pfsense – Boot failure after upgrade to 2.4.0

Scenario

Upgraded from pfsense 2.3x to 2.4.0

Upon reboot, I was unable to ssh to the box.

Once at the physical console, I noticed pfsense had encountered a panic condition,
barking about not being able to mount /dev/ad0s1a

Troubleshooting

At the prompt, I typed in "?" to review the available block devices (disks and the like)

I saw in the output the device /dev/ada0s1a, a slightly different device path from what the error message referred to.

I then entered in: ufs:/dev/ada0s1a, and boom, pfsense kicked off its regular routines (although it did keep barking about this or that package needing to be cleaned and such)

The permanent fix was to correct the mount references in /etc/fstab.

I changed any reference to ad0 to ada0, rebooted, and voila.

Next time I upgrade pfsense, I'll read up on any known issues and the like.

Hint Hint:
2.4 New Features and Changes:

https://doc.pfsense.org/index.php/2.4_New_Features_and_Changes#Known_Issues

---

Kubernetes Deployment Error – PodToleratesNodeTaints

Scenario

You have a single node (master) kubernetes deployment and you want to schedule standard pods.

The master name is your hostname: $(hostname).

Upon your attempt at deploying a service, you notice the state of the resulting pod remains in *Pending*.

Further investigation via kubectl describe pod {{ YOUR_POD_NAME }} reveals an error similar to
No nodes are available that match all of the following predicates:: PodToleratesNodeTaints

Due Diligence:

- All kubernetes nodes are in a 'Ready' status: kubectl get nodes
- All kubernetes nodes have sufficient resources for pod deployment: kubectl describe nodes
- Your image is available on the docker registry you've specified in your kubernetes manifest (.yaml)

Troubleshooting

According to this post:

“No nodes are available that match all of the following predicates:: PodFitsHostPorts (1), PodToleratesNodeTaints” https://github.com/kubernetes/kubernetes/issues/49440

The troubleshooting methodology was to review the kubernetes codebase:

- Navigate to the kubernetes github repo
- Search the repository for the relevant function
- Kubernetes is written in golang, so search for “func PodToleratesNodeTaints”

As such, the following block of code:

```
if
v1helper.TolerationsTolerateTaintsWithFilter(pod.Spec.Tolerati
ons, taints, filter) {
return true, nil, nil
}
```

Will not be executed, which will trigger the next line of code:

```
return                                                  false,
[]algorithm.PredicateFailureReason{ErrTaintsTolerationsNotMatc
h}, nil
```

Effectively returning false, hence the original error

Further investigation on your master:

```
kubectl describe node $(hostname) | grep -i taint
```

If the command returns something similar to:

```
Taints: node-role.kubernetes.io/master:NoSchedule
```

Then your node is unschedulable.

The fix would be to remove this taint, as follows:

kubectl taint nodes $(hostname) node-role.kubernetes.io/master:NoSchedule-

You should see a confirmation similar to:

node {{ NODE_NAME }} untainted

You should now be able to schedule pods on this node

Notes

I came across the github issue description by Googling the following search term:

gls*"No nodes are available that match all of the following predicates" "PodToleratesNodeTaints"

Kubernetes, Docker volume mounts, and autofs

Environment details

- Machine_Type: Virtual
- OS: Oracle Enterprise Linux 7.x
- Software: Docker 1.12.6, Kubernetes 1.7.1

Scenario: Can’t Login via ssh public key

Unable to login to docker host using public key authentication
Able to login to the host using my password
Once at the console, I observed an error similar to:
Could not chdir to home directory /home/myuser: Too many levels of symbolic links
-bash: /home/myuser/.bash_profile: Too many levels of symbolic links

Hmm wtf …

Troubleshooting Steps

A fellow admin suggested I check for docker mapped volumes that point to /home

Here’s the command I used to query for that:

```
sudo docker ps --filter volume=/opt --format "Name:\n\t{{.Names}}\nID:\n\t{{.ID}}\nMounts:\n\t{{.Mounts}}\n"
```

Boom, looks like the kubernetes weaver container is using that mapping:
Name:
k8s_weave_weave-net-ljzn9_kube-system_740c10c5-d6b8-11e7-838f-005056b5384e_0
ID:
dc95801e4442
Mounts:
/opt/kubernetes,/lib/modules,/run/xtables.lo,
/var/lib/kubele,/var/lib/weave,/etc,/var/lib/dbus,/var/lib/kubele,/opt

Ok, so why would a docker volume mapped to /home induce such a

problem?

