Compromise-as-a-Service

Our PleAZURE

Felix Wilhelm & Matthias Luft
{fwilhelm, mluft}@ernw.de

Agenda

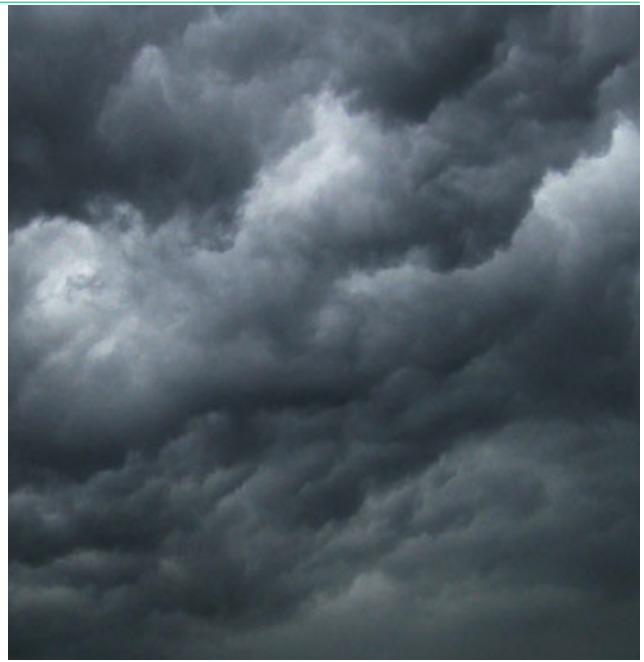
- Azure & Hyper-V:
  The research project
- Clouds & Hypervisors:
  A taxonomy of attack vectors
- Hyper-V:
  Architecture & research

ERNW
providing security.

Azure & Hyper-V

Windows Azure™

Background



- Government research project
  - together with *http:/www.thinktecture.com/*
    - Thanks, guys!
- Scope: Overall security posture of the Azure Cloud
  - Network security
  - VM Isolation
  - …
  - Management interfaces

Security Objectives

Technical objectives, ignoring privacy here

- Availability
- Isolation!
  - Or, in a more classical way, safeguarding the confidentiality and integrity of clients against each other.

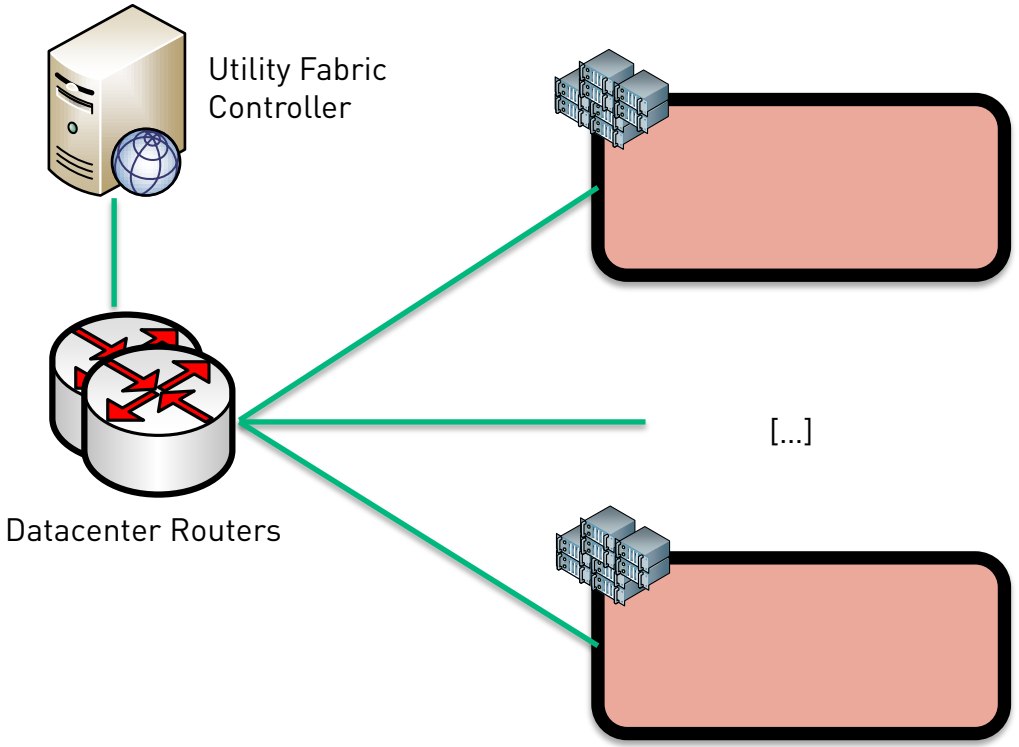
Taxonomy of Technical Cloud Attack Vectors

