

Errata for Linux Quick Fix Notebook

By Peter Harrison

Sunday, April 09, 2006

This is the unofficial errata for the Linux Quick Fix Notebook based on corrections submitted by me to Prentice Hall. Special thanks to all the visitors to the website who have assisted in this effort.

Table of Contents

| | |
|------------------|---|
| Chapter 3..... | 1 |
| Chapter 10..... | 1 |
| Chapter 14..... | 2 |
| Chapter 15..... | 5 |
| Chapter 18..... | 6 |
| Chapter 21..... | 6 |
| Chapter 23..... | 6 |
| Chapter 26..... | 7 |
| Chapter 27..... | 7 |
| Appendix II..... | 7 |

Chapter 3

1. pp 51. Replace

```
ETHTOOL_OPTS = "speed 100 duplex full autoneg off"
```

with

```
ETHTOOL_OPTS="speed 100 duplex full autoneg off"
```

Chapter 10

1. pp 159. Table 10-2 Replace "DOMAIN" in the first column with "WORKGROUP"
2. pp 161. Replace

```
[root@bigboy tmp]# chmod -r 0755 /home/samba
```

With

```
[root@bigboy tmp]# chmod -R 0755 /home/samba
```

3. At the top of pp 154, add

Before SWAT can be used, the xinetd program which controls it must be activated in advance. You can start/stop/restart xinetd after boot time using the xinetd initialization script as in the examples below:

```
[root@bigboy tmp]# service xinetd start
[root@bigboy tmp]# service xinetd stop
[root@bigboy tmp]# service xinetd restart
```

Just like most Linux systems applications, you can configure xinetd to start at boot time using the chkconfig command:

```
[root@bigboy tmp]# chkconfig xinetd on
```

4. In the middle of pp 153, replace

- The default configuration only allows SWAT web access from the VGA console only as user `root` on port 901 with the Linux `root` password. This means you'll have to enter "http://127.0.0.0:901" in your browser to get the login screen.

With

- The default configuration only allows SWAT web access from the VGA console only as user `root` on port 901 with the Linux `root` password. This means you'll have to enter "http://127.0.0.1:901" in your browser to get the login screen.

Chapter 14

5. pp 225. Replace

```
iptables --policy INPUT -j DROP
iptables --policy OUTPUT -j DROP
iptables --policy FORWARD -j DROP
```

With

```
iptables --policy INPUT DROP
iptables --policy OUTPUT DROP
iptables --policy FORWARD DROP
```

6. pp 228. Replace

```
iptables -A OUTPUT -j ACCEPT -m state --state NEW \
-o eth0 -p tcp -m multiport --dport 80,443 -m multiport \
--sport 1024:65535
```

With

```
iptables -A OUTPUT -j ACCEPT -m state \
--state NEW,ESTABLISHED,RELATED -o eth0 -p tcp \
-m multiport --dport 80,443 -m multiport --sport 1024:65535
```

7. pp 228. Replace

If you want all TCP traffic originating from the firewall to be accepted, then remove the line:

```
-m multiport --dport 80,443 --sport 1024:65535
```

With

If you want all TCP traffic originating from the firewall to be accepted, then remove the line:

```
-m multiport --dport 80,443 -m multiport --sport 1024:65535
```

8. pp 234. Replace

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 \  
-j SNAT -o eth1 --to-source 97.158.253.29
```

With

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 \  
-j SNAT -o eth0 --to-source 97.158.253.29
```

9. pp 214. Updated [Figure 14.1](#)

10. pp 213. Replace:

The packet is first examined by your rules in the mangle table's PREROUTING chain, if any. It is then inspected by the rules in the `nat` table's PREROUTING chain to see whether the packet requires DNAT. It is then routed.

If the packet is destined for a protected network, then it is filtered by the rules in the FORWARD chain of the `filter` table and, if necessary, the packet undergoes SNAT before arriving at Network B. When the destination server decides to reply, the packet undergoes the same sequence of steps.

If the packet is destined for the firewall itself, then it is filtered by the rules in the INPUT chain of the `filter` table before being processed by the intended application on the firewall. At some point, the firewall needs to reply. This reply is inspected by your rules in the OUTPUT chain of the mangle table, if any. The rules in the OUTPUT chain of the `nat` table determine whether address translation is required and the rules in the OUTPUT chain of the `filter` table are then inspected before the packet is routed back to the Internet.

