Network Security Using IOS

Brett Eldridge
beldridg@pobox.com
http://pobox.com/~beldridg

Copyright (c) 2001 Brett Eldridge
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.1.
A copy of the license is available at http://www.gnu.org/licenses/fdl.txt

Goals and Outline

- Learn how to:
  - Secure the router itself (bastion router).
  - Use access lists and CBAC to secure hosts behind the router.
- Assumes you know:
  - Basic IOS CLI.
  - Basic TCP/IP.
- Outline
  - Bastion Router Principles
  - SNMPv3
  - Access Lists
  - Questions

Bastion Router Principles

- Securing a router is very similar to securing a Unix host:
  - Turn off unnecessary services
  - Create strong passwords/user accounts
  - Configure NTP
  - Configure syslog
  - Be a good Internet citizen
  - Other miscellaneous steps
  - Configure/use SSH
  - Limit access to local services

Turn off unnecessary services

- By default, IOS has a lot of unnecessary services running and we need to disable them.
- A lot of these services have had security problems in the past.
- Here are some examples:

```
no service udp-small-servers (IOS <= 11.2)
no service tcp-small-servers (IOS <= 11.2)
no service finger
no ip bootp server
no ip http server
no cdp run
no ip proxy-arp (on all interfaces)
```

User accounts/passwords

- IOS has the concept of user accounts and passwords like any Unix system. It also has the concept of privilege levels.
  - level 1 is the lowest level (user EXEC)
  - level 15 is the highest level (privileged EXEC)
- Create an unprivileged user account for yourself in addition to the "standard" enable password. This creates an audit trail for configuration changes.
- Use ’service password encryption’ to ensure that the passwords in the config are at least scrambled and not plain text.
- Use ’enable secret’ instead of ’enable password’ to hash the enable password using MD5.

Enable NTP

- NTP should be used to synchronize your clock. This will aid immensely in the event of a forensic analysis.
- Take the following steps when setting up NTP:
  - Set the timezone

```
clock timezone PST -8’
```

  - Use MD5 where possible for authentication

```
ntp authenticate
ntp authentication key 1 MD5 ntpk3y
ntp trusted-key 1
ntp server ntp_server1_ip key 1
```

  - Implement a standard access list to limit connections

```
access-list 20 permit host ntp_server1_ip
access-list 20 deny any
ntp access-group peer 20
```

Setup syslog

- It is important to log at least drops from access lists and other system events such as configuration changes, etc.

- Use the following to send syslog messages to a centralized log server(s):

```
service timestamps log datetime localtime
logging syslog_host1_ip
logging syslog_host2_ip
logging facility facility_name
```

Be a Good Internet Citizen

- Use Ingress Filtering
  - Ingress filtering (RFC 2267) is simple and helps prevent insiders from launching attacks which rely upon spoofed source addresses.
  - The theory is simple : only allow packets out with a source address that is allocated or used on your inside network.

```
ip access-list extended eth0-in
 permit ip our_net wildcard any
!eg permit ip 192.168.0.0 0.0.0.255 any
!
interface ethernet 0
 ip access-group eth0-in in
```

- Don’t respond to broadcast traffic

```
interface Serial0
 no ip directed-broadcast
```

Miscellaneous Protection Steps

- Don’t send ICMP unreachable messages
  - Helps thwart most UDP scans
  - Don’t let them know we are using ACLs
  - Must be enabled on a per-interface basis

```
no ip unreachables
```

- Drop packets with the source route options set using the global option:

```
no ip source-route
```

SSH

- Newer IOS versions support SSH (protocol ver 1 only).
  - At this point, I see absolutely no reason to use telnet.
- You need to configure an authorization scheme first (router based username/password, tacacs, etc).
  - Public key authentication is not supported.
- The latest IOS versions provide the ability to scp config files from the router to avoid using TFTP.
- Some gripes
  - Cisco recently finally admitted their implementation was susceptible to some well-known protocol 1.5 attacks.
  - They have no plans to support SSH protocol ver 2.
  - Can’t limit connections to only 3DES on 3DES images (crippled clients will connect with DES).