Cloud infrastructure != hosted environments

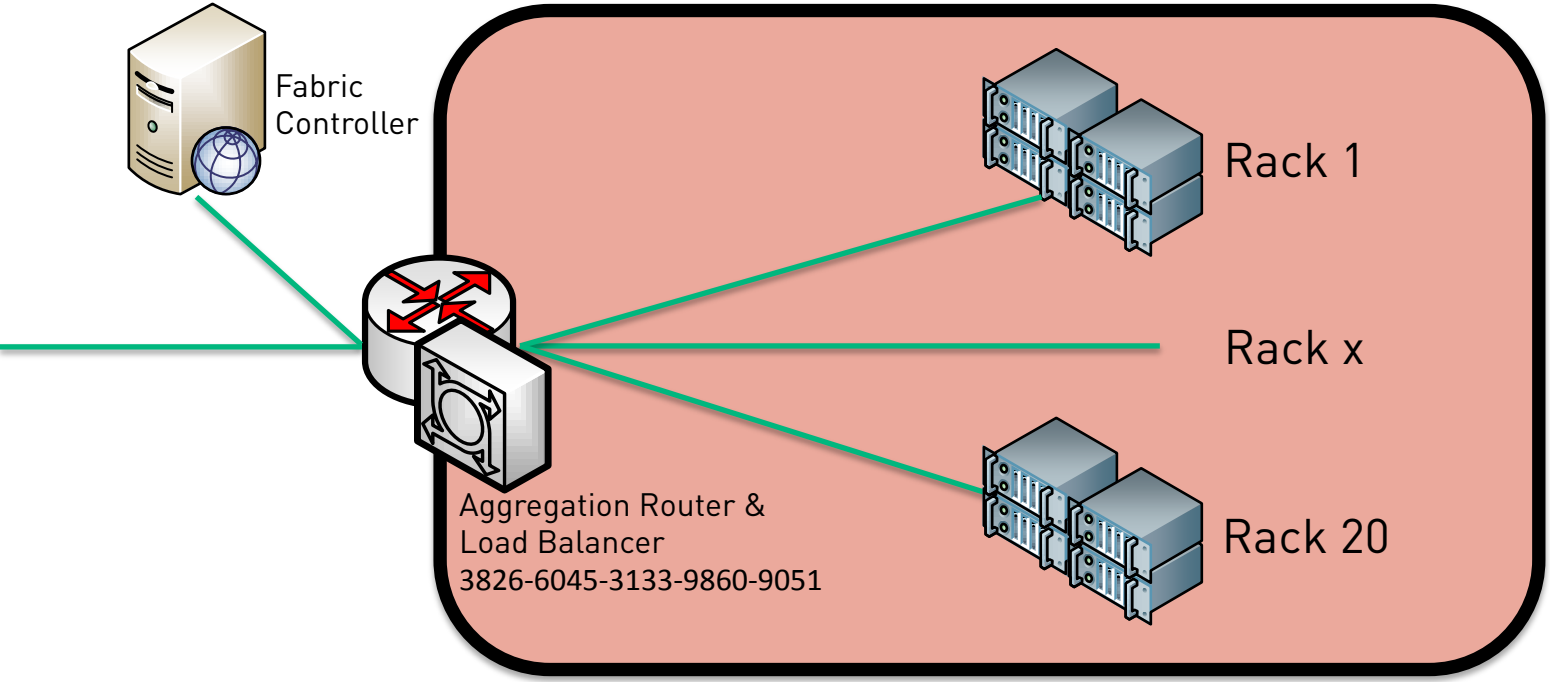
- Runtime Breakout
  - Covering CPU, memory, devices. See later.
- Storage breakout
  - Think: Directory traversal/weak ACLs on backend storage.
- Network isolation failure
  - Think: Eavesdropping on traffic of other VMs.
- Management interfaces
  - Think: BO/SQLi in web service, weak passwords.

- Scope of this talk: Runtime breakouts ;)

