

# Building Bastion Routers Using Cisco IOS

Brett Beldridge

## Abstract

Members of the firewall and network security community are generally clueful when it comes to the topic of bastion hosts, and the various approaches and issues involved in constructing them on different platforms. However, less attention has been paid to the subject of securing routers that are exposed to attack--or building bastion routers.

Routers, and in particular Cisco routers, are often deployed in various parts of a firewall system, for example as border and choke packet filters. As such, they can be high-value targets for attackers. This paper provides a simple methodology and specific examples for securing Cisco routers running IOS. Our focus and examples are based upon the IOS versions we are most familiar with: 11.2 and 11.3. However, the principles we present may also apply to older and newer IOS versions (e.g., 12.0, other 11.X versions and 10.X), and possibly to other vendors' gear.

## What is a Bastion Router Anyway?

Routers previously did just that: route IP. However, modern routers have features that permit them to be used as static packet screens, security (VPN) gateways, and other key components in security systems. There is even an IOS variant called the Firewall Feature Set (this is different than the PIX firewall), which we don't cover here because we haven't used it, that supports stateful packet filtering, intrusion detection features and other stuff. We use the term bastion [0] router to refer to a router that requires some level of special configuration to secure it against attack.

We generally focus on two areas: protecting the router itself and protecting hosts behind the router (or possibly on other sides).

## Basic Methodology

Our methodology is relatively simple. We want to disable features and services that are on by default and that we are not using. In other words: if we're not using something, we turn it off. We enable features that may aid in protecting the router or the networks behind the router. If we need a feature we try to protect it as best we can using the protection mechanisms that IOS provides, for example VTY filters. We use ACLs on each interface that permit the specific traffic that we have decided to permit and deny everything else (the "default deny" stance).

IOS supports many, many features; and there are many different releases and a number of feature sets available. Our examples assume IOS version 11.2 and 11.3, with the IP Only feature set, though we will point out exceptions (e.g., TCP Intercept and the Enterprise feature set) as they come up. Also, we can't possibly cover all the various ways to configure something-- our goal is to present some of the things we've learned and some of the methods by which we configure bastion routers.

So the basic methodology we will follow is:

1. Password protection
2. Limit remote access
3. Limit local access
4. Display login banner
5. Configure SNMP
6. Configure logging and NTP
7. Other protection mechanisms
8. Anti-spoofing
9. Mitigate Denial of Service attacks
10. Protect hosts behind the router
11. Verify the configuration

# Building Bastion Routers Using Cisco IOS

Brett Belldridge

For purposes of the examples, we will use a sample network with the following topology. We will also assume that 192.168/16 is routable.

```
Eth0 192.168.0.0/16          Eth1 172.16.1.0/30
      .1 +-----+ .1 .2
private net -----| Router |----- ISP
                  +-----+
access-list e0-in -->          <-- access-list e1-in
```

The final complete configuration will be given at the end of the paper in Appendix A.

## Background

### Brief Introduction to the IOS Command Line Interface

Cisco's IOS (Internetworking Operating System) thankfully supports a Command Line Interface which Cisco calls CLI. The command line interface can be accessed via the console port, a modem, or a TELNET connection. A command line session is referred to as an EXEC session, and it's similar to a Unix shell process. There are two different kinds of traditional EXEC sessions: user EXEC level and privileged EXEC level. User EXEC level can be considered similar to a non-root login account on a Unix system, and privileged EXEC level somewhat like the super-user account, or a UID 0 process. The prompt even changes to end in a pound sign when you switch to privileged EXEC level:

```
reeb>enable
Password:
reeb#
reeb#disable
reeb>
```

You can also customize privilege levels. We'll cover this a bit more later on.

Context sensitive help is also available. Typing a question mark will provide a list of available commands and options that may be entered at that point. For example,

```
reeb#debug ip r?
rip routing rsvp rtp

reeb#debug ip rip ?
events RIP protocol events

reeb#debug ip rip
```

CLI also supports a mini Emacs-like editing mode and command history by default. So you have C-n for next line, C-p for previous line, C-a for beginning of line, C-e for end of line, C-u to erase the line, C-w for erase previous word, and also TAB to finish a partial command. The arrow keys should also work.

