Subject: Iptables port opening error CentOS
Posted by whatever on Tue, 03 Oct 2006 05:53:24 GMT

Hello,
I am using CentOS as my VPS OS. After installation of CentOS the IPtables are on by default.
how can we just allow port 22 to open in iptables
I modify the iptables rules and when I restarted the VPS it get error

Perhaps iptables or your kernel needs to be upgraded.
Warning: wierd character in interface `venet0:0' (No aliases, :, ! or *).
iptables v1.2.11: can't initialize iptables table `filter': Table does not exist (do you need to insmod?)
Perhaps iptables or your kernel needs to be upgraded.
iptables v1.2.11: can't initialize iptables table `filter': Table does not exist (do you need to insmod?)
 -- -
I tried apf firewall too but there also I get error

iptables v1.2.11: can't initialize iptables table `filter': Table does not exist (do you need to insmod?)
Perhaps iptables or your kernel needs to be upgraded.
iptables v1.2.11: can't initialize iptables table `filter': Table does not exist (do you need to insmod?)
Perhaps iptables or your kernel needs to be upgraded.
iptables v1.2.11: can't initialize iptables table `filter': Table does not exist (do you need to insmod?)
Perhaps iptables or your kernel needs to be upgraded.

Please suggest.
Thanks

Subject: Re: Iptables port opening error CentOS
Posted by dim on Tue, 03 Oct 2006 07:30:39 GMT

Please, provide:
1) iptables rules, which you are trying to insert to kernel
2) /etc/vz/vz.conf
3) /etc/vz/conf/<VPS_ID>.conf
4) lsmod before VPS start

Subject: Re: Iptables port opening error CentOS
Posted by whatever on Tue, 03 Oct 2006 07:54:49 GMT

hello,
Below are the out of commands you asked for. I can't get how to 4) lsmod before VPS start
Thanks.