Turns out that in some cases, binding autofs-mounted paths to docker containers can cause problems on the docker host.

This is due to the way in which kubernetes performs the volume mapping, which utilizes docker volume binds under the hood.

And, depending on how you map a volume to a docker container, you might conflict with autofs volume mounting.

For insight into a similar issue, see:

- Issue with AutoFS mounts and docker 1.11.2: https://github.com/moby/moby/issues/24303

According to the above issue description, the problem we're seeing might be fixed by adjusting the bind propagation for the volume mount in question,
see:
https://docs.docker.com/engine/admin/volumes/bind-mounts/#choosing-the-v-or-mount-flag

However, there's no way to control that setting via a kubernetes manifest, not at present at least, since HostPath bind propagation is currently a proposed feature in kubernetes,
see:
https://github.com/kubernetes/community/blob/master/contributors/design-proposals/node/propagation.md

So the best course of action is to simply change hostPath setting in the weave-kube manifest, e.g.

- Change:
  hostPath:
  path: /home
- To:
  hostPath:
  path: /opt/kubernetes/bind-mounts/weave-kube/home

You can then simply redeploy the offending container (sudo docker stop ecfa204283d3 && sudo docker rm ecfa204283d3 && kubectl apply -f net.yaml)

Note: You'll have to perform similar changes to the weave manifest according to whatever other autofs mounts its hostPath(s) might conflict with.

Ensure you review your autofs settings!

Managing Virtual Machines on Ubuntu KVM

This article is a dump of my experience with setting up a viable virtual machine management platform on an Ubuntu Hypervisor with following specs:

- OS: Ubuntu 14.04.2 LTS
- HDD:
- Memory:

Preflight

Check for Virtualization Support

egrep -c '(vmx|svm)' /proc/cpuinfo

If 0 it means that your CPU doesn't support hardware virtualization.

If 1 or more it does – but you still need to make sure that virtualization is enabled in the BIOS.

Issue Package Updates

- sudo apt-get update
- sudo apt-get upgrade

It is assumed you have the following packages installed

git
python-pip
If No:

sudo apt-get install git python-pip

Setup libvirt and KVM

Manually

sudo apt-get install qemu-kvm libvirt-bin virtinst bridge-utils sasl2-bin

Via bootstrap script

curl http://retspen.github.io/libvirt-bootstrap.sh | sudo sh

Once the installation is complete, add a designated user account

sudo adduser `id -un` libvirtd

Add the option -l in the file */etc/default/libvirt-bin*:
It should look like:

```
libvirtd_opts="-d -l"
```

In the file */etc/libvirt/libvirtd.conf* uncomment the line (Remove #):

```
listen_tls = 0
listen_tcp = 1
tcp_port = "16509"
```

Create a *saslpassword:*

sudo saslpasswd2 -a libvirt [username] // where [username] is the designated libvirt user account

```
    Password: xxxxxx
    Again (for verification): xxxxxx
```

Add firewall rule for *TCP port 16509:*
Create a file /etc/ufw/applications.d/libvirtd and it add the following lines:

```
    [Libvirt]
    title=Virtualization library
    description=Open port for libvirt
    ports=16509/tcp
```

Add a firewall rule in the chain

```
    sudo ufw allow from any to any app Libvirt
```

Install Administration Package

```
    sudo apt-get install git python-pip python-libvirt python-libxml2 novnc supervisor nginx
```

Validate Installation

```
virsh -c qemu+tcp://127.0.0.1/system nodeinfo
```

Please enter your authentication name: [username]
Please enter your password: [password]

