

Introduction to Helix

Helix is a self-contained bootable Linux CD that provides a known-good platform from which to use a wide range of security-related tools. Cornell's Security Team has significantly improved and expanded the operation of Helix and the toolset that it includes, and has made it more user-friendly for non-Linux users. This handout covers getting started and using some of the more common tools, and is designed to accompany the "Introduction to Helix" presentation. Topics include:

- Booting
- The Desktop
- Finding and mounting drives
- Network configuration
- Virus scanning
- Capturing disk images
- Wiping disks to DoD standards
- Acquiring local user names
- Acquiring NTLM hashes out of SAM files
- Reading and Writing NTFS filesystems
- Viewing registry files

Getting started

Booting on the Helix CD will result in a GUI desktop environment. On the left are status monitors. To access a menu of options, right-click anywhere on the desktop. Menu options include mounting drives, setting up the network, updating antivirus definitions, running applications, and opening terminal windows for command line tools. Note that Helix loads much of its functionality into a ramdisk to speed operation.

Sometimes Helix may not recognize certain hardware. In this case, it may be helpful to use the boot options screen. At the pre-boot splash screen, the word "boot:" will appear briefly. Press F2 at that prompt to enter the boot options screen. F3 produces additional options, and F1 returns to the splash screen. At any time, type the desired option and press Enter to boot. Options include, among other things:

- Booting with the newer Linux 2.6.7 kernel, which includes newer drivers
- Using framebuffer mode, which may be useful with laptops
- Custom screen resolution and color depth
- Booting without dma acceleration
- Turning off hardware detection
- Booting in text mode

Note that the alternate language and alternate desktop options are not functional, as alternate language and desktops have been removed in order to use the space for additional tools.

Configuring the network

Helix will automatically configure the network if dhcp is available. If dhcp is not available, most of the configuration can be done from the desktop menu. Right-click the background and choose "Configure network interface." Fill in the three blanks with the desired IP, netmask and router, and click "OK".

Setting the DNS in a static environment is a little trickier:

1. Rightclick the background
2. Choose "Root Shell – Logged"
3. Execute the command "xedit" in the shell to open a basic text editor (for advanced users - vi and nedit are also available.)
4. Click the "Load" button in xedit
5. Open "/etc/resolv.conf" by entering the filename to the right of the Load button and pressing Enter
6. Add the line: nameserver 132.236.56.250

7. Additional nameserver lines can be added, one per line if desired
8. Click the "Save" button
9. Click the "Quit" button

Using the computer's native hard drives

Helix can read and write to Unix/Linux ext2/ext3 and Windows FAT/FAT32/NTFS file systems as well as many other common and uncommon file systems (see below.) In Unix, drives are "mounted" into the file system in order to read them. Unlike Knoppix, a popular bootable Linux, Helix does not automatically mount drives, but they can be mounted through the desktop menu. Simply right-click the background, and choose "mount local disk volumes" from the menu.

Since Helix is, first and foremost, a platform used for forensics of all kinds, it is important to be able to inspect data while preserving it untouched. Therefore Helix deliberately makes it difficult to accidentally change the contents of the host system. *All drives are mounted read-only using the above method.* Drives must be mounted read/write manually as described below.

Helix can recognize the following file systems:

- NTFS - all variants
- SFS - NTFS on hardware RAID systems
- UFS - UNIX filesystem; BSD, Solaris, etc.
- HFS/HFS+ - Apple
- ext2/ext3 - Common Linux filesystems
- reiserfs - another Linux HP filesystem
- XFS - Linux filesystem descended from the IRIX creation
- JFS - Linux journaling filesystem
- loop-crypto and loop-AES - loopback cryptographic filesystems, if keys are available

To list the available drives in the system and their partitioning scheme, including all recognized partitions, use **fdisk -l**. Use **lsusb** to list attached USB devices including external hard drives and thumb drives.

