vulnerabilities in program code itself. According to the *SANS Top Ten List,* "[t]here is compelling evidence that the vast majority of the distributed denial of service attacks launched during 1999 and early 2000 were executed by systems that had been victimized because they had the RPC vulnerabilities."

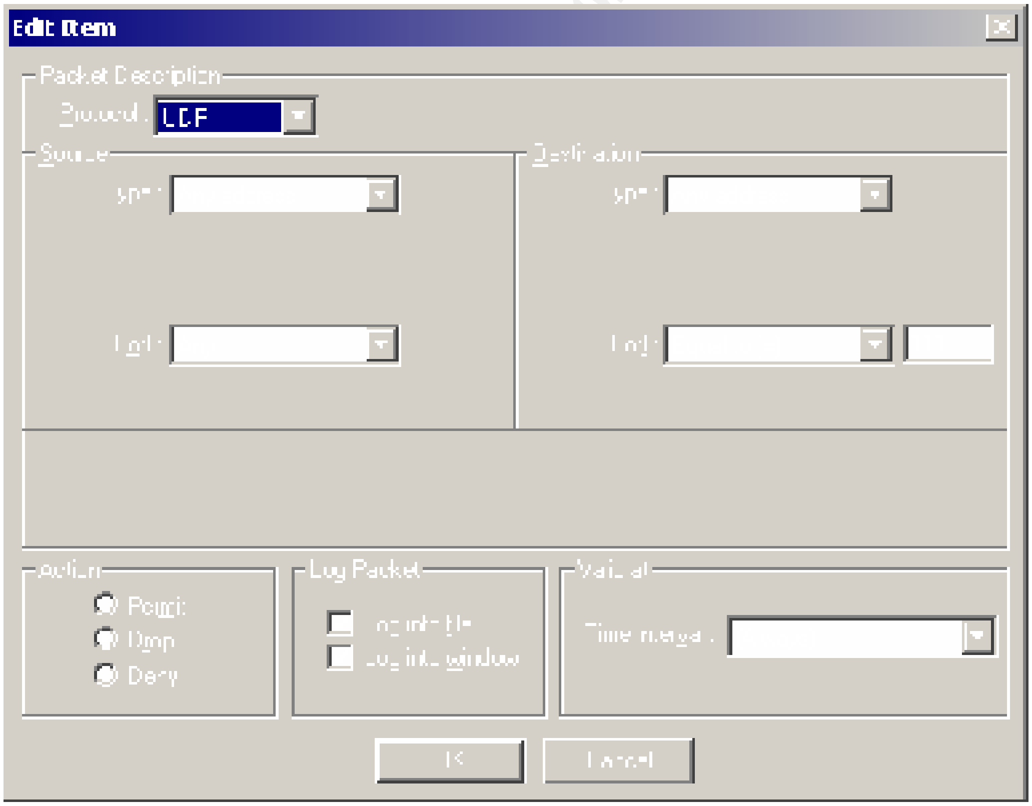


Figure 4 – RPC example: portmap

In Figure 4, we block attempts to 111/udp (portmap), drop the packet and log.

4. NetBIOS in Windows NT -- 135 (tcp and udp), 137 (udp), 138 (udp), 139 (tcp).
Windows 2000 - earlier ports plus 445(tcp and udp)

NetBIOS allows for network file sharing on Microsoft systems. On Windows NT, NetBIOS also can be used to gather information regarding users, groups, registry keys, hidden shares and other important system information. NetBIOS can easily be misconfigured and expose systems to a serious threat. We want to completely eliminate the possibility of connection to any system that is misconfigured. Also, NetBIOS is very fond of UDP broadcasts; we do not want these packets escaping out into the wild.

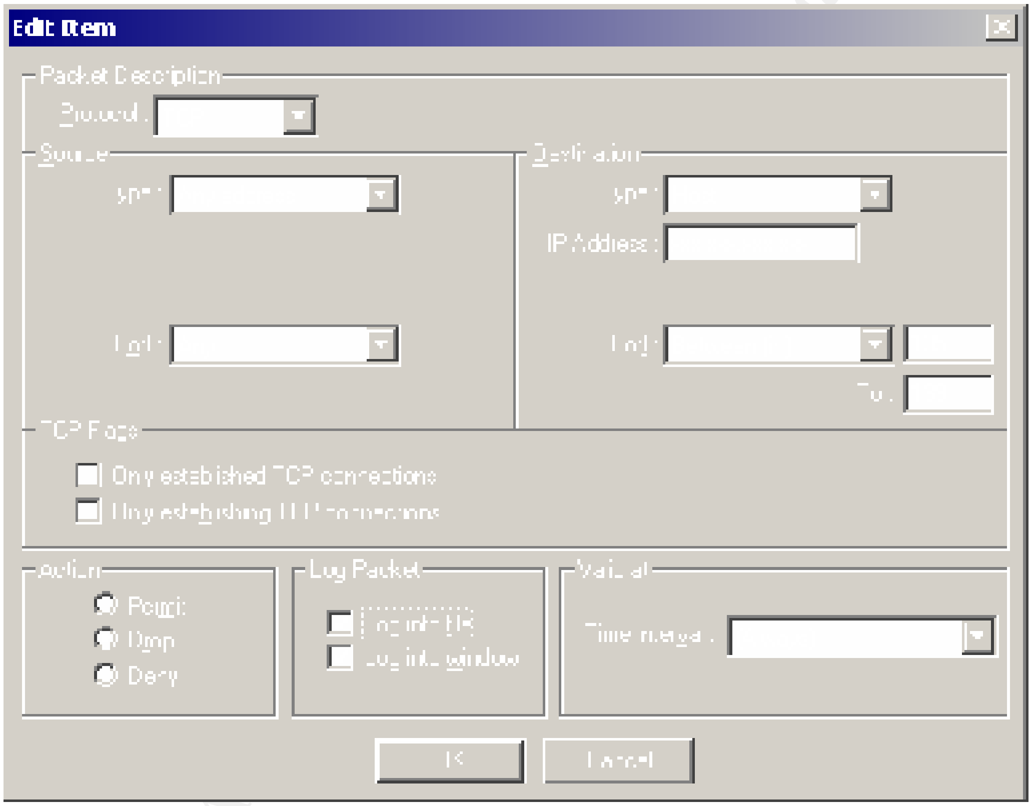


Figure 5 – NetBIOS example: multiple tcp ports

In Figure 5, I'm blocking a range of ports 135-139/tcp. I drop the packet and log it.

5. X Windows -- 6000/tcp through 6255/tcp

There are many dangers associated with unprotected X (pun intended). X Windows is a wonderful interface to the Unix system for those who find the black console so drab and boring. An X Windows Server will allow a user to open an X Windows session on one system from a separate system. However, if misconfigured, X Windows can pose a serious threat to any system. According to the CIAC document *Securing X Windows*[2],

> Any client that can access a server can potentially access and change any X communications that take place on it. This could include the following:
>
> - Modifying session parameters.
> - Create/destroy windows - Was that document saved before the window mysteriously disappeared?
> - Capture X events - For example, reading keystrokes on an `xterm` window, which include a login and password.
> - Create X events - For example, sending keystroke sequences to an emacs window, or an `xterm` window, to execute a command.