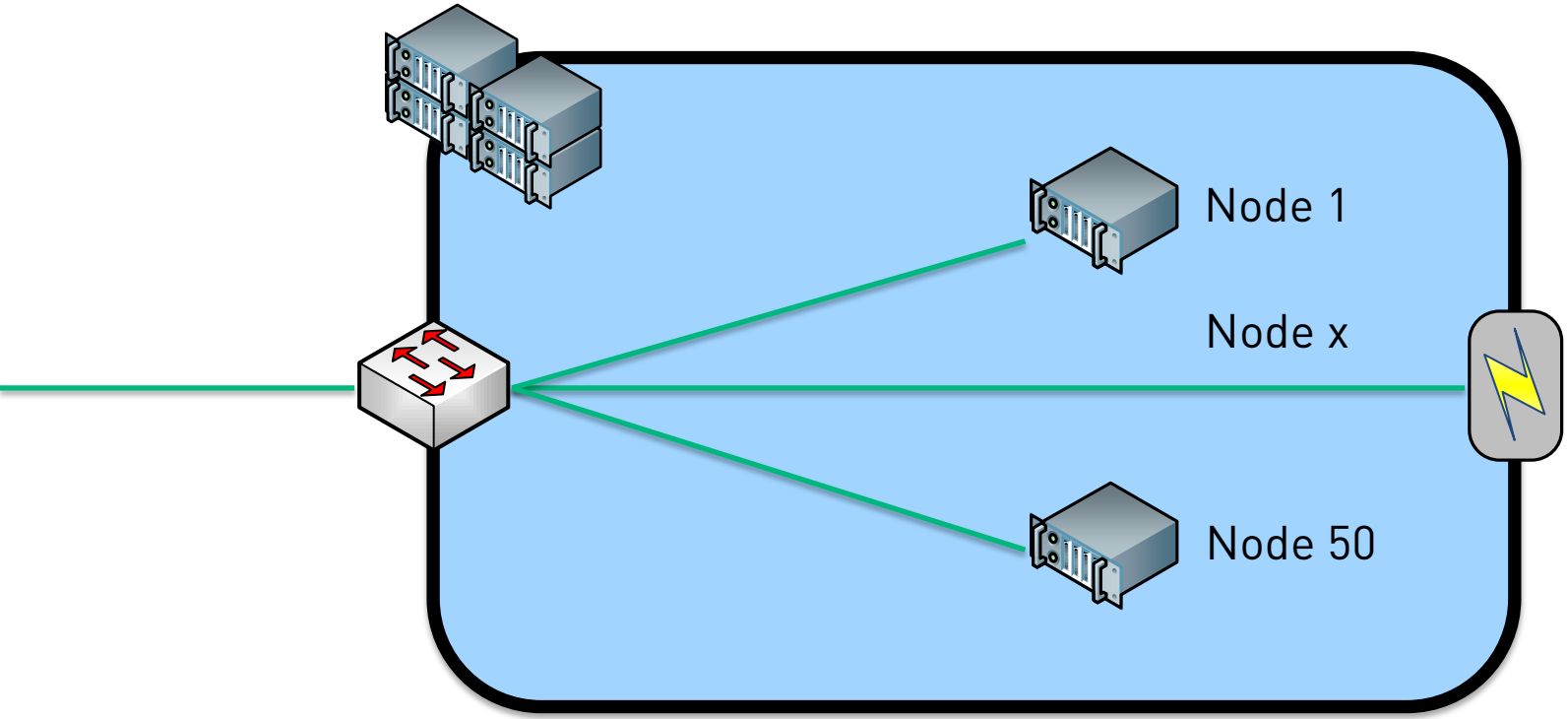
Fabric



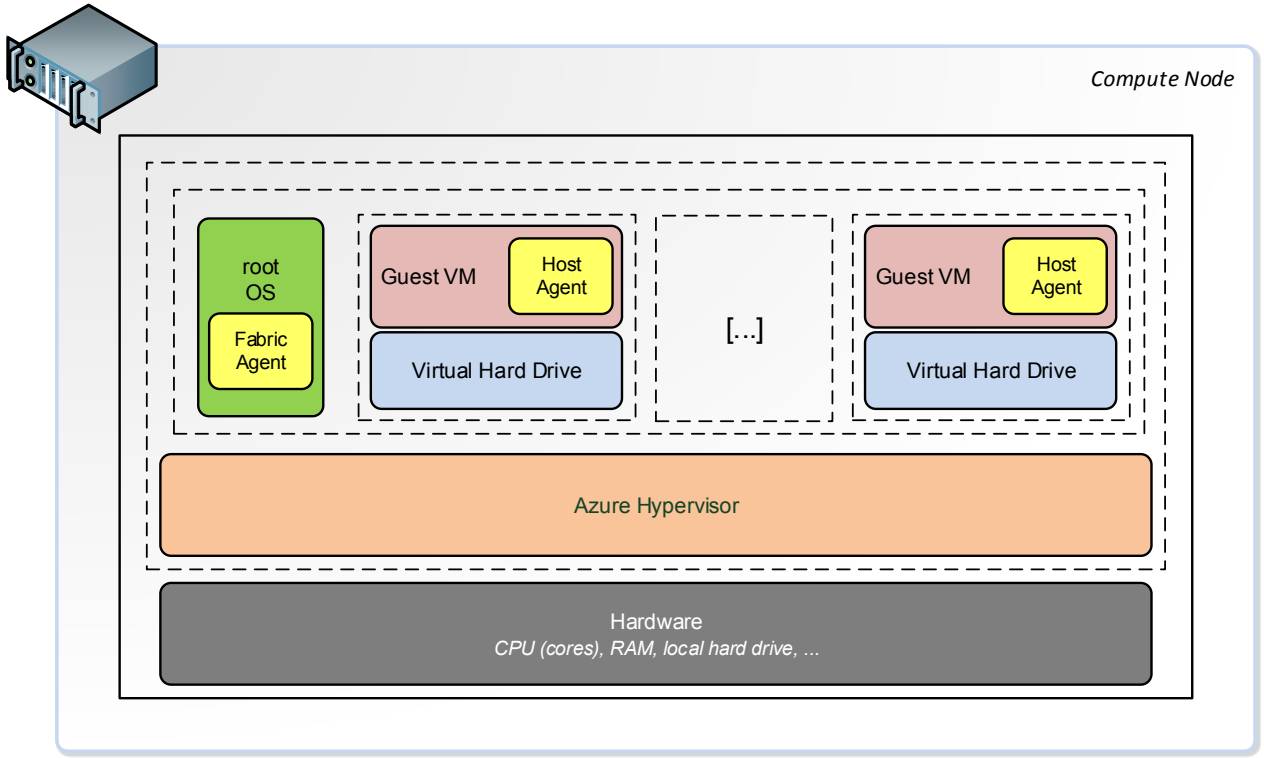
Cluster


Fabric
Controller
Rack 1
Rack x
Rack 20
Aggregation Router &
Load Balancer
3826-6045-3133-9860-9051

