

BACKTRACK 4 - BOOTABLE USB THUMB DRIVE WITH "FULL" DISK ENCRYPTION

This is a step-by-step guide showing how to create a encrypted bootable Backtrack 4 USB thumb drive.

Before we get started, here are a few housekeeping items:

- I also made a video of the process. It is [here](#). (The video is a bit out of date too)
- Finally, if you want to be notified of updates to this page, subscribe to my RSS feed [here](#).

I put quotes around full in the title because technically the whole disk isn't encrypted. We use LVM and the native encryption routines included in Ubuntu 8.10 to encrypt all partitions except for a small boot partition that never contains any data.

This is a fairly involved process, but I have done my best to document each detail. Please let me know if I missed anything or you have any questions. I can be reached via the contact form on the 'About' page of this website or via the comments below.

I strongly recommend you read through this guide at least once before starting. I will be making a PDF available in the near future.

As in all my how-tos, user entered text is **bold** and comments are preceded by a # sign and generally not part of the output of a command.

Finally, a couple of posts from the Ubuntu Community Documentation site were instrumental in getting this working.

<https://help.ubuntu.com/community/EncryptedFilesystemOnIntrepid>

<https://help.ubuntu.com/community/EncryptedFilesystemLVMHowto>

WARNING: Before you start, please be aware that you can cause the system you are using to build this with to not boot correctly. During the install process below there is a warning about indicating where you want the boot loader to be installed. Be very careful at this point.

First we are going to need some stuff.

Tools and Supplies

1. A USB thumbdrive for the install - minimum capacity 8GB for Backtrack Final or 16GB for Backtrack 2
2. A Backtrack 4 DVD or an additional USB thumbdrive (minimum 2GB, must be Backtrack 4)
3. Optional: [UNetbootin](#) - A tool to transfer an iso image to a USB drive.
4. Working internet connection once Backtrack 4 is booted.

Let's get started!

Let's grab a copy of the Backtrack 4 R2 ISO.

BackTrack 4 R2 Release ISO

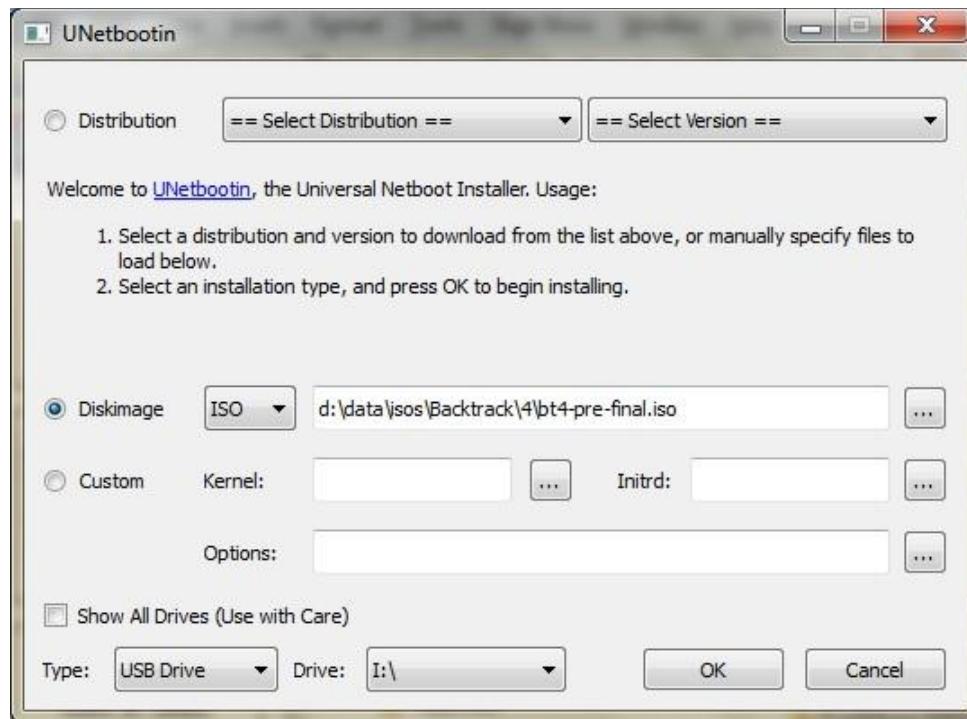
Last Update: 22.11.2010

Name: bt4-r2.iso **Size:** 2000 MB

MD5: 9a94caa0e980a7331e9abc1d4c42c9a9

[Download](#)[Torrent](#)

Now that we have the goods in hand, we can get to cooking. This tutorial is based on booting Backtrack 4 first. This means that you need some form of bootable Backtrack 4 media. This can be a virtual machine, DVD, or USB drive. Use your favorite method of creating a DVD or USB drive or you can use UNetBootin to create the thumb drive. Below is a screenshot of using UnetBootin to install Backtrack 4 on a USB drive.



It is as simple as selecting the image we want to write to the USB drive, the drive to write it to, and then clicking the 'OK' button. Warning: Make sure you pick the correct destination drive. You don't want to shoot yourself in the foot. :)

Partitioning

The first step is the physical partitioning of the drive.

Boot up Backtrack 4 from your DVD or USB drive. We will need both networking and the graphical interface running. The following commands will get us there.

```
/etc/init.d/networking start  
startx
```

We will also need to figure out which drive is our target drive. The following command will show the drives available and you can determine from that which is the new USB drive. Open a terminal windows and execute the following.

```
dmesg | egrep hd.\|sd.
```

We need to physically partition the target drive as follows:

1. The first partition needs to be a primary partition, 120 MB in size set to type ext3. Also remember to make this partition active when you are creating it. Otherwise you might have some boot problems.
2. The rest of the drive should be configured as an extended partition and then a logical partition created on top of it.

Below are the steps to take to get the drive partitioned. A '# blah blah' indicates a comment and is not part of the command and user typed commands are **bolded**. One note, we will need to delete any existing partitions on the drive. Final note, the cylinder numbers below are specific to my test machines/thumb drives, yours may be different.

fdisk /dev/sdb # use the appropriate drive letter for your system

delete existing partitions. There may be more than one.