## Configuration Settings

One of the things that can be very confusing with IOS is how configuration settings are presented to the user. A default setting is not displayed when you view the router configuration. And the default setting can change across different IOS versions. For example, in IOS 11.2, the services `udp-small-servers` and `tcp-small-servers` are enabled by default. So when you disable UDP and TCP small servers you will see the following in the configuration:

# Building Bastion Routers Using Cisco IOS

Brett Beldridge

```
version 11.2
no service udp-small-servers
no service tcp-small-servers
```

And by default you would see no configuration setting. However, the defaults changed in 11.3 to "no service" for both. So when no configuration setting is displayed, UDP and TCP small servers are disabled. You will see the following when they are enabled:

```
version 11.3
service udp-small-servers
service tcp-small-servers
```

You need to keep this in mind when building bastion Cisco's, and it may take some investigation and detective work to determine which services and features are enabled.

## Step 1 : Password protection

One of the first things we can do is configure and protect the passwords. These include routing protocol and NTP authentication secrets, login, and enable (privileged EXEC mode) passwords.

## Passwords And Privileges

There are many options available for user authentication; for example, overriding access classes and TACACS support that we won't go into here. However, there are some important things we wanted to mention regarding passwords and privilege support. First, different types of passwords have different construction and length requirements. For example, an OSPF simple password can be any continuous string of characters that can be entered from the keyboard up to 8 bytes in length, while an OSPF MD5 key can be any alphanumeric password up to 16 bytes long. A line password can be up to 80 characters in length and can contain any alphanumeric characters including spaces. An enable secret and username password can be up to 25 characters and can contain embedded spaces. In some cases the construction requirements are not clearly documented so you'll have to experiment to come up with a "good" password depending on your environment.

Earlier we mentioned "traditional" user EXEC and privileged EXEC. There are actually 16 privilege levels, numbered 0 through 15. Level 1 is the normal user EXEC mode and 15 is the default privileged EXEC mode. To expand on the sample earlier:

```
reeb>show privilege
Current privilege level is 1
reeb>enable
Password:
reeb#show privilege
Current privilege level is 15
reeb#disable
reeb>show privilege
Current privilege level is 1
```

You can use the privilege mechanism to tailor the authentication configuration to your specific environment.

For sample purposes, we will use separate, unique, personal login names for each of the administrators granted access to the router for audit trail purposes. We will start with two users:

# Building Bastion Routers Using Cisco IOS

Brett Beldridge

```
username variablek password st0rk
username brett password r0ddag
```

## Service Password-Encryption

By default, anyone with privileged access can view all of the passwords on the router. If somebody is watching you configure the router, they can "shoulder surf" and capture passwords.

You can use the "service password-encryption" command to encode or scramble most of the various router password strings. These scrambled passwords are also known as type 7 passwords because of the digit that precedes the encoded password string. Note that while technically the passwords are encrypted, this service provides minimal protection and only serves to hide the passwords from casual observation. The scrambled passwords can be decoded trivially by a simple shell script [1] or on a bar napkin while munching on a plate of nachos or (in our case) drinking a Guinness.

Note that for some reason the password-encryption service does not encode SNMP community names.

Granted this adds little in terms of password security, but we guess it doesn't hurt. We mainly point it out because its name has led to confusion regarding its purpose and strength.

## Enable Secret

The IOS equivalent of root access is privileged EXEC mode which is protected by the enable password. There are two methods of protecting the enable password. The first method is to use "enable password" which only uses the trivial Cisco encoding mechanism.

The second method is to use the "enable secret" command which uses MD5, a one-way cryptographic hash function. Passwords protected with MD5 are also known as type 5 passwords. To use the enable secret command you can specify the enable secret then disable the enable password if you have one:

```
reeb(config)#enable secret s3kr3t
reeb(config)#no enable password
reeb(config)#exit

reeb#sh running-config
Building configuration...

enable secret 5 $1$k2gM$4W2tuuTUqxuRd.LQxsh/v.
```

You might ask why not protect all passwords and secrets with MD5? This won't work because MD5 is a one-way hash, and IOS needs to be able to access clear text strings for stuff like the MD5-based MAC secret that NTP can use for authentication, or OSPF simple authentication strings and so on.