Helix, like other Unix/Linux variants, refers to drives in a seemingly cryptic but consistent format:

- `/dev/hda`, `/dev/sda`, `/dev/hdb`, etc - raw drive devices.
- `/dev/hda1`, `/dev/hda2`, `/dev/hdb1`, etc - logical volumes
 - `/dev/hda#` - first IDE drive
 - `/dev/hdb#` - 2nd
 - `/dev/hdc#` - 3rd, etc

 - `/dev/sda#` - 1st SCSI device
 - `/dev/sdb#` - 2nd SCSI device

 - `/dev/fd0` - floppy drive

Linux maps both CDROM devices and USB/Firewire devices as SCSI devices. They'll show up as `/dev/sd##`

Although Helix will mount native drives on command using the method described above, it is important to know how to mount drives manually in the event that drives need to be mounted read/write, as opposed to read-only. Several examples of the mount command follow:

The basic read-only (ro) mount command, mounts partition `/dev/hda1` into the mount point (directory) `/mnt/hda1`:

```
mount -o ro /dev/hda1 /mnt/hda1
```

A filesystem type can be specified if Helix has a problem correctly determining the filesystem:

```
mount -o ro -t vfat /dev/hda1 /mnt/hda1
```

Mounting a partition read-write:

```
mount -o rw /dev/hda1 /mnt/hda1
```

Mounting USB drives:

1. **lsusb** - is it there?
2. **fdisk -l** - look for /dev/sd##
3. <wait for automounter to create a mount point>
4. **mount -o rw -t vfat /dev/sda1 /mnt/sda1** (assuming drive is at sda1)

Virus Scanning:

There are three different virus scanners on Helix:

ClamAV has signatures for, among other things, some interesting quasi-threats like HTML tricks, phishing, and javascript exploits, and as such is useful versus IE caches, Eudora folders, and the like. Signature updates are extremely fast when new nasties appear, but Clam lags behind in dealing with obfuscated executables such as HackerDefender. On the upside, it's able to unpack nearly any archive format that ever existed.

BitDefender is a commercial product and by far the most fully featured and capable.

F-Prot is 1990s technology, and requires a full replacement to get new signatures. It includes some powerful heuristic functionality that makes it useful under rare circumstances, but it almost never finds recent conventional nasties.

The first step in using these products is to update their virus definitions files, since obviously the definitions will only be as new as the version of Helix that is in use.

freshclam updates ClamAV

/opt/bdc/bdc -update updates BitDefender

F-Prot is used mainly for heuristics, not for definitions, and so would not normally be updated. To update it however, would require downloading and installing the latest version on the Helix ramdisk.

To virus scan a drive with BitDefender, assuming you've got the drive mounted under /mnt/hda1:

```
/opt/bdc/bdc /mnt/hda1/*
```

To virus scan a drive with ClamAV, assuming you've got the drive mounted under /mnt/hda1: **clamscan -r -i /mnt/hda1/***

Note that some malware uses packed executables and may encrypt executables with Morphine, an executable file encryptor. BitDefender reports packed executables based on the methodology used, generally with a '(UpX)' after the filename. These bear careful examination. Clamscan reports something similar, and will report 'invalid PE header' when it stumbles across obfuscation methods. It's accuracy versus Morphine is currently spotty, but anything that gets an 'invalid PE header' report ought to be looked at closely.

Although Clam and BitDefender both have "removal" options, it is strongly recommended that virus files be removed manually, using "captive" described later in this document.

Capturing disk images

The server has been compromised and the dean wants the system back online pronto. You however, want to know how the compromise occurred, by whom, and if any data was touched. Using the system itself to copy its own drives will cause the system files to be modified. In order to take a snapshot of the

system exactly as it is and copy it to another drive for later inspection, use Helix and either the “grab” application, or the “dd” command. We’ll look at both of these. Although grab may seem more convenient, it is simply a front-end for the dd process, and it is important to understand what it is doing under the hood. First the command line...

Using dd

First, locate the drive with fdisk -l as described above. You do not need to mount the source drive, however, you will need to mount the destination drive read/write. Note - you can image either the logical partition or the raw drive. Imaging the logical partition is simpler when it comes to later inspection, and is generally “good enough” for most applications. However, for advanced users, imaging the raw drive will get all the partition tables, master boot record, etc., in which small amounts of data can hide. Imaging the logical volume omits that info, but it does get all the slack, unallocated space, etc.

You will need to know the true sector size of the drive/partition. For all practical purposes, we can assume we’re dealing with 512-byte sectors on any conventional hard disk we’re likely to encounter. Note however, that there are ways of finding out the true sector size, which is particularly valuable when dealing with floppies, where varying such things is fairly easy.