Rack



Compute Node

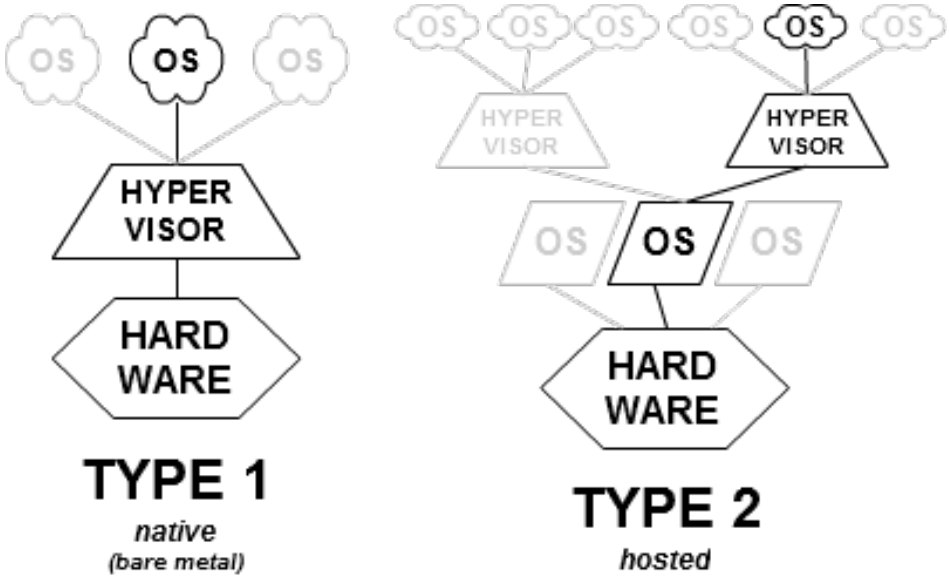


... the Cloud?

ERNW
providing security.

Hypervisor > *

Or: Runtime Breakouts



Type 1 vs. Type 2

Source: Wikipedia

Full vs. Para vs. HW

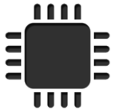
- Full Virtualization
  - A.k.a. Binary Translation

- Para-Virtualization

- Hardware-assisted Virtualization

ERNW
providing security.

Parts of an Hypervisor

- Device virtualization 
- Memory management & isolation 
- CPU scheduling & isolation 
- Management interfaces 
- Additional APIs 

Vulnerability History

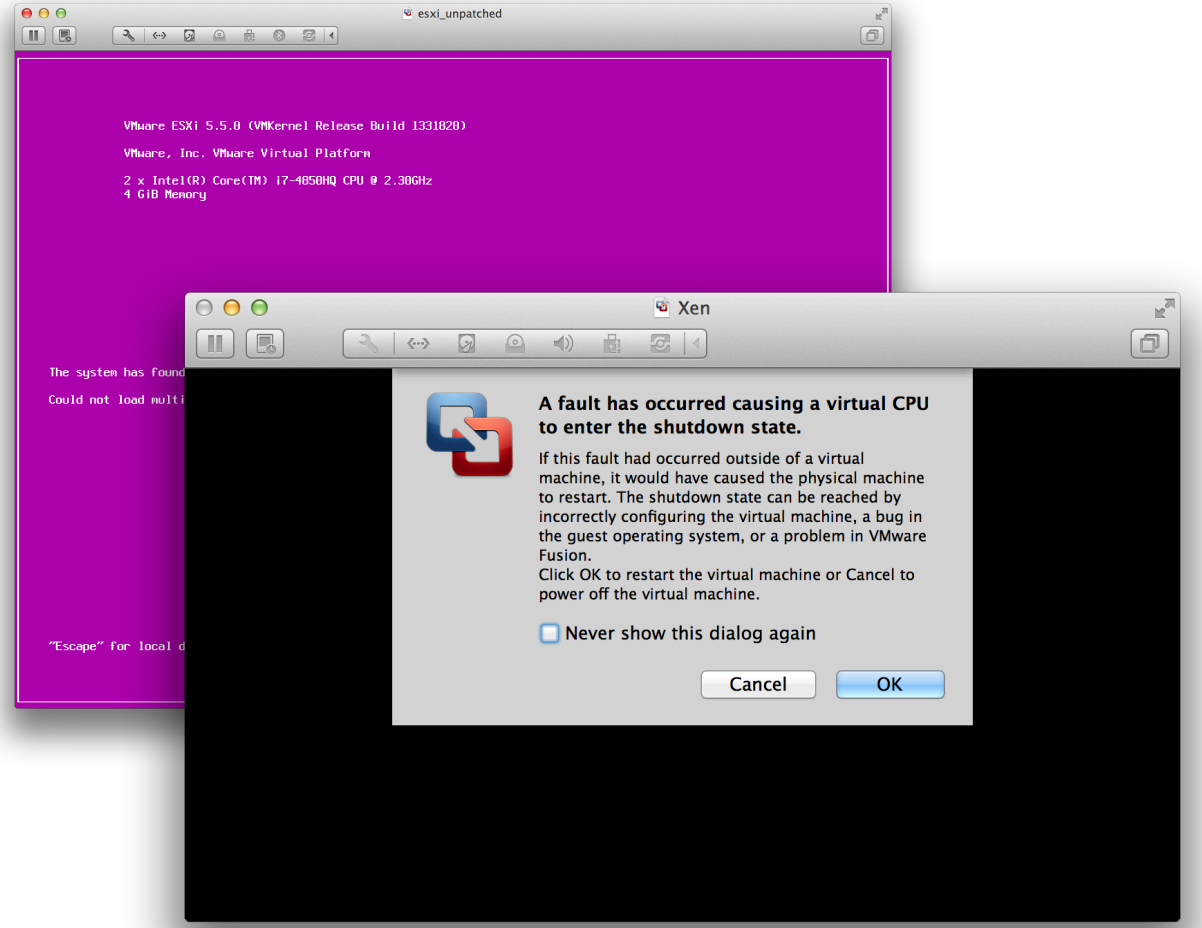
Hypervisor Breakout

- 2007: VMftp
- 2009: VMware Cloudburst
- 2011: Virtunoid – KVM Breakout
- 2012: VMSA-12-009
- 2012: VMDK Has Left The Building
- 2012: Xen SYSRET
- *2013: MS13-092*
- 2014: VirtualBox HGCM
- 2014: VirtualBox Chromium Breakout