With

The packet is first examined by your rules in the mangle table's PREROUTING chain, if any. It is then inspected by the rules in the `nat` table's PREROUTING chain to see whether the packet requires DNAT. It is then routed.

If the packet is destined for a protected network, then it is filtered by the rules in the FORWARD chain of the `filter` table and, if necessary, the packet undergoes SNAT in the POSTROUTING chain before arriving at Network B. When the destination server decides to reply, the packet undergoes the same sequence of steps. Both the FORWARD and POSTROUTING chains may be configured to implement quality of service (QoS) features in their `mangle` tables, but this is not usually done in SOHO environments.

If the packet is destined for the firewall itself, then it passes through the `mangle` table of the INPUT chain, if configured, before being filtered by the rules in the INPUT chain of the `filter` table before. If it successfully passes these tests then it is processed by the intended application on the firewall.

At some point, the firewall needs to reply. This reply is routed and inspected by the rules in the OUTPUT chain of the `mangle` table, if any. Next, the rules in the OUTPUT chain of the `nat` table determine whether DNAT is required and the rules in the OUTPUT chain of the `filter` table are then inspected to help restrict unauthorized packets. Finally, before the packet is sent back to the Internet, SNAT and QoS mangling is done by the POSTROUTING chain.

Chapter 15

1. pp 247. Replace

```
# Allow anonymous FTP?  
# anonymous_enable=YES
```

With

```
# Allow anonymous FTP?  
anonymous_enable=NO
```

Chapter 18

1. pp 284. Replace

```
zone "1.168.196.in-addr.arpa" {
```

with

```
zone "1.168.192.in-addr.arpa" {
```

2. pp 285. Table 18.3 under "Serial-no" should read "You can use the date format YYYYMMDD with an incremented single digit number tagged to the end. This will allow you to do multiple edits each day with a serial number that both increments and reflects the date on which the change was made".

Chapter 21

1. pp 362. Replace

```
[root@bigboy tmp]# chkconfig pop3 on
```

with

```
[root@bigboy tmp]# chkconfig ipop3 on
```

2. pp 362. Replace

```
[root@bigboy tmp]# chkconfig pop3 off
```

With

```
[root@bigboy tmp]# chkconfig ipop3 off
```

Chapter 23

1. On pp 396. The Memory Monitoring (Percentage usage) section should read like this:

```

#
# Memory Monitoring (Percentage usage)
#
Title[server.mempercent]: Percentage Free Memory
PageTop[server.mempercent]: <H1>Percentage Free Memory</H1>
Target[server.mempercent]: ( memAvailReal.0&memAvailReal.0:craz33guy@localhost
) * 100 / ( memTotalReal.0&memTotalReal.0:craz33guy@localhost )
options[server.mempercent]: growright,gauge,transparent,nopercent
Unscaled[server.mempercent]: ymwd
MaxBytes[server.mempercent]: 100
YLegend[server.mempercent]: Memory %
ShortLegend[server.mempercent]: Percent
LegendI[server.mempercent]: Free
LegendO[server.mempercent]: Free
Legend1[server.mempercent]: Percentage Free Memory
Legend2[server.mempercent]: Percentage Free Memory

```

Chapter 26

1. Replace " The data is distributed across the drives in 32 MB chunks:" with " The data is distributed across the drives in 32 KB chunks:"

Chapter 27

1. pp 441. Replace

```
sh-2.05b# cp -r /var/transactions-save /mnt/hdb1/transactions
```

With

```
sh-2.05b# cp -a /var/transactions-save/* /mnt/hdb1
```

Appendix II

1. pp 605. Replace

```

#-----
# Allow port 80 (www) and 443 (https) connections from the firewall
#-----
iptables -A OUTPUT -j ACCEPT -m state --state NEW \
-o $EXTERNAL_INT -p tcp --dport 80 --sport 1024:65535

iptables -A OUTPUT -j ACCEPT -m state --state NEW \
-o $EXTERNAL_INT -p tcp --dport 443 --sport 1024:65535