Command (m for help): **d**

Partition number (1-4): **1**

create the first partition

Command (m for help): **n**

Command action

e extended

p primary partition (1-4)

p

Partition number (1-4): **1**

First cylinder (1-1044, default 1): **<enter>**

Using default value 1

Last cylinder, +cylinders or +size{K,M,G} (1-1044, default 1044): **+120M**

#create the extended partition

Command (m for help): **n**

Command action

e extended

p primary partition (1-4)

e

Partition number (1-4): **2**

First cylinder (15-1044, default 15): **<enter>**

Using default value 15

Last cylinder, +cylinders or +size{K,M,G} (15-1044, default 1044): **<enter>**

Using default value 1044

Create the logical partition.

Command (m for help): **n**

Command action

l logical (5 or over)

p primary partition (1-4)

l

First cylinder (15-1044, default 15): **<enter>**

Using default value 15

Last cylinder, +cylinders or +size{K,M,G} (15-1044, default 1044): **<enter>**

Using default value 1044

Setting the partition type for the first partition to ext3

Command (m for help): **t**

Partition number (1-4): **1**

Hex code (type L to list codes): **83**

```
# Setting the first partition active
```

```
Command (m for help): a
```

```
Partition number (1-4): 1
```

```
Command (m for help): w
```

It is now time to get a couple additional packages installed that we need for LVM and encryption. First we need to update the local repositories and then install lvm2 and hashalot. Output has been omitted.

```
apt-get update
```

```
apt-get install hashalot lvm2
```

Our next step is to enable encryption on the logical partition we created above and make it available for use.

Before we do that though, there is an optional step we can take if we want to make sure no one can tell where our data is on the drive. It isn't really necessary since anything written will be encrypted, but if we want to be thorough and make sure no one can see where our data even sits on the drive, we can fill the logical partition with random data before enabling encryption on it. This will take some time, as much as a couple hours or more. Execute the following command:

```
dd if=/dev/urandom of=/dev/sdb5
```

The following commands will setup encryption services for the partition and open it for use. There are several ciphers that can be used, but the one indicated in the command is supposed to be the most secure and quickest for Ubuntu 8.10. Please note that the case of the command luksFormat is required.

```
cryptsetup -y --cipher aes-xts-plain --key-size 512 luksFormat /dev/sdb5
```

```
WARNING!
```

```
=====
```

```
This will overwrite data on /dev/sdb5 irrevocably.
```

```
Are you sure? (Type uppercase yes): YES
```

```
Enter LUKS passphrase: (enter passphrase) [type passphrase]
```

```
Verify passphrase: (repeat passphrase) [type passphrase]
```

```
Command successful.
```

```
cryptsetup luksOpen /dev/sdb5 pvcrypt
```

```
Enter LUKS passphrase: [type passphrase]
```

```
key slot 0 unlocked.
```

```
Command successful.
```

Now that that's all done, we can create our root and swap partitions using LVM. Again, the commands below will do so. 7.3 GB was the largest I could make my root partition. Play around with it a little and you may be able to make it a bit larger or you may have to make it a bit smaller.

```
pvcreate /dev/mapper/pvcrypt
```

```
Physical "volume /dev/mapper/pvcrypt" successfully created
```

```
vgcreate vg /dev/mapper/pvcrypt
```

```
Volume group "vg" successfully created
```

