Superstes WIKI

Superstes

May 14, 2022

CONTENTS

1 IT-Infrastructure **3**
1.1 IT-Infrastructure 3

2 Network **11**
2.1 Network 11

3 Coding **19**
3.1 Coding 19

Knowledge for everyone.

CHAPTER

ONE

IT-INFRASTRUCTURE

1.1 IT-Infrastructure

1.1.1 LVM

Intro

LVM is a disk-management system that adds some nice functionality in comparison to using bare partitions.

It creates a layer between the physical disks and partitions.

This allows you to:

- Work around the common problem of resizing partitions that are not in the last position on the physical disk
- Creation of volume snapshots
- Thin-Provisioned volumes
- Options to configure RAID
- Implementation of write caches

More information can be found in the RedHat documentation!

Usage

Navigation

How to get the status and information of your current setup.

Overview

The ‘lsblk’ command shows you an overview of your current storage configuration.

```
root@superstes:~# lsblk
> NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
> sda            8:0    0   30G  0 disk
> |-sda1         8:1    0  487M  0 part /boot
> |-sda2         8:2    0    1K  0 part
> `-sda5         8:5    0 29.5G  0 part
>   |-vg0-root 254:1    0  9.7G  0 lvm  /
>   |-vg0-var  254:2    0  9.7G  0 lvm  /var
>   `-vg0-swap 254:3    0  1.9G  0 lvm  [SWAP]
> sdb            8:16   0  100G  0 disk
> `-vg1-data   254:0    0   50G  0 lvm  /mnt/data
```

Physical volumes

These commands show you how LVM sees your physical disks.

```
root@superstes:~# pvs
> PV         VG  Fmt  Attr PSize    PFree
> /dev/sda5  vg0 lvm2 a--   <29.52g  <8.35g
> /dev/sdb   vg1 lvm2 a--  <100.00g <50.00g

root@superstes:~# pvdisplay
# prettified
> --- Physical volume ---
> PV Name               /dev/sdb
> VG Name               vg1
> PV Size               100.00 GiB / not usable 4.00 MiB
> Total PE              25599
> Free PE               12799
> Allocated PE          12800
>
> --- Physical volume ---
> PV Name               /dev/sda5
> VG Name               vg0
> PV Size               29.52 GiB / not usable 2.00 MiB
> Total PE              7557
> Free PE               2137
> Allocated PE          5420
```

That can be useful if you resize the (*virtual*) physical disk and are not sure if LVM realized the changes.

Volume groups

These commands show you the existing Volume Groups.

```
root@superstes:~# vgs
> VG  #PV #LV #SN Attr   VSize    VFree
> vg0   1   3   0 wz--n-  <29.52g  <8.35g
> vg1   1   1   0 wz--n- <100.00g <50.00g

root@superstes:~# vgdisplay
# prettified
> --- Volume group ---
> VG Name               vg1
> VG Access             read/write
> VG Status             resizable
> MAX LV                0
> Cur LV                1
> Open LV               1
> Max PV                0
> Cur PV                1
> Act PV                1
> VG Size               <100.00 GiB
> Alloc PE / Size       12800 / 50.00 GiB
> Free  PE / Size       12799 / <50.00 GiB
>
> --- Volume group ---
> VG Name               vg0
> VG Access             read/write
> VG Status             resizable
> VG Size               <29.52 GiB
> Alloc PE / Size       5420 / 21.17 GiB
> Free  PE / Size       2137 / <8.35 GiB
```

Logical Volumes

These commands show you the existing Logical Volumes.

```
root@superstes:~# lvs
> LV   VG  Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
> root vg0 -wi-ao---- <9.66g
> swap vg0 -wi-ao---- <1.86g
> var  vg0 -wi-ao---- <9.66g
> data vg1 -wi-ao---- 50.00g

root@superstes:~# lvdisplay
# prettified
> --- Logical volume ---
> LV Path                /dev/vg1/data
> LV Name                data
> VG Name                vg1
> LV Write Access        read/write
> LV Status              available
```

(continues on next page)

(continued from previous page)

```
> LV Size                50.00 GiB
>
> --- Logical volume ---
> LV Path                /dev/vg0/root
> LV Name                root
> VG Name                vg0
> LV Size                <9.66 GiB
>
> --- Logical volume ---
> LV Path                /dev/vg0/var
> LV Name                var
> VG Name                vg0
> LV Size                <9.66 GiB
>
> --- Logical volume ---
> LV Path                /dev/vg0/swap
> LV Name                swap
> VG Name                vg0
> LV Size                <1.86 GiB
```

Create

OS Setup

If you set-up Ubuntu or Debian you need to create your LVM-config using the graphical setup!