“Virtual Air Gap”

“VMware isn't an additional security layer: It's just another layer to find bugs in”

Kostya Kortchinsky/Immunity/Cloudburst, 2009


esxi_unpatched
VMware ESXi 5.5.0 (VMKernel Release Build 1331820)
VMware, Inc. VMware Virtual Platform
2 x Intel(R) Core(TM) i7-4850HQ CPU @ 2.30GHz
4 GiB Memory
The system has found
Could not load multi
"Escape" for local d
Xen
A fault has occurred causing a virtual CPU to enter the shutdown state.
If this fault had occurred outside of a virtual machine, it would have caused the physical machine to restart. The shutdown state can be reached by incorrectly configuring the virtual machine, a bug in the guest operating system, or a problem in VMware Fusion.
Click OK to restart the virtual machine or Cancel to power off the virtual machine.
Never show this dialog again
Cancel
OK

SCNR ;-)

Taxonomy

As for Hypervisors

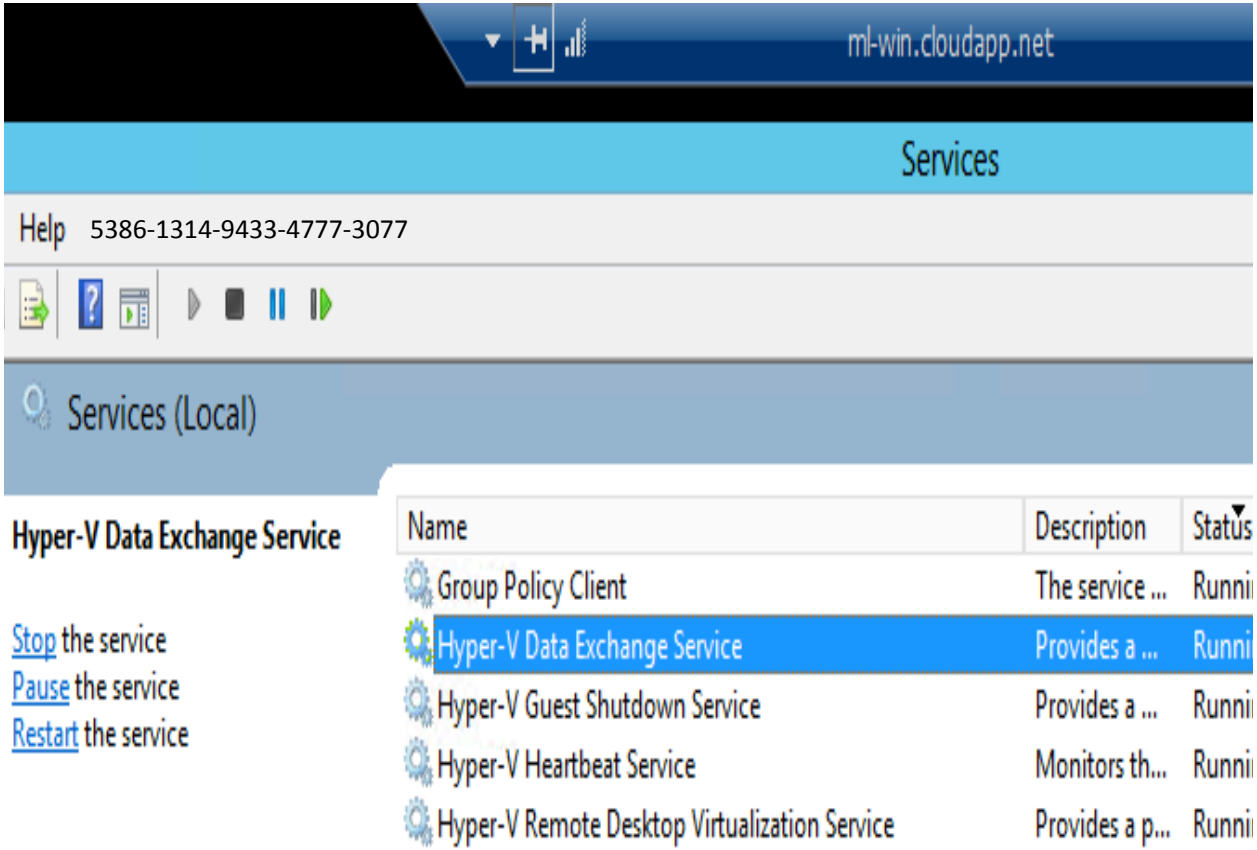
- Instruction set
  - Full support of x86 necessary – which is difficult!
  - What about processor errata?
- Management & additional channels
  - Look at the previous examples…
- Monitoring & Caches
  - Lots of side channels
- Memory management & devices
  - Ever looked at the QEMU CVE history? ;-)

Hyper-V

Seriously, there is just no Hyper-V Logo available.

Why Hyper-V?