It makes sense to block any and all access to X Windows (in both directions).

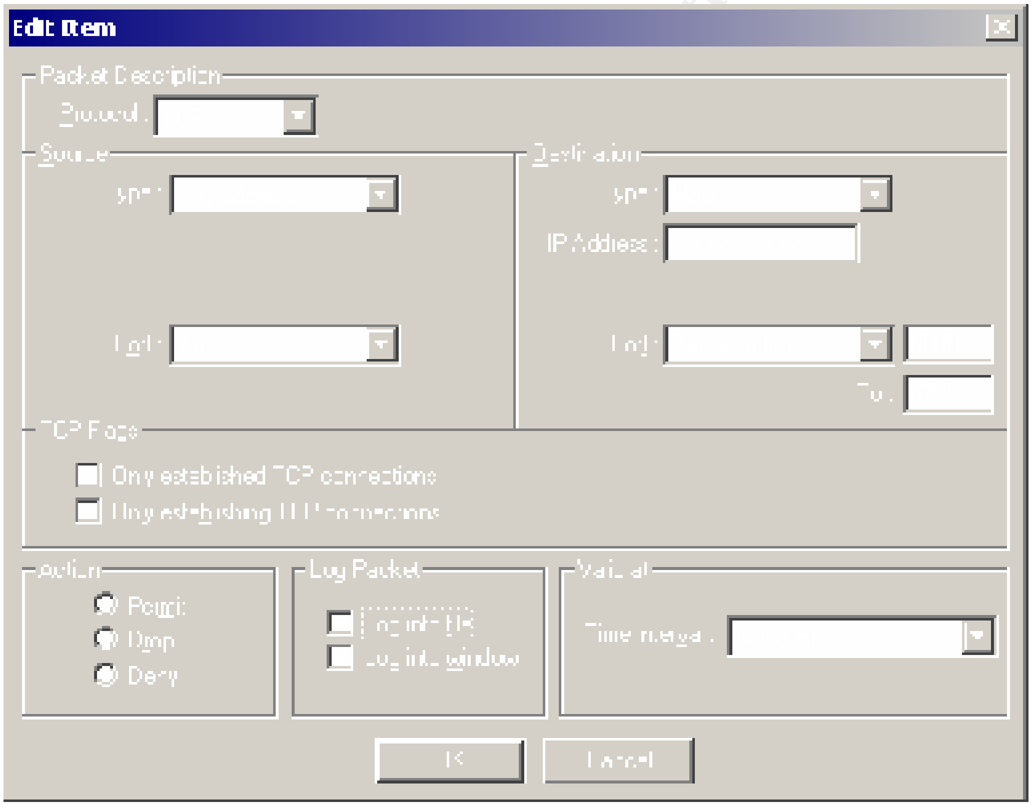


Figure 6 – Blocking X Windows traffic

Again we use a port range in Figure 6, blocking 6000-6255/tcp (X Windows). We drop the packet and log it once again.

6. Naming services-- DNS (53/udp) to all machines that are not DNS servers, DNS zone transfers (53/tcp) except from external secondaries, LDAP (389/tcp and 389/udp)

DNS is the service that translates www.sans.org into 167.216.133.33. Humans like names, computers like numbers. DNS can give an attacker a wealth of information (like the name and IP address of every host in your organization). This is why DNS access should only be allowed to external DNS servers that only contain the relevant information needed for your external systems. Zone transfers should be eliminated, as well. LDAP acts like a phone book. How many outsiders do you allow to look at your company directory? Not many is my guess. Since there is no externally accessible DNS server, we disallow all access to DNS services inbound. We will do the same for LDAP.

