Global Information Assurance Certification Paper

John W. Jordan

GSEC Practical, Version 1.4b, Option 1

WATCH YOUR BIT-BUCKET

A Quick and Easy Alert to Some Things that Shouldn't Be

bit bucket *n.* The universal data sink (originally, the mythical receptacle used to catch bits when they fall off the end of a register during a shift instruction). Discarded, lost, or destroyed data is said to have 'gone to the bit bucket'[1].

The Jargon Dictionary - **http://info.astrian.net/jargon/terms/b/bit_bucket.html**

Contents

1 The Jargon Dictionary

1. Introduction

The Bit Bucket is the mythical destination for all discarded data. But what should we discard? The answer from a network perspective is: everything except desired traffic. That is a bit trite, and far harder to execute flawlessly than it sounds. There are several methods for discarding unwanted packets; firewall rules, router Access Control Lists (packet filtering), and routing. Each of these has their place in a "defense in depth" strategy. In addition to using careful design and configuration to discard all unwanted traffic (and only unwanted traffic) it is also important to monitor these discards.

The two areas I will specifically discuss in this paper are "Bit Buckets" to prevent and detect spoofing (network (IP) traffic with an invalid source address), and worm scanning (packets with a destination address not valid on our internal network). Monitoring these discarded packets can provide an early warning of malicious activity. I will provide two examples of how to set up for and monitor these discarded packets, and also provide some quick ways to identify and locate infected or misconfigured hosts.

2. Assumptions, Limitations, and Caveats (see Drawing 1)

- All examples apply to a private network behind a firewall

- We are using private (RFP 1918[2]) address space throughout our internal (private) network.
- All traffic between our private network and the internet goes through a proxy, with network address translation to/from our public (ISP attached) addresses.
- All network traffic is IP (although the same bit bucket principals apply to other protocols).
- All routers are Cisco (though other brands have similar capabilities).
- The example network is very simplified. All cited hosts and addresses are fictitious.
- The reader is invited to use due care if choosing to implement any network change.

3. Spoofing

Misconfigured hosts, mobile users, and spoofing are the three main causes of internal network traffic with an invalid source address. Misconfigured hosts tend to be quickly corrected, since they will be unable to set up a TCP session, or in fact receive any traffic from remote subnets, so the affected user calls in someone who knows how to un-misconfigure their machine. Mobile users who move from the conference room subnet to their desk subnet to the patio wireless often transmit a few packets using their previously assigned DHCP address before automatically getting a valid address on their current subnet. Spoofing, however, has been used in a number of attacks, including Smurf, Fraggle, Teardrop, and Land Attack. Most spoofing attacks are "denial of service" or "distributed denial of service" (DoS or DDoS)[3]. A spoofing host, on the other hand, will often use two source addresses, one genuine address so it can receive traffic, and one or more fake addresses for nefarious activity, such as denial-of-service attacks, session hijacking, or third-party port scanning.

Spoofing is the deliberate use of a source address not assigned to that host. Spoofing is very difficult to use in most exploits because the attacking host generally needs to see replies from the targeted machine to set up a TCP session, for example. A targeted machine would reply to the source (spoofed) address, and not to the true source. A machine launching a denial-of-service attack, though, may not need to see replies from the target, and by spoofing its source address, becomes much harder to locate. In fact it can spoof the address of the true targeted machine to a large number of hosts in an attempt to overwhelm the target with a huge number of bogus replies. A denial-of-service attack from within our network can be devastating because it would not be limited by our internet connection bandwidth, but would be able to attack with almost the full rate of the local LAN connection.

Spoofing must be blocked (and detected) as close to the source as possible. That's

---

2 Y. Rekhter, et al - Address Allocation for Private Internets

3 Tom Dunnigan - Backtracking Spoofed Packets

why RFC 2827[4] recommends all ISPs filter all traffic from their customers so that only packets with source addresses assigned to that customer are accepted for forwarding to the internet. To protect against internal spoofing on our private network, however, we need to block this activity at the local subnet router.