Once your main partitions are placed on non-LVM volumes - it is hard to move them.

It might be easier to just reinstall the machine.

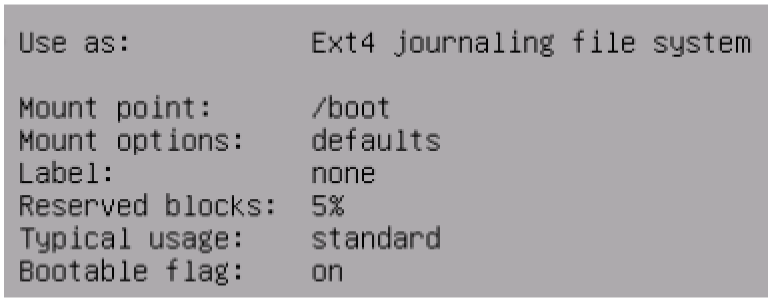
Bootable

You might need to create your **boot-partition 'outside' the LVM**!

I found that EFI boot did not really work when put inside a LVM volume.

Therefor you might want to create a 512MB primary partition at the begin of your disk-layout that acts as 'EFI bootable' or '/boot'.

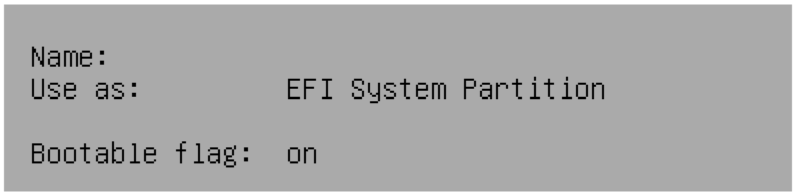
Example for BIOS:



```
Use as:            Ext4 journaling file system

Mount point:       /boot
Mount options:     defaults
Label:             none
Reserved blocks:   5%
Typical usage:     standard
Bootable flag:     on
```

Example for UEFI:



```
Name:
Use as:          EFI System Partition

Bootable flag:   on
```

Add physical volume

1. Add physical volume
2. Make sure LVM recognized the new disk:

   ```
   pvscan
   ```

   Remember the name of your new disk.
3. Create a new volume group or add the physical volume to an existing one:

   ```
   # create new one
   vgcreate <NAME-OF-NEW-VOLUME-GROUP> /dev/<NAME-OF-NEW-DISK>
   #  example: vgcreate vg1 /dev/sdb

   # add to existing one (ADVANCED USAGE)
   vgextend <NAME-OF-EXISTING-VOLUME-GROUP> /dev/<NAME-OF-NEW-DISK>
   #  example: vgextend vg0 /dev/sdb
   ```

4. Create a new logical volume - if needed:

   ```
   lvcreate -n <NAME-OF-NEW-VOLUME> -L <SIZE-OF-NEW-VOLUME> <NAME-OF-VOLUME-GROUP>
   #  example: lvcreate -n data -L 20G vg1
   ```

5. Create file-system: (*'ext4' in this case*)

   ```
   mkfs.ext4 /dev/mapper/<NAME-OF-VOLUME-GROUP>-<NAME-OF-NEW-VOLUME>
   #  example: mkfs.ext4 /dev/mapper/vg1-data
   ```

6. Mount volume:

   ```
   # 1. create mount directory
   mkdir -p /<PATH-TO-MOUNTPOINT>
   #  example: mkdir -p /mnt/data

   # 2. mount permanently
   # 2.1. edit fstab-config
   sudo nano /etc/fstab

   # 2.2. append line
   /dev/mapper/<NAME-OF-VOLUME-GROUP>-<NAME-OF-NEW-VOLUME> /<PATH-TO-MOUNTPOINT>
   ↪<NAME-OF-FILESYSTEM> defaults 0 2
   #  example: /dev/mapper/vg1-data /mnt/data ext4 defaults 0 2

   # 2.3. save and exit
   ```

(continues on next page)

(continued from previous page)

```
# 2.4. mount
sudo mount -a

# check if volume was mounted correctly (yes - if shown in output)
mount | grep "/<PATH-TO-MOUNTPOINT>"
#  example: mount | grep "/mnt/data"
```

Resize

Warning: If you don't know what you are doing => you should not make changes like these on an important system!