Iptables

```
#!/bin/sh
#
 #############################################################
###################
#
# Local Settings
#

# sysctl location.  If set, it will use sysctl to adjust the kernel parameters.
# If this is set to the empty string (or is unset), the use of sysctl
# is disabled.

SYSCTL="/sbin/sysctl -w"

# To echo the value directly to the /proc file instead
# SYSCTL=""

# IPTables Location - adjust if needed

IPT="/sbin/iptables"
IPTS="/sbin/iptables-save"
IPTR="/sbin/iptables-restore"

# Internet Interface
INET_IFACE="venet0"
INET_ADDRESS="x.x.x.x"

# Localhost Interface

LO_IFACE="lo"
LO_IP="127.0.0.1"

# Save and Restore arguments handled here
if [ "$1" = "save" ]
then
 echo -n "Saving firewall to /etc/sysconfig/iptables ... "
 $IPTS > /etc/sysconfig/iptables
 echo "done"
 exit 0
```

```
elif [ "$1" = "restore" ]
then
 echo -n "Restoring firewall from /etc/sysconfig/iptables ... "
 $IPTR < /etc/sysconfig/iptables
 echo "done"
 exit 0
fi


 ##############################################################
###################
#
# Load Modules
#

echo "Loading kernel modules ..."

# You should uncomment the line below and run it the first time just to
# ensure all kernel module dependencies are OK.  There is no need to run
# every time, however.

# /sbin/depmod -a

# Unless you have kernel module auto-loading disabled, you should not
# need to manually load each of these modules.  Other than ip_tables,
# ip_conntrack, and some of the optional modules, I've left these
# commented by default.  Uncomment if you have any problems or if
# you have disabled module autoload.  Note that some modules must
# be loaded by another kernel module.

# core netfilter module
/sbin/modprobe ip_tables

# the stateful connection tracking module
/sbin/modprobe ip_conntrack

# filter table module
# /sbin/modprobe iptable_filter

# mangle table module
# /sbin/modprobe iptable_mangle

# nat table module
# /sbin/modprobe iptable_nat

# LOG target module
# /sbin/modprobe ipt_LOG

# This is used to limit the number of packets per sec/min/hr
```

```
# /sbin/modprobe ipt_limit

# masquerade target module
# /sbin/modprobe ipt_MASQUERADE

# filter using owner as part of the match
# /sbin/modprobe ipt_owner

# REJECT target drops the packet and returns an ICMP response.
# The response is configurable.  By default, connection refused.
# /sbin/modprobe ipt_REJECT

# This target allows packets to be marked in the mangle table
# /sbin/modprobe ipt_mark

# This target affects the TCP MSS
# /sbin/modprobe ipt_tcpmss

# This match allows multiple ports instead of a single port or range
# /sbin/modprobe multiport

# This match checks against the TCP flags
# /sbin/modprobe ipt_state

# This match catches packets with invalid flags
# /sbin/modprobe ipt_unclean

# The ftp nat module is required for non-PASV ftp support
/sbin/modprobe ip_nat_ftp

# the module for full ftp connection tracking
/sbin/modprobe ip_conntrack_ftp

# the module for full irc connection tracking
/sbin/modprobe ip_conntrack_irc


 #############################################################
###################
#
# Kernel Parameter Configuration
#
# See  http://ipsysctl-tutorial.frozentux.net/chunkyhtml/index.html
# for a detailed tutorial on sysctl and the various settings
# available.

# Required to enable IPv4 forwarding.
# Redhat users can try setting FORWARD_IPV4 in /etc/sysconfig/network to true
```

```
# Alternatively, it can be set in /etc/sysctl.conf
#if [ "$SYSCTL" = "" ]
#then
#    echo "1" > /proc/sys/net/ipv4/ip_forward
#else
#    $SYSCTL net.ipv4.ip_forward="1"
#fi

# This enables dynamic address hacking.
# This may help if you have a dynamic IP address \(e.g. slip, ppp, dhcp\).
#if [ "$SYSCTL" = "" ]
#then
#    echo "1" > /proc/sys/net/ipv4/ip_dynaddr
#else
#    $SYSCTL net.ipv4.ip_dynaddr="1"
#fi

# This enables SYN flood protection.
# The SYN cookies activation allows your system to accept an unlimited
# number of TCP connections while still trying to give reasonable
# service during a denial of service attack.
if [ "$SYSCTL" = "" ]