Limit Remote Access to Local Services

- This is usually done through the use of standard access lists.
- For example, if you want to limit access to the SSH server on a terminal line, you can apply a standard access list:

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

SNMPv3 Security Objectives

- Primary objectives of SNMPv3 are to provide:
  - Message integrity (MD5 or SHA)
  - User authentication and authorization

- Secondary objectives of SNMPv3 are to provide limited protection against:
  - Data privacy (DES only)
  - Message replay attacks

- Threats that SNMPv3 doesn’t protect against:
  - Denial of Service
  - Traffic Analysis

[Reference: RFC2274]

SNMP Security Levels

Model	Level	Auth	Enc	What Happens
v1	noAuthNoPriv	Community	No	Uses a community string match for authentication.
v2c	noAuthNoPriv	Community	No	Uses a community string match for authentication.
v3	noAuthNoPriv	Username/ Password	No	Uses a username/password match for authentication.
v3	authNoPriv	MD5/SHA	No	Provides authentication based on HMAC-MD5 or HMAC-SHA algorithms.
v3	authPriv	MD5/SHA	DES	Provides authentication based on HMAC-MD5 or HMAC-SHA. Provides DES 56-bit encryption in addition to authentication based on the CBC-DES (DES-56) standard.

[Reference: Cisco]

Cisco SNMPv3 Information

- SNMPv3 is supported in IOS version 12.0(3)T higher.
- Cisco implementation features and terminology
  - Each user belongs to a group.
  - The group defines the access policy for a set of users. The access policy consists of three components:
    - The View determines the MIB objects that can be accessed.
    - The Model is the version of SNMP (e.g., v3)
    - The Level is the authentication and encryption scheme for that group (e.g., authNoPriv).

Quick Guide to Cisco SNMPv3 Server Config

```
!snmp-server group <name> <model> <level> <read/write> <view> <acl>
snmp-server group test v3 priv read leeloo access 5

snmp-server view leeloo sysUpTime included

access-list 5 permit host 192.168.3.5
access-list 5 deny any log

!snmp-server user <name> <group> <model> <level> auth <type> <secret>
!priv <level> <secret>
snmp-server user beldridg test v3 auth sha vip5er82 priv des56 par56agon
```

Check the Configuration:

```
filter#show snmp group
groupname: test                                   security model:v3 priv
readview :leeloo                                  writeview: <no writeview specified>
notifyview: <no notifyview specified>
row status: active      access-list: 5

filter#show snmp user
User name: beldridg
Engine ID: 00000009020000D058BF7B80
storage-type: nonvolatile        active

filter#show snmp view
*ilmi system - included permanent active
*ilmi atmForumUni - included permanent active
leeloo sysUpTime - included nonvolatile active
v1default internet - included volatile active
v1default internet.6.3.15 - excluded volatile active
v1default internet.6.3.16 - excluded volatile active
```

Test using snmpwalk from UCD-SNMP

- UCD-SNMP has supported SNMPv3 since version 4.0 and can be used to build "secure" management systems.

- This example shows both authentication and encryption and the limited view of the MIB.

```
[beldridg@rush ~]$ snmpwalk cisco -v 3 -u beldridg -l authPriv -a
SHA -A vip5er82 -x DES -X para56gon

system.sysUpTime.0 = Timeticks: (57061) 0:09:30.61
```

Protection Against Common Attacks

- SNMPv3 provides some protection against replay attacks by including a timestamp in the encrypted packet.
  - If packets are received past a certain window they are dropped.
  - If packets are received out of order, they are dropped.
  - This means that you should keep clock synchronized.
- This has not been tested by the author. :)

Recent SNMP Security Advisories

- There are two current security advisories regarding the IOS SNMP implementation.
  - Feb 2001: Multiple SNMP Community String Vulnerabilities
    - Read-Write string exposed during a read only SNMP MIB walk
    - Read-Only community string silently added
  - Feb 2001: Read-Write ILMI Community String
    - Undocumented ILMI community strings allows some minor MIB variables to be written.