You might **BREAK YOU SYSTEM**!!

Try it on an useless VM and play around with it or leave it to the pros.

We need to go through these steps:

1. Extend the physical drive or partition

   1.1. Drive

   You might need to resize the disk in your virtual environment/hypervisor.

   1.2. Partition - entering fdisk

```
# start fdisk targeting the modified disk
fdisk /dev/<NAME-OF-DISK>
#  example: fdisk /dev/sdb

# show current partition layout
p
```

   1.3. Partition - direct vs nested

   Sometimes the target partition might be encapsulated inside an 'extended' partition.

   It might look like this:

```
fdisk -l /dev/sda
> Device     Boot   Start      End  Sectors  Size Id Type
> /dev/sda1  *       2048   999423   997376  487M 83 Linux
> /dev/sda2       1001470 25163775 24162306 11.5G  5 Extended
> /dev/sda5       1001472 25163775 24162304 11.5G 8e Linux LVM
```

   In that case you will have to delete and re-add both of these partitions to extend them.

```
# delete partition you want to extend
d
=> number of partition

# delete the extended partition
```

(continues on next page)

(continued from previous page)

```
d
=> number of partition

# re-create the extended partition
n
=> choose 'extended'
e
=> enter or choose custom partition number
=> enter
=> enter

# re-create the target partition
n
=> enter
=> enter
=> remove the Signature?
n  # else your LVM config will be gone

# modify partition type
t
=> enter partition number
8e  # for LVM disk

# verify the layout is the same as before (except being bigger)
p

# save and write
w

# example:
fdisk /dev/sda
d ENTER 5 ENTER
d ENTER 2 ENTER
n ENTER e ENTER 2 ENTER ENTER ENTER
n ENTER ENTER n ENTER
t ENTER 5 ENTER 8e ENTER
w
```

If that is not the case it is a little easier:

```
# delete partition you want to extend
d
=> number of partition to increase

# re-create the partition
n
=> enter or choose custom partition number
=> enter
=> enter
=> remove the Signature?
n  # else your LVM config will be gone

```

(continues on next page)

(continued from previous page)

```
# modify partition type
t
=> enter partition number
8e  # for LVM disk

# verify the layout is the same as before (except being bigger)
p

# save and write
w

# example:
fdisk /dev/sda
d ENTER 2 ENTER
n ENTER p ENTER 2 ENTER ENTER ENTER n ENTER
t ENTER 2 ENTER 8e ENTER
w
```

2. Resize the LVM physical volume

```
pvresize /dev/sdX
```

3. Extend the LVM volume group

```
vgextend vg0 /dev/sdX
```

4. Extend the LVM logical volume

```
lvextend /dev/vg0/lv1 -L 20GB
```

5. Update the partition size

```
resize2fs /dev/mapper/vg0-lv1
```

```
#!/bin/bash
PD='sda'
LVM_PV="${PD}2"
LVM_VG='vg0'
LVM_LV='lv1'
EXT='20GB'

fdisk "/dev/${PD}"
pvresize "/dev/${LVM_PV}"
vgextend "${LVM_VG}" "/dev/${LVM_PV}"
lvextend "/dev/${LVM_VG}/${LVM_LV}" -L "${EXT}"
resize2fs "/dev/mapper/${LVM_VG}-${LVM_LV}"
```

CHAPTER

TWO

NETWORK

2.1 Network

2.1.1 ProxyShark

Warning: This application let's you intercept and modify network traffic.

That can be illegal => you are warned.

Introduction

I came across a challenge that needed me to modify tcp packages as man-in-the-middle.

I found a nice tool to do that; but without much information on how to set it up or use it.

Therefore I will provide those info's to you:

The tool I'm talking about is called 'proxyshark'. It's a python2 program/script written by the company '*CONIX Cybersecurity*' from france.

Just so I acknowledged it:

1. I know you could code this from scratch by only using the netfilterqueue and scapy modules
2. There are cleaner, faster and more up-to-date projects out there.. (p.e. pypacker)

Source

Original

- GitHub

My fork containing the fixes as described below:

- ProxyShark Fork

Dependencies

At first I had some problems getting started with this script since the dependencies are ‘a little’ outdated.