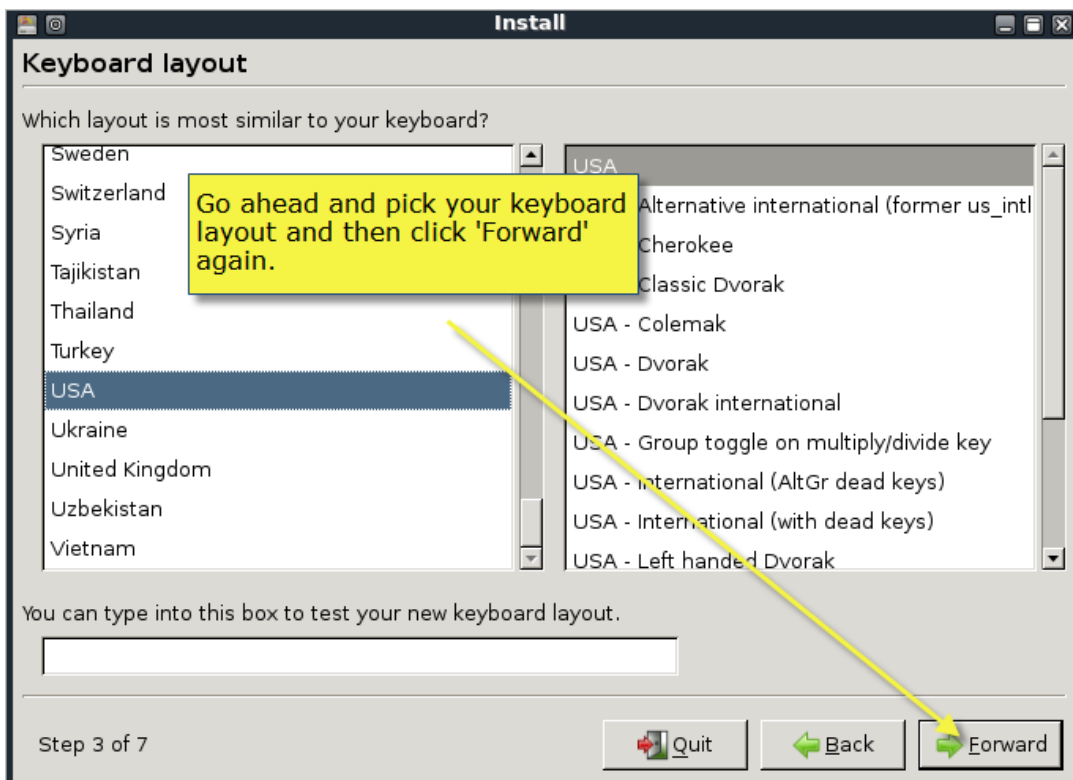
```
lvcreate -n root -l 100%FREE vg
```

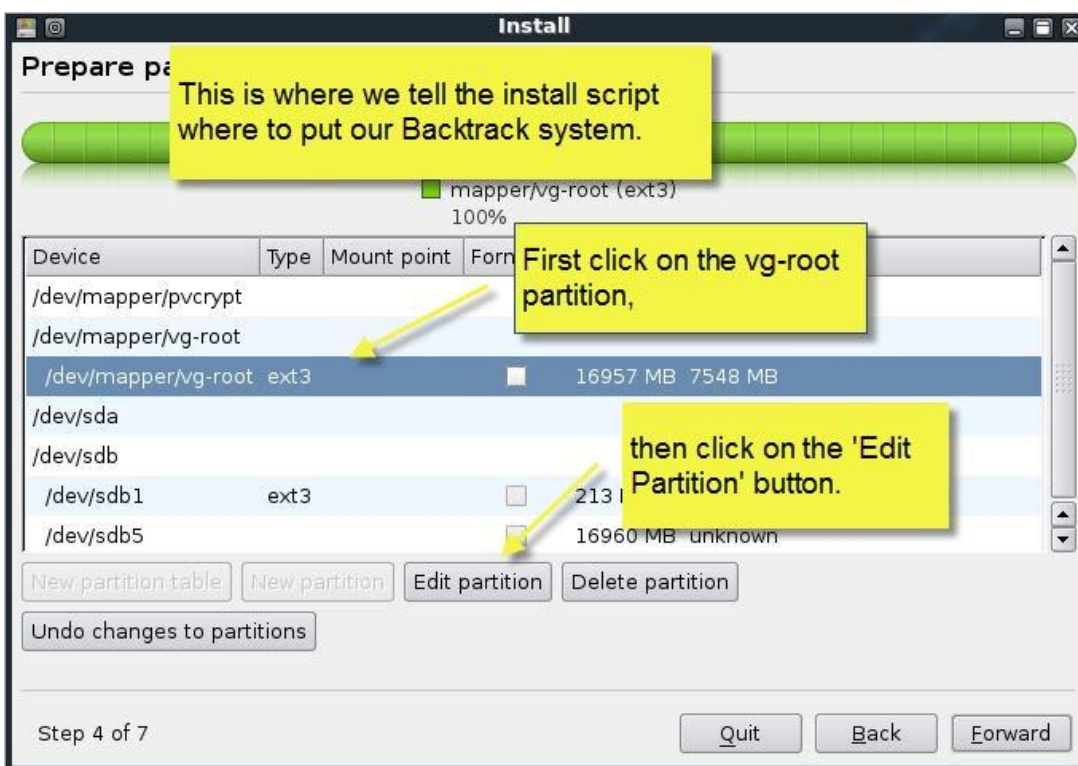
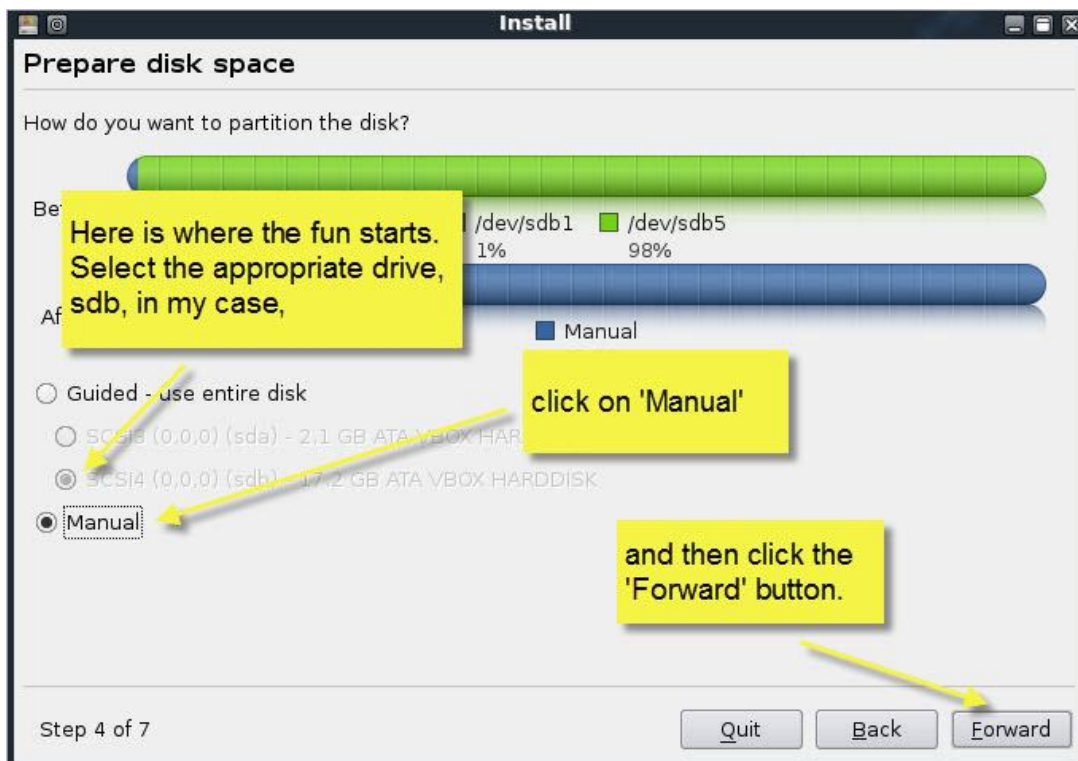
```
Logical volume "root" created.
```