then
    echo "1" > /proc/sys/net/ipv4/tcp_syncookies
else
    $SYSCTL net.ipv4.tcp_syncookies="1"
fi

# This enables source validation by reversed path according to RFC1812.
# In other words, did the response packet originate from the same interface
# through which the source packet was sent?  It's recommended for single-homed
# systems and routers on stub networks.  Since those are the configurations
# this firewall is designed to support, I turn it on by default.
# Turn it off if you use multiple NICs connected to the same network.
if [ "$SYSCTL" = "" ]
then
    echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
else
    $SYSCTL net.ipv4.conf.all.rp_filter="1"
fi

# This option allows a subnet to be firewalled with a single IP address.
# It's used to build a DMZ.  Since that's not a focus of this firewall
# script, it's not enabled by default, but is included for reference.
# See: http://www.sjdjweis.com/linux/proxyarp/
#if [ "$SYSCTL" = "" ]
#then
#    echo "1" > /proc/sys/net/ipv4/conf/all/proxy_arp
```

```
#else
#    $SYSCTL net.ipv4.conf.all.proxy_arp="1"
#fi

# The following kernel settings were suggested by Alex Weeks. Thanks!

# This kernel parameter instructs the kernel to ignore all ICMP
# echo requests sent to the broadcast address.  This prevents
# a number of smurfs and similar DoS nasty attacks.
if [ "$SYSCTL" = "" ]
then
    echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
else
    $SYSCTL net.ipv4.icmp_echo_ignore_broadcasts="1"
fi

# This option can be used to accept or refuse source routed
# packets.  It is usually on by default, but is generally
# considered a security risk.  This option turns it off.
if [ "$SYSCTL" = "" ]
then
    echo "0" > /proc/sys/net/ipv4/conf/all/accept_source_route
else
    $SYSCTL net.ipv4.conf.all.accept_source_route="0"
fi

# This option can disable ICMP redirects.  ICMP redirects
# are generally considered a security risk and shouldn't be
# needed by most systems using this generator.
#if [ "$SYSCTL" = "" ]
#then
#    echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects
#else
#    $SYSCTL net.ipv4.conf.all.accept_redirects="0"
#fi

# However, we'll ensure the secure_redirects option is on instead.
# This option accepts only from gateways in the default gateways list.
if [ "$SYSCTL" = "" ]
then
    echo "1" > /proc/sys/net/ipv4/conf/all/secure_redirects
else
    $SYSCTL net.ipv4.conf.all.secure_redirects="1"
fi

# This option logs packets from impossible addresses.
if [ "$SYSCTL" = "" ]
then
```

```
    echo "1" > /proc/sys/net/ipv4/conf/all/log_martians
else
    $SYSCTL net.ipv4.conf.all.log_martians="1"
fi


 #############################################################
####################
#
# Flush Any Existing Rules or Chains
#

echo "Flushing Tables ..."

# Reset Default Policies
$IPT -P INPUT ACCEPT
$IPT -P FORWARD ACCEPT
$IPT -P OUTPUT ACCEPT
$IPT -t nat -P PREROUTING ACCEPT
$IPT -t nat -P POSTROUTING ACCEPT
$IPT -t nat -P OUTPUT ACCEPT
$IPT -t mangle -P PREROUTING ACCEPT
$IPT -t mangle -P OUTPUT ACCEPT

# Flush all rules
$IPT -F
$IPT -t nat -F
$IPT -t mangle -F

# Erase all non-default chains
$IPT -X
$IPT -t nat -X
$IPT -t mangle -X

if [ "$1" = "stop" ]
then
 echo "Firewall completely flushed!  Now running with no firewall."
 exit 0
fi


 #############################################################
####################
#
# Rules Configuration
#

 #############################################################
###################
```

```
#
# Filter Table
#
 #############################################################
###################

# Set Policies

$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP


 #############################################################
##################
#
# User-Specified Chains
#
# Create user chains to reduce the number of rules each packet
# must traverse.

echo "Create and populate custom rule chains ..."