4. Local Anti-Spoofing Methods

The simplest method to insure that cross-subnet spoofing does not leave the originating host's subnet is with an ingress Access Control List (ACL) on each local subnet interface. This is effective, but can be a drain on router resources, especially with today's very fast and heavily loaded LANs. A preferred method, recommended by Cisco, is "Unicast Reverse Path Forwarding" on each local subnet interface. This method, however, requires Cisco Express Forwarding (CEF) (a proprietary layer-3 switching capability) be turned on. I will cover both of these methods below.

Note in Figure 2 that router **LANrouter-1** can easily detect incoming traffic with a bogus source address on each local subnet interface, by comparing the network portion of the source address with the network portion of that interface address. Unfortunately, neither of these methods will detect or stop spoofing that uses a false address on the same subnet as the true address. Implementing one of these local anti-spoofing methods can still be of enormous benefit anyway, by guaranteeing that the new "worm infected host" that we just detected spraying our entire network with infectious packets is in fact located on the subnet that we think it is. More on detecting that worm later.

a. Filtering with Access Control Lists

To stop cross-subnet spoofing with an access control list we simply configure a standard ACL on each inbound interface, permitting packets with source addresses on

4 P. Ferguson and D. Senie - RFC 2827, Network Ingress Filtering

the local subnet, and DHCP broadcasts, and denying everything else. In configuration mode, enter:

```
ip access-list standard spoof1 permit 10.10.1.0 0.0.0.255
ip access-list standard spoof1 permit 0.0.0.0  0.0.0.0
<implicit deny all>
```

and apply this to interface Ethernet 1:

```
interface ethernet1
ip address 10.10.1.1 255.255.255.0
access-group spoof1 in
no shutdown
```

This turns on interface ethernet 1, assigns an ip address of 10.10.1.1 with a 24 bit subnet mask, and installs access list "spoof1" on the inbound direction. This access list allows all traffic entering that interface from hosts with source addresses from 10.10.1.2 through 10.10.1.254, and denies inbound traffic from all other source addresses. The "implicit deny" supplied by the operating system (IOS) in all Cisco routers works fine, but gives no indication of when it has been invoked. If we are interested in these denied (discarded) packets (and we should be, since that's the whole point of this paper), we can place an explicit deny statement at the end of the access control list:

```
ip access-list standard spoof1 permit 10.10.1.0 0.0.0.255
ip access-list standard spoof1 permit 0.0.0.0  0.0.0.0
ip access-list standard spoof1 deny all
```

Now, the router console command *show access-list spoof1* will tell us how many times each line has been invoked:

```
Standard IP access list spoof1
permit 10.10.1.0 0.0.0.255 (2757956 matches)
permit 0.0.0.0 0.0.0.0 (2 matches)
deny all (457 matches)
```

To make this traffic even easier to see, we will send it to the router log, and the syslog server (10.10.254.5):

```
ip access-list standard spoof1 permit 10.10.1.0 0.0.0.255
ip access-list standard spoof1 permit 0.0.0.0  0.0.0.0
ip access-list standard spoof1 deny all log
logging 10.10.254.5
```

We would then add access list ***spoof2*** to the interface for subnet 10.10.2.0, and so on for all interfaces on all routers servicing local subnets. Note that we would not do this for interfaces connecting to other routers, since that would block traffic forwarded from other subnets.

b. Using Unicast Reverse Path Forwarding

Unicast Reverse Path Forwarding (RPF) is a much more efficient method (in terms of router resource requirements) of ensuring that ingress traffic packets are arriving on the expected router interface.

Cisco gives the following explanation of how Unicast RPF works:

> When Unicast RPF is enabled on an interface, the router examines all packets received as input on that interface to make sure that the source address and source interface appear in the routing table and match the interface on which the packet was received. This "look backwards" ability is available only when Cisco express forwarding (CEF) is enabled on the router, because the lookup relies on the presence of the Forwarding Information Base (FIB). CEF generates the FIB as part of its operation.
>
> Unicast RPF checks to see if any packet received at a router interface arrives on the best return path (return route) to the source of the packet. Unicast RPF does this by doing a reverse lookup in the CEF table. If the packet was received from one of the best reverse path routes, the packet is forwarded as normal. If there is no reverse path route on the same interface from which the packet was received, it might mean that the source address was modified. If Unicast RPF does not find a reverse path for the packet, the packet is dropped or forwarded, depending on whether an access control list (ACL) is specified in the `ip verify unicast reverse-path` interface configuration command.[5]