## Step 2 : Limit Remote Access

Cisco routers can be remotely managed via a TELNET connection. It is a good idea to limit, or even disable, TELNET access. To limit access you can specify an access class on the VTY lines:

```
access-list 99 permit host mgmt_ip
access-list 99 deny any
!
line vty 0 4
access-class 99 in
```

# Building Bastion Routers Using Cisco IOS

Brett Beldridge

```
login local
```

In addition, if you are using access lists with a default deny, you will need to allow connections to tcp/23 from specific source IP addresses on the inside:

```
!  
interface Ethernet0  
  ip access-group e0-in in  
!  
ip access-list extended e0-in  
  permit tcp host mgmt_ip host 192.168.0.1 eq 23
```

If we want to disable the TELNET listener completely (a good idea for exposed routers that are high visibility targets), the following will work:

```
line vty 0 4  
  transport input none
```

An ultra-paranoid configuration might even be something like:

```
access-list 99 deny any  
!  
line vty 0 4  
  access-class 99 in  
  exec-timeout 0 1  
  login local  
  transport input none
```

This configuration may be a bit overboard but it:

- sets a deny any access class on the VTY
- disables the TELNET listener
- sets the EXEC session timeout to 1 second

There have been requests to add SSH support to IOS, apparently from as long as 3 years ago. There was even a rumor that IOS 12.0 would contain SSH support, but it didn't make it in. There is also Kerberos support in IOS, and a way to do Kerberized TELNET to the router, but we haven't used that.

## Step 3 : Limit Local Access

By default, when you connect to the console or AUX port, you are given user EXEC mode access without a password. If the router cannot be physically secured, it is a good idea to set a user EXEC password on these ports. Even if the router is in a secured environment, like a locked machine room, it doesn't hurt.

```
line con 0  
  login local  
  ! logout idle console access after 2 min  
  exec-timeout 2 0  
line aux 0  
  ! Uncomment below to disable logins on the AUX port  
  ! no exec  
  ! Or allow password access
```

# Building Bastion Routers Using Cisco IOS

Brett Beldridge

```
login local
```

This will not stop a determined attacker from gaining access to the router. If an attacker has physical access to the box, they can use well-known password recovery techniques to gain access. [2]

## Step 4 : Display Login Banner

It's a good idea to configure a login banner that warns users against unauthorized access. This may help in the event of legal action against an intruder. We tend to use something like the following:

```
banner motd #
```

```
    This is a private system operated for and by  
    Big Phreaking Bank (BPB).
```

```
    Authorization from BPB management is required to use  
    this system.
```

```
    Use by unauthorized persons is prohibited.
```

```
#
```

Though you should tailor it to meet your local requirements. BPB might also be considered an "inviting" target. For examples and more detailed information on the topic of login banners refer to [3].

## Step 5 : SNMP

Another common method of router management is to use the Simple Network Management Protocol (SNMP). IOS supports SNMPv1 and SNMPv2. SNMPv1 was not designed with authentication and data privacy features. Some implementations of SNMPv2 contain security enhancements. SNMPv3 apparently contains more security enhancements.

We generally leave SNMP disabled on our bastion routers, however if you must enable it, we recommend the following protective steps:

- Use a hard-to-guess community name
- Make the MIB read only
- Permit access only from specific hosts

These precautions can be implemented using the following configuration:

```
! allow SNMP reads from hosts in access-list 10  
snmp-server community h4rd2gu3ss ro 10  
!  
! access list for SNMP reads  
access-list 10 permit host snmp_mgmt_ip  
access-list 10 deny any  
!  
! send traps with community names  
snmp-server trap-authentication  
! send all traps to the management host on the inside interface  
snmp-server trap-source Ethernet0  
snmp-server host snmp_mgmt_ip h4rd2gu3ss  
!  
interface Ethernet0
```

# Building Bastion Routers Using Cisco IOS

Brett Belldridge

```
ip access-group e0-in in
!
ip access-list extended e0-in
! allow access from a specific machine on the inside
permit udp host snmp_mgmt_ip host 192.168.0.1 eq snmp
```

## Step 6 : Logging Data

If your security policy requires that logs be generated for access list drops or other security events, you can use the IOS syslog facility. Since syslog uses UDP, which is not a reliable transport mechanism, it can be good idea to log messages to more than one host, which may reduce the occurrence of lost messages due to packet loss or other weirdness (and it's a simple way to automatically create a backup of your logs). Also, using NTP to synchronize all of the clocks greatly aids forensic log analysis in the event of an attack or break in.