Sample Output:

```
  CPU model:           x86_64
  CPU(s):              2
  CPU frequency:       3611 MHz
  CPU socket(s):       1
  Core(s) per socket:  2
  Thread(s) per core:  1
  NUMA cell(s):        1
  Memory size:         3019260 kB
```

Install WebvirtMgr

Install required Packages

```
     sudo apt-get install python-libvirt python-libxml2
```

supervisor nginx

Clone webvirtmgr project from github

```
cd /var/www
git clone git://github.com/retspen/webvirtmgr.git
```

Set permissions

```
sudo chown -R www-data:www-data /var/www/webvirtmgr
```

Install requirements

```
cd webvirtmgr
sudo pip install -r requirements.txt
```

Update Django Settings

```
./manage.py syncdb
./manage.py collectstatic
```

Enter the user information when prompted:

You just installed Django's auth system, which means you don't have any superusers defined.

Would you like to create one now? (yes/no): yes (Put: yes)

Username (Leave blank to use 'admin'): admin (Put: your username or login)

E-mail address: username@domain.local (Put: your email)

Password: xxxxxx (Put: your password)

Password (again): xxxxxx (Put: confirm password)

Superuser created successfully.

Adding additional superusers

./manage.py createsuperuser

(Optional) Enable remote access to the WebUI via Nginx or SSH Tunnel

Usually WebVirtMgr is only available from localhost on port 8000

You can connect via ssh tunnel, like so:

ssh user@server:port -L localhost:8000:localhost:8000 -L localhost:6080:localhost:6080

You should then be able to access WebVirtMgr by typing localhost:8000 in your browser after completing the install. Port 6080 is forwarded to make noVNC work.

or

You can configure a redirect to have the WebUI accessible via nginx:

if not already done:

sudo mv webvirtmgr /var/www/

sudo vim /etc/nginx/conf.d/webvirtmgr.conf

```
server {
        listen 80 default_server;
        server_name $hostname;
        #access_log /var/log/nginx/webvirtmgr_access_log;
        location /static/ {
             root /var/www/webvirtmgr/webvirtmgr; # or /srv instead of /var
            expires max;
        }
        location / {
            proxy_pass http://127.0.0.1:8000;
            proxy_set_header X-Real-IP $remote_addr;
                      proxy_set_header  X-Forwarded-for $proxy_add_x_forwarded_for;
            proxy_set_header Host $host:$server_port;
            proxy_set_header X-Forwarded-Proto $remote_addr;
            proxy_connect_timeout 600;
            proxy_read_timeout 600;
            proxy_send_timeout 600;
            client_max_body_size 1024M; # Set higher depending on your needs
        }
    }
```

Comment the Server Section as it is shown in the example:

Either:

sudo vim /etc/nginx/sites-available/

or

sudo vim /etc/nginx/nginx.conf

Note:

The path may differ

The end result should look like this:

```
        #    server {
```

```
        #        listen       80 default_server;
        #        server_name  localhost;
        #        root         /usr/share/nginx/html;
        #
        #        #charset koi8-r;
        #
         #        #access_log  /var/log/nginx/host.access.log
main;
        #
         #        # Load configuration files for the default
server block.
        #        include /etc/nginx/default.d/*.conf;
        #
        #        location / {
        #        }
        #
         #        # redirect server error pages to the static
page /40x.html
        #        #
        #        error_page  404              /404.html;
        #        location = /40x.html {
        #        }
        #
         #        # redirect server error pages to the static
page /50x.html
        #        #
        #        error_page   500 502 503 504  /50x.html;
        #        location = /50x.html {
        #        }
        #    }
```