# Create a chain to filter INVALID packets

$IPT -N bad_packets

# Create another chain to filter bad tcp packets

$IPT -N bad_tcp_packets

# Create separate chains for icmp, tcp (incoming and outgoing),
# and incoming udp packets.

$IPT -N icmp_packets

# Used for UDP packets inbound from the Internet
$IPT -N udp_inbound

# Used to block outbound UDP services from internal network
# Default to allow all
$IPT -N udp_outbound

# Used to allow inbound services if desired
# Default fail except for established sessions
$IPT -N tcp_inbound

# Used to block outbound services from internal network
# Default to allow all
```

```
$IPT -N tcp_outbound

 ##########################################################
###################
#
# Populate User Chains
#

# bad_packets chain
#

# Drop INVALID packets immediately
$IPT -A bad_packets -p ALL -m state --state INVALID -j LOG \
   --log-prefix "Invalid packet: "

$IPT -A bad_packets -p ALL -m state --state INVALID -j DROP

# Then check the tcp packets for additional problems
$IPT -A bad_packets -p tcp -j bad_tcp_packets

# All good, so return
$IPT -A bad_packets -p ALL -j RETURN

# bad_tcp_packets chain
#
# All tcp packets will traverse this chain.
# Every new connection attempt should begin with
# a syn packet.  If it doesn't, it is likely a
# port scan.  This drops packets in state
# NEW that are not flagged as syn packets.


$IPT -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG \
   --log-prefix "New not syn: "
$IPT -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP

$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL NONE -j LOG \
   --log-prefix "Stealth scan: "
$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL NONE -j DROP

$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL ALL -j LOG \
   --log-prefix "Stealth scan: "
$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL ALL -j DROP

$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL FIN,URG,PSH -j LOG \
   --log-prefix "Stealth scan: "
$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL FIN,URG,PSH -j DROP
```

```
$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j LOG \
    --log-prefix "Stealth scan: "
$IPT -A bad_tcp_packets -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j DROP

$IPT -A bad_tcp_packets -p tcp --tcp-flags SYN,RST SYN,RST -j LOG \
    --log-prefix "Stealth scan: "
$IPT -A bad_tcp_packets -p tcp --tcp-flags SYN,RST SYN,RST -j DROP

$IPT -A bad_tcp_packets -p tcp --tcp-flags SYN,FIN SYN,FIN -j LOG \
    --log-prefix "Stealth scan: "
$IPT -A bad_tcp_packets -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP

# All good, so return
$IPT -A bad_tcp_packets -p tcp -j RETURN

# icmp_packets chain
#
# This chain is for inbound (from the Internet) icmp packets only.
# Type 8 (Echo Request) is not accepted by default
# Enable it if you want remote hosts to be able to reach you.
# 11 (Time Exceeded) is the only one accepted
# that would not already be covered by the established
# connection rule.  Applied to INPUT on the external interface.
#
# See: http://www.ee.siue.edu/~rwalden/networking/icmp.html
# for more info on ICMP types.
#
# Note that the stateful settings allow replies to ICMP packets.
# These rules allow new packets of the specified types.

# ICMP packets should fit in a Layer 2 frame, thus they should
# never be fragmented.  Fragmented ICMP packets are a typical sign
# of a denial of service attack.
$IPT -A icmp_packets --fragment -p ICMP -j LOG \
    --log-prefix "ICMP Fragment: "
$IPT -A icmp_packets --fragment -p ICMP -j DROP

# Echo - uncomment to allow your system to be pinged.
# Uncomment the LOG command if you also want to log PING attempts
#
# $IPT -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j LOG \
#    --log-prefix "Ping detected: "
# $IPT -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT

# By default, however, drop pings without logging. Blaster
# and other worms have infected systems blasting pings.
# Comment the line below if you want pings logged, but it
# will likely fill your logs.
```