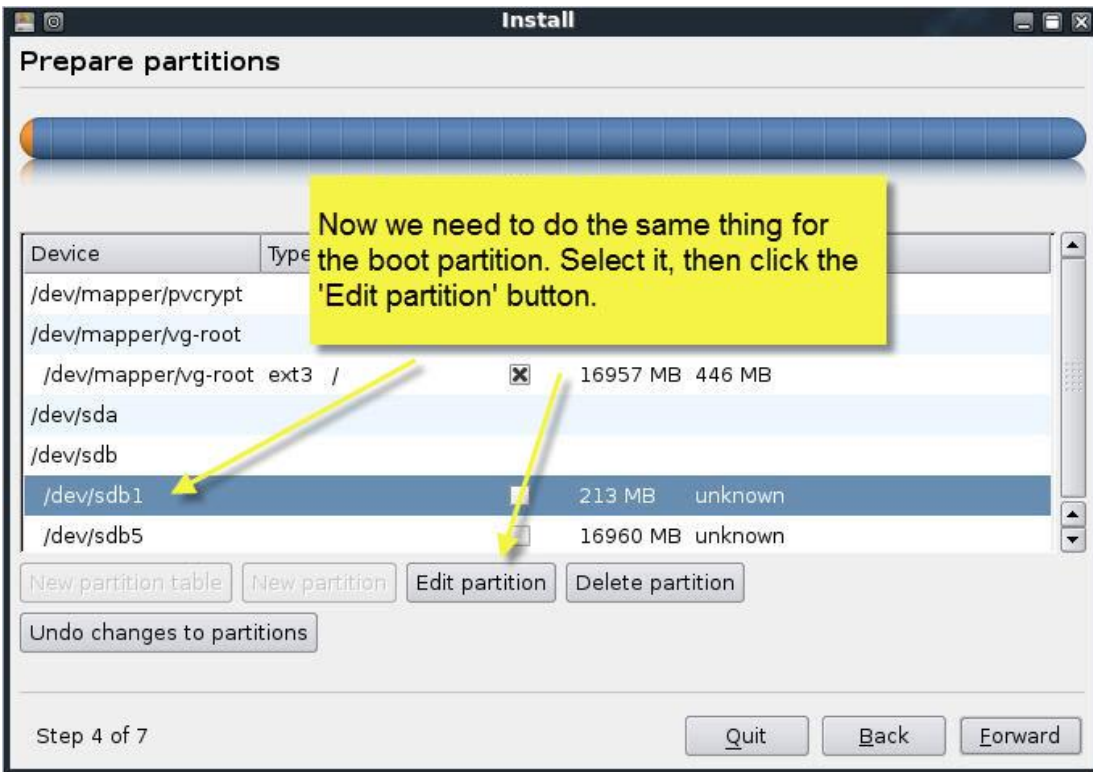
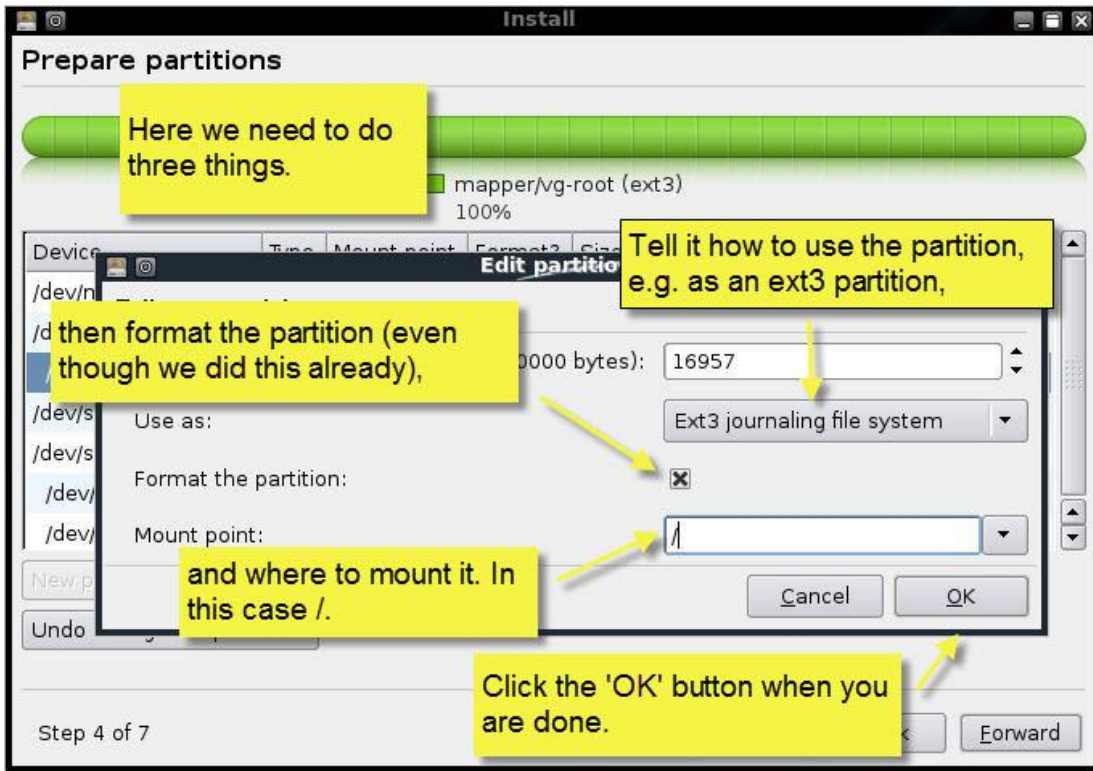
The final step is to format the logical volumes we just created. I have not included the output below for brevity's sake.