## NTP Configuration

Without synchronized time on the various hosts within your firewall complex and network, event correlation from log message timestamps is nearly impossible. The NTP protocol and the Cisco NTP implementation support cryptographic authentication using MD5 (DES is also supported by the protocol as the authentication hash but MD5 doesn't suffer from US export bogosity). This allows the NTP client to authenticate its time sources, and should prevent attackers from spoofing NTP servers and playing with the system clock. If your budget can handle it, consider a network-based GPS stratum 1 NTP time server that supports MD5 authentication. Below we configure NTP to allow updates only from our internal time servers and authenticate the messages using MD5 for the message authentication code (MAC).

```
! Setup our clock environment
clock timezone PST -8
clock summer-time zone recurring
! Configure NTP
ntp authenticate
ntp authentication-key 1 md5 ntpk3y
ntp trusted-key 1
ntp access-group peer 20
ntp server ntp_server1_ip key 1 prefer
ntp server ntp_server2_ip key 1
!
! Allow selected ntp hosts
access-list 20 permit host ntp_server1_ip
access-list 20 permit host ntp_server2_ip
access-list 20 deny any
```

## Syslog Setup

In this case, we will send syslog messages to two hosts and stamp the messages with the local date and time:

```
! Send syslog messages to the mgmt host and log with localtime
service timestamps log datetime localtime
logging syslog1_ip
logging syslog2_ip
```

By default, the router will send syslog messages with a local7 facility. If you want to store router messages in a separate file, your syslog.conf should include the line:

# Building Bastion Routers Using Cisco IOS

Brett Beldridge

```
# router messages
local7.* /var/adm/router.log
```

The exact syntax and log file location may vary depending upon the syslogd you are using.

You can change the facility using:

```
logging facility facility-type
```

## Step 7 : Other Protection Mechanisms

### no ip source-route

Some attacks use the IP source route option. The attacks rely on the ability of the attacker to specify the path a packet will take. An attacker can send a source routed packet to a victim host behind the router which will then send back packets along the same path. This allows replies to spoofed packets to return to the attacker. Many modern operating systems allow you to drop IP packets with source route options set. However, it is a good idea to drop these packets at the edge using the "no ip source-route" option.

### Limiting ICMP

Several DoS attacks use the ICMP protocol. It is a good idea to limit what types of ICMP messages are allowed. At a minimum, in order to allow for Path MTU discovery (PMTU), you should consider permitting packet-too-big messages. The other types of ICMP messages allowed will depend upon the local security policy.

```
ip access-list extended e1-in
! Allow fragmentation needed messages (type 3 code 4)
  permit icmp any 192.168.0.0 0.0.255.255 packet-too-big
! Allow outbound ping and MS style traceroute (type 0)
  permit icmp any 192.168.0.0 0.0.255.255 echo-reply
! Uncomment to allow ping to the inside net (type 8)
  permit icmp any 192.168.0.0 0.0.255.255 echo
! Uncomment to allow traceroute
  permit icmp any 192.168.0.0 0.0.255.255 ttl-exceeded
```

### Disable Unnecessary Services

Next we can disable unnecessary services. By default, IOS has some services enabled which will allow attackers to gain information and perform Denial of Service attacks (though see above for issues with changing defaults in newer IOS versions and determining what is really enabled).

We will disable these:

```
no service udp-small-servers
no service tcp-small-servers
no service finger
no ip bootp server
! not enabled by default but be paranoid
no ip http server

no cdp run
```

# Building Bastion Routers Using Cisco IOS

Brett Beldridge

Cisco Discovery Protocol (CDP) is a media independent protocol which, by default, runs on all Cisco equipment. The protocol is used for network management and to discover other Cisco devices. The Cisco documentation says:

"CDP allows network management applications to discover Cisco devices that are neighbors of already known devices, in particular, neighbors running lower-layer, transparent protocols."