Adding Unicast RPF to local subnet interfaces is quite simple. In configuration mode we would make sure CEF is enabled, and configure the interface:

```
ip cef
interface ethernet1
ip address 10.10.1.1 255.255.255.0
ip verify unicast reverse-path
no shutdown
```

This configuration will cause Unicast RPF to examine the source address of each inbound packet on the ethernet 1 interface, and discard it if it doesn't match the local subnet network address. If we add an ACL, we can specify if the packet is dropped or forwarded, and add logging the same way we did with just plain ACL filtering. Unicast RPF is still more efficient, since it only consults the ACL if the source address does not match the reverse path.

```
~~ip access-list standard spoof1 permit 10.10.1.0 0.0.0.255~~
~~ip access-list standard spoof1 permit 0.0.0.0  0.0.0.0~~
ip access-list standard spoof1 deny all log
logging 10.10.254.5
```

Note that we don't need the first line, since packets with proper source addresses are automatically forwarded by Unicast RPF, and on Cisco Internet Operating System (IOS) version 12 and later, the second line that permits BootP and DHCP broadcasts is not needed, because Unicast RPF handles that. Our entire Unicast RPF configuration for interfaces ethernet 1 and 2 is:

```
ip cef
```

---

5 Unknown Author - Cisco Tech Notes - Configuring Unicast Reverse Path Forwarding

```
interface ethernet1
ip address 10.10.1.1 255.255.255.0
ip verify unicast reverse-path spoof1
no shutdown
interface ethernet2
ip address 10.10.2.1 255.255.255.0
ip verify unicast reverse-path spoof2
no shutdown
exit
ip access-list standard spoof1 deny all log
ip access-list standard spoof2 deny all log
logging 10.10.254.5
```

We would then add Unicast RPF to all interfaces on all routers servicing local subnets. Note that we would not do this for interfaces connecting to other routers, since that would block traffic forwarded from other subnets.

For further details and some caveats, consult Cisco's excellent Tech Note in reference 5

5. Malware Scanning

To replicate, worms must find other vulnerable hosts. The two most popular methods currently are via email broadcasts, or direct network probing, though IRC, Instant Messaging, and Peer-to-Peer are gaining in popularity. Email, IRC, and Peer-to-Peer count on humans at least clicking on the message, if not an attachment, so they are relatively slow to spread. Network propagating worms can spread with breath-taking speed as SQL Slammer proved. Such "network active" worms must probe network addresses. Since they do not have a way to know the bounds of the local network, many scan the entire internet address space.

An internet address is composed of four "octets" or 8-bit binary words (commonly called "bytes"), each of which can represent any of 256 values, from 0 to 255. Therefore the entire address space runs from 0.0.0.0 to 255.255.255.255, for a theoretical maximum of 4,294,967,296 addresses (256 X 256 X 256 X 256).

Different worms handle this search for vulnerable hosts in different ways. SQL Slammer, for example, generated a pseudo-random number and translated that to an address, so it sprayed over the entire address range[6]. Blaster[7], Donk, and Nachi (Welchia[7]) scan upward either sequentially, or with a stutter-step in an apparent attempt to avoid detection by network intrusion detection systems (NIDS). Some worms are even smart enough to start in the local subnet[8].

Slammer (aka Sapphire Worm and W32.SQLExp.Worm), used the UDP protocol instead of TCP, so it did not have to wait for session setup, It was a single self-contained worm in each 376 byte packet. SQL Slammer generated a pseudo-random

---

6 Joanne Pilker - MS SQL Slammer/Sapphire Worm

7 Frederic Perriot - Detecting network traffic that may be due to RPC worms

8 Kaoru Hayashi and Serghei Sevcenco - W32.HLLW.Nebiwo

number based on the local clock of the infected host, translated that to an IP address, and then sent itself out to that address. It then used a pseudo-random number generation algorithm to generate further almost random numbers to translate to addresses (It's one bug was in this process. Each host would not generate all possible addresses, but with enough hosts infected, all addresses were repeatedly probed)[6]. Since Slammer was sending to random addresses, most of the packets would end up in the routing bit bucket. So much so, that many routers were too busy sending ICMP Unreachable messages back to the infected hosts that they would not respond to a console log-in attempt (personal experience). This would not happen if the router had a default route available (see Advantages of a Default Route below).