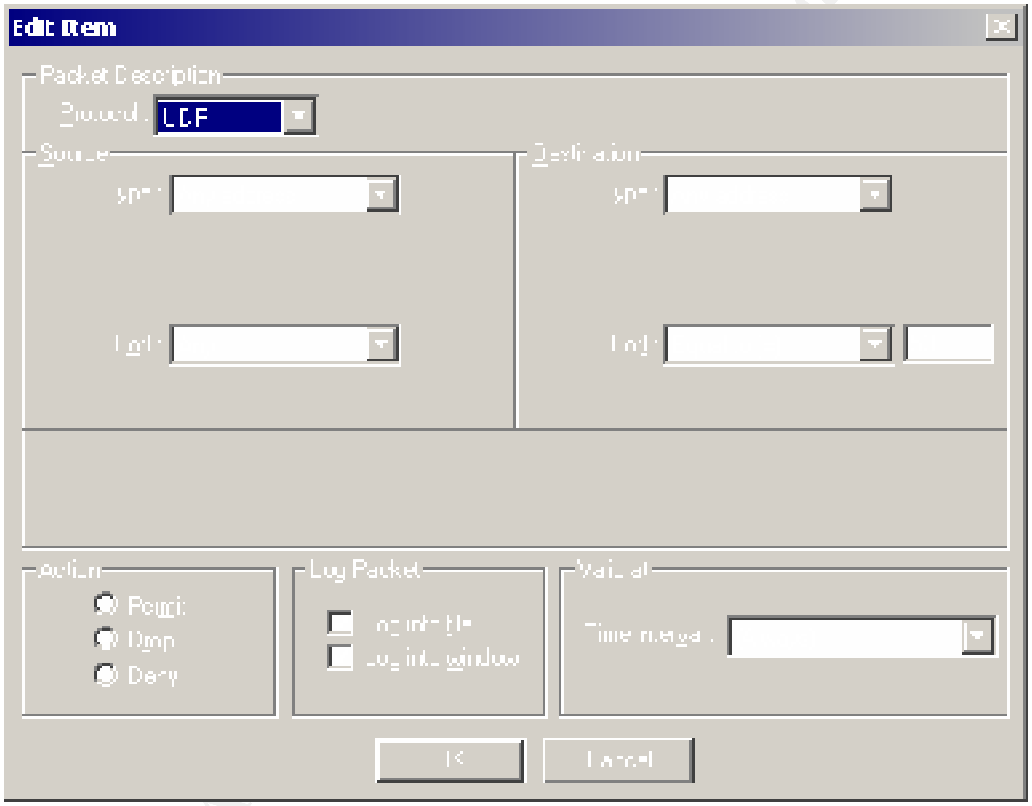

Protocol: UDP

Figure 7 – Blocking 53/udp: dns

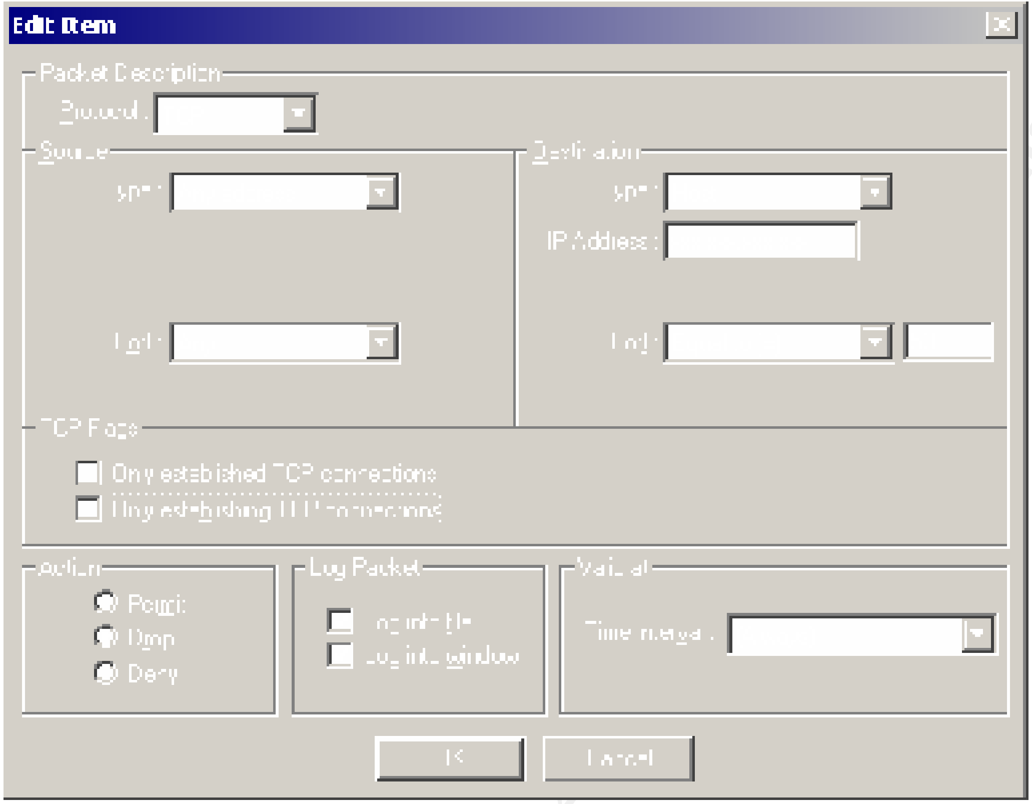


Figure 8 – Blocking 53/tcp: dns zone transfers

In Figures 7 and 8, we block access to both 53/tcp and 53/udp. Drop and log is the order of the day.

7. Mail-- SMTP (25/tcp) to all machines, which are not external mail relays, POP (109/tcp and 110/tcp), IMAP (143/tcp)

e-Mail is a wonderful tool. However, we need to block access to systems that are not external mail relays. This is because other internal mail systems listening on port 25 may be vulnerable to spam-relaying, for example. The external mail relay should be as resistant as possible. Certain implementations of POP and IMAP servers have programming vulnerabilities. They are also susceptible to brute force attempts. I don't like the idea of someone having the ability to brute force my e-mail account. In this example, external access will only be allowed to our external mail relay.

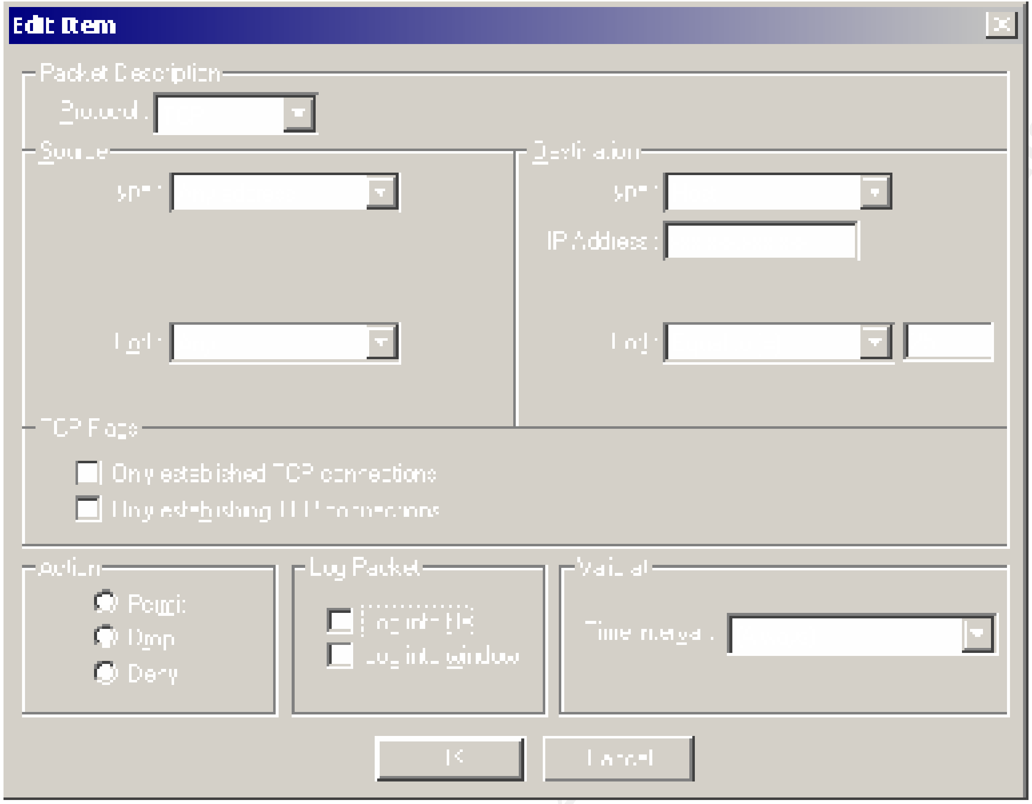


Figure 9 – Permit port 25/tcp – smtp

Figure 9 has us allowing port 25 inbound. However, Figure 10 will show you what we do with the packets.

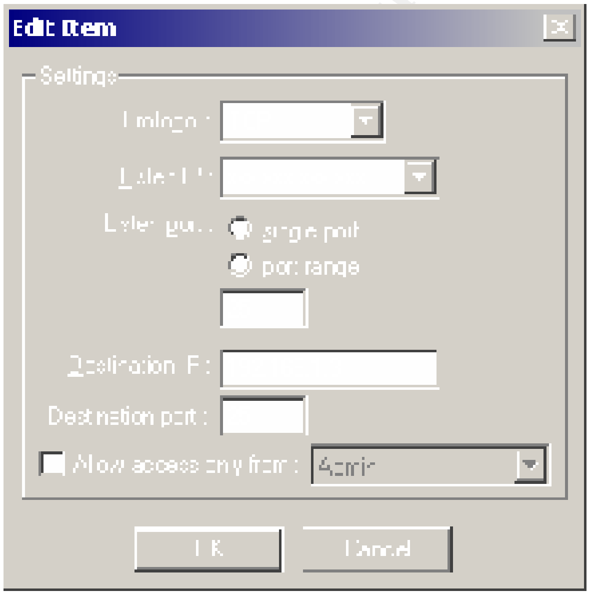


Figure 10 – Port mapping port 25 to external mail server

In Figure 10, we say all port 25/tcp traffic should be routed to port 25/tcp on 192.168.1.3.

