

[About](#) [Archives](#) [Home](#)

Search for:

Search

August 2010

M	T	W	T	F	S	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

« May

Links & Feeds

Subscribe in a reader

Entries RSS Feed

Entries Atom Feed

Follow me on Twitter

Share / Save

Blogroll

Ben Rockwood
 Chad Sakac
 Duncan Epping
 Eric Siebert
 Eric Sloof
 Forbes Guthrie
 Jason Bosche
 Ken Cline
 Ryan Arneson
 Scott Lowe
 Stu Radnidge
vmprofessional.com

Tags

aggregate automated
 automation backup best
 practice clone clones comstar
 cron Dedup design disaster
 esx fc fcoe features gateway
 howto ISCSI iser multi NFS
 opensolaris
 performance presentations
 protocol provision recovery
 replication reprovision script
 scripts Security
 snapshot Storage
 summit sun syslinux target
 undelete vexperts vmfs
 VMware vSphere
 zfs

Stats

Cluster Map



« Sun's S7000 Storage Systems with Fishworks is Really Awesome

Multi Protocol Storage Provisioning with COMSTAR »

vyatta
 NETWORK VIRTUALIZATION

**vROUTER
vFIREWALL**
 SECURE VMware, XenServer & Hyper-V

FREE DOWNLOAD

www.vyatta.com

Ads by Google

Understanding VMFS volumes

Understanding VMFS volumes is an important element within VMware ESX environments. When storage issues surface we need to correctly evaluate the VMFS volume states and apply the appropriate corrective actions to remediate undesirable storage events. VMFS architecture is not publically available and this certainly adds to the challenge when we need to correct a volume configuration or change issue. So lets begin to look at the components of a VMFS from what I have been able to decrypt using direct analysis.

All VMFS volume partitions will have a partition ID value of fb. Running fdisk can identify any partitions that are flagged as VMFS as shown here.

```
[root@vhl ]# fdisk -lu /dev/sdc
```

```
Disk /dev/sdc: 274.8 GB, 274877889536 bytes
255 heads, 63 sectors/track, 33418 cylinders, total 536870878 sectors
Units = sectors of 1 * 512 = 512 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sdc1		128	536860169	268430021	fb	Unknown

What's important to note here is the sector size = 512 and the starting/ending blocks.

Many VMFS volume configuration elements are visible in the /vmfs mount folder. Within the directory the are two subdirectories, the volumes directory and the devices directory. The volumes directory provisions the mount point location and the devices directory holds configuration elements. Within the devices directory the are several subdirectories of which I can explain the disks and lvm folders, the others are not known to me outside of theory only.

A key part of a VMFS volume is it's **UUID** (aka Universally Unique Identifier) and as the name suggests it used to ensure uniqueness when more than one volume is in use. The UUID is generated on the initial ESX host that created the VMFS volume based on the UUID creation standards. You can determine which ESX host the initial VMFS volume was created on by referring to the last 6 bytes of the UUID. This value is the same as the last six bytes of the ESX host's system UUID found in the /etc/vmware/esx.conf file.

By far one of the most critical elements on a VMFS volume is the GUID. The GUID is integral within the volume because it is used to form the vml path (aka virtual multipath link). The GUID is stored within the VMFS volume header and begins at address 0x10002E.

The format of the GUID can vary based on different implementations of SCSI transport protocols but generally you will see some obvious length variances of the vml path identifiers which stem from the use of T11 and T10 Standard SCSI address formats like **EUI-64**, and **NAA-64**. Regardless of those variables there are components outside of the GUID within the vml that we should take notice of. The vml construct contains references to the LUN and partition values and these are useful to know about. The following illustrates where these elements appear in some real examples.

When we issue an ls -l from the /vmfs/devices/disks directory the following info is observed.

```
vhbba#:Target:LUN:Partition -> vml:??_LUN_??_GUID:Partition
```

	LUN ^	GUID ^	PARTITION ^
vmhba0:1:0:0 ->	vml.0200000000	5005076719d163d844544e313436	
vmhba0:1:0:1 ->	vml.0200000000	5005076719d163d844544e313436:1	
vmhba32:1:3:0 ->	vml.0200030000	600144f07ed404000000496ff8cd0003434f4d535441	
vmhba32:1:3:1 ->	vml.0200030000	600144f07ed404000000496ff8cd0003434f4d535441:1	

As well the issuing ls -l on the /vmfs/volumes list the VMFS UUID's and the link name which is what we see displayed in the GUI client. In this example we will follow the UUID shown in blue and the named ss2-cstar-zs0.2 volume.

```
ss2-cstar-zs0.2 -> 49716cd8-ebcbbf9a-6792-000d60d46e2e
```

Additionally we can use esxcfg-vmkbadevs -m to list the vmhba, dev and UUID associations.