To turn CDP off on a specific interface, you can use:

```
interface Ethernet1
  no cdp enable
```

To disable CDP on all interfaces, you can use the global command:

```
no cdp run
```

## no ip unreachableables

By default, when an access list drops a packet, the router returns a type 3, code 13 ICMP (administratively prohibited) message. This allows potential attackers to know that the router implements access list filters. Also, most UDP scans rely on the target sending back unreachable messages. To thwart UDP scans we can prevent the router from sending any ICMP type 3 (unreachable) messages by specifying the following on each interface:

```
no ip unreachableables
```

## no ip proxy-arp

By default, IOS enables proxy ARP on all interfaces. Since we don't need the service, we will disable it:

```
interface Ethernet0
  no ip proxy-arp
interface Ethernet1
  no ip proxy-arp
```

## no ip redirects

In cases where we have no need to send redirects, we will disable them:

```
interface Ethernet0
  no ip redirects
interface Ethernet1
  no ip redirects
```

## Step 8 : Anti-Spoofing

The idea behind anti-spoofing is that nobody from the outside network should be sending packets to you with a source address of either your inside network address, or certain well-known and reserved addresses. We will use access lists to drop and log any of these packets. A recent Internet draft is available (draft-manning-dsua-00.txt) which discusses the reserved netblocks that should be blocked at the edge.

```
ip access-list extended e1-in
  ! Anti-spoofing: no packets with a src address = our inside net
  ! Normally, this would not be a RFC 1918 net
```

# Building Bastion Routers Using Cisco IOS

Brett Beldridge

```
deny ip 192.168.0.0 0.0.255.255 any log
!
! Deny first octet zeros, all ones, and loopback network
deny ip 0.0.0.0 0.255.255.255 any log
deny ip host 255.255.255.255 any log
deny ip 127.0.0.0 0.255.255.255 any log
!
! Deny class D (multicast) and class E (reserved for future use)
deny ip 224.0.0.0 15.255.255.255 any log
deny ip 240.0.0.0 7.255.255.255 any log
!
! Deny RFC 1918 addresses
deny ip 10.0.0.0 0.255.255.255 any log
deny ip 172.16.0.0 0.15.255.255 any log
! included above in this example
! deny ip 192.168.0.0 0.0.255.255 any log
!
! Deny test-net
deny ip 192.0.2.0 0.0.0.255 any log
!
! Deny end node autoconfig
deny ip 169.254.0.0 0.0.255.255 any log
```

What you really want is a switch that will drop packets arriving on an interface with a source address that is not routed out that interface. Some IOS releases have the ability to do this by using something called Cisco Express Forwarding (CEF) in conjunction with the "ip verify unicast reverse-path" interface command. This requires strictly symmetric routing patterns and a 7500 Series (any 7000 with IOS 11.3) or a 12000 Gigabit switch router to run CEF.

## Step 9 : Mitigating Denial Of Service Attacks

There have been a rash of new Denial of Service (DoS) attacks over the past few years. We can use access lists and other mechanisms to prevent or at least increase our ability to withstand some common DoS attacks.

### SYN Floods

A SYN flood occurs when an attacker sends a TCP SYN segment with an unreachable spoofed source address to an open port on the target. The victim responds with a SYN,ACK to the unreachable host and the TCP handshake never completes. The victim's connection queue quickly gets filled with half-open connections in the SYN\_RCVD state. At some point, the server TCP will start to drop new SYNs.

SYN floods are discussed in the Cisco publication "Defining Strategies to Protect Against TCP SYN Denial of Service Attacks" [4]. Cisco IOS has a mechanism called TCP Intercept [5] which can be used to help protect against SYN floods. TCP Intercept was introduced in IOS 11.3 and requires a specific feature set; it's in the Enterprise feature set and we hear some service provider feature sets and maybe others.

We have found that TCP Intercept works well in practice (protecting against real SYN floods); however, configuring it can be very confusing and the specifics will vary depending on a number of factors. We recommend reading the Cisco documentation and if you are susceptible to SYN floods you may consider implementing TCP Intercept to mitigate the effects.