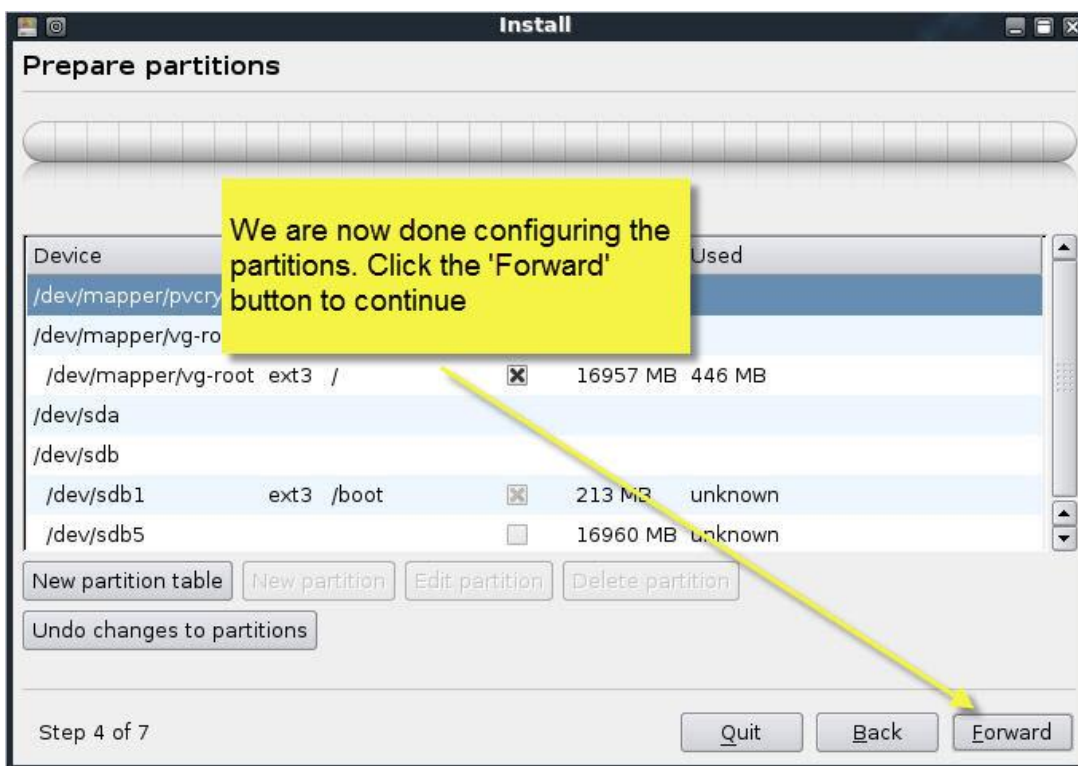
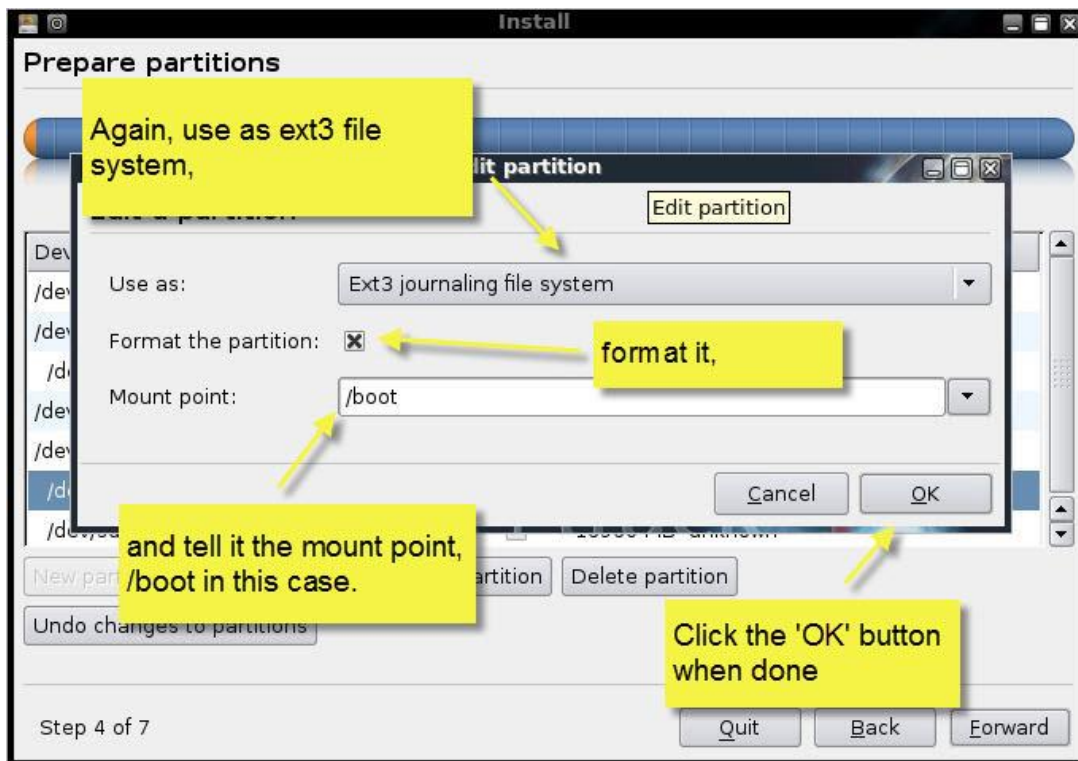
mkfs.ext3 /dev/mapper/vg-root

Believe it or not, we are finally ready to start installing Backtrack. To do, click on the install.sh icon on the desktop. This will start the graphical installer.