```
[root@vhl ]# esxcfg-vmkbadevs -m
vmhba0:1:0:1 /dev/sdd1 48a3b0f3-736b896e-af8f-00025567144e
vmhba32:1:3:1 /dev/sdf1 49716cd8-ebcbbf9a-6792-
```

000d60d46e2e

As you can see we indeed have different GUID lengths in this example. We also can see that the vmhba device is linked to a vml construct and this is how the kernel defines paths to a visible SCSI LUN. The vml path hosts the LUN ID, GUID and partition number information and this is also stored in the volumes VMFS header. As well the header contains a UUID signature but this is not the VMFS UUID.

If we use hexdump as illustrated below we can see these elements in the VMFS header directly.

```
[root@vhl root]# hexdump -C -s 0x100000 -n 800 /dev/sdf1
00100000 0d d0 01 c0 03 00 00 00 10 00 00 00 02 16 03 00 |          | <-
LUN ID
00100010 00 06 53 55 4e 20 20 20 20 20 43 4f 4d 53 54 41 | SUN      COMSTA| <-
Target Label
00100020 52 20 20 20 20 20 20 20 20 20 31 2e 30 20 60 01 |R        1.0 ` | <-
LUN GUID
00100030 44 f0 7e d4 04 00 00 00 49 6f f8 cd 00 03 43 4f |D ~      Io    CO|
00100040 4d 53 54 41 00 00 00 00 00 00 00 00 00 00 00 00 |MSTA     |
00100050 00 00 00 00 00 00 00 00 00 00 02 00 00 00 fc |          | <-
Volume Size
00100060 e9 ff 18 00 00 00 01 00 00 00 8f 01 00 00 8e 01 |          |
00100070 00 00 91 01 00 00 00 00 00 00 00 00 10 01 00 00 |          |
00100080 00 00 d8 6c 71 49 b0 aa 97 9b 6c 2f 00 0d 60 d4 | lqI      l/    `|
00100090 6e 2e 6e 89 19 fb a6 60 04 00 a7 ce 20 fb a6 60 |n n      `     `|
001000a0 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |          |
001000b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |          |
*
00100200 00 00 00 f0 18 00 00 00 90 01 00 00 00 00 00 00 |          |
00100210 01 00 00 00 34 39 37 31 36 63 64 38 2d 36 30 37 | 49716cd8-607| <-
SEG UUID in ASCII
00100220 35 38 39 39 61 2d 61 64 31 63 2d 30 30 30 64 36 |5899a-ad1c-000d6|
00100230 30 64 34 36 65 32 65 00 00 00 00 00 00 00 00 00 |0d46e2e     |
00100240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |          |
00100250 00 00 00 00 d8 6c 71 49 9a 89 75 60 1c ad 00 0d |          lqI  u` | <-
SEG UUID
00100260 60 d4 6e 2e 01 00 00 00 e1 9c 19 fb a6 60 04 00 |` n      `     |
00100270 00 00 00 00 8f 01 00 00 00 00 00 00 00 00 00 00 |          |
00100280 8e 01 00 00 00 00 00 00 64 cc 20 fb a6 60 04 00 |          d     `|
00100290 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |          |
001002a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |          |
```

In addition to the VMFS header block we have the hidden metadata files of the volume which you can list using `ls -al`. The `vh.sf` contains the UUID of the VMFS store and any member segments info. (I would presume the name `vh` stands for Volume Header ... ;D)

```
[root@vhl]# hexdump -C -s 0x200000 -n 256 /vmfs/volumes/49716cd8-ebcbbf9a-6792-000d60d46e2e/.vh.sf
00200000 5e f1 ab 2f 04 00 00 00 1f d8 6c 71 49 9a bf cb |^ / lqI      | <-
VMFS UUID
00200010 eb 92 67 00 0d 60 d4 6e 2e 02 00 00 00 73 73 32 | g ` n      ss2| <-
Volume Name
00200020 2d 63 73 74 61 72 2d 7a 73 30 2e 32 00 00 00 00 |-cstar-zs0.2  |
00200030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |          |
*
00200090 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 |          |
002000a0 00 00 00 10 00 00 00 00 00 d8 6c 71 49 01 00 00 |          lqI  |
002000b0 00 d8 6c 71 49 9a 89 75 60 1c ad 00 0d 60 d4 6e | lqI  u` ` n| <-
SEG UUID
002000c0 2e 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |          |
002000d0 00 00 00 01 00 20 00 00 00 00 00 01 00 00 00 00 |          |
002000e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |          |
```

And of course we can not leave out the partition entry block data for the device.

`hexdump -C -n 256 /dev/sdf1`