- Supposedly, very little research so far
  - Only one DoS bug in 2009, reverse engineered from a patch.
- Used in a variety of corporate environments.
  - Including Azure!



Azure Hypervisor

A.k.a. Hyper-V?

ERNW
providing security.

Versions

```
azureuser@ernw-s1:~$ ./cpuid
Checking for Hyper-V: True :)
Max leaf number: 0x40000006
Looks sane
Build number: 9200
Version: 6.2
Service Pack: 20
Service Branch: 20539
CreatePartitions: 0
AccessPartitionId: 0
AccessMemoryPool: 0
AdjustMessageBuffers: 0
PostMessages: 1
SignalEvents: 1
CreatePort: 0
ConnectPort: 1
AccessStats: 0
RsvdZ: 0
RsvdZ: 0
Debugging: 1
CpuPowerManagement: 0
```

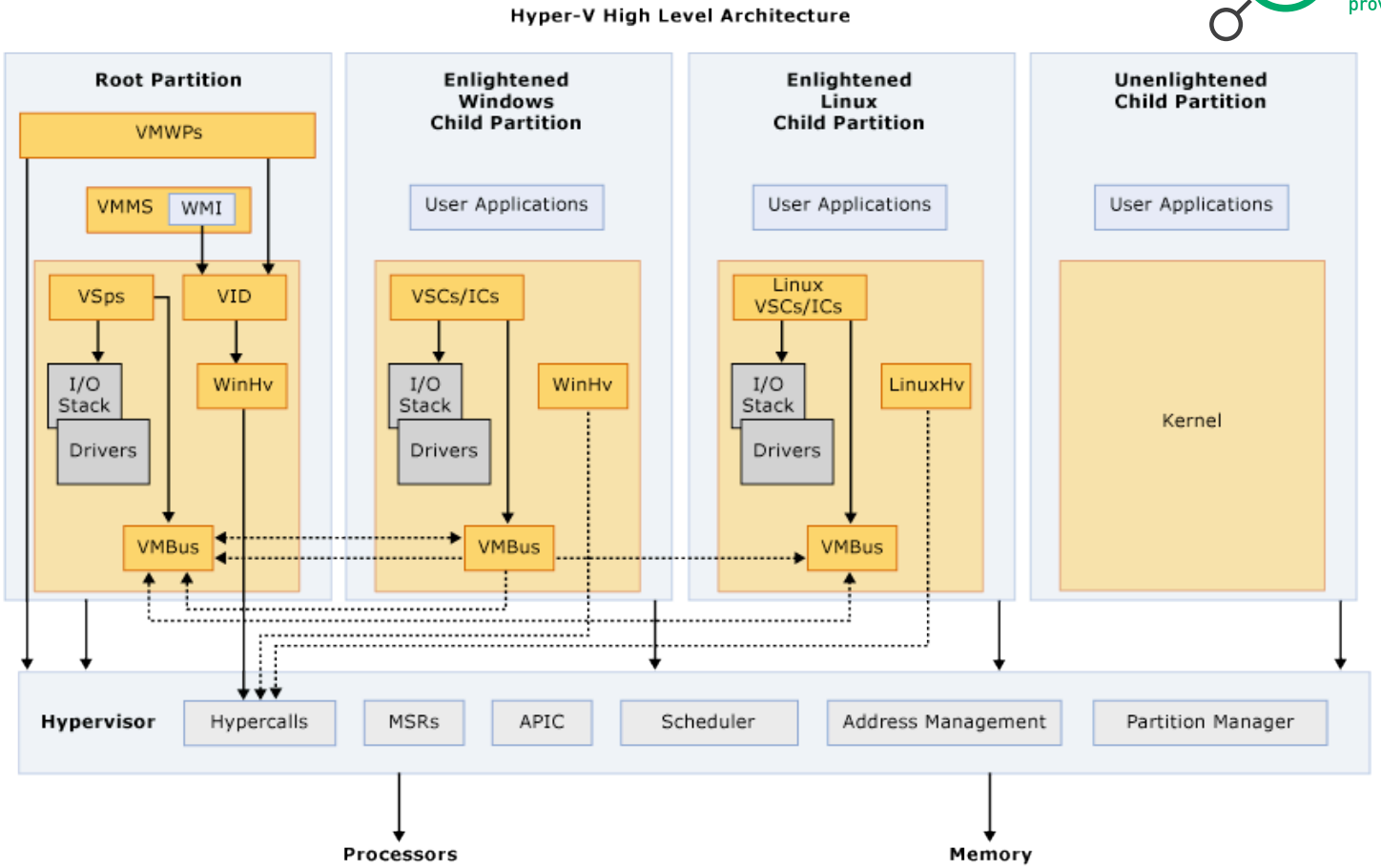
Azure Hypervisor

```
root@virtual-linux:/home/ernw# ./cpuid
Checking for Hyper-V: True :)
Max leaf number: 0x40000006
Looks sane
Build number: 9200
Version: 6.2
Service Pack: 16
Service Branch: 16384
CreatePartitions: 0
AccessPartitionId: 0
AccessMemoryPool: 0
AdjustMessageBuffers: 0
PostMessages: 1
SignalEvents: 1
CreatePort: 0
ConnectPort: 1
AccessStats: 0
RsvdZ: 0
RsvdZ: 0
Debugging: 1
CpuPowerManagement: 0
```

Hyper-V

Hyper-V

- Type 1 hypervisor
  - „Bare metal“
- VMs are called „partitions“
  - Root Partition performs management duties.
    - Creation, Configuration, Destruction
    - Logging
- Uses Intel VT instruction set for hardware based virtualization
- Support for „unenlightened“ + „englightened“ systems
  - Enlightened = Explicit support for Hyper-V. Requires guest modification.