Testing SNMP Vulnerabilities

- On a default install of 12.0(6), you can see the information available:

```
[beldridg@rush beldridg]$ snmpwalk cisco ILMI
system.sysDescr.0 = Cisco Internetwork Operating System Software
IOS (tm) C2600 Software (C2600-IS-M), Version 12.0(6), RELEASE
SOFTWARE (fc1)
Copyright (c) 1986-1999 by cisco Systems, Inc.
Compiled Wed 11-Aug-99 00:16 by phanguye
system.sysObjectID.0 = OID: enterprises.9.1.186
system.sysUpTime.0 = Timeticks: (86128) 0:14:21.28
system.sysContact.0 =
system.sysName.0 = filter.home.net
system.sysLocation.0 =
system.sysServices.0 = 78
system.sysORLastChange.0 = Timeticks: (0) 0:00:00.00
End of MIB
```

Testing (Cont’d)

- Now, let’s set a few parameters:

```
[beldridg@rush ~]$ snmpset cisco ILMI system.sysName.0 s "router.hacked.net"
system.sysName.0 = router.hacked.net

[beldridg@rush ~]$ snmpset cisco ILMI system.sysContact.0 s "hax0r"
system.sysContact.0 = hax0r
```

- And test the result:

```
[beldridg@rush ~]$ snmpget cisco ILMI system.sysName.0 system.sysContact.0
system.sysName.0 = router.hacked.net
system.sysContact.0 = hax0r
```

- Use Access Lists and install security patches.

Access Lists

- Access lists on a router provide the basis for constructing security perimeter systems.

- ACLs provide the capability to perform packet filtering based upon fields in the TCP/IP header. This includes source IP, destination port, destination IP address, etc.

Summary of Access List Types

- Standard
  - Allows filtering based solely upon the the source IP.
  - Numbered 1 to 99 or 1300 to 1999.
- Extended
  - Initially available in 1993 with the introduction of 10.0
  - Allows filtering based upon many values in the TCP/IP headers but not all.
  - Numbered 100 to 199 or 2000 to 2699 or using named access lists.
- Dynamic (Lock-and-Key)
  - Introduced in Mar 1996 with the release of 11.1
  - Works in conjunction with extended access lists.
  - Allows user authentication for connections.
  - Users authenticate via telnet to the router (ugh) or TACACS
  - Invoked via ’access-list 101 dynamic ...’

Access List Types (cont’d)

- Reflexive
  - Introduced in Dec 1997 with the release of 11.3
  - Works in conjunction with extended access lists.
  - Allows temporary reply filters to be dynamically created based upon request traffic.
  - Invoked via ’access-list 101 permit ... reflect <list_name>
  - Works for "standard" TCP/UDP protocols (telnet, web, etc) and ICMP but not for complex protocols like FTP (except passive) and H323, etc.
  - Available in all releases

Context-Based Access Control (CBAC)

- CBAC Information
  - Introduced in Dec 1996 with the release of 11.2P.
  - Available with the firewall feature set.
  - Works independent of access lists (both are consulted before traffic is allowed).
  - Packets with an interface IP address as the source or destination are not inspected by CBAC.
  - Doesn’t inspect ICMP but does work for complex protocols that reflexive access lists don’t handle (FTP, RPC, etc).
  - Invoked via the ’ip inspect ...’ command.

Access Lists cont’d

- Access lists can be applied to just about any service or interface (virtual or physical). For example:
  - Restrict access to a vty by creating a standard access list and then apply using the access-class command.
  - Restrict route distribution based upon a standard access list which can help limit what you announce to prevent route leaks.
  - Limit access to the SNMP service by creating a standard access list and then specify the access list number in the snmp-server command.