```
00000000 fa b8 00 10 8e d0 bc 00 b0 b8 00 00 8e d8 8e c0 |          |
00000010 fb be 00 7c bf 00 06 b9 00 02 f3 a4 ea 21 06 00 |          ! |
00000020 00 be be 07 38 04 75 0b 83 c6 10 81 fe fe 07 75 | 8 u      u |
00000030 f3 eb 16 b4 02 b0 01 bb 00 7c b2 80 8a 74 01 8b |          | t |
00000040 4c 02 cd 13 ea 00 7c 00 00 eb fe 00 00 00 00 00 |L         |
00000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |          |
*
000001b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 |          |
000001c0 03 00 fb fe ff ff 80 00 00 00 72 ef bf 5d 00 00 |          r ] |
Type Start End
000001d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |          |
```

With this detailed information it is possible to solve some common security issues with VMware stores like volume deletion and unintentional LUN ID changes.

Recently VMware added a some what useful command line tool named `vmfs-undelete` which exports metadata to a recovery log file which can restore `vmrk` block addresses in the event of deletion. It's a simple tool and at present it's experimental and unsupported and is not available on ESXi. The tool of course demands that you were proactive and ran it's backup function in order to use it. Well I think this falls well short of what we need here. What if you have no previous backups of the VMFS configuration, so we really need to know what to look for and how to correct it and that's exactly why I created this blog.

The volume deletion event is quite easy to fix and thats simply because the VMFS volume header is not actually deleted. The partition block data is what gets trashed and you can just about get way with murder when it comes to recreating that part. Within the first 128 sectors is the peice we need to fix. One method is to create a

store with the same storage volume and then block copy the partition to a file which can be block copied to the deleted device partition data blocks and this will fix the issue.

For example we create a new VMFS store on the same storage backing with the same LUN size as the original and it shows up as a LUN with a device name of /dev/sdd we can use `esxcfg-vmhbadevs -m` to find it if required

The deleted device name was /dev/sdc

We use the `dd` command to do a block copy from the new partition to a file or even directly in this case.

Remember to back it up first!

```
dd if=/dev/sdc of=/var/log/part-backup-sdc-1st.hex bs=512 count=1
```

then issue

```
dd if=/dev/sdd of=/dev/sdc bs=512 count=1
```

or

```
dd if=/dev/sdd of=/var/log/part-backup-sdd.hex bs=512 count=1
```

```
dd if=/var/log/part-backup-sdd.hex of=/dev/sdc bs=512 count=1
```

I personally like using a file to perform this function as this becomes a future backup element which you can move to a safe location. The file can actually be edited with other utilities to provide more flexibility. e.g. `hexedit` etc. Additionally you could use `fdisk` to directly edit the partition table and provide the correct start and end addresses. This is something you should only do if you are well versed in it's usage.

As as an additional level of protection we could even include making backups of the `vh.sf` metadata file and the VMFS header:

```
cp /vmfs/volumes/49716cd8-ebcbbf9a-6792-000d60d46e2e/.vh.sf /var/log/vh.sf.bu
```