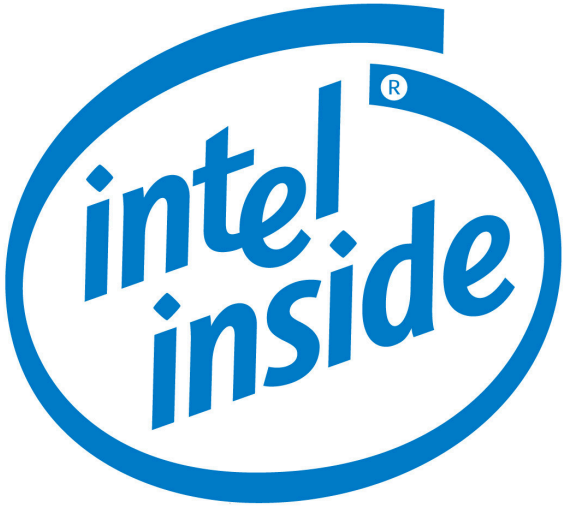
Source: http://msdn.microsoft.com/de-de/library/cc768520(en-us).aspx

Attack Surface

Attack Surface

- Concentrated on mapping the attack surface
- Three interesting attack vectors:
  - VM Exits
  - Hypercalls
  - VMBus

Hyper V – Hardware-assisted Virtualization



- Intel version is based on Intel VMX (virtual-machine extensions)
- Adds an “additional privilege ring”
  - VMX root vs. VMX non root
- Transitions between hypervisor and guest system
  - VM exits
  - VM entries

VM Exits

- Can be triggered on purpose
  - VMCALL
- Or as effect of executing “privileged” instructions
  - Which instructions are considered privileged depends on the hypervisor
    - Access to system registers
    - VMX instructions
    - RDRAND 
- And much more..
  - Interrupts
  - APIC

VM Exits

```
root@virtual-linux:/home/ernw# ./cpui
Checking for Hyper-V: True :)
Max leaf number: 0x40000006
Looks sane
Build number: 9200
Version: 6.2
Service Pack: 16
Service Branch: 16384
CreatePartitions: 0
AccessPartitionId: 0
AccessMemoryPool: 0
AdjustMessageBuffers: 0
PostMessages: 1
SignalEvents: 1
CreatePort: 0
ConnectPort: 1
AccessStats: 0
RsvdZ: 0
RsvdZ: 0
Debugging: 1
CpuPowerManagement: 0
root@virtual-linux:/home/ernw#
```

- VM Exits need to be transparent to guest
  - Requires emulation of privileged instructions
- “Enlightenment” requires way to query Hyper-V version and features
  - Implemented using “cpuid” instruction
  - Return values contain Hyper-V version + guest permissions

Hypercalls

5.6.1 HvCreatePartition

The HvCreatePartition hypercall allows an authorized guest to create a new partition.

Wrapper Interface

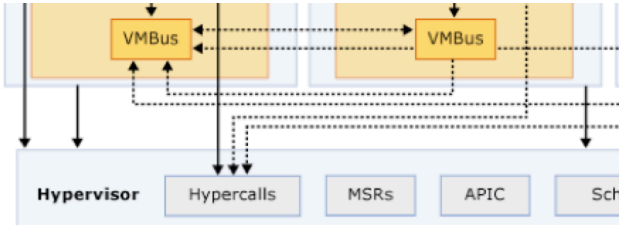
```
HV_STATUS
HvCreatePartition(
      __in  UINT64                          Flags,
      __in  HV_PROXIMITY_DOMAIN_INFO        ProximityDomainInfo,
      __out PHV_PARTITION_ID                NewPartitionId
);
```

Native Interface

HvCreatePartition		
	Call Code = 0x0040	
➡ Input Parameters		
0	Flags (8 bytes)	
8	ProximityDomainInfo (8 bytes)	
⬅ Output Parameters		
0	NewPartitionId (8 bytes)	1747-8640-2533-9505-2870

- Like system calls but for communication between VM kernel and hypervisor
- Documented interface
  - And known security boundary
- Used by enlightened partitions to improve performance
- Root partition manages other partition using hypercalls