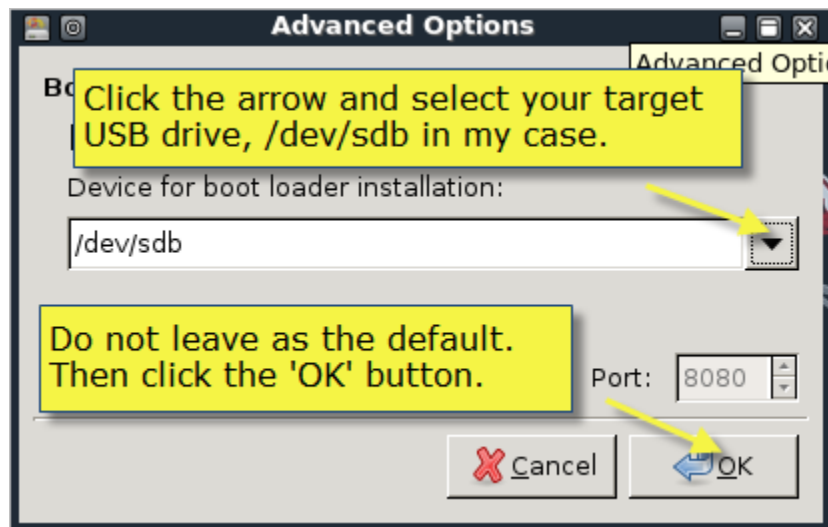
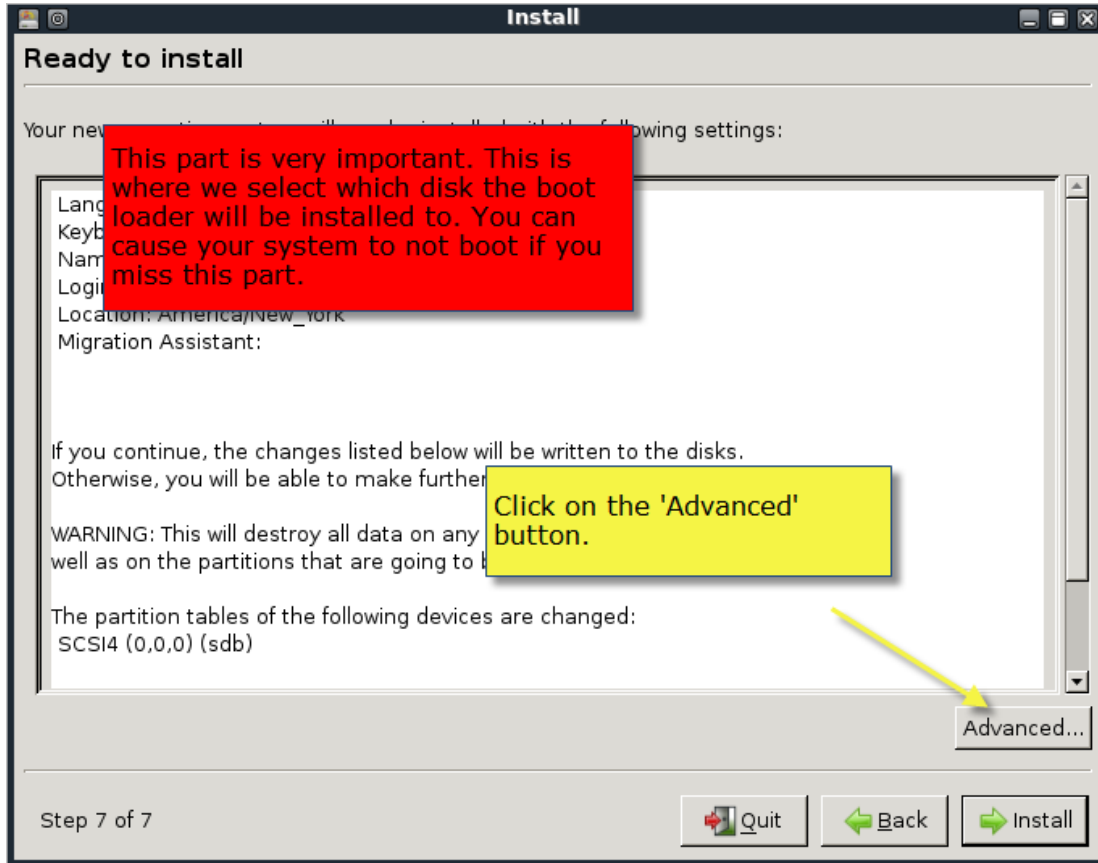


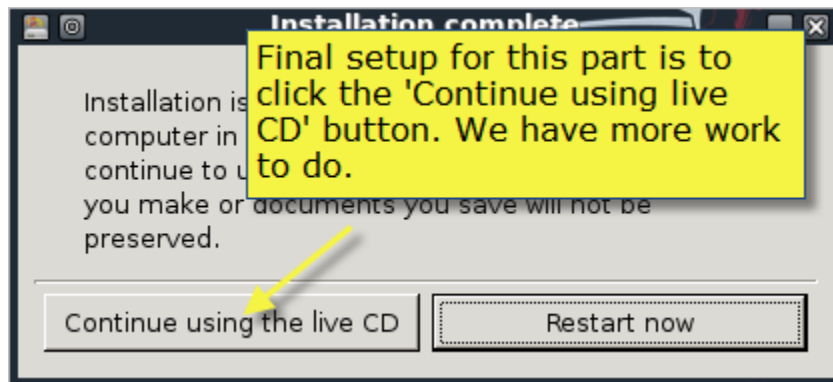
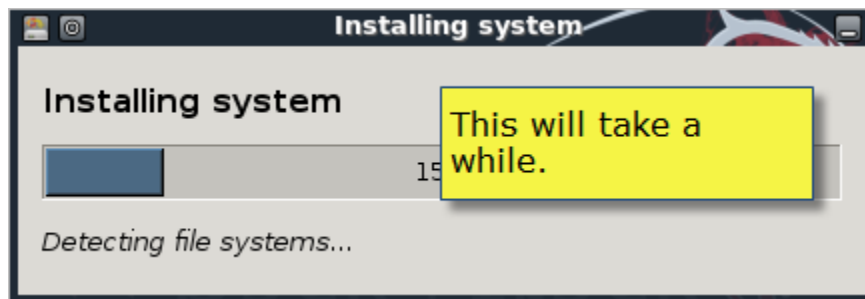
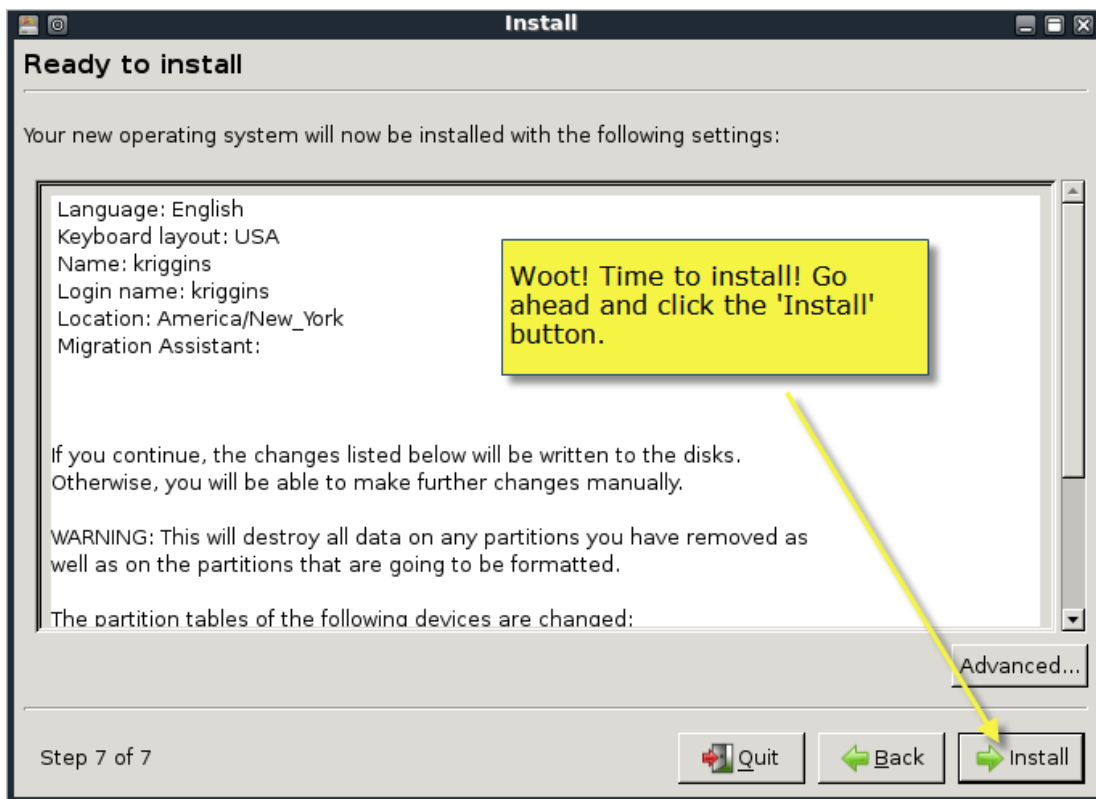