Restart nginx service:

sudo service nginx restart

Setup novnc

vi /etc/init.d/novnc

```
  #!/bin/sh
  ### BEGIN INIT INFO
  # Provides:          nova-novncproxy
  # Required-Start:    $network $local_fs $remote_fs $syslog
  # Required-Stop:     $remote_fs
  # Default-Start:     2 3 4 5
```

```
  # Default-Stop:      0 1 6
  # Short-Description: Nova NoVNC proxy
  # Description:       Nova NoVNC proxy
  ### END INIT INFO
   # PATH should only include /usr/* if it runs after the
mountnfs.sh script
  PATH=/sbin:/usr/sbin:/bin:/usr/bin
  DESC="WebVirtMgr NoVNC proxy"
  NAME='webvirtmgr-console'
  DAEMON_PREFIX='/var/www/webvirtmgr/console'
  PIDFILE="/run/${NAME}.pid"
  #SCRIPTNAME="/etc/init.d/${NAME}"
  SCRIPTNAME="/etc/init.d/novnc"
  LOCK_DIR="/run/lock/${NAME}"
  USER='www-data'
  GROUP='www-data'
  # read in defaults if available
  [ -f "/etc/default/${NAME}" ] && . "/etc/default/${NAME}"
  DAEMON="${DAEMON_PREFIX}/${NAME}"
  # Exit if the package is not installed
  [ -x $DAEMON ] || exit 0
  mkdir -p ${LOCK_DIR}
  chown "${USER}:${GROUP}"  ${LOCK_DIR}
  . /lib/lsb/init-functions
  do_start()
  {
     start-stop-daemon --start --background --quiet --chuid
"${USER}:${GROUP}" --make-pidfile --pidfile $PIDFILE --startas
$DAEMON --test > /dev/null \
      || return 1
     start-stop-daemon --start --background --quiet --chuid
"${USER}:${GROUP}" --make-pidfile --pidfile $PIDFILE --startas
$DAEMON -- \
      $DAEMON_ARGS \
      || return 2
  }
  do_stop()
  {
    start-stop-daemon --stop --quiet --retry=TERM/30/KILL/5 --
pidfile $PIDFILE
    RETVAL="$?"
```

```
  rm -f $PIDFILE
  return "$RETVAL"
}
case "$1" in
  start)
    log_daemon_msg "Starting $DESC " "$NAME"
    do_start
    case "$?" in
  0|1) log_end_msg 0 ;;
  2) log_end_msg 1 ;;
  esac
  ;;
  stop)
  log_daemon_msg "Stopping $DESC" "$NAME"
  do_stop
  case "$?" in
    0|1) log_end_msg 0 ;;
    2) log_end_msg 1 ;;
  esac
  ;;
  status)
       status_of_proc "$DAEMON" "$NAME" && exit 0 || exit $?
       ;;
  restart|force-reload)
  log_daemon_msg "Restarting $DESC" "$NAME"
  do_stop
  case "$?" in
    0|1)
    do_start
    case "$?" in
      0) log_end_msg 0 ;;
      1) log_end_msg 1 ;; # Old process is still running
      *) log_end_msg 1 ;; # Failed to start
    esac
    ;;
    *)
    # Failed to stop
    log_end_msg 1
    ;;
  esac
  ;;
```

```
    *)
    echo "Usage: $SCRIPTNAME {start|stop|status|restart|force-
reload}" >&2
    exit 3
    ;;
  esac
```

Setup Supervisor

sudo service novnc stop

sudo insserv -r novnc

sudo vi /etc/insserv/overrides/novnc

```
  #!/bin/sh
  ### BEGIN INIT INFO
  # Provides:          nova-novncproxy
  # Required-Start:    $network $local_fs $remote_fs $syslog
  # Required-Stop:     $remote_fs
  # Default-Start:
  # Default-Stop:
  # Short-Description: Nova NoVNC proxy
  # Description:       Nova NoVNC proxy
  ### END INIT INFO
```

sudo vi /etc/supervisor/conf.d/webvirtmgr.conf

```
  [program:webvirtmgr]
    command=/usr/bin/python  /var/www/webvirtmgr/manage.py
run_gunicorn -c /var/www/webvirtmgr/conf/gunicorn.conf.py
  directory=/var/www/webvirtmgr
  autostart=true
  autorestart=true
  stdout_logfile=/var/log/supervisor/webvirtmgr.log
  redirect_stderr=true
  user=www-data
  [program:webvirtmgr-console]
                                  command=/usr/bin/python
/var/www/webvirtmgr/console/webvirtmgr-console
  directory=/var/www/webvirtmgr
  autostart=true
  autorestart=true
  stdout_logfile=/var/log/supervisor/webvirtmgr-console.log
```

```
  redirect_stderr=true
  user=www-data
```