```
$IPT -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j DROP

# Time Exceeded
$IPT -A icmp_packets -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT

# Not matched, so return so it will be logged
$IPT -A icmp_packets -p ICMP -j RETURN

# TCP & UDP
# Identify ports at:
#    http://www.chebucto.ns.ca/~rakerman/port-table.html
#    http://www.iana.org/assignments/port-numbers

# udp_inbound chain
#
# This chain describes the inbound UDP packets it will accept.
# It's applied to INPUT on the external or Internet interface.
# Note that the stateful settings allow replies.
# These rules are for new requests.
# It drops netbios packets (windows) immediately without logging.

# Drop netbios calls
# Please note that these rules do not really change the way the firewall
# treats netbios connections.  Connections from the localhost and
# internal interface (if one exists) are accepted by default.
# Responses from the Internet to requests initiated by or through
# the firewall are also accepted by default.  To get here, the
# packets would have to be part of a new request received by the
# Internet interface.  You would have to manually add rules to
# accept these.  I added these rules because some network connections,
# such as those via cable modems, tend to be filled with noise from
# unprotected Windows machines.  These rules drop those packets
# quickly and without logging them.  This prevents them from traversing
# the whole chain and keeps the log from getting cluttered with
# chatter from Windows systems.
$IPT -A udp_inbound -p UDP -s 0/0 --destination-port 137 -j DROP
$IPT -A udp_inbound -p UDP -s 0/0 --destination-port 138 -j DROP

# DNS Server
# Configure the server to use port 53 as the source port for requests
# Note, if you run a caching-only name server that only accepts queries
# from the private network or localhost, you can comment out this line.
$IPT -A udp_inbound -p UDP -s 0/0 --destination-port 53 -j ACCEPT

# If you don't query-source the server to port 53 and you have problems,
# uncomment this rule.  It specifically allows responses to queries
# initiated to another server from a high UDP port.  The stateful
# connection rules should handle this situation, though.
```

```
# $IPT -A udp_inbound -p UDP -s 0/0 --source-port 53 -j ACCEPT


# Not matched, so return for logging
$IPT -A udp_inbound -p UDP -j RETURN

# udp_outbound chain
#
# This chain is used with a private network to prevent forwarding for
# UDP requests on specific protocols.  Applied to the FORWARD rule from
# the internal network.  Ends with an ACCEPT


# No match, so ACCEPT
$IPT -A udp_outbound -p UDP -s 0/0 -j ACCEPT

# tcp_inbound chain
#
# This chain is used to allow inbound connections to the
# system/gateway.  Use with care.  It defaults to none.
# It's applied on INPUT from the external or Internet interface.

# DNS Server - Allow TCP connections (zone transfers and large requests)
# This is disabled by default.  DNS Zone transfers occur via TCP.
# If you need to allow transfers over the net you need to uncomment this line.
# If you allow queries from the 'net, you also need to be aware that although
# DNS queries use UDP by default, a truncated UDP query can legally be
# submitted via TCP instead.  You probably will never need it, but should
# be aware of the fact.
# $IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 53 -j ACCEPT

# Web Server

# HTTP
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 80 -j ACCEPT

# FTP Server (Control)
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 21 -j ACCEPT

# FTP Client (Data Port for non-PASV transfers)
$IPT -A tcp_inbound -p TCP -s 0/0 --source-port 20 -j ACCEPT

# Passive FTP
#
# With passive FTP, the server provides a port to the client
# and allows the client to initiate the connection rather
# than initiating the connection with the client from the data port.
# Web browsers and clients operating behind a firewall generally
```