8. Web-- HTTP (80/tcp) and SSL (443/tcp) except to external Web servers, may also want to block common high-order HTTP port choices (8000/tcp, 8080/tcp, 8888/tcp, etc.)

Everyone knows and loves the Web, however multiple vulnerabilities are associated with web servers. It is best to block these ports, with exception to secured Web servers. Since none are contained in this example, we do not have to address this issue.

9. "Small Services"-- ports below 20/tcp and 20/udp, time (37/tcp and 37/udp)

The small services, e.g. echo, chargen, discard, can be utilized for testing purposes. The time services retrieves the time from a system. One vulnerability of the small services is to spoof a packet containing the target host's own IP address. The source port is chargen and the destination port is echo. The result is a denial of service and that is a bad thing.

10. Miscellaneous-- TFTP (69/udp), finger (79/tcp), NNTP (119/tcp), NTP (123/tcp), LPD (515/tcp), syslog (514/udp), SNMP (161/tcp and 161/udp, 162/tcp and 162/udp), BGP (179/tcp), SOCKS (1080/tcp)

Each of these services has some vulnerability. Rather than go into detail for each one, I will explain a few. Back in the old days, finger served a purpose. This was when people trusted each other. Today, finger will allow an attacker to find out account names and login status for a user. TFTP is a connectionless file transfer protocol. It does not have the overhead associated with FTP, however it does not have the authentication either. If misconfigured, I can tftp your /etc/passwd (or /etc/shadow) file. SNMP, in my experience, is very easily misconfigured. SNMP is a good tool, however, if misconfigured, it can give away a whole lot of information. We will block all of these miscellaneous services.

11. ICMP-- block incoming echo request (ping and Windows traceroute), block outgoing echo replies, time exceeded, and unreachable messages

ICMP is a mixed blessing. On one hand it is a wonderful protocol, on the other hand it can cause nightmares. Have you ever heard of the Ping o' Death? ICMP.

By blocking incoming ICMP echo requests, we nullify the possibility of our systems being used as smurf amplifiers (a spoofed ICMP echo request packet is sent to a internal broadcast address, the internal hosts reply back to the victim host). Blocking outgoing echo replies acts as back up so we are positive that we are not aiding and abetting a DoS attack. Outgoing time-exceeded messages and outgoing ICMP unreachable messages allow attackers to do reverse mappings of networks. It's best to block these ICMP types.

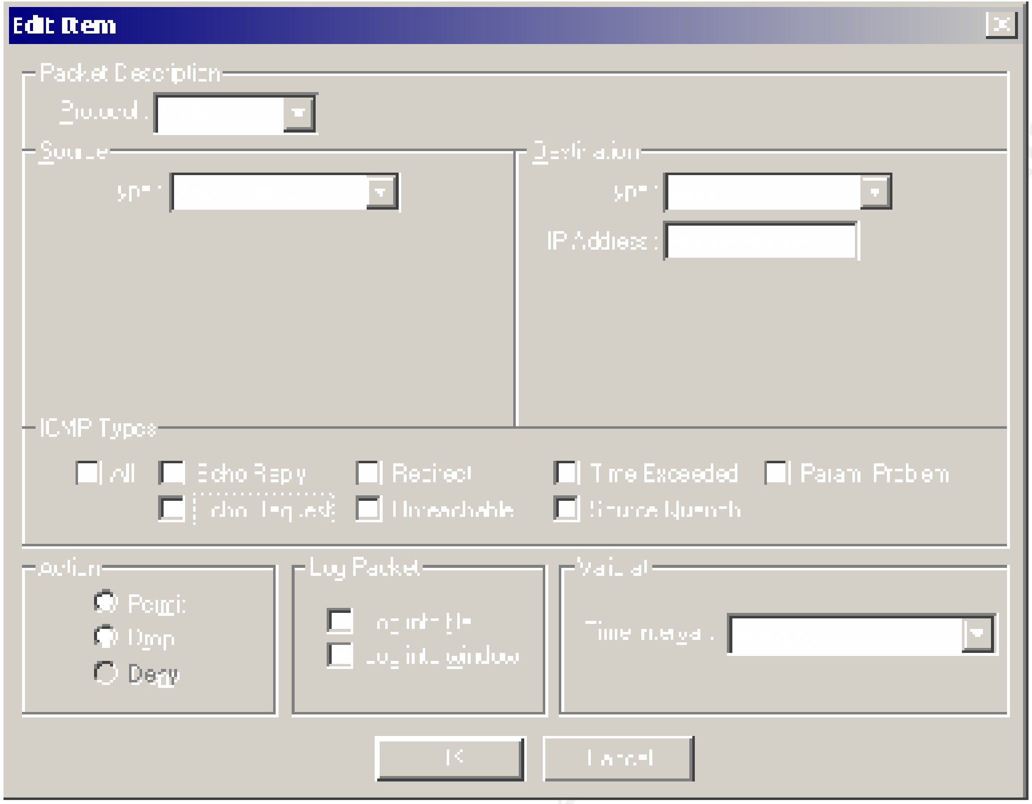


Figure 11 – Blocking Incoming Echo Request

Figure 11 is a different look we have a Source and a Destination, but now we have ICMP Types listed. Notice also that Deny is grayed out. This is because we can’t send an ICMP packet in response.

Figure 11 illustrates the method of blocking incoming ICMP echo requests. We drop the packets and log them.