Restart supervisor daemon

sudo service supervisor restart

WebVirtMgr Post-Installation

I provide the below for additional considerations:

Networking

Before libvirt was installed, virbr0 did not exist. We only had interfaces for loopback and eth0. virbr0 means “virtual bridge 0” and was automatically created by libvirt during installation. virbr0 was configured as a NAT-only interface. This means virtual machine hosts that use this bridge can get out to the network via the eth0 interface but any devices on the other side cannot initiate requests into virbr0 clients.

Here’s my networking configuration:

```
/etc/network/interfaces
    auto eth0
    iface eth0 inet dhcp
    auto br0
    iface br0 inet dhcp
        bridge_ports eth0
        bridge_stp off
    auto eth1
    iface eth1 inet dhcp
    auto eth2
    iface eth2 inet dhcp
```

Troubleshooting The WebVirtMgr Installation

Debugging the webapp

cd /var/www/webvirtmgr

Enable debug mode in the local_settings.py

sudo ./manage.py runserver 0:8000

Errors:

“webvirtmgr” “authentication failed:”

Ensure that any users specified in the connections matches what is listed in the local database

Review administrative users

sasldblistusers2 -f /etc/libvirt/passwd.db

saslpasswd2 -cf /etc/libvirt/passwd.db

If no users configured:

add user

saslpasswd2 -a libvirt <username>

Installing Vagrant

Preflight

Install prerequisites

sudo apt-get install

sudo apt-get install gcc libvirt-dev ruby-libvirt

Download vagrant package

wget https://dl.bintray.com/mitchellh/vagrant/vagrant_1.7.2_x86_64.deb

Install vagrant

sudo dpkg -i vagrant_1.7.2_x86_64.deb

Install vagrant libvirt/kvm provider

sudo vagrant plugin install vagrant-libvirt

Recovering a Failed QNAP Raid Volume

How to recover data from QNAP drives using testdisk from SystemRescueCd

Pre-flight

Given the following scenario:

QNAP server was factory reset, clearing the software RAID information on the QNAP OS.
As such, all drives in the RAID were essentially orphaned.
Data on the drives remained intact.

Recovery Options:

In order to recover the information, we could proceed via many troubleshooting pathways, two of which I list below:

– Rebuilding the software RAID
– Recovering the data directly from the drives

I chose the second option, since I wasn't too handy with administration of the Linux Multiple Device Driver (**MD**), aka software RAID.
In this article, we will be recovering the data from **ONE** drive at a time, so it is best to plug in **ONLY ONE** of drives to be recovered, along with a **spare** drive on which the recovered data will be copied to.

Recovery Software:
We will be using **SystemRescueCD** to perform the data recovery

I assume the following:
You've already booted the SystemRescueCD

You either have console or ssh access (or whatever other means) to the SystemRescueCD shell
You have the drive to be recovered and a spare plugged in to your system

Lastly, this is key in **Understanding QNAP volumes:**
QNAP utilizes Logical Volume Management (**LVM**) and the Linux **MD** software RAID technologies to manage its storage devices.
Partition 3 Holds all the **data** on any given drive
Keep this in mind as you start digging for your data on the QNAP drives.

Identify the Destination Drive

Before going through the recovery, you must prep the directory on which you will be copying the recovered data to.
With the specs on your hard drive already in mind, issue the list hardware command (**lshw**) to determine the device name to the drive:
lshw -short -c disk
Once you match the device information to that of the spare drive, you can proceed to initialize (wipe/clean) the drive or mount it if it's already prepared.