Disclaimer: You might want to use this on a vm - since the dependencies will foul your system a little.

```
# install pip2 dependencies
apt install build-essential python-dev libnetfilter-queue-dev tshark git python2

# install pip2
wget https://bootstrap.pypa.io/pip/2.7/get-pip.py
sudo python2 get-pip.py

# install pip packages
python2 -m pip install dnspython pyparsing

# scapy workaround
python3 -m pip install scapy
sudo mkdir /usr/lib/python2.7/dist-packages/scapy
cd /usr/lib/python3/dist-packages/
sudo cp -avr scapy/* /usr/lib/python2.7/dist-packages/scapy
```

Bugs & Fixes

Code

Note: The fixes are included in my fork of this project => link can be found above.

The capture bugged out on me.

Therefore I fixed some ‘bugs’ that broke it:

```
# Line 1159 => comment out the existing regex and replace it with this one:

# regex = r'^.*(\d+\.\d+) +([^ ]+) +-> +([^ ]+) +([^ ]+) +[^ ]+ +(.*)$'
regex = r'.*?(\d+\.\d+)\s*((?:[0-9]{1,3}\.){3}[0-9]{1,3}).*?((?:[0-9]{1,3}\.){3}[0-91,3}
↪)\s*([A-Z]{3,20})\s*(.*)'  # might be problematic in edge-cases

# Line 1674 => add those lines before the 'if item_showname' matching:
if type(item_show) == 'unicode':
    item_show = item_show.encode('utf8')
if type(item_showname) == 'unicode':
    item_showname = item_showname.encode('utf8')
```

Execution

Tshark

Run it with the '-t /usr/bin' argument to use the newer apt-version of **tshark** => the one from their repo bugged out on me..

Packet modification

Somehow the 'direct' packet modification did not work.

I found the list object '_items_to_commit' and modified the value in it => that worked

```
bpkt._items_to_commit[22]['value'] = 'aa02'
```

Usage

Some basic commands can be found in the ReadMe of the repository!

```
python2 ps1/proxyshark.py -v --capture-filter 'dst host 8.8.8.8' --packet-filter 'udp' -
↪t /usr/bin/
```

- Start the capture from the **interactive mode** by typing 'run'
- You should now see packages matching the filter by **typing 'packet'** (*last one*) or **'queue'** (*all*)
- Those packages can be 'caught' by defining a **breakpoint**.
- Captured packets can be **interacted** with in the default **python2 syntax**.
- **All possible attributes** etc. can be displayed by entering **'bpkt.__dict__'** (*after capturing some packet with a breakpoint*)
- Breakpoints can either pause the capture, so the packet can be edited manually, or modify it automatically with defined action

At first you might want to play around with the attributes of captures packets.

After that you can write automated actions to become a fully-grown m.i.t.m.

Examples

- show tcp data attributes

  ```
  bpkt['tcp.data']
  ```

- show tcp flags

  ```
  bpkt['tcp.flags']
  ```

- show ip destination

```
bpkt['ip.dst']
```

- rewrite tcp-reserved bits in auto-mode:

```
a add ta1 to tb1 "_flags = bpkt['tcp.flags'][0]['unmaskedvalue'][:1] + 'a'␣
↪+ bpkt['tcp.flags'][0]['unmaskedvalue'][2:]" "bpkt._items_to_commit[22][
↪'value'] = _flags" "print _flags" "bpkt.accept()"
```

2.1.2 Tor

Warning: Tor SHOULD NOT be used to masquerade the source of malicious network requests.

That can be illegal => you are warned.

Introduction

Tor is used to anonymize network requests and host/allow access to hidden (*.onion*) services.

Warning: Traffic you send over tor should **ALWAYS be encrypted**!

Else malicious tor nodes might be able to read and even modify your requests.

Just using the tor network won't be enough to stay anonymous!

You should also always follow the basic guidelines on how to stay anonymous.

If you are serious about your anonymity you might even go a set further and:

- Set-up a virtual machine to use for accessing tor.

  Per example: whonix

- Connect that vm to a network that has no access to internal resources. (*only internet access, maybe a guest-network or 'dmz'*)

Route traffic over tor

Tor uses the protocol SOCKS5.

A technical limitation of the tor implementation is that only TCP-traffic can be sent over those connections.

DNS

DNS-requests can be tunneled specially to prevent dns-leaking. (*your provider [and so on..] will know what you are accessing*)

Implementation examples:

- Cloudflare hidden resolver
- local tor dns-port

  Basically:

```
# example on ubuntu/debian

sudo -i
#   installing
apt update
apt install tor -y
#     if you want to start it on system boot
systemctl enable tor

#   configuration
echo 'DNSPort 5353' > /etc/tor/torrc
systemctl start tor
echo 'nameserver 127.0.0.1' > /etc/resolv.conf  # will not be persistent
```

- You might want to enable DNS-over-HTTPS or DNS-over-TLS to keep your requests secure.