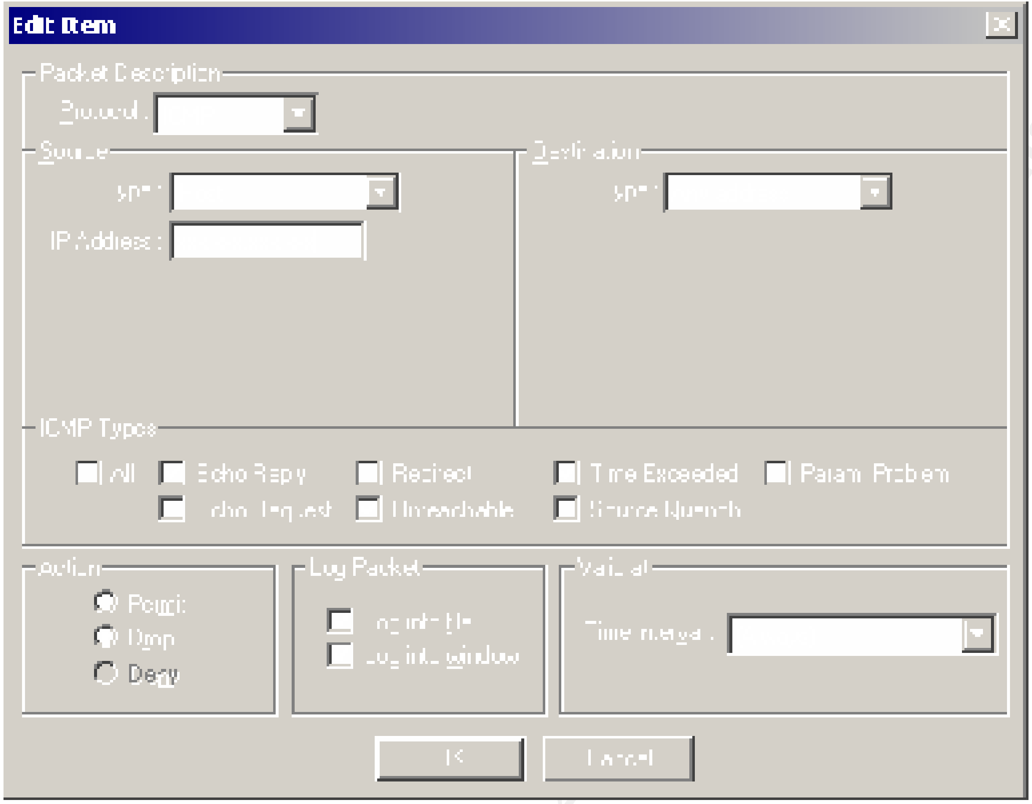


Figure 12 – Blocking outgoing ICMP echo reply, time exceeded and unreachable

In Figure 12, we block the outgoing ICMP echo replies, time exceeded and unreachable messages. The packets are dropped and logged.

Ronald Ross
SANS DC 2000
GIAC Firewall and Perimeter Protection Practical
August 15, 2000

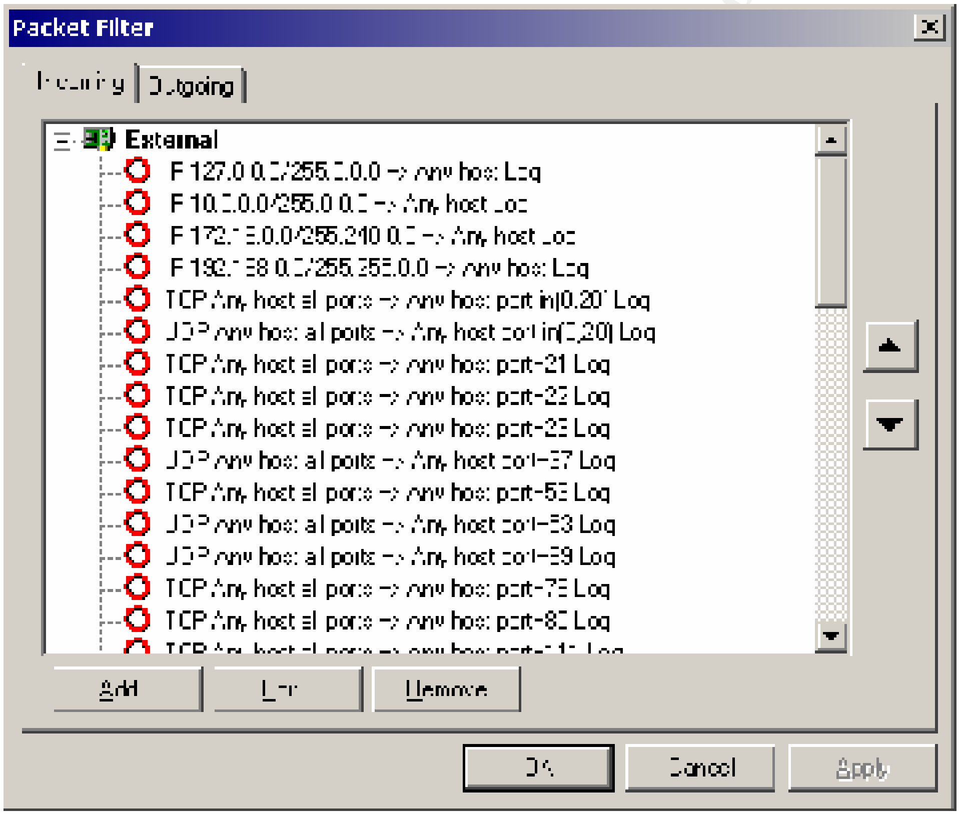
III. Rule Order

Rules are shown in the following manner

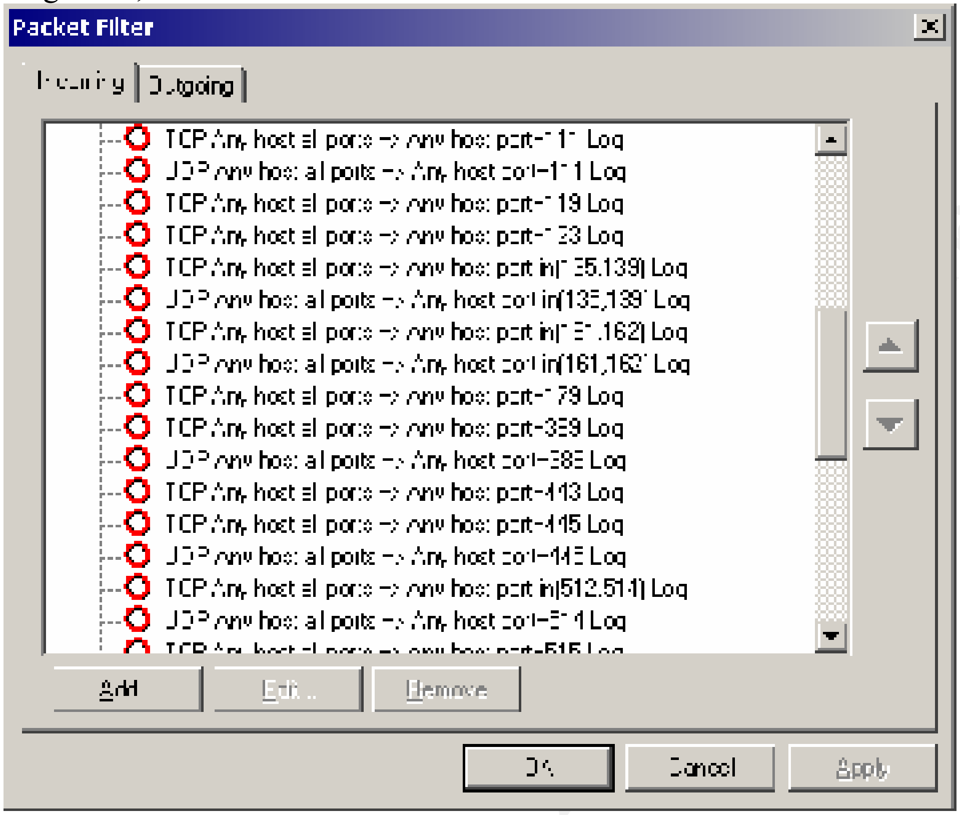
ActionIcon Protocol Source SourcePort => Destination Destination Port (Log)

Here are multiple screen shots to illustrate rule order. For the blocked packets, I chose to order them in Ascending order (so that I could find them easier). I block the packets first, then I put in the permit rules, then I block all other traffic (the catch all rule). WinRoute checks packets from the top down.