To dd to an external USB drive where:

/dev/hda1 is the source on the compromised box

/dev/sda1 is an attached USB drive

Standard flags include “if” (input file), “of” (output file), and “bs” (block size)

dd if=/dev/hda1 of=/mnt/sda1/filename_hda1_dd_512 bs=512

(to dd a floppy to a file: 'dd if=/dev/fd0 of=some_file_dd_bs1024 bs=1024')

For advanced users, the dd Command of Champions™:

(This assumes that there is a previously prepared remote host with capacity to store the dd image. Note that the file naming convention here includes the originating host, the imaging tool, the logical volume, and the blocksize.)

dd if=/dev/hda1 bs=512 | ssh -l user some.host.cornell.edu "(cd /opt/dropoff;dd of=name.of.host.hda1_dd_512)"

md5sum is tool that produces a unique hash number for any given input. It is an excellent way to verify that two files are identical. Hence, the following two hashes should match:

md5sum /dev/hda1

md5sum /mnt/sda1/filename.dd.512

Using grab

Helix includes a GUI tool called 'grab' that will do all this for you, including sending images over an encrypted connection, comparing hashes, etc. To use it:

1. Open a root terminal and type “grab”
2. Click the Source menu at the top and choose the drive you wish to image
3. Click the Dest menu at the top and choose the drive you wish to put the image on
4. Choose other options as you wish, such as compression and verification
5. Click Start

dd/grab images can also be used (though with a large performance hit) as though they were actual mountable volumes, and can be virus-scanned, and otherwise inspected.

Restoring from an image so that it will boot

Assuming that /dev/hda1 is the new target drive and the dd image exists on an external USB drive on partition /dev/sda1 at mount point /mnt/sda1.

First, it will be necessary to establish a bootable first partition on the drive of the proper type for the

filesystem you're restoring. Do this with:
fdisk /dev/hda1

Now set up a loopback mounting for dd images of logical volumes:

```
mkdir /mnt/dd  
mount -o loop,ro /path/to/image.dd /mnt/dd
```

Note for advanced users - it is possible to loopback mount logical volumes from within dd images of raw devices, but it requires a little screwing around with start/end sector offsets from the start of the file so the loop device can find the volume correctly.

Restore the image:

```
dd if=/mnt/sda1/some_dd_image.blah.blah.blah of=/dev/hda1 bs=512
```

Now boot up on the restored drive!

Wiping drives securely

Use Helix to completely and permanently erase either a logical volume or an entire drive with a full 7-pass, zeroes/ones/random/verify DoD wipe. If you wipe the entire drive, you will need to repartition it before it will be useful again. Wipe works in such a way that the zeroing pass happens last, which leaves little evidence that a wipe has occurred.

Assume /dev/hda1 is the drive to be wiped.

```
wipe -k /dev/hda1
```

Another option, that will completely blast all data off the drive and render the partition itself unrecognizable is as follows:

1. **fdisk -l** - figure out the number of sectors on the drive
2. **dd if=/dev/zero of=/dev/hda1 bs=512 count=###**

Grabbing NTLM hashes out of SAM files/checking for local users

Windows NT and above store password hashes that can be used by a password cracking program to determine passwords. The following section will demonstrate a couple of ways to do this, and also how to produce a list of local users from the SAM (the file that holds the user names and password hashes.) Note that if the computer is a member of a domain, domain accounts and hashes will not appear in the list.

First off, mount the drive that had Windows on it if it isn't already mounted:

```
mount -o ro -t ntfs /dev/hda1 /mnt/hda1
```

The SAM lives in %WINDIR%/system32/config. Linux will randomly interpret Windows filename case through some obscure mechanism, so if 'system32' isn't there, try 'SYSTEM32':

```
cd /mnt/hda1/WINDOWS/system32/config
```

Helix has three utilities that can dump hashes and local users: chntpw, bkhive, and samdump2. Use any of them, depending on your objective. A note about SYSKEY – Windows 2000/sp4 and later use a key called SYSKEY (which uses RC4 encryption) to encrypt the hashes stored in the SAM. As the system has to recover the key at startup, it's stored in bits and pieces in a file called SECURITY. It is possible to configure the system to look for the SYSKEY key on removable media at boot time, but almost no one uses that feature.

To list users:

```
chntpw -l SAM
```

To list users and dump hashes, there are two options. bkhive finds, reconstructs, and dumps the SYSKEY. samdump2 does that and also does the user and hash dump:
samdump2 SAM SECURITY

If you just want the SYSKEY:
bkhive SECURITY /tmp/key.out

Note that samdump2 can use key.out as well as the SECURITY file:
samdump2 SAM /tmp/key.out

You can dump the output of samdump2 to a file. That file is exactly the same format used by the pwdump/smbpasswd utilities, which are used to dump the password hashes. The first hash is the LANMAN one (case insensitive) and the 2nd is the NT one (case sensitive). For local logins we only care about the 2nd one.