Simple Example : Allow Outbound SSH

```
                  +-------------+
             Eth0 |             | Ser0
     -------------+             +---------------
    inside        |             |    outside
  10.0.3.0/24     +-------------+
```

```
! allow outbound connection
ip access-list extended eth0-in
 permit tcp 10.0.3.0 0.0.0.255 any eq 22
 deny   ip any any log
!
! allow replies to outbound ssh connections
ip access-list extended ser0-in
 permit tcp any eq 22 10.0.3.0 0.0.0.255 established
 deny   ip any any log
!
interface ethernet 0
 ip access-group eth0-in in
!
interface serial 0
 ip access-group ser0-in in
```

Source Port Filtering

- Implementing security policy by trusting the source port is not a good security practice.
- On the plus side:
  - Can help document odd protocols and explain how/why a specific ACL is built for that protocol.
  - Can increase security when you perform filtering of outbound connections from servers (example later).
- On the negative side:
  - Usually doesn’t increase the security and adds to the complexity of the configuration.
  - Configurations can get unwieldy if you allow out a lot of outbound protocols and create ACL reply entries with source port filtering for each.

The ’established’ keyword

- The ’established’ key word first appeared in the 10.3 release (April 1995).
- Cisco defines ’established’ as either the ACK or the RST flag are set in the TCP header (the connection is not new).
- This is useful for allowing inbound replies for connections that have been established outbound.

Outbound FTP Example

```
                  +-------------+
             Eth0 |             | Ser0
     -------------+             +---------------
    inside        |             |    outside
   10.0.3.0/24    +-------------+

! outbound. first rule is for the data channel, second is for
! passive FTP and third is for active data.
ip access-list extended eth0-in
 permit tcp 10.0.3.0 0.0.0.255 any eq 21
 permit tcp 10.0.3.0 0.0.0.255 gt 1023 any gt 1023
 permit tcp 10.0.3.0 0.0.0.255 gt 1023 any eq 20 established
 deny   ip any any log
!
! inbound. fear and loathing. since server initiates data in
! active, we allow inbound SYN to tcp > 1023. this rule will
! also match data channel and passive.
ip access-list extended ser0-in
 permit tcp any 10.0.3.0 0.0.0.255 gt 1023
 deny   ip any any log
```

Inbound Web Server Example

```
                  +-------------+
             Eth0 |             | Ser0
     -------------+             +----------------
    web farm      |             |    outside
   10.0.3.0/24    +-------------+
```

```
! allow inbound connections to the web server
access-list 102 permit tcp any 10.0.3.0 0.0.0.255 eq 80
access-list 102 permit tcp any 10.0.3.0 0.0.0.255 eq 443
access-list 102 deny any any log
!
interface serial 0
 ip access-group 102 in
!
! be smart and limit outbound connections from the web servers
! prevents infected code red web servers from spreading
! note source port filtering
access-list 101 permit 10.0.3.0 0.0.0.255 eq 80 any established
access-list 101 permit 10.0.3.0 0.0.0.255 eq 443 any established
access-list 101 deny any any log
!
interface ethernet 0
 ip access-group 101 in
```

Sample Reflexive Access Lists

```
                  +-------------+
             Eth0 |             | Ser0
     -------------+             +----------------
    inside        |             |    outside
  10.0.3.0/24     +-------------+

! be stingy. only allow http and https, dns out
ip access-list extended permitout
 permit tcp 10.0.3.0 0.0.0.255 any eq 80 reflect tcpreply
 permit tcp 10.0.3.0 0.0.0.255 any eq 443 reflect tcpreply
 permit udp 10.0.3.0 0.0.0.255 any eq 53 reflect udpreply
 deny   ip any any log
!
interface ethernet 0
 ip access-group permitout in
!
! allow the replies in
! reflexive ACL tables are "nested" inside extended access-lists
ip access-list extended permitreply
 evaluate tcpreply
 evaluate udpreply
 deny   ip any any log
!
interface serial 0
 ip access-group permitreply in
```