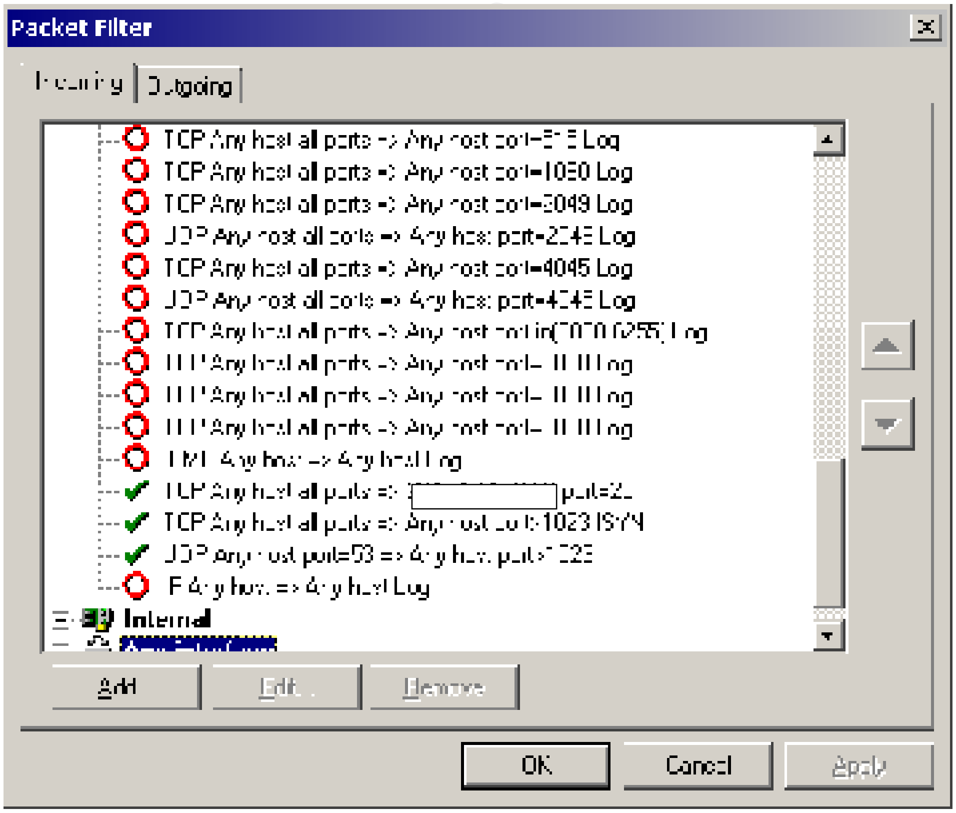
Incoming Rules



```
External
  IP 127.0.0.0/255.0.0.0 -> Any host Log
  IP 10.0.0.0/255.0.0.0 -> Any host Log
  IP 172.16.0.0/255.240.0.0 -> Any host Log
  IP 192.168.0.0/255.255.0.0 -> Any host Log
  TCP Any host all ports -> Any host port in[0,20] Log
  UDP Any host all ports -> Any host port in[0,20] Log
  TCP Any host all ports -> Any host port-21 Log
  TCP Any host all ports -> Any host port-22 Log
  TCP Any host all ports -> Any host port-23 Log
  UDP Any host all ports -> Any host port-37 Log
  TCP Any host all ports -> Any host port-53 Log
  UDP Any host all ports -> Any host port-53 Log
  UDP Any host all ports -> Any host port-69 Log
  TCP Any host all ports -> Any host port-79 Log
  TCP Any host all ports -> Any host port-80 Log
```



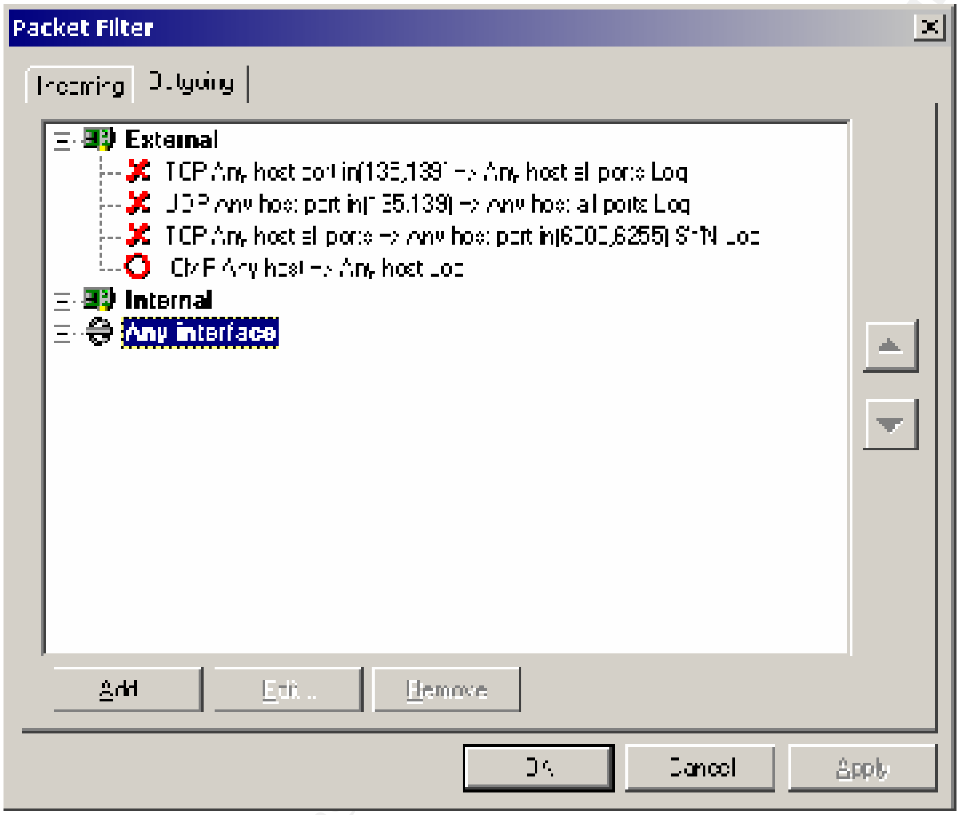
```
TCP Any host all ports => Any host port=111 Log
UDP Any host all ports => Any host port=111 Log
TCP Any host all ports => Any host port=119 Log
TCP Any host all ports => Any host port=123 Log
TCP Any host all ports => Any host port in(135,139) Log
UDP Any host all ports => Any host port in(135,139) Log
TCP Any host all ports => Any host port in(161,162) Log
UDP Any host all ports => Any host port in(161,162) Log
TCP Any host all ports => Any host port=179 Log
TCP Any host all ports => Any host port=389 Log
UDP Any host all ports => Any host port=389 Log
TCP Any host all ports => Any host port=443 Log
TCP Any host all ports => Any host port=445 Log
UDP Any host all ports => Any host port=445 Log
TCP Any host all ports => Any host port in(512,514) Log
UDP Any host all ports => Any host port=514 Log
TCP Any host all ports => Any host port=515 Log
```



```
TCP Any host all ports => Any host port=515 Log
TCP Any host all ports => Any host port=1080 Log
TCP Any host all ports => Any host port=2049 Log
UDP Any host all ports => Any host port=2049 Log
TCP Any host all ports => Any host port=4045 Log
UDP Any host all ports => Any host port=4045 Log
TCP Any host all ports => Any host port in(6000,6255) Log
[illegible] Any host all ports => Any host port=[illegible] Log
[illegible] Any host all ports => Any host port=[illegible] Log
[illegible] Any host all ports => Any host port=[illegible] Log
ICMP Any host => Any host Log
TCP Any host all ports => [redacted] port=25
TCP Any host all ports => Any host port>1023 !SYN
UDP Any host port=53 => Any host port>1023
IP Any host => Any host Log
Internal
```