Web access

To access websites anonymously you can use the tor browser.

Basics:

- Use DuckDuckGo as **search engine**!

  Other engines like Google will compromise your anonymity!

- Don't log-in with your **personal accounts** => it will compromise your anonymity.

- Don't enable **JavaScript** (*disabled via NoScript by default*)

  Some websites won't work correctly => but it keeps you safe(r).

- You might see **websites blocking your requests** as the most providers are blocking or limiting requests from tor exit-nodes.

  Maybe pressing 'Ctrl+Shift+L', to use a new 'route' for accessing the current page, might help.

- Using **hidden services**

  The 'default' websites you use daily using your normal browser are in a domain of the internet called 'clear-net'.

  You can browse them without worrying too much - they might 'just' compromise your anonymity.

  Hidden services are in the 'deep-net' => those are hidden for the usual user and only reachable using tor.

  For clarification: tor is only ONE network that hosts hidden services - there are more out there

  Hidden services have their own search engines but they have not listed many of the existing services!

  For the most** part you need to know the unique **.onion** address of a service to access it.

  Warning: Be aware:

  - You might see/find disturbing and/or illegal content on those hidden services.
  - You need to have a basic technical understanding on how to interact with those services securely - else you might even get hacked.

Specific program

Whenever you want a program to use tor as 'gateway' for its connection you need to 'proxy' it.

That proxy needs a tor 'SOCKS' socket to connect to.

SOCKS socket:

- The easiest way of starting such a socket is by opening the tor browser

  It starts such a socket in the background!

  ```
  socks5 127.0.0.1 9150
  ```

- Another way is to install & start tor as service

  ```
  socks5 127.0.0.1 9050
  ```

Linux

On linux I would recommend using the application 'proxychains4' to achieve that.

You just need to set the SOCKS target to use.

```
# example: tor browser SOCKS
sudo -i
echo 'socks5  127.0.0.1 9150' > /etc/proxychains4.conf
```

After that you can just start the application that should connect over tor by prepending 'proxychains4' to its command:

```
# without tor
curl https://ipinfo.io/city
# using tor
proxychains4 curl https://ipinfo.io/city
```

SSH

You can also set a proxy for ssh-connections.

Another program called 'netcat' is needed to archive that.

You will need to install the variant 'netcat-openbsd' as the 'default' one does not implement the needed options.

```
# example on ubuntu/debian using tor browser SOCKS
#   install dependencies
sudo apt update
sudo apt install openssh netcat-openbsd -y

#   use
ssh -p PORT -o ProxyCommand="nc -X5 -x127.0.0.1:9150 %h %p" USER@SERVER
```

All Traffic

There are options to send all **TCP-Traffic and DNS-Requests** over the tor network.

> **Warning:** This should only be used if you really know what you are doing - as there are many ways you might compromise your anonymity!

Linux

Here is the official guide to proxying.

I won't go into the details on how to set this up - as I have not got experience with it.

It is done something like this: (*copied from the guide*)

```
# example for 'middlebox' on ubuntu/debian
sudo -i
#   installing
apt update
apt install tor -y
#     if you want to start it on system boot
systemctl enable tor

#   writing config
echo 'VirtualAddrNetworkIPv4 10.192.0.0/10' > /etc/tor/torrc
echo 'AutomapHostsOnResolve 1' >> /etc/tor/torrc
echo 'TransPort 192.168.1.1:9040' >> /etc/tor/torrc
echo 'DNSPort 192.168.1.1:5353' >> /etc/tor/torrc
systemctl restart tor

#   adding traffic redirection
_trans_port="9040"
_inc_if="eth1"  # you need to update the interface
iptables -t nat -F  # WARNING: will remove all existing NAT-rules
iptables -t nat -A PREROUTING -i $_inc_if -p udp --dport 53 -j REDIRECT --to-ports 5353
iptables -t nat -A PREROUTING -i $_inc_if -p udp --dport 5353 -j REDIRECT --to-ports 5353
iptables -t nat -A PREROUTING -i $_inc_if -p tcp --syn -j REDIRECT --to-ports $_trans_
↪port
```

Windows

You can use a tool like OnionFruit.

CHAPTER

THREE

CODING

3.1 Coding