VMBus



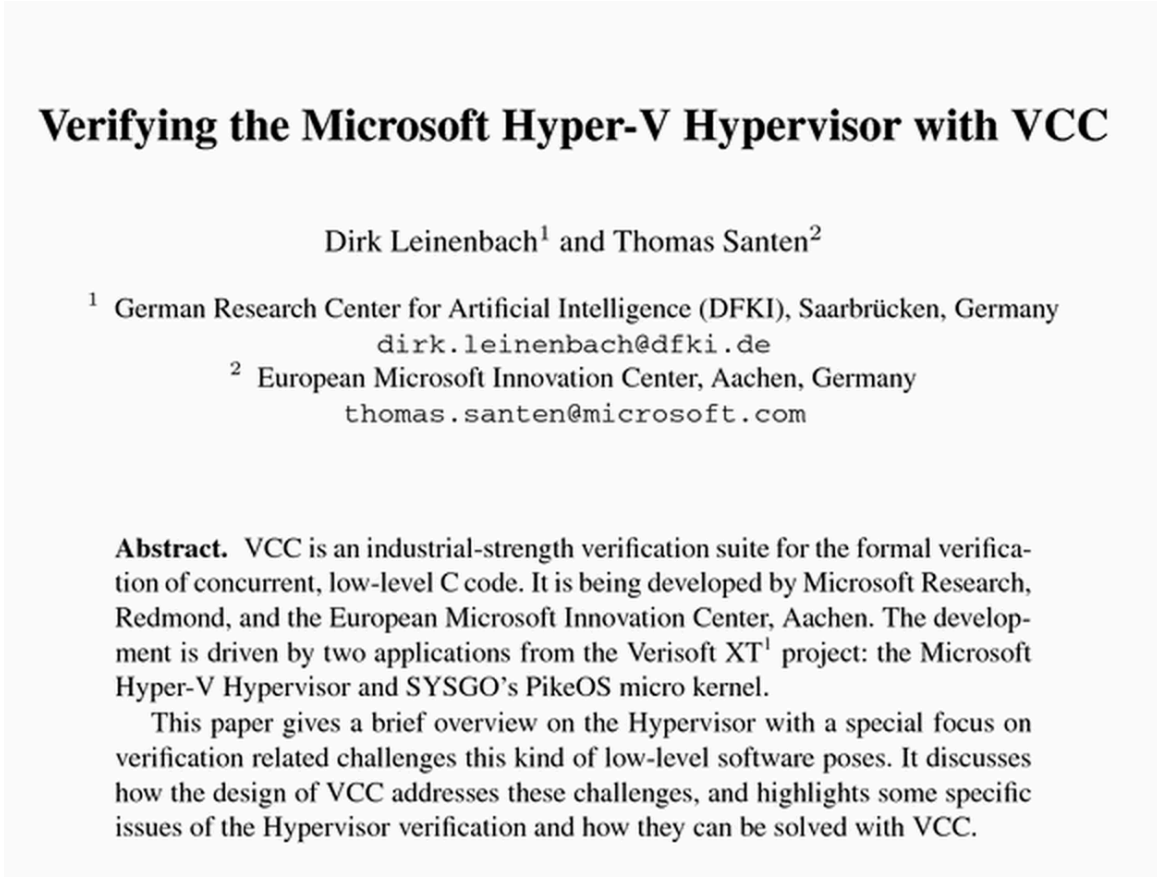
- Message bus for communication between partitions
- Default configuration:
  - Only communication to and from root partition allowed
- Memory pages that are mapped for multiple partitions
  - Heavily used for performance critical tasks in enlightened partitions
    - Think network devices
- Large attack surfaces

Attack Surface

- VMBus as a very promising target
  - But, possibly large differences between Azure and “Normal” implementation
- VM Exit handling
  - Especially special cases and error conditions
  - Requires high insight into processor internals
    - We are not Gal Diskin ☺
- **Hypercalls**
  - Documented API
  - Isolated functions
  - Low hanging fruits ☺

Expectations

- Minimal feature set inside Hypervisor itself
  - Small code base (2MB)
  - Around 60k lines of C + assembly code
  - Compared to 50M LOC in Windows Server
- Only 1 serious (public) vulnerability until 2013
  - VMBus Denial-of-Service



Verifying the Microsoft Hyper-V Hypervisor with VCC

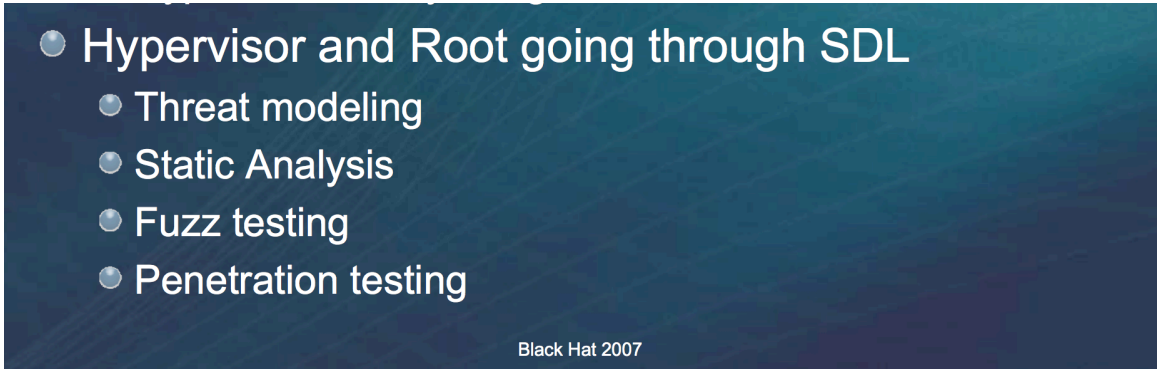
Dirk Leinenbach[1] and Thomas Santen[2]

[1] German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
dirk.leinenbach@dfki.de
[2] European Microsoft Innovation Center, Aachen, Germany
thomas.santen@microsoft.com

Abstract. VCC is an industrial-strength verification suite for the formal verification of concurrent, low-level C code. It is being developed by Microsoft Research, Redmond, and the European Microsoft Innovation Center, Aachen. The development is driven by two applications from the Verisoft XT[1] project: the Microsoft Hyper-V Hypervisor and SYSGO's PikeOS micro kernel.