As can be seen in the incoming rules, after specifying the packets to block, I permit in port 25 for mail. I then allow all communications in from internally initiated traffic (SYN bit is not the only bit set - !SYN), notice that it is only for the ephemeral ports greater than 1023. For UDP I only allow client communications to external DNS servers. I then block everything not specified.

Outgoing rules



External
- ✗ TCP Any host port in[135,139] -> Any host all ports Log
- ✗ UDP Any host port in[135,139] -> Any host all ports Log
- ✗ TCP Any host all ports -> Any host port in[6000,6255] SYN Log
- ○ ICMP Any host -> Any host Log

Internal

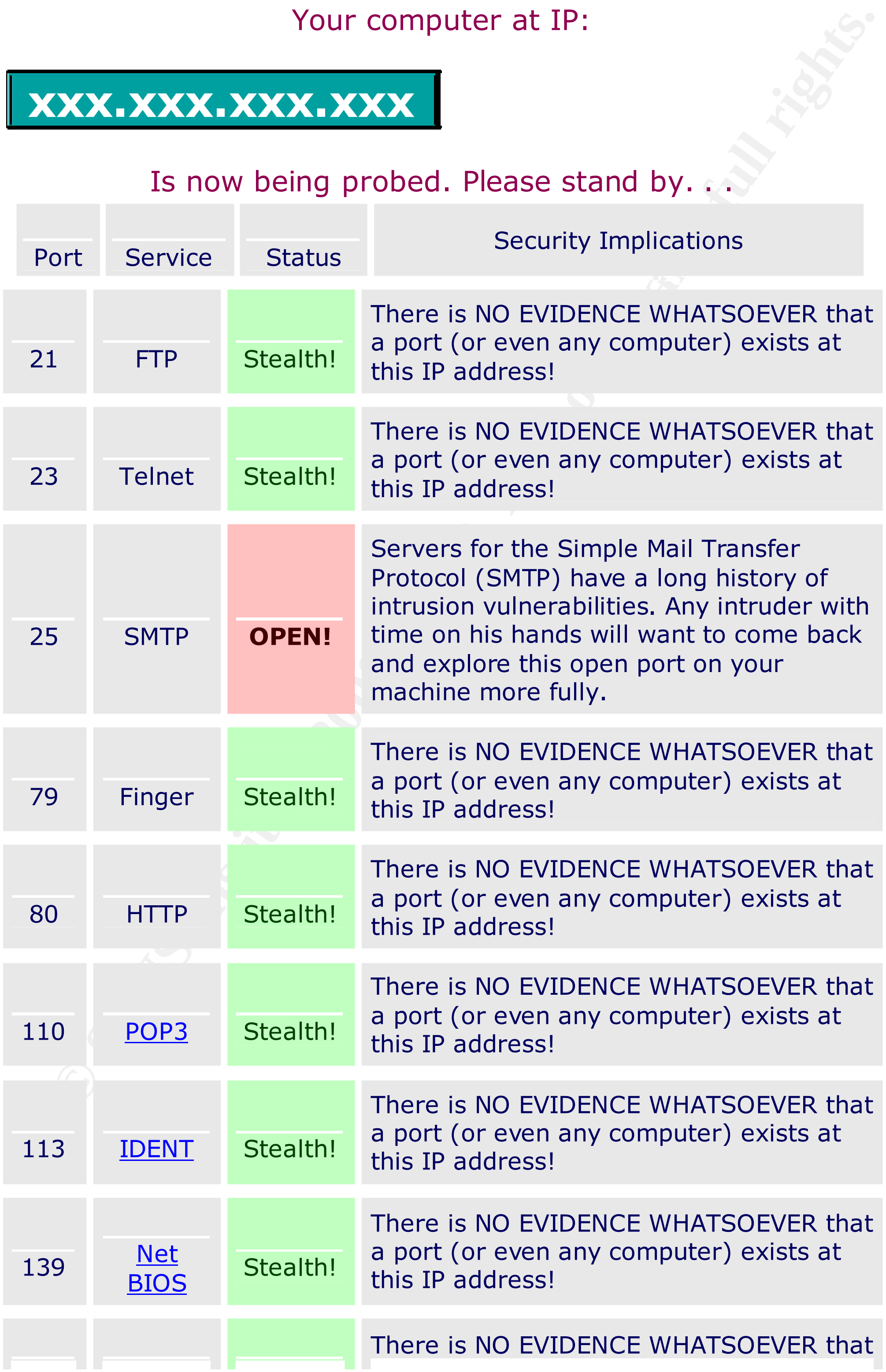
Any interface

For the Outgoing rules I deny all NetBIOS traffic generated internally (tcp and udp). I also deny access to X Windows, to be safe. I also drop the ICMP traffic detailed in Figure 12.

How does it work? Quite well, actually. Below is a scan from Shields UP! (http://grc.com).

Quickly Check for Connectable Listening Internet Ports

Port Probe attempts to establish standard TCP/IP (Internet) connections on a handful of standard, well-known, and often vulnerable Internet service ports on **YOUR** computer. Since this is being done from **our** server, successful connections demonstrate which of your ports are "open" and actively soliciting connections from passing Internet port scanners.



Your computer at IP:

xxx.xxx.xxx.xxx

Is now being probed. Please stand by. . .

Port	Service	Status	Security Implications
21	FTP	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
23	Telnet	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
25	SMTP	**OPEN!**	Servers for the Simple Mail Transfer Protocol (SMTP) have a long history of intrusion vulnerabilities. Any intruder with time on his hands will want to come back and explore this open port on your machine more fully.
79	Finger	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
80	HTTP	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
110	POP3	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
113	IDENT	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
139	Net BIOS	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!
			There is NO EVIDENCE WHATSOEVER that

143	IMAP	Stealth!	a port (or even any computer) exists at this IP address!
443	HTTPS	Stealth!	There is NO EVIDENCE WHATSOEVER that a port (or even any computer) exists at this IP address!