use passive ftp transfers. A general purpose FTP server
will need to support them.
#
However, by default an FTP server will select a port from the entire
range of high ports. It is not particularly safe to open all
high ports. Fortunately, that range can be restricted. This
firewall presumes that the range has been restricted to a specific
selected range. That range must also be configured in the ftp server.
#
Instructions for specifying the port range for the wu-ftpd server
can be found here:
http://www.wu-ftpd.org/man/ftpaccess.html
(See the passive ports option.)
#
Instructions for the ProFTPD server can be found here:
http://proftpd.linux.co.uk/localsite/Userguide/linked/x861.h tml

Sample Rule
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 62000:64000 -j ACCEPT

Email Server (SMTP)
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 25 -j ACCEPT

Email Server (POP3)
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 110 -j ACCEPT

Email Server (IMAP4)
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 143 -j ACCEPT

sshd
$IPT -A tcp_inbound -p TCP -s 0/0 --destination-port 22 -j ACCEPT

Not matched, so return so it will be logged
$IPT -A tcp_inbound -p TCP -j RETURN

tcp_outbound chain
#
This chain is used with a private network to prevent forwarding for
requests on specific protocols. Applied to the FORWARD rule from
the internal network. Ends with an ACCEPT

No match, so ACCEPT
$IPT -A tcp_outbound -p TCP -s 0/0 -j ACCEPT

 ##
####################

```
#
# INPUT Chain
#

echo "Process INPUT chain ..."

# Allow all on localhost interface
$IPT -A INPUT -p ALL -i $LO_IFACE -j ACCEPT

# Drop bad packets
$IPT -A INPUT -p ALL -j bad_packets

# DOCSIS compliant cable modems
# Some DOCSIS compliant cable modems send IGMP multicasts to find
# connected PCs.  The multicast packets have the destination address
# 224.0.0.1.  You can accept them.  If you choose to do so,
# Uncomment the rule to ACCEPT them and comment the rule to DROP
# them  The firewall will drop them here by default to avoid
# cluttering the log.  The firewall will drop all multicasts
# to the entire subnet (224.0.0.1) by default.  To only affect
# IGMP multicasts, change '-p ALL' to '-p 2'.  Of course,
# if they aren't accepted elsewhere, it will only ensure that
# multicasts on other protocols are logged.
# Drop them without logging.
$IPT -A INPUT -p ALL -d 224.0.0.1 -j DROP
# The rule to accept the packets.
# $IPT -A INPUT -p ALL -d 224.0.0.1 -j ACCEPT


# Inbound Internet Packet Rules

# Accept Established Connections
$IPT -A INPUT -p ALL -i $INET_IFACE -m state --state ESTABLISHED,RELATED \
    -j ACCEPT

# Route the rest to the appropriate user chain
$IPT -A INPUT -p TCP -i $INET_IFACE -j tcp_inbound
$IPT -A INPUT -p UDP -i $INET_IFACE -j udp_inbound
$IPT -A INPUT -p ICMP -i $INET_IFACE -j icmp_packets

# Drop without logging broadcasts that get this far.
# Cuts down on log clutter.
# Comment this line if testing new rules that impact
# broadcast protocols.
$IPT -A INPUT -m pkttype --pkt-type broadcast -j DROP

# Log packets that still don't match
$IPT -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
```

```
        --log-prefix "INPUT packet died: "

 ###############################################################
####################
#
# FORWARD Chain
#

echo "Process FORWARD chain ..."

# Used if forwarding for a private network


 ###############################################################
####################
#
# OUTPUT Chain
#

echo "Process OUTPUT chain ..."

# Generally trust the firewall on output

# However, invalid icmp packets need to be dropped
# to prevent a possible exploit.
$IPT -A OUTPUT -m state -p icmp --state INVALID -j DROP

# Localhost
$IPT -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPT -A OUTPUT -p ALL -o $LO_IFACE -j ACCEPT

# To internet
$IPT -A OUTPUT -p ALL -o $INET_IFACE -j ACCEPT

# Log packets that still don't match
$IPT -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
    --log-prefix "OUTPUT packet died: "

 ###############################################################
####################
#
# nat table
#
 ###############################################################
####################

# The nat table is where network address translation occurs if there
# is a private network.  If the gateway is connected to the Internet
```

with a static IP, snat is used. If the gateway has a dynamic address,
masquerade must be used instead. There is more overhead associated
with masquerade, so snat is better when it can be used.
The nat table has a builtin chain, PREROUTING, for dnat and redirects.
Another, POSTROUTING, handles snat and masquerade.

echo "Load rules for nat table ..."

 ##
###################
#
PREROUTING chain
#

 ##
###################
#
POSTROUTING chain
#

 ##
###################
#
mangle table
#
 ##
###################

The mangle table is used to alter packets. It can alter or mangle them in
several ways. For the purposes of this generator, we only use its ability
to alter the TTL in packets. However, it can be used to set netfilter
mark values on specific packets. Those marks could then be used in another
table like filter, to limit activities associated with a specific host, for
instance. The TOS target can be used to set the Type of Service field in
the IP header. Note that the TTL target might not be included in the
distribution on your system. If it is not and you require it, you will
have to add it. That may require that you build from source.

echo "Load rules for mangle table ..."