If the drive is already initialized, skip the next step, otherwise proceed …

Prepare the Destination Drive

You can initialize the drive for use on the SystemRescueCD as follows:
fdisk <device_name>, e.g. **fdisk /dev/sda**
Follow the prompts to create a **Linux Partition**
Note: Once the partition is created, the device you'll actually be acting against is <device_namelogical_partition_number>, e.g. **/dev/sda1**
Once you've written the changes to the disk, you can proceed

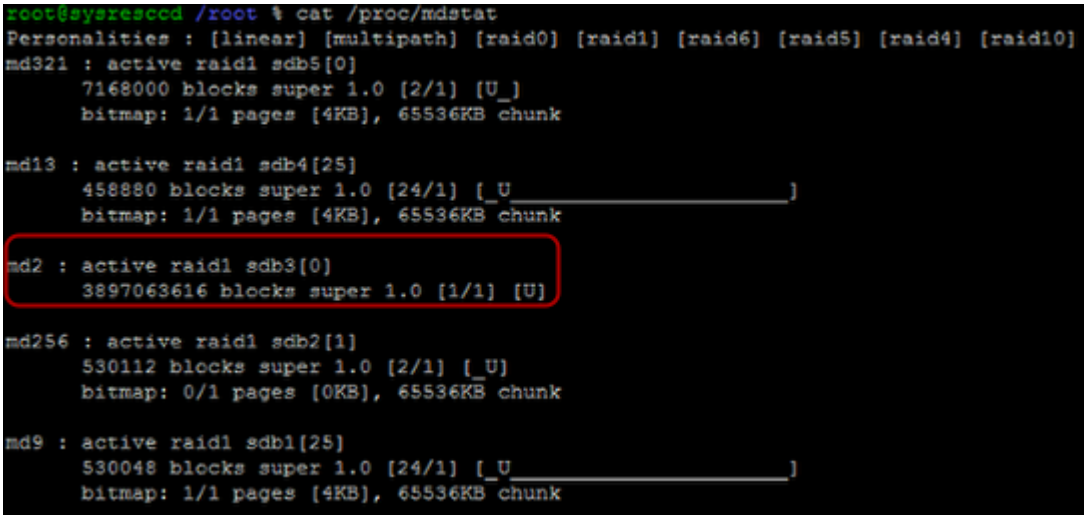
to create the filesystem on the drive:
mkfs -t <fs_type> <device_namelogical_partition_number>, e.g.
mkfs -t ext4 /dev/sda1
or
mkfs.<fstype> <device_namelogical_partition_number>, e.g.
mkfs.ext4 ext4 /dev/sda1
Once the filesystem has been created, you can mount it.
Do so first by creating a directory on which the drive will be mounted, e.g.:
mkdir /mnt/recovery

Mount the Destination Drive

Mounting the drive is quite easy, simply invoke the **mount** command, e.g.:
mount -t ext4 /dev/sda1 /mnt/recovery

Your destination drive is now ready to be used!

Identify the Data Partition on the Source Drive



The following commands are to be issued from the SystemRescueCD session:

First, we need to determine what MD volumes the SystemRescueCD

has detected.
You can do so by displaying the contents of the mdstat file under /proc as follows:

```
cat /proc/mdstat
```