This is exactly what I expected to see. Attempts from the scan were all logged to the firewall's logs and console.

```
[15/Aug/2000 22:52:13] Packet filter: ACL 1:24 External: drop packet in: TCP 207.71.92.221:2998 ->
xxx.xxx.xxx.xxx:21

[15/Aug/2000 22:52:16] Packet filter: ACL 1:24 External: drop packet in: TCP 207.71.92.221:2998 ->
xxx.xxx.xxx.xxx:21

[15/Aug/2000 22:52:23] Packet filter: ACL 1:24 External: drop packet in: TCP 207.71.92.221:2998 ->
xxx.xxx.xxx.xxx:21

[15/Aug/2000 22:53:02] Packet filter: ACL 1:4 External: drop packet in: TCP 207.71.92.221:3192 ->
xxx.xxx.xxx.xxx:23

[15/Aug/2000 22:53:05] Packet filter: ACL 1:4 External: drop packet in: TCP 207.71.92.221:3192 ->
xxx.xxx.xxx.xxx:23

[15/Aug/2000 22:53:12] Packet filter: ACL 1:4 External: drop packet in: TCP 207.71.92.221:3192 ->
xxx.xxx.xxx.xxx:23

[15/Aug/2000 22:53:25] Packet filter: ACL 1:4 External: drop packet in: TCP 207.71.92.221:3192 ->
xxx.xxx.xxx.xxx:23

[15/Aug/2000 22:54:01] Packet filter: ACL 1:38 External: drop packet in: TCP 207.71.92.221:3531 ->
xxx.xxx.xxx.xxx:79

[15/Aug/2000 22:54:05] Packet filter: ACL 1:38 External: drop packet in: TCP 207.71.92.221:3531 ->
xxx.xxx.xxx.xxx:79

[15/Aug/2000 22:54:11] Packet filter: ACL 1:38 External: drop packet in: TCP 207.71.92.221:3531 ->
xxx.xxx.xxx.xxx:79

[15/Aug/2000 22:54:24] Packet filter: ACL 1:38 External: drop packet in: TCP 207.71.92.221:3531 ->
xxx.xxx.xxx.xxx:79

[15/Aug/2000 22:54:50] Packet filter: ACL 1:32 External: drop packet in: TCP 207.71.92.221:3800 ->
xxx.xxx.xxx.xxx:80
```

```
[15/Aug/2000 22:54:54] Packet filter: ACL 1:32 External: drop packet in: TCP 207.71.92.221:3800 ->
xxx.xxx.xxx.xxx:80

[15/Aug/2000 22:55:00] Packet filter: ACL 1:32 External: drop packet in: TCP 207.71.92.221:3800 ->
xxx.xxx.xxx.xxx:80

[15/Aug/2000 22:55:13] Packet filter: ACL 1:32 External: drop packet in: TCP 207.71.92.221:3800 ->
xxx.xxx.xxx.xxx:80

[15/Aug/2000 22:55:40] Packet filter: ACL 1:48 External: drop packet in: TCP 207.71.92.221:4086 ->
xxx.xxx.xxx.xxx:110

[15/Aug/2000 22:55:43] Packet filter: ACL 1:48 External: drop packet in: TCP 207.71.92.221:4086 ->
xxx.xxx.xxx.xxx:110

[15/Aug/2000 22:55:50] Packet filter: ACL 1:48 External: drop packet in: TCP 207.71.92.221:4086 ->
xxx.xxx.xxx.xxx:110

[15/Aug/2000 22:56:03] Packet filter: ACL 1:48 External: drop packet in: TCP 207.71.92.221:4086 ->
xxx.xxx.xxx.xxx:110

[15/Aug/2000 22:56:29] Packet filter: ACL 1:48 External: drop packet in: TCP 207.71.92.221:4277 ->
xxx.xxx.xxx.xxx:113

[15/Aug/2000 22:56:32] Packet filter: ACL 1:48 External: drop packet in: TCP 207.71.92.221:4277 ->
xxx.xxx.xxx.xxx:113

[15/Aug/2000 22:56:39] Packet filter: ACL 1:48 External: drop packet in: TCP 207.71.92.221:4277 ->
xxx.xxx.xxx.xxx:113

[15/Aug/2000 22:56:52] Packet filter: ACL 1:48 External: drop packet in: TCP 207.71.92.221:4277 ->
xxx.xxx.xxx.xxx:113

[15/Aug/2000 22:57:18] Packet filter: ACL 1:5 External: drop packet in: TCP 207.71.92.221:4559 ->
xxx.xxx.xxx.xxx:139

[15/Aug/2000 22:57:21] Packet filter: ACL 1:5 External: drop packet in: TCP 207.71.92.221:4559 ->
xxx.xxx.xxx.xxx:139

[15/Aug/2000 22:57:41] Packet filter: ACL 1:5 External: drop packet in: TCP 207.71.92.221:4559 ->
xxx.xxx.xxx.xxx:139

[15/Aug/2000 22:58:07] Packet filter: ACL 1:48 External: drop packet in: TCP 207.71.92.221:4777 ->
xxx.xxx.xxx.xxx:143

[15/Aug/2000 22:58:11] Packet filter: ACL 1:48 External: drop packet in: TCP 207.71.92.221:4777 ->
xxx.xxx.xxx.xxx:143

[15/Aug/2000 22:58:17] Packet filter: ACL 1:48 External: drop packet in: TCP 207.71.92.221:4777 ->
xxx.xxx.xxx.xxx:143
```

```
[15/Aug/2000 22:58:30] Packet filter: ACL 1:46 External: drop packet in: TCP 207.71.92.221:4777 ->
xxx.xxx.xxx.xxx:143

[15/Aug/2000 22:58:57] Packet filter: ACL 1:24 External: drop packet in: TCP 207.71.92.221:1085 ->
xxx.xxx.xxx.xxx:443

[15/Aug/2000 22:59:00] Packet filter: ACL 1:24 External: drop packet in: TCP 207.71.92.221:1085 ->
xxx.xxx.xxx.xxx:443
```

In addition, running nmap from an external IP would give more benefit, as you can test all ports.

WinRoute is not that difficult to configure, with some knowledge of TCP/IP. It can help to defend the perimeter of a small site quite well.

Ronald Ross
SANS DC 2000
GIAC Firewall and Perimeter Protection Practical
August 15, 2000

References

SANS Institute, *How To Eliminate The Ten Most Critical Internet Security Threats, Version 1.25*. http://www.sans.org/topten.htm, July 12, 2000.

Fisher, John. *Securing X Windows*, CIAC, August, 1995.

Shields UP! Port Scanning Online Utility. http://grc.com. August, 2000.