WARNING: You must click on the advanced tab on the next page and select your USB drive as the target for installing the bootloader. You will break your system if you do not.





We have now installed the main distribution to our thumb drive. The next step is to configure the newly installed system to use LVM and open the encrypted partition.

However, before we do that we need to figure out the UUID of our encrypted volume. We want to do this so that we don't run into problems if the device name of the drive changes from machine to machine. The command `vol_id` will give us the information we need. So execute `vol_id` as below.

```
vol_id /dev/sdb5
```

```
ID_FS_USAGE=crypto
```

```
ID_FS_TYPE=crypto_LUKS
```

```
ID_FS_VERSION=2
```

```
ID_FS_UUID=09330b5a-5659-4efd-8e9d-0abc404c5162  
ID_FS_UUID_ENC=09330b5a-5659-4efd-8e9d-0abc404c5162  
ID_FS_LABEL=  
ID_FS_LABEL_ENC=  
ID_FS_LABEL_SAFE=
```

Make a note of the `ID_FS_UUID` value which is in italics above. We will need it later. Note: your output will be different than mine.

Now time to configure our newly installed system. The first thing we have to do is make the newly installed system active so we can make changes to it. We do that by mounting the partitions and chrooting to it.

```
mkdir /mnt/backtrack4  
mount /dev/mapper/vg-root /mnt/backtrack4  
mount /dev/sdb1 /mnt/backtrack4/boot  
chroot /mnt/backtrack4  
mount -t proc proc /proc  
mount -t sysfs sys /sys
```

To make everything truly operational, we can mount `/dev/pts`, but every time I try I have problems unless I reboot first. That is a real pain, so I just don't mount `/dev/pts`. We will get a couple warnings/errors as we go along, but they do not affect our install.

The magic to making all this work is to rebuild the `initrd` image that is used to boot our system. We need to include some things, load some modules, and tell it to open the encrypted volume, but first we have to go through the whole process of installing software again. We have to do this because we are essentially right back where we started when we booted the live cd. Do the following again.

```
apt-get update  
apt-get install hashalot lvm2
```

The next step is to configure how `initramfs-tools` will create our `initrd` file.

There are two ways to do this, an easy way and a slightly harder way.

The Easy Way

The easy way involves editing two files, the `/etc/crypttab` file and the `/etc/fstab` file. I use the `vi` editor, but you can use your favorite editor.

```
vi /etc/crypttab
```

We need to add the following line to the file. If you are new to `vi`, hit the `o` key and then type the following:

```
pvcrypt    /dev/disk/by-uuid/<uuid from above>    none    luks
```

When you are done typing that line, hit the `esc` key and then type `':wq'` without the quotes to save and exit `vi`. The file should look like this. The `uuid` is unique to my case. Make sure yours matches your system.

```
# <target device>    <source device>    <key file>    <options>  
pvcrypt    /dev/disk/by-uuid/09330b5a-5659-4efd-8e9d-0abc404c5162    none    luks
```

Then we need to edit the `/etc/fstab` file and change the device name for `root`. We will also change the mount type at this time to `ext2` so we can reduce the number of writes to our USB drive.

Again, use your favorite editor or vi.

vi /etc/fstab

The file will look something like below. The UUIDs will be different though.

```
# /etc/fstab: static file system information.
#
# <file system> <mount point> <type> <options> <dump> <pass>
proc /proc proc defaults 0 0
# /dev/mapper/vg-root
UUID=c8d9b9a0-2198-4966-bc3a-39259df6a2c2 / ext3 relatime,errors=remount-ro 0 1
# /dev/sdb1
UUID=6af425ad-99b8-44a5-9ee1-0349141f9b1f /boot ext3 relatime 0 2
/dev/hdc /media/cdrom0 udf,iso9660 user,noauto,exec,utf8 0 0
/dev/fd0 /media/floppy0 auto rw,user,noauto,exec,utf8 0 0
```

The only line we need to change is the line for vg-root which is bolded above. For those new to vi, position the cursor on first 'U' of the line using your arrow keys and type 'dd', then move the cursor to the '#' in the line above and type the letter o, then type the line below, hit the esc key and type ':wq' without the quotes to save the file. The line needs to look like below when done:

```
/dev/mapper/vg-root / ext2 defaults 0 1
```

Once that is done, execute the following commands.

update-initramfs -u

If all goes well, you are now ready to cross your fingers and reboot. The system will start to boot then ask you for your LUKS passphrase. Type that bad boy in and, if all goes well, your system will boot.