Donk (aka W32/Donk, W32.HLLW.Donk, W32/Sdbot.worm, W32/Sdbot.worm.gen, W32/Sdbot.worm.gen.b, worm_donk, and Win32.Sdbot) primarily exploits a non-patchable vulnerability, i. e. poor security on target systems' drive sharing, or the credentials of the user logged on to an infected system are sufficient to access other systems on the network[9]. It scans for machines that have open drive shares that it can access by requesting it's host to establish a drive mapping to a random address. If there is a live host at that address, the obliging host sends a tcp syn back to the infected host to set up a session. If the session setup fails, the TCP stack of the infected (attacking) host reports that back to the application (Donk worm) which then generates a new address near the last one (a few addresses up or down, but always trending upward) and instructs the host to try again. If the session setup succeeds, the worm tries to log in to one of the default shares (Admin, Owner, or Guest) using a list of hard-coded (weak) passwords. If successful, it copies itself over, and schedules a job to run on the victim machine to start its new clone[9]. If the infected host is on our private network, the worm will spend most of it's time scanning public address space, not directly reachable from inside. This traffic ends up in the routing bit bucket. Below is an actual capture of Donk traffic. The first two octets of the destination address are masked with "nn" to hide the actual public network it was attempting to scan:

DONK:

```
0.0   10.14.128.152  nn.nn.110.205    TCP 3656  > 139 [SYN] Seq=559208343  Ack=0   Win=16384 Len=0
0.001 10.14.128.152  nn.nn.110.207    TCP 3657  > 139 [SYN] Seq=559257957  Ack=0   Win=16384 Len=0
0.005 10.14.128.152  nn.nn.110.211    TCP 3662  > 139 [SYN] Seq=559516496  Ack=0   Win=16384 Len=0
0.099 10.14.128.152  nn.nn.110.209    TCP 3669  > 139 [SYN] Seq=559869316  Ack=0   Win=16384 Len=0
0.106 10.14.128.152  nn.nn.110.217    TCP 3675  > 139 [SYN] Seq=560153234  Ack=0   Win=16384 Len=0
0.109 10.14.128.152  nn.nn.110.222    TCP 3679  > 139 [SYN] Seq=560382655  Ack=0   Win=16384 Len=0
0.111 10.14.128.152  nn.nn.110.225    TCP 3682  > 139 [SYN] Seq=560528224  Ack=0   Win=16384 Len=0
0.2   10.14.128.152  nn.nn.110.223    TCP 3683  > 139 [SYN] Seq=560611542  Ack=0   Win=16384 Len=0
0.202 10.14.128.152  nn.nn.110.234    TCP 3685  > 139 [SYN] Seq=560721974  Ack=0   Win=16384 Len=0
0.204 10.14.128.152  nn.nn.110.236    TCP 3687  > 139 [SYN] Seq=560845414  Ack=0   Win=16384 Len=0
```

Nachi (aka nachia, Win32.Nachi.Worm, Welchia, W32.Welchia.Worm, W32.Nachi.worm, W32/Nachi-A) is another network worm that scans network addresses for vulnerable hosts. The worm author though, found a faster way to scan. Instead of calling the infected host to set up a TCP session (requiring the famous syn, syn/ack, and ack handshake), this worm calls ping, which requires much less resources

---

6 Joanne Pilker - MS SQL Slammer/Sapphire Worm, pp. 5-6

9 Unknown Author – Network Associates - W32/Sdbot.worm

on the host machine, so it can scan non-responsive addresses much quicker. So quick, in fact, that a few infected machines can cause an effective denial-of-service on a small network[10]. Like Donk, though, most of Nachi's pings (ICMP Echo Requests) end up in the bit bucket, because they are bound for external addresses.