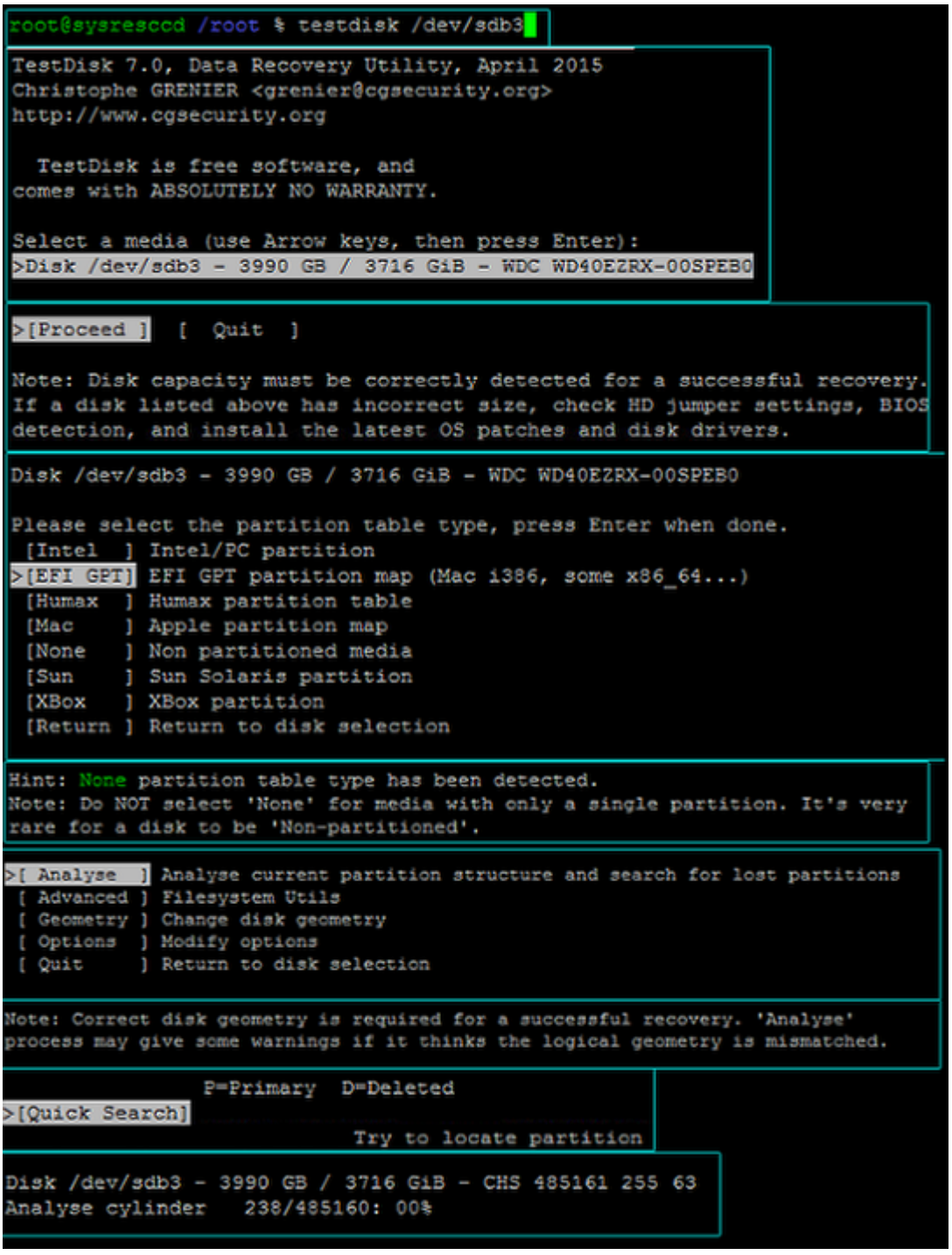
Samlpe Output:

```
        Personalities : [linear] [multipath] [raid0] [raid1]
[raid6] [raid5] [raid4] [raid10]
       md321 : active raid1 sdb5[0]
             7168000 blocks super 1.0 [2/1] [U_]
             bitmap: 1/1 pages [4KB], 65536KB chunk

       md13 : active raid1 sdb4[25]
                        458880 blocks super 1.0 [24/1]
[_U_____________________]
             bitmap: 1/1 pages [4KB], 65536KB chunk

       md2 : active raid1 sdb3[0]
             3897063616 blocks super 1.0 [1/1] [U]

       md256 : active raid1 sdb2[1]
             530112 blocks super 1.0 [2/1] [_U]
             bitmap: 0/1 pages [0KB], 65536KB chunk

       md9 : active raid1 sdb1[25]
                        530048 blocks super 1.0 [24/1]
[_U_____________________]
             bitmap: 1/1 pages [4KB], 65536KB chunk
```

As you can see from the above output, there is a disk with a 3rd partition that is most likely an MD LVM volume.
I'd say there is a 90% chance that this is the drive and partition we're interested in.