Reflexive Access List results

```
filter#sh ip access-list
Extended IP access list permitout
    permit tcp 10.0.3.0 0.0.0.255 any eq www reflect tcpreply
    permit tcp 10.0.3.0 0.0.0.255 any eq 443 reflect tcpreply
    permit udp 10.0.3.0 0.0.0.255 any eq domain reflect udpreply
    deny ip any any
Extended IP access list permitreply
    evaluate tcpreply
    evaluate udpreply
    deny ip any any log (5 matches)
Reflexive IP access list tcpreply
    permit tcp host 192.168.0.10 eq www host 10.0.3.2 eq 1196
    (25 matches) (time left 295)
Reflexive IP access list udpreply
    permit udp host 192.168.0.10 eq domain host 10.0.3.2 eq 1195
    (3 matches) (time left 293)
```

abbreviated netstat output from client side showing matching port numbers:

```
[beldridg@rush beldridg]$ netstat -an --inet |grep 80
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
tcp        0      0 10.0.3.2:1196           192.168.0.10:80
```

Context-Based Access Control Example

```
                  +-------------+
             Eth0 |             | Ser0
     -------------+             +---------------
    inside        |             |    outside
  10.0.3.0/24     +-------------+

! be stingy. only allow http, dns out
ip access-list extended permitout
 permit tcp 10.0.3.0 0.0.0.255 any eq 80
 permit udp 10.0.3.0 0.0.0.255 any eq 53
 deny   ip any any log
!
! inspect outbound tcp packets & apply to interface
ip inspect name cbac-out tcp
ip inspect name cbac-out udp
!
interface ethernet 0
 ip access-group permitout in
 ip inspect cbac-out in
!
! default deny on inbound access list & apply inbound
ip access-list extended ser0-in
 deny   ip any any log
interface serial 0
 ip access-group ser0-in in
```

Results

```
filter#show ip access-lists
Extended IP access list ser0-in
    deny ip any any log
Extended IP access list permitout
    permit tcp 10.0.3.0 0.0.0.255 any eq www
    permit udp 10.0.3.0 0.0.0.255 any eq domain
    deny ip any any

filter#show ip access-lists
Extended IP access list ser0-in
    permit tcp host 192.168.3.2 eq www host 10.0.3.10 eq 1354 (5 matches)
    permit udp host 192.168.3.2 eq domain host 10.0.3.10 eq 1353 (1 match)
    permit tcp host 192.168.3.5 eq www host 10.0.3.10 eq 1355
    deny ip any any log (2 matches)
Extended IP access list permitout
    permit tcp 10.0.3.0 0.0.0.255 any eq www (8 matches)
    permit udp 10.0.3.0 0.0.0.255 any eq domain (3 matches)
    deny ip any any

filter#sh ip inspect sessions
Established Sessions
 Session 822645D4 (10.0.3.10:1353)=>(192.168.3.2:53) udp SIS_OPEN
 Session 8261D4DC (10.0.3.10:1354)=>(192.168.3.2:80) tcp SIS_OPEN
Half-open Sessions
 Session 8226261C (10.0.3.10:1355)=>(192.168.3.5:80) tcp SIS_OPENING
```

Common Access Lists You Should Implement

- Anti-Spoofing
  - Reject packets on the outside interface which have a source address of our inside network.
- Limit Inbound ICMP
  - Some attacks use ICMP so we need to limit what we accept into our internal network (make sure you allow necessary ICMP types like packet-too-big).
- Drop Reserved Addresses
  - We shouldn’t see any packets which have RFC1918 addresses all zeros, etc. on the outside interface.

Topics we didn’t cover

- Authentication, Authorization, and Accounting
- Tacacs+, Radius
- IPsec
- Kerberos
- IOS IDS
- Securing Dynamic Routing Protocols
- PIX
    - It’s not IOS, but it is similar.

Questions

- Open for questions.
- Available after the session as well.