```

With

```
#-----  
# Allow port 80 (www) and 443 (https) connections from the firewall  
#-----  
  
iptables -A OUTPUT -j ACCEPT -m state --state NEW,ESTABLISHED,RELATED \  
-o $EXTERNAL_INT -p tcp --dport 80 --sport 1024:65535  
  
iptables -A OUTPUT -j ACCEPT -m state --state NEW,ESTABLISHED,RELATED \  
-o $EXTERNAL_INT -p tcp --dport 443 --sport 1024:65535
```

2. Add the Subnet Calculator Script

```
#!/usr/bin/perl  
#  
# subnet-calc.sh - Calculates subnets given an IP address and subnet mask  
# The subnet mask must begin with "255.255.255" or in  
# the range /24 to /30  
#  
# Usage:  
#  
# subnet-calc.sh IP-address subnet-mask  
#  
# (c) SiliconValleyCCIE.com  
#  
#  
# Get the IP address, the subnet mask and broadcast address of  
# the external interface  
#  
  
    &main;  
  
sub main {  
  
    my $ip                  = $ARGV[0];  
    my $netmask             = $ARGV[1];  
    my $subnet_size        = 0;  
    my $subnet_base        = 0;  
  
    #  
    # Convert "/" notation to dotted decimal  
    #  
  
    if ( $netmask eq "/30" || $netmask eq "255.255.255.252" ) {  
        $long_netmask = "255.255.255.252";  
        $short_netmask = "/30";  
    }  
    elsif ( $netmask eq "/29" || $netmask eq "255.255.255.248" ) {  
        $long_netmask = "255.255.255.248";  
        $short_netmask = "/29";  
    }  
    elsif ( $netmask eq "/28" || $netmask eq "255.255.255.240" ) {  
        $long_netmask = "255.255.255.240";  
        $short_netmask = "/28";  
    }  
    elsif ( $netmask eq "/27" || $netmask eq "255.255.255.224" ) {
```

```

    $long_netmask = "255.255.255.224";
    $short_netmask = "/27";
}
elseif ( $netmask eq "/26" || $netmask eq "255.255.255.192" ) {
    $long_netmask = "255.255.255.192";
    $short_netmask = "/26";
}
elseif ( $netmask eq "/25" || $netmask eq "255.255.255.128" ) {
    $long_netmask = "255.255.255.128";
    $short_netmask = "/25";
}
elseif ( $netmask eq "/24" || $netmask eq "255.255.255.0" ) {
    $long_netmask = "255.255.255.0";
    $short_netmask = "/24";
}
}
else{
    &usage;
}

#
# Get the last octet of the subnet mask
#

my @netmask_octet = split (/\.\/,$long_netmask);

#
# Get first three octets of the IP address
#

my @ip_octet = split(/\.\/,$ip);

#
# Check for IP address errors
#

if( ($ip_octet[0] >= 255) || ($ip_octet[0] <= 0) ) {&usage};
if( ($ip_octet[1] >= 255) || ($ip_octet[1] <= 0) ) {&usage};
if( ($ip_octet[2] >= 255) || ($ip_octet[2] <= 0) ) {&usage};

#
# Get the size of the subnet
#

$subnet_size = (256 - $netmask_octet[3]);

#
# Get the last octet of the network address
#
if ($netmask_octet[3] eq "0" ) {
    $subnet_base = "0";
}
else {
    $subnet_base = $ip_octet[3] - ($ip_octet[3] % $subnet_size);
}

$network_address = $ip_octet[0].".".$ip_octet[1].".".$ip_octet[2].".".$subnet_base;

```

```

    $broadcast_address = $ip_octet[0].".$ip_octet[1].".$ip_octet[2].".$subnet_base -1 +
    $subnet_size);

    print ("\n");
    print ( "IP Address \t\t: $ip\n");
    print ( "Network Base Address \t: $network_address\n");
    print ( "Broadcast Address \t: $broadcast_address\n\n");
    print ( "Subnet Mask \t\t: $long_netmask or $short_netmask\n");
    print ( "Subnet Size \t\t: $subnet_size IP Addresses\n\n");
}

sub usage {

print STDERR << "EOF";

    This program calculates full subnet information for any IP address with a class C network mask or
less
    Subnet masks and IP addresses must be valid

    usage:  $0 <ip_address> <subnet_mask>

    example: $0 216.10.119.241 /28
    example: $0 216.10.119.241 255.255.255.240

EOF
    exit;
}

```