NACHI

```
0.04    10.2.7.184    nn.nn.254.180    ICMP    Echo (ping) request
0.05    10.2.7.184    nn.nn.254.182    ICMP    Echo (ping) request
0.06    10.2.7.184    nn.nn.254.186    ICMP    Echo (ping) request
0.1     10.2.7.184    nn.nn.254.187    ICMP    Echo (ping) request
0.15    10.2.7.184    nn.nn.254.188    ICMP    Echo (ping) request
0.16    10.2.7.184    nn.nn.254.189    ICMP    Echo (ping) request
0.25    10.2.7.184    nn.nn.254.192    ICMP    Echo (ping) request
0.26    10.2.7.184    nn.nn.254.198    ICMP    Echo (ping) request
0.36    10.2.7.184    nn.nn.254.199    ICMP    Echo (ping) request
0.39    10.2.7.184    nn.nn.254.201    ICMP    Echo (ping) request
0.4     10.2.7.184    nn.nn.254.204    ICMP    Echo (ping) request
0.49    10.2.7.184    nn.nn.254.205    ICMP    Echo (ping) request
```

Most other network propagating worms use similar stratagies for locating and infecting vulnerable hosts, and like the three examples given, most of their probes end up in the bit bucket,

Mimail is another interesting worm that sends traffic that ends up in the bit bucket. Mimail is not a network scanning worm. It uses email to propagate, and counts on another non-patchable vulnerability, the unsuspecting user, who must be lured into launching the attachment. As well as mailing itself out to email addresses that it finds on the newly infected host, the worm also launches a denial-of-service attack directly on some web sites[11]. Since all of our internal browsers are configured to use a proxy to access the internet, this worm traffic will end up in the bit bucket of a properly isolated private network.

6. Routing and the Default Route

It's a router's job to know what to do with every packet it receives. Since a router is a "layer 3" device, it makes these decisions based on the IP destination address. Each router sets up a forwarding table or routing table listing all of the networks it knows about, and the interface or next hop it should forward the packet to. If it does not have the destination network listed, it drops the packet into the bit bucket and sends an ICMP unreachable message back to the source address. To populate its routing table the router must discover routes to destination networks. First, and easiest, is locally attached networks (see Drawing 2). Router LANrouter-1 knows that network 10.10.1.0 with a 24 bit network mask is locally attached, because it is told that when it is configured:

```
interface ethernet1
ip address 10.10.1.1 255.255.255.0
```

It likewise knows about each of the other attached networks, 10.10.2.0, etc. But to find

10 Unknown Author - Cisco Security Notice: Nachi Worm Mitigation Recommendations

11 Atli Gudmundsson and Scott Gettis - W32.Mimail.A@mm

out about the way to get to other networks, it must learn the routes from other routers.

Routers exchange routing information with each other using routing protocols such as RIP, EIGRP, OSPF, or BGP[12]. (For an excellent explanation of how routing tables and routing works, see reference 12)

Since we are using EIGRP for our site routing protocol, we will turn on the protocol in the router configuration, and tell it to advertise its local networks:

```
router eigrp 10
network 10.10.0.0
```

All of the site routers (including **Site1WAN-1**) participate in the same EIGRP Autonomous System (AS) 10, so each will know the route to every internal subnet on site (Autonomous System is a Cisco EIGRP term for a group of routers that exchange routing information with each other without going through a border router to another autonomous System or routing protocol). For all other subnets we will use the default route, advertised from **Site1WAN-1**. The default route is the route to "every address", and is used only if the router does not know, or cannot learn a more specific route (one with a longer subnet mask).

Advantages of a default route:

- Whenever a router discards a packet because it has no route to forward it, the router sends an ICMP unreachable (network unreachable) message back to the originating host. If the originating application is a worm, it ignores the reply, but this action by the router has the effect of increasing the network traffic caused by the worm as well as increasing the CPU load on the router. With a default route

12 Unknown Author – Cisco Tech Notes - Route Selection in Cisco Routers

specified within the network, there is always a valid route to any address, so an ICMP unreachable message never need be sent back to the worm.