 -- --------------------------

[root@v1 ~]# cat /etc/vz/vz.conf
Global parameters

```
VIRTUOZZO=yes
LOCKDIR=/vz/lock
DUMPDIR=/vz/dump
VE0CPUUNITS=1000

## Logging parameters
LOGGING=yes
LOGFILE=/var/log/vzctl.log
LOG_LEVEL=0

## Disk quota parameters
DISK_QUOTA=yes
VZFASTBOOT=no

# The name of the device whose ip address will be used as source ip for VE.
# By default automatically assigned.
#VE_ROUTE_SRC_DEV="eth0"

## Template parameters
TEMPLATE=/vz/template

## Defaults for VEs
VE_ROOT=/vz/root/$VEID
VE_PRIVATE=/vz/private/$VEID
CONFIGFILE="vps.basic"
DEF_OSTEMPLATE="fedora-core-4"

## Load vzwdog module
VZWDOG="no"

#IPTABLES="ipt_REJECT ipt_tos ipt_limit ipt_multiport iptable_filter iptable_man
          gle ipt_TCPMSS ipt_tcpmss ipt_ttl
ipt_length"
IPTABLES="ipt_REJECT
ipt_tos ipt_TOS ipt_LOG ip_conntrack ipt_limit ipt_multiport
iptable_filter iptable_mangle ipt_TCPMSS ipt_tcpmss ipt_ttl ipt_length ipt_state
          iptable_nat ip_nat_ftp
```

--- ----------------------------
 cat /etc/vz/conf/100.conf

```
#
#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either version 2 of the License, or
#  (at your option) any later version.
#
```

```
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
#

ONBOOT="yes"

# UBC parameters (in form of barrier:limit)
# Primary parameters
AVNUMPROC="40:40"
NUMPROC="999999:999999"
NUMTCPSOCK="999999:999999"
NUMOTHERSOCK="999999:999999"
VMGUARPAGES="128000:128000"
# Secondary parameters
KMEMSIZE="299997000:299997000"
TCPSNDBUF="3494912:26214400"
TCPRCVBUF="3494912:26214400"
OTHERSOCKBUF="3494912:26214400"
DGRAMRCVBUF="3494912:26214400"
OOMGUARPAGES="6144:2147483647"
# Auxiliary parameters
LOCKEDPAGES="32:32"
SHMPAGES="8192:8192"
PRIVVMPAGES="255999744:255999744"
NUMFILE="999999:999999"
NUMFLOCK="999999:999999"
NUMPTY="16:16"
NUMSIGINFO="256:256"
DCACHESIZE="1023989760:1023989760"

PHYSPAGES="0:2147483647"
NUMIPTENT="400:400"

# Disk quota parameters (in form of softlimit:hardlimit)
DISKSPACE="61440000:61440000"
DISKINODES="6144000:6144000"
QUOTATIME="0"

# CPU fair sheduler parameter
CPUUNITS="1000"
VE_ROOT="/vz/root/$VEID"
VE_PRIVATE="/vz/private/$VEID"
```

OSTEMPLATE="centos-4-i386-afull"
ORIGIN_SAMPLE="vps.basic"
CPULIMIT="100"
IP_ADDRESS="x.x.x.x"

Subject: Re: Iptables port opening error CentOS
Posted by dim on Tue, 03 Oct 2006 11:44:35 GMT
View Forum Message <> Reply to Message

Probably issue is due to that you have no closed " in IPTABLES definition in /etc/vz/vz.conf.

BTW, could you find exact iptables rules, which cause some of the above errors (it is too much rules in your config, I can't determine visually which could be the reasons)?

Subject: Re: Iptables port opening error CentOS
Posted by whatever on Tue, 03 Oct 2006 12:01:23 GMT
View Forum Message <> Reply to Message

IPTABLES definition in /etc/vz/vz.conf is closed with "
Can you copy paste the firewall rule which only allow port 21,22,53,80,110 open only.
Thanks.

Subject: Re: Iptables port opening error CentOS
Posted by dim on Tue, 03 Oct 2006 12:21:54 GMT
View Forum Message <> Reply to Message

the best way is to run bash -x /etc/init.d/iptables restart inside VE and see, which rule cause error.