# Building Bastion Routers Using Cisco IOS

Brett Beldridge

## Land Attack

The land program sends a packet to the victim with identical source and destination port, and identical IP addresses. This causes many network devices with to panic, including Unix hosts, Windows hosts, routers, etc.

We recommend that you run one of the newer IOS releases which contains fixes for this defect. A Cisco field notice provides details on which IOS versions are vulnerable. [6] If you can't update to a newer IOS, the field notice also contains information on how to configure access lists for protection.

## Stop Malicious Insiders (Ingress Filtering)

If the inside network has untrusted hosts or users, you might want to use Ingress Filtering [7]. By denying packets with spoofed source addresses, Ingress Filtering prevents malicious inside users from launching some Denial of Service attacks.

In our case, this would be achieved by allowing the valid inside addresses out and then denying all others:

```
! Ingress filter: only allow our net outbound
ip access-list extended e0-in
  permit ip 192.168.0.0 0.0.255.255 any
  deny ip any any log
!
! apply to inbound packets on the inside interface
interface Ethernet0
  ip access-group e0-in in
```

## Smurf Attacks

Smurf attacks continue to plague the Internet. If you don't take appropriate steps, you can be either a victim or an amplifier in a Smurf attack. Craig Huegen has written a paper that details Smurf attacks and defenses [8].

To prevent your network from being used as a smurf amplifier, you need to filter packets sent to the broadcast address of your network.

```
interface Ethernet0
  no ip directed-broadcast

interface Ethernet1
  no ip directed-broadcast
```

## Step 10 : Protect Hosts Behind The Router

The router can also provide additional protection to any hosts behind it. This may include bastion hosts running web, FTP, mail, and DNS servers. As an example, we will implement access lists to screen access to an HTTP server host (192.168.0.5). We think it is generally a good idea to filter both inbound and outbound packets (using inbound "in" access lists of each interface--we rarely come across cases where we use outbound "out" access lists).

```
ip access-list extended e1-in
! allow tcp/80 to the web server
  permit tcp any host 192.168.0.5 eq www
!
interface Ethernet1
```

# Building Bastion Routers Using Cisco IOS

Brett Beldridge

```
ip access-group e1-in in

ip access-list extended e0-in
! allow established connections from the web server
permit tcp host 192.168.0.5 eq www any established
!
interface Ethernet0
ip access-group e0-in in
```

Note that this will not protect against command channel attacks directed at the permitted services.

## Step 11 : Verify the configuration

As mentioned earlier, depending upon the IOS version, a "sh running-config" might not display whether TCP and UDP small-servers are enabled. You should, at a minimum, run a port scan against the router to verify the basic configuration. Note that if you have disabled IP unreachable, you will have to temporarily re-enable them to perform a UDP scan.

You can use Fyodor's nmap program [9] to perform the scans.

## TCP Scan

```
[root@fuel src]# nmap -sT 192.168.0.1 -p 1-65535
```

```
Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
```

```
Interesting ports on (192.168.0.1):
```

| Port | State | Protocol | Service |
|------|-------|----------|---------|
| 23   | open  | tcp      | telnet  |

If you do not allow VTY access, there shouldn't be any ports open. In this case, we are allowing TELNET access from the same host that performed the scan.

## UDP Scan

```
[root@fuel config]# nmap -sU 192.168.0.1
```

```
WARNING: -sU is now UDP scan -- for TCP FIN scan use -sF
```

```
Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
```

```
Interesting ports on (192.168.0.1):
```

| Port | State | Protocol | Service  |
|------|-------|----------|----------|
| 123  | open  | udp      | ntp      |
| 161  | open  | udp      | snmp     |
| 387  | open  | udp      | aurp     |
| 611  | open  | udp      | npmp-gui |
| 727  | open  | udp      | unknown  |
| 910  | open  | udp      | unknown  |

Note: We have seen false positives when using nmap for router UDP scans. It can be a good approach to use multiple scanners for these tests. Below we point udp\_scan from SATAN at the router. In this case, it turns out that 611/udp and 727/udp are not really open:

```
[root@fuel bin]# ./udp_scan 192.168.0.1 1-1024
```

```
123:ntp:
```

```
161:snmp:
```

```
387:UNKNOWN:
```

# Building Bastion Routers Using Cisco IOS

Brett Beldridge

910:UNKNOWN:

Also, we have noticed that IOS versions 11.2 and 11.3 have 387/udp and 910/udp open. If someone at Cisco could explain this, we sure would like to hear it. We don't have Appletalk enabled so that doesn't explain the udp/387. We tested IOS 12.0 with the exact same configuration and they are not open.