- If traffic to the default route is all forwarded out a router interface that is dedicated to this route, the interface may be monitored very easily using the methods outlined below, and virus activity will be clearly and quickly seen. IP accounting will show infected host addresses, and a sniffer connected to this interface has a very clear view of traffic that "shouldn't be there".

To set up a default route, the configuration line is ***"ip route 0.0.0.0 0.0.0.0 [next hop address | forwarding interface] [administrative distance]"***[12, 13] with the first 0.0.0.0 being the base address and the second 0.0.0.0 being the mask. This means that the specified route (next hop address, or forwarding interface) is a valid route to the entire internet address range (network zero with a mask of zero, so all addresses are host addresses). Any learned routes will be "more specific" (have a longer network mask) so they will take precedence over the default or "all other" route. This will be configured on **Site1WAN-1** (note: for the example network, we are using EIGRP as our WAN routing protocol, and AS 254 is our WAN AS):

```
ip classless
ip route 0.0.0.0 0.0.0.0 10.11.1.1

router eigrp 10
redistribute static
redistribute eigrp 254
network 10.0.0.0 0.0.255.255
.
.
```

Note that the default route will be sent to all local site routers from **Site1WAN-1**, but will not normally be propagated out onto the WAN, because this could cause overloading of the WAN links by worm traffic from other sites. Each site should have its own "bit bucket" setup.

We will also want to increase our IP accounting threshold on **Site1WAN-1**, since worms usually send one or at most two packets to each non-responsive address. This statement will allow us to record up to 20,000 source/destination pairs:

```
ip accounting-threshold 20000
```

Another source of traffic that is addressed directly to external addresses is rude applications. Every time they are launched, they try to check for updates by "calling home" directly to a hard-coded address. To keep this rude but probably not dangerous traffic out of our "bit bucket' monitor, we will add an input access list (let's call it "rude") to **Site1WAN-1**'s input interface(s) from the site to deny it. We will learn specific addresses and networks to exclude by monitoring the bit bucket segment and after

13 Unknown Author – Cisco Tech Notes - Configuring a Gateway of Last Resort Using IP Commands

determining that they are relatively benign (by doing a reverse DNS lookup to a known vendor's site, for example), we will add it to the list of deny statements in access list "rude". With some experience, we can eliminate most of the routine background noise from the "bit bucket" segment.

7. Setting up a "Bit Bucket" Router.

Refer to Drawing 3. The interface from the WAN router carrying the default route traffic is connected to the "bit bucket" router, whose sole purpose is to throw away packets bound for addresses that we don't have good valid production routes to. It can be a low-end router, and in fact doesn't even have to be a router as long as it answers an ARP request for its address, and can handle the traffic on its interface that is not addressed to it. For a Cisco router, it would be configured:

```
interface ethernet0
ip address 10.11.1.1 255.255.255.0
no shutdown
exit
ip route 0.0.0.0 0.0.0.0 null0
```

The last line tells the Bit Bucket Router to send all packets to its internal bit bucket, null0, except traffic bound for its attached subnet.

8. Monitoring Methods

Now that we have our imaginary network configured, how do we monitor these "bit buckets"? Fortunately there are numerous tools and techniques available to automate this process, from behemoth enterprise "Security Event Managers" to home-grown custom scripts and programs that will watch over our discards and give us an alert.

First a word about alerts. If we have someone dedicated to watching our network, then an audible alarm and a message popping up or icon changing color on the console would probably be preferred. For those of us that must run a network without such a resource, email messages or paging would be a more appropriate choice. Most of the

tools mentioned below have the capability to provide any or all of these alerts.

a. Custom Programs and Scripts

Custom scripts or programs are by far the most versatile monitoring tool, since we can write them to watch whatever we want, alert us however we want, and they work just the way we want (right?). Writing a Perl script to send an email when syslog entrys that contain the string "***spoof***" exceeds so many per minute is relatively easy, and would alert us to a spoofing worm. Even writing a script to log into all routers and count the ACL hits (parse the ***show access-list*** command output), or the rate of packets on the "bit bucket" interface is possible (parse the ***show interface*** *[bit bucket interface]* command output).