If you want to crack the dumped output, John the Ripper is in our most recent Helix build and speaks pwdump file. The whole process would look like this:

1. **samdump2 SAM SECURITY /tmp/passwd**
2. **mkdir /tmp/run**
3. **cp /opt/john-1.6.36/run/* /tmp/run**
4. **cd /tmp/run**
5. **./john -format=NT /tmp/passwd**

Press <Enter> every now and then to show John's progress.

Reading and Writing NTFS filesystems

In order to reset passwords, clean virus infections, edit (or even view) the registry, or do any other activity that requires read/write access, the drives need to be mounted read/write.

The latest, greatest thinking in NTFS read/write is the Captive project. Linux kernel 2.6 has experimental NTFS read/write support but there are two problems: First, Helix modifies its two kernels specifically not to write to drives except under cumbersome circumstances and second, Linux lags far behind in being able to comfortably write to NTFS partitions.

Captive uses the WINE Windows emulation infrastructure under Linux to run enough NT kernel to write NTFS safely. It uses real Microsoft NTFS drivers and takes precautions (chroots/setuid to user 'captive') to keep Windows code from causing problems with the local Linux image, in this case, Helix.

First off, find the NTFS drivers as shown below. Do ***not*** have the local Windows machine mounted at this point, lest you find potentially compromised drivers:

captive-install-acquire

Click "Next ->" and ignore errors until you get a valid "OK" button. Click that and you're done. (The errors are caused by the fact that the software wants to do a quasi-WindowsUpdate to find newer drivers, even though it doesn't need them.)

Mount the partition read/write:

```
mount -t captive-ntfs -o rw /dev/hda1 /mnt/hda1
```

A message will appear about Captive 1.5 while it launches apps from the CD, which can be ignored. Captive isn't necessarily stable, so don't expect too much here. Get in, clean up, get out. Use "**mount / -o remount /mnt/hda1**" to remount the drive if Captive loses track of itself.

Once you have the partition mounted read/write, you can reset SAM passwords with chntpw, fix virus problems, edit the registry, etc.

It is imperative that the filesystem be properly unmounted prior to rebooting, as Captive doesn't sync when init sends it the dreaded TERM signal. **Failure to properly unmount will more than likely corrupt the partition beyond repair**.

umount /mnt/hda1

Viewing registry files

Both chntpw and regviewer can dump registry entries. regviewer is a graphical application, but it doesn't know how to reassemble the two dozen odd hive locations that constitute the modern NT and above "registry". You'll have to pick through these piecemeal with regviewer. However, regviewer is useful for looking at the SAM for security settings, or hunting through user NTUSER.DAT files. regviewer requires a writable file system even if its just reading the registry, so be sure to use Captive as described above before proceeding. Let's assume the captive Windows host is mounted off /mnt/hda1:

1. **cd /mnt/hda1/Documents\ and \Settings/wm63**
2. **cp NTUSER.DAT /tmp**
3. **cd /tmp**
4. **regviewer &**

Useful Linux commands

For those not familiar with a Unix-like environment, the following commands, not found elsewhere in this document, will likely prove useful. Remember that there are MANY variations on these commands. In order to learn more about any command, type "man" and the command, as in "man grep". "man" stands for "manual".

- **cat** dumps a file to the screen, analogous to "type" in dos
- **cp** copies files/directories (-r recurses directories)
- **df** displays information about all mounted file systems
- **diff** displays differences between two files, as in "diff file1 file2"
- **du** displays disk usage for directories
- **fls** lists file info, (including deleted) from /dev/<device> or from dd image
- **grep** filter, analogous to "find" in dos
- **kill** kills a process, using a process id (PID) found using ps
- **less** displays a file one line/page at a time, arrows scroll by line, space by page
- **ls -alt** lists files, including hidden ones (a) with more info (l), by time (t)
- **ls -la** lists files, including hidden ones (a) with more info (l)
- **ls** lists files, analogous to "dir" in dos
- **mactime** translates from machine time to normal time
- **mkdir** creates a directory
- **mv** moves files/directories (-r recurses directories)
- **ps -aux** lists running processes
- **rm** removes a file (-r recurses directories)
- **rmdir** removes a directory
- **sort** displays and sorts file contents, as in "sort myfile.txt"
- **wc** counts lines in a file

Linux (and Windows as well) can "pipe" one command to another, using the "|" symbol. This allows one program's output to be used as another program's input. Hence, "cat myfile.txt|grep jpg|wc -l" will count the number of lines in myfile.txt in which the letters "jpg" appear.