## References

### General References

Increasing Security on IP Networks is at

<http://www.cisco.com/univercd/cc/td/doc/cisintwk/ics/cs003.htm>

Cisco Internet Security Advisories can be found at

<http://www.cisco.com/warp/public/707/advisory.html>

### Specific References

[0] Marcus J. Ranum, "Thinking About Firewalls V2.0: Beyond Perimeter Security"

[1] Decoding type 7 passwords

[2] Password Recovery Techniques

[3] CIAC bulletin on login banners

<http://ciac.llnl.gov/ciac/bulletins/j-043.shtml>

[4] "Defining Strategies to Protect Against TCP SYN Denial of Service Attacks"

[5] Information on TCP Intercept

[6] Information on land attacks

[7] RFC 2267: Network Ingress Filtering: Defeating Denial of Service Attacks by P. Ferguson and D. Senie

[8] Craig Huegen's paper Cisco has a paper Minimizing the Effects of "Smurfing" Denial of Service (DOS) Attacks

[9] Fyodor's nmap

## Appendix A

The complete router configuration is given below.

```
<+> P55/Bastion-router/cisco-conf.txt !75510e67
! We have replaced the mnemonic names with the following addresses:
!
! ntp_server1_ip: 192.168.1.100
! ntp_server2_ip: 192.168.1.101
! syslog1_ip: 192.168.1.102
! syslog1_ip: 192.168.1.103
! mgmt_ip: 192.168.1.104
! snmp_mgmt_ip: 192.168.1.105
!
version 11.3
```

# Building Bastion Routers Using Cisco IOS

Brett Belldridge

```
service timestamps debug uptime
service timestamps log datetime localtime
!
! protect passwords
service password-encryption
enable secret 5 $1$k2gM$4W2tuuTUqxuRd.LQxsh/v.
!
username variablek password 7 110F0B0012
username brett password 7 15190E1A0D24
ip subnet-zero
!
hostname reeb
!
interface Ethernet0
  description Inside Interface
  ip address 192.168.0.1 255.255.0.0
  ip access-group e0-in in
  no ip directed-broadcast
  no ip unreachable
  no ip proxy-arp
  no ip redirects
!
interface Ethernet1
  description Outside Interface
  ip address 172.16.1.1 255.255.255.252
  ip access-group e1-in in
  no ip directed-broadcast
  no ip unreachable
  no ip proxy-arp
  no ip redirects
!
! turn off unnecessary services
no ip bootp server
! the http server is disabled by default. but be paranoid.
no ip http server
no service tcp-small-servers
no service udp-small-servers
no service finger
no cdp run
!
! disable source routed packets
no ip source-route
!
! setup the clock
clock timezone PST -8
clock summer-time zone recurring
! setup NTP
ntp authenticate
ntp authentication-key 1 md5 151C1F1C0F7932 7
ntp trusted-key 1
ntp access-group peer 20
ntp server 192.168.1.100 key 1 prefer
ntp server 192.168.1.101 key 1
!
! configure logging
service timestamps log datetime localtime
logging buffered 4096 informational
logging console informational
```