If you have problems, you can use the troubleshooting directions below to get back to the state where you can try to figure out how what went wrong.

System All Booted

Once you have a booting system, you are ready to login. The default userid is **root** and the default password is **toor**. You are now ready to login and being playing. Don't forget to change the root password as soon as you login the first time.

That's it. You can make some final tweaks if you want like setting the network to start automatically and starting KDE at boot, but for all intents and purposes you have successfully installed Backtrack 4 to USB drive and don't have to worry about sensitive information being intercepted if it gets lost or stolen.

Slightly Harder Way

This method involves adding two scripts and editing the modules file. I have added the text of the scripts here, but also provided a command that will grab them from my website.

The first script we need to create is /etc/initramfs-tools/hooks/pvcrypt. This script will copy the needed files for the initrd image. Executing the following will get the script where it needs to be.

```
cd /etc/initramfs-tools/hooks
wget -O pvcrypt http://www.infosecramblings.com/hooks-pvcrypt
```

The contents of the script should look like this.

```

PREREQ=""

prereqs()
{
    echo "$PREREQ"
}

case $1 in
prereqs)
    prereqs
    exit 0
    ;;
esac

if [ ! -x /sbin/cryptsetup ]; then
    exit 0
fi

. /usr/share/initramfs-tools/hook-functions

mkdir -p ${DESTDIR}/etc/console-setup
cp /etc/console-setup/boottime.kmap.gz ${DESTDIR}/etc/console
copy_exec /bin/loadkeys /bin
copy_exec /bin/chvt /bin
copy_exec /sbin/cryptsetup /sbin
copy_exec /sbin/vol_id /sbin

```

The next script we need to create is `/etc/initramfs-tools/scripts/local-top/pvcrypt`. This script tells the system to open the encrypted volume and requests the passphrase. Executing the following will get the script where it needs to be.

```

cd /etc/initramfs-tools/scripts/local-top
wget -O pvcrypt http://www.infosecramblings.com/local-top-pvcrypt

```

Unlike the first script, you will need to edit this script to point to your encrypted volume. This is where the UUID we found earlier comes in. Replace the word `UUID` with the value you noted above.

```

PREREQ="udev"

prereqs()
{
    echo "$PREREQ"
}

case $1 in
# get pre-requisites
prereqs)
    prereqs
    exit 0
    ;;
esac

/bin/loadkeys -q /etc/console-setup/boottime.kmap.gz
modprobe -Qb dm_crypt
modprobe -Qb sha256
modprobe -Qb aes_i586
modprobe -Qb xts

# The following command will ensure that the kernel is aware of
# the partition before we attempt to open it with cryptsetup.
/sbin/udevadm settle

sleep 10

if grep -q splash /proc/cmdline; then

```

```
/bin/chvt 1
fi
/sbin/cryptsetup luksOpen /dev/disk/by-uuid/UUID pvcrypt
```

Both scripts need to be executable.

```
chmod +x /etc/initramfs-tools/hooks/pvcrypt
chmod +x /etc/initramfs-tools/scripts/local-top/pvcrypt
```

The final change we need to make before rebuilding initrd is to edit the `/etc/initramfs-tools/modules` file and add a couple encryption modules. This will make sure they are copied into the initrd image. We can do this one of two ways. We can use our favorite editor and add the following lines to the bottom of the file and save it.

```
aes_i586
xts
```

or use a `wget` command like above.

```
cd /etc/initramfs-tools
wget -O modules http://www.infosecramblings.com/initramfs-modules
```

Either way, your `/etc/initramfs-tools/modules` file should look like this:

```
# List of modules that you want to include in your initramfs.
#
# Syntax:  module_name [args ...]
#
# You must run update-initramfs(8) to effect this change.
#
# Examples:
#
# raid1
# sd_mod
fbcon
vesafb
aes_i586
xts
```

Now it's time to rebuild our initrd image.

```
update-initramfs -u
```

If all goes well, you are now ready to cross your fingers and reboot. The system will start to boot then ask you for your LUKS passphrase. Type that bad boy in and, if all goes well, your system will boot.

System All Booted

Once you have a booting system, you are ready to login. The default userid is **root** and the default password is **toor**. You are now ready to login and being playing. Don't forget to change the root password as soon as you login the first time.

That's it. You can make some final tweaks if you want like setting the network to start automatically and starting KDE at boot, but for all intents and purposes you have successfully installed Backtrack 4 to USB drive and don't have to worry about sensitive information being intercepted if it gets lost or stolen.

Troubleshooting

If you run into any problems, you don't have to start over. As long as your encrypted volume is built correctly and you have the correct LUKS passphrase, you can get back to the place you were with the Live CD. Simply boot with the original Live CD/USB drive and enter the following.

```
/etc/init.d/networking start  
apt-get update  
apt-get instal hashalot lvm2  
cryptsetup luksOpen /dev/[your logical partition] pvcrypt  
mkdir /mnt/backtrack4  
mount /dev/mapper/vg-root /mnt/backtrack4  
mount /dev/[boot partition] /mnt/backtrack4/boot  
chroot /mnt/backtrack4  
mount -t proc proc /proc  
mount -t sysfs sys /sys  
mount -t devpts devpts /dev/pts
```

You can now do any trouble shooting you need to do and try to reboot again. One note, if you want to check the UUID of your partition, do it before you chroot.

-Kevin



Backtrack 4 – Bootable USB Thumb Drive with “Full” Disk Encryption by [Kevin Riggins](#) is licensed under a [Creative Commons Attribution-NonCommercial-Share Alike 3.0 United States License](#). Permissions beyond the scope of this license may be available at <http://www.infosecramblings.com/about/>.