The big disadvantage with custom scripts, is that we have to write, debug, and maintain them ourselves, which takes time that is increasingly hard to come by.

b. Syslog Monitoring Tools

Entering "***syslog monitoring tool***" and "***syslog monitoring tools***" into my favorite search engine yielded a long list of free and not-so-free software tools to watch and sort syslogs, with various pattern matching and alerting capabilities. LogScanner, Logcheck, Swatch, and Logsurfer were the most mentioned. Virtually every flavor of Unix and Windows OS are supported. Though far to many to go into, I will mention two that are highly rated by Network World Fusion online magazine:

> **Kiwi Syslog Daemon** . . . is a terrific syslog monitoring tool, perhaps the best we've seen so far[14]!
>
> **Kiwi Syslog Daemon** is a freeware Windows Syslog Daemon. It receives, logs, displays and forwards Syslog messages from hosts such as routers, switches, Unix hosts and any other syslog enabled device. There are many customisable options available[15].
>
> **ArcSight 2.5** wins our Blue Ribbon Award based on its ease of use, flexibility and administration interface[16].
>
> **ArcSight** offers a suite of software solutions designed to bridge the enterprise security gap and deliver the protection and productivity that large organizations require from their security investments. The ArcSight architecture is comprised of a data collection and storage system to consolidate network-wide alarms and alerts and a display and report function to manage the results[17].

c. SNMP Managers

---

14 Mark Gibbs- Network World

15 Kiwi Enterprises – Kiwi Syslog Daemon

16 Mandy Andress - Network World Global Test Alliance

17 ArcSight Product Information - http://www.arcsight.com/product.htm

Simple Network Management Protocol provides an easy way to monitor network equipment and traffic flow. It is an ideal vehicle to watch the volume of traffic flowing into our "bit bucket" segment.

Probably the best way is to monitor this traffic is by using an SNMP manager to watch the output rate, either in octets or preferably packets per monitoring interval on the bit bucket interface of the **Site1WAN-1** router. Most SNMP managers can be configured to graph this rate for a good visual indicator, and a threshold can be set for alerting via icon, email or paging. If we don't have an SNMP manager already available, an excellent SNMP graphing tool (MRTG) is available with a free (gnu) license from http://people.ee.ethz.ch/~oetiker/webtools/mrtg/index-2.html[18].

The Multi Router Traffic Grapher is widely used, with a lot of on-line discussion on configuration and customization tips.

9. Tracking Down the "Bogie"

Allright, now that we are watching our spoofing and default route bit buckets, what do we do when we get an alert?

For our spoofing alert, syslog will tell us which router is sending the alerts, and ***show arp*** will tell us which interface the bad guy is on, and its MAC address. If we have a switched network, the switch attached to that router interface will tell us which port that MAC address is on.

A host infected with malware that is attempting a scan or denial-of-service attack against external addresses will show up clearly on a sniffer or Network Intrusion Device on our "bit bucket" segment. If these are not available, probably the first thing to do would be to set up IP accounting on the bit bucket interface of the **Site1WAN-1** router, if this was not already done:

```
interface ethernet 2
ip accounting output-packets
```

If it was already set up, then we would just clear it:

```
clear ip accounting
```

We would then let it run or a minute or two, and examine the output:

```
show ip accounting
```

We will typically see a single host (hopefully) sending one or at most two packets to many destinations (scanning), or sending many packets to one or two hosts (DOS). This should be obvious to the eye, but if we have multiple infected hosts, or we want to take a closer look, we would save the output as a .TXT file and open the IP accounting data with a spreadsheet. This will provide all the tools we need to sort and compare. Once identified by source address, we just log in to the router with that subnet on it, get the infected host's MAC address, etc, and then take whatever action is dictated by our Security Policy, or Incident Commander, from sending a polite note, to bringing back the

18 Tobias Oetiker, Dave Rand, et al - Multi Router Traffic Grapher

death penalty. Having a written Security Policy, and following it is of utmost importance here. The most logical immediate action is to shut off the switch port connected to the infected host. Shutting off a laptop user is one thing, but disconnecting a critical application server in the middle of a "beat-the-clock" production run, without the authority of a good Security Policy could be "update the resume" time!