# Building Bastion Routers Using Cisco IOS

Brett Belldridge

```
logging 192.168.1.102
logging 192.168.1.103
!
! configure SNMP
! allow SNMP reads from hosts in access-list 10
snmp-server community h4rd2gu3ss ro 10
! send traps with community names
snmp-server trap-authentication
! send all traps to the management host on the inside interface
snmp-server trap-source Ethernet0
snmp-server host 192.168.1.105 h4rd2gu3ss
!
! simple static routing. default to the ISP
ip route 0.0.0.0 0.0.0.0 172.16.1.2
ip route 192.168.0.0 255.255.0.0 192.168.0.2
!
! standard ip access-lists
!
! allowed hosts for SNMP reads
no access-list 10
access-list 10 permit host 192.168.1.105
access-list 10 deny any
!
! ntp hosts
no access-list 20
access-list 20 permit host 192.168.1.100
access-list 20 permit host 192.168.1.101
access-list 20 deny any
!
! hosts allowed to telnet to the router
no access-list 99
access-list 99 permit host 192.168.1.104
access-list 99 deny any
!
! extended ip access-lists
!
no ip access-list extended e1-in
ip access-list extended e1-in
! Anti-spoofing
! Deny packets on outside with src address = our inside nets
! This normally wouldn't be a RFC 1918 network
deny ip 192.168.0.0 0.0.255.255 any log
!
! Deny first octet zeros, all ones, and loopback
deny ip 0.0.0.0 0.255.255.255 any log
deny ip host 255.255.255.255 any log
deny ip 127.0.0.0 0.255.255.255 any log
!
! Deny class D (multicast) and class E (reserved for future use)
deny ip 224.0.0.0 15.255.255.255 any log
deny ip 240.0.0.0 7.255.255.255 any log
!
! Deny RFC 1918 addresses
deny ip 10.0.0.0 0.255.255.255 any log
deny ip 172.16.0.0 0.15.255.255 any log
! included above in this example
! deny ip 192.168.0.0 0.0.255.255 any log
!
```

# Building Bastion Routers Using Cisco IOS

Brett Belldridge

```
! Deny test-net
deny ip 192.0.2.0 0.0.0.255 any log
! Deny end node autoconfig
deny ip 169.254.0.0 0.0.255.255 any log
!
! ICMP allows
! Allow fragmentation needed messages (type 3 code 4)
permit icmp any 192.168.0.0 0.0.255.255 packet-too-big
! Allow outbound ping and MS style traceroute (type 0)
permit icmp any 192.168.0.0 0.0.255.255 echo-reply
! Uncomment to allow ping to the inside net (type 8)
! permit icmp any 192.168.0.0 0.0.255.255 echo
! Uncomment to allow traceroute
! permit icmp any 192.168.0.0 0.0.255.255 ttl-exceeded
!
! permit certain connections
! example: permit connections from the outside to a web server
permit tcp any host 192.168.0.5 eq 80
!
! explicit default deny
deny ip any any log
!
no ip access-list extended e0-in
ip access-list extended e0-in
!
! our policy is only allow replies from the inside web server,
! some ICMP and specific inside hosts to access the router.
!
! permit certain connections
! example: allow responses from the web server
permit tcp host 192.168.0.5 eq www any established
!
! allow connections from ntp, mgmt, etc. to the router
permit udp host 192.168.1.105 host 192.168.0.1 eq snmp
permit udp host 192.168.1.100 host 192.168.0.1 eq ntp
permit udp host 192.168.1.101 host 192.168.0.1 eq ntp
permit tcp host 192.168.1.104 host 192.168.0.1 eq telnet
!
! allow specific ICMP out
permit icmp 192.168.0.0 0.0.255.255 any packet-too-big
permit icmp 192.168.0.0 0.0.255.255 any echo
! Uncomment to allow inbound ping responses
! permit icmp 192.168.0.0 0.0.255.255 any echo-reply
! Uncomment to allow traceroute
! permit icmp 192.168.0.0 0.0.255.255 any ttl-exceeded
!
! Ingress filtering: uncomment to deny connections to router and
! then allow outbound if source address = our net. In this case,
! we don't allow any traffic out and go directly to explicit deny.
! deny ip any host 192.168.0.1 log
! permit ip 192.168.0.0 0.0.255.255 any
!
! explicit deny
deny ip any any log
!
!
line con 0
login local
```

# Building Bastion Routers Using Cisco IOS

Brett Belldridge

```
! logout idle console access after two min
  exec-timeout 2 0
line aux 0
! Uncomment below to disable logins on the AUX port
! no exec
! Or allow password access
  login local
line vty 0 4
! uncomment to disable telnet listener
! transport input none
  access-class 99 in
  login local
end
```