Take note of the device information, in this case **/dev/sdb3**

Invoke Testdisk Partiton Scan



So, again, we've detetermined the data to be on device **/dev/sda3**

The next step is to run testdisk against this device:

testdisk /dev/sdb3

In the ensuing dialog, choose the following order of actions:

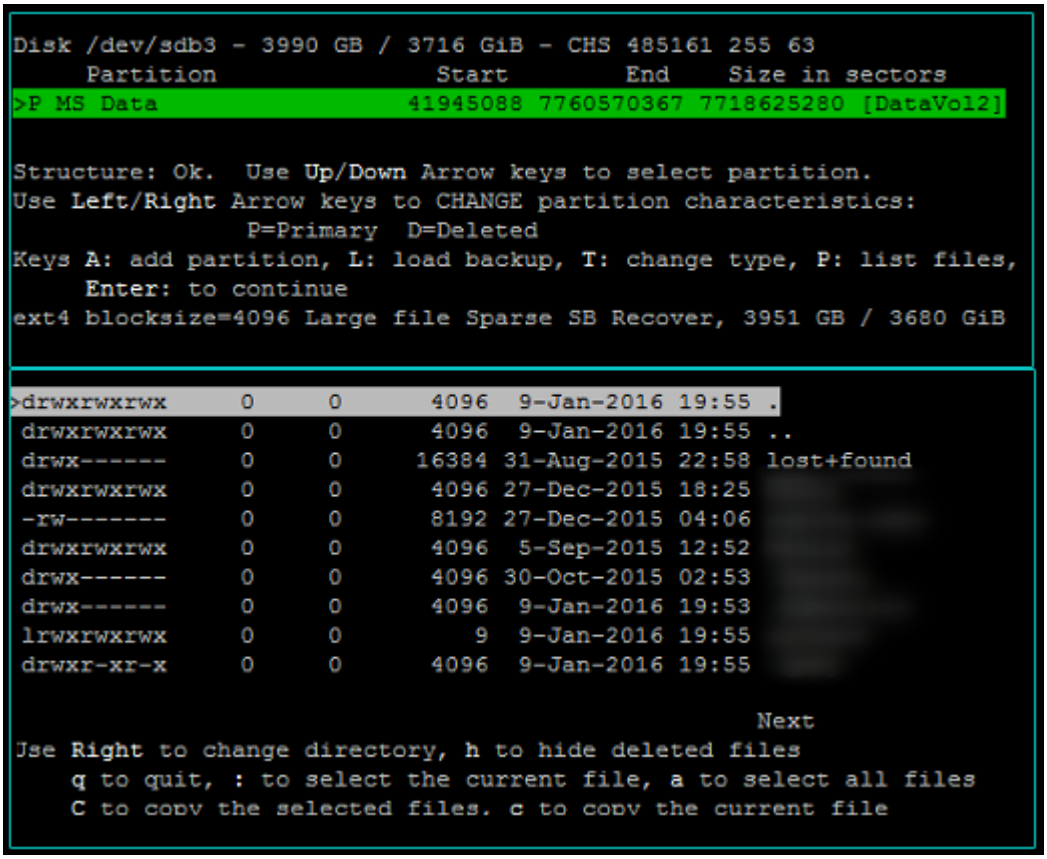
Select a media ...: **(choose the device, in this case /dev/sdb3)**

Proceed
Please select a partition table type ...: **(choose EFI GPT)**
Analyze
Quick Search

At this point, the drive scan will commence.

Once it completes, you'll be presented with a partition table as detected by testdisk.

List Files for Recovery & Copy



In the resulting partition table option, select the partition you think contains the data
Press **shift + P**
This will print the files on the partition
Read the instructions at the bottom of the file listing …

q to quit
: to select the current file
a to select all files

shift + C to copy the selected files
c to copy the current file

Once you invoke the copy action, you will be prompted to navigate to the destination path.

Hopefully you've already completed that in steps '**Prepare the Destination Drive**' and '**Mount the Destination Drive**'

Once the copy process is started, you'll be presented with a progress indication.

Sit tight. The wait is worth it.

Sources

[SMB] HOW-TO RECOVER data from LVM volume on a PC (UX-500P)
http://forum.qnap.com/viewtopic.php?t=93862