```
dd if=/dev/sdc of=/var/log/vmfsheader-bu-sdc.hex bs=512 count=4096
```

This would grant the ability for support to examine the exact details of the VMFS configuration and potentially allow recovery from more complex issues.

One of the most annoying security events is when a VMFS LUN get's changed inadvertently. If a VMFS volume LUN ID changes and is presented to an ESX host then the presented volume will be treated as a potential snapshot LUN. If this event occurs and the ESX servers advanced LVM parameter settings are at default the ESX host will not mount the volume. This behaviour is to prevent the possibility of corruption and downing the host since it can not determine which VM metadata inventory is correct.


If you are aware that the LUN ID has changed then the best course of action is to re-establish the correct LUN ID at the storage server first and rescan the affected vmhba's. This is important because if you need to resignature the VMFS volume it will also require that the VMs be imported back into inventory. Virtual Center logging and other various settings will be lost when this action is performed. This is a result of now having an incorrect UUID between the metadata, mount location and the `vmx` file UUID value.

If the storage change cannot be reverted back then a VMFS resignature method is the only option for reprovisioning a VMFS volume mount.

This is invoked by setting the `LVM.DisallowSnapshotLun = 0` and `LVM.EnableResignature = 1` and these should be reverted back once the VMFS resignature operation is complete.

Regards,

Mike

[Share / Save](#) 

Tags: [deleted](#), [esx](#), [esxi](#), [fdisk](#), [guid](#), [header](#), [hex](#), [lun](#), [lvm](#), [partition](#), [path](#), [recover](#), [resignature](#), [sector](#), [Storage](#), [uuid](#), [vml](#), [VMware](#), [volume](#)

This entry was posted on Wednesday, January 21st, 2009 at 5:16 PM and is filed under [Security](#), [Storage](#), [VMware](#). You can follow any responses to this entry through the [RSS 2.0](#) feed. You can [leave a response](#), or [trackback](#) from your own site.

Site Contents: © 2009 [Mike La Spina](#)

10 Comments

Dai Nan says:

August 15, 2009 at 10:01 AM



Hi , I want to know is there anyway to restore vmk file ? i delete the whole vm files using VC client , no vmk-flat and nothing else, Is it possible to restore these files someway ? thank you very much

Mike La Spina says:

August 15, 2009 at 1:33 PM



Hi,

I'm sorry.

You can only restore from a backup or use `vmfs-undelete` and both methods require a proactive function prior to a delete call.

There are currently no tools that I am aware of which can recover the metadata and block allocation info after the security event occurs without some proactive steps prior to the incidents occurrence.

Regards,

Mike

Arno says:



September 26, 2009 at 5:56 AM

Hi Mike, thanks for the info, it's great. I have a question about the distributed file system nature of VMFS. In a VI environment, do you know if when a LUN gets detected as a snapshot for any reason by a host, say with no VMs running on it, is there a security mechanism in VMFS which would prevent say fdisk being run or vmkfstools -C to create a new VMFS on it when that VMFS is being accessed by other ESX hosts which have running VMs on it?

Mike La Spina says:

September 26, 2009 at 9:23 PM



Hi Arno,

The LUN will be at the mercy of the Administrator, fdisk will be able to write to it and you will be able to overwrite any VMFS header that is exposed to the host in question.

Regards,

Mike

Utiliser dd sur ESXi 4.0 - Hypervisor.fr says:

April 7, 2010 at 11:48 AM

[...] expliqué il y a longtemps par Mike La Spina sur son blog et lors d'une session dédiée au VMFS au VMworld 2007, il est possible de sauvegarder les [...]

Raza says:

May 31, 2010 at 7:46 AM



This is really a very good Article, Thank Mike for sharing.....
All the best wishes for you..... ☐

Mike Andreiev says:

July 22, 2010 at 2:49 AM



hello!

we have a problem with vmdk file on vmfs volume – after hard reset (ESX server hang up) we could not access vmdk file – when we try copying it on other host (storage) we got "invalid argument" message. We try remove lock from this vmdk file, but owner of file creator is dead, so how we could change UUID of vmdk file or remove lock from it ?

Mike La Spina says:

July 22, 2010 at 7:12 AM



Hi Mike,

Try editing the VM's vmx file and set the following param:
disk.locking=off
Then see if it starts.

As well I would try removing the VM from inventory and adding it back again. Another approach would be to connect your vCenter client to an ESX host directly which would avoid vCenter Server safety checks on the vmdk operations.

It should not be necessary to change the uuid.

Regards,

Mike

Mike Andreiev says:

July 23, 2010 at 4:01 AM



Mike, here more details of our problem:

Host were was that storage is dead now ☹

We install new ESX server and attach drive from dead ESX to new.

Do with locked storage:

vmkfstools -D 1C-SQL_1-flat.vmdk

in /var/log/vmkernel we see –

Lock [type 10c00001 offset 61200384 v 369, hb offset 3600384
gen 11, mode 1, owner 4c36c4a5-981d4706-2cc6-0025900154ad mtime 1094]

but old ESX host is gone....

UUID of our ESX -

Current /system/uuid = "4c45a358-adb8-8358-c665-0025900154ac"

How we can unlock vmdk ?

read file direct is impossible – we got "invalid argument"

1) Could we change UUID of current disk on UUID which was in moment of died ? – it was 4c36c4a5-981d4706-2cc6-0025900154ad

2) Could I directly open fb VMware VMFS, find place with locker owner and write zeros than ?

3) Could we install new ESX and change it UUID on UUID which was on old ESX when file locked?

Regards,

Mike.

Mike La Spina says:

July 24, 2010 at 8:52 AM



Hi Mike,

The lock is not issued on the vmdk file directly. VMFS volumes use the metadata files to maintain cluster state info. Since the lock state is the result of an unfinished clustered file operation you need to issue a breaklock at the VMFS device level.

This operation is available using the vmkfstools as follows:

```
# vmkfstools -B /vmfs/device/disk/vml.uuid
```

Regards,
Mike

Leave a Reply

Name (required)

Mail (will not be published) (required)

Website

XHTML: You can use these tags:

 <abbr title=""> <acronym title=""> <blockquote cite=""> <cite> <code>
<del datetime=""> <i> <q cite=""> <strike>

Submit Comment