10. Conclusion

Configuring our local subnet interfaces for anti-spoofing on every local LAN interface takes a bit of work up front, but provides an instrumented bit bucket with protection against potentially devastating denial-of-service attacks from an internal cross-subnet spoofing host, and the assurance that traffic from any scanning worms that we detect on our network is really coming from the subnet we think it is.

Setting up a "bit bucket" segment not only reduces the impact of worm traffic by providing a viable default route, it provides a convenient location for monitoring a clean flow of "bad" packets. Most network worms will show up here, either first, or in short order. MiMail is a mass mailing worm that also launches DOS attacks against some external addresses. A MiMail infected host showed up clearly in our bit bucket monitor before we had the mail gateway or NIDS signatures for it, and saved us a significant amount of work, and risk.

Firewalls are the bit bucket everyone thinks of from a security standpoing, but there are many others that can be monitored, such as server login and database access failures. The point is, don't just throw it away; Watch Your Bit Bucket.

11. References

1. The Jargon File (The Jargon Dictionary), version 4.2.2, 20 August 2000 - http://info.astrian.net/jargon/terms/b/bit_bucket.html

2. Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear - RFC 1918, Address Allocation for Private Internets, February 1996 - Internet RFC/STD/FYI/BCP Archives - http://www.ietf.org/rfc/rfc1918.txt

3. Tom Dunnigan - Backtracking Spoofed Packets - June 10, 2001 - http://www.csm.ornl.gov/~dunigan/oci/bktrk.html

4. P. Ferguson and D. Senie - RFC 2827, Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing, May 2000 - Internet RFC/STD/FYI/BCP Archives - http://www.faqs.org/rfcs/rfc2827.html

5. Unknown Author - Configuring Unicast Reverse Path Forwarding, August 21, 2003 - Cisco Tech Note - http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fsecur_c/fothersf/scfrpf.htm

6. Joanne Pilker - MS SQL Slammer/Sapphire Worm, August, 2003 - GSEC - SANS Security Essentials Certified Graduates - http://www.giac.org/practical/GSEC/Joanne_Pilker_GSEC.pdf

7. Frederic Perriot - Detecting network traffic that may be due to RPC worms, September 12, 2003 - Symantec Security Response - http://securityresponse.symantec.com/avcenter/venc/data/detecting.traffic.due.to.rpc.worms.html

8. Kaoru Hayashi and Serghei Sevcenco, W32.HLLW.Nebiwo, July 18, 2003 - Symantec Security Response - http://securityresponse.symantec.com/avcenter/venc/data/w32.hllw.nebiwo.html

9. Unknown Author - W32/Sdbot.worm, October 6, 2003 - Network Associates - http://vil.nai.com/vil/content/v_100454.htm

10. Unknown Author - Cisco Security Notice: Nachi Worm Mitigation Recommendations, October 14, 2003 - http://www.cisco.com/warp/public/707/cisco-sn-20030820-nachi.shtml

11. Atli Gudmundsson and Scott Gettis - W32.Mimail.A@mm, Symantec Security Response Center, October 24, 2003 - http://securityresponse.symantec.com/avcenter/venc/data/w32.mimail.a@mm.html

12. Unknown Author - Route Selection in Cisco Routers, Jun 06, 2003 – Cisco Tech Note Document ID: 8651 -

http://www.cisco.com/en/US/tech/tk365/tk207/technologies_tech_note09186a0080094823.shtml

13. Unknown Author - Configuring a Gateway of Last Resort Using IP Commands, Jun 06, 2003 - Cisco Tech Note Document ID: 16448 - http://www.cisco.com/en/US/tech/tk365/tk554/technologies_tech_note09186a0080094374.shtml

14. Mark Gibbs- Network World, 06/24/02 - http://www.nwfusion.com/columnists/2002/0624gearhead.html

15. Kiwi Enterprises – Kiwi Syslog Daemon - http://www.kiwisyslog.com/products.htm

16. Tobias Oetiker, Dave Rand, et al - Multi Router Traffic Grapher - http://people.ee.ethz.ch/~oetiker/webtools/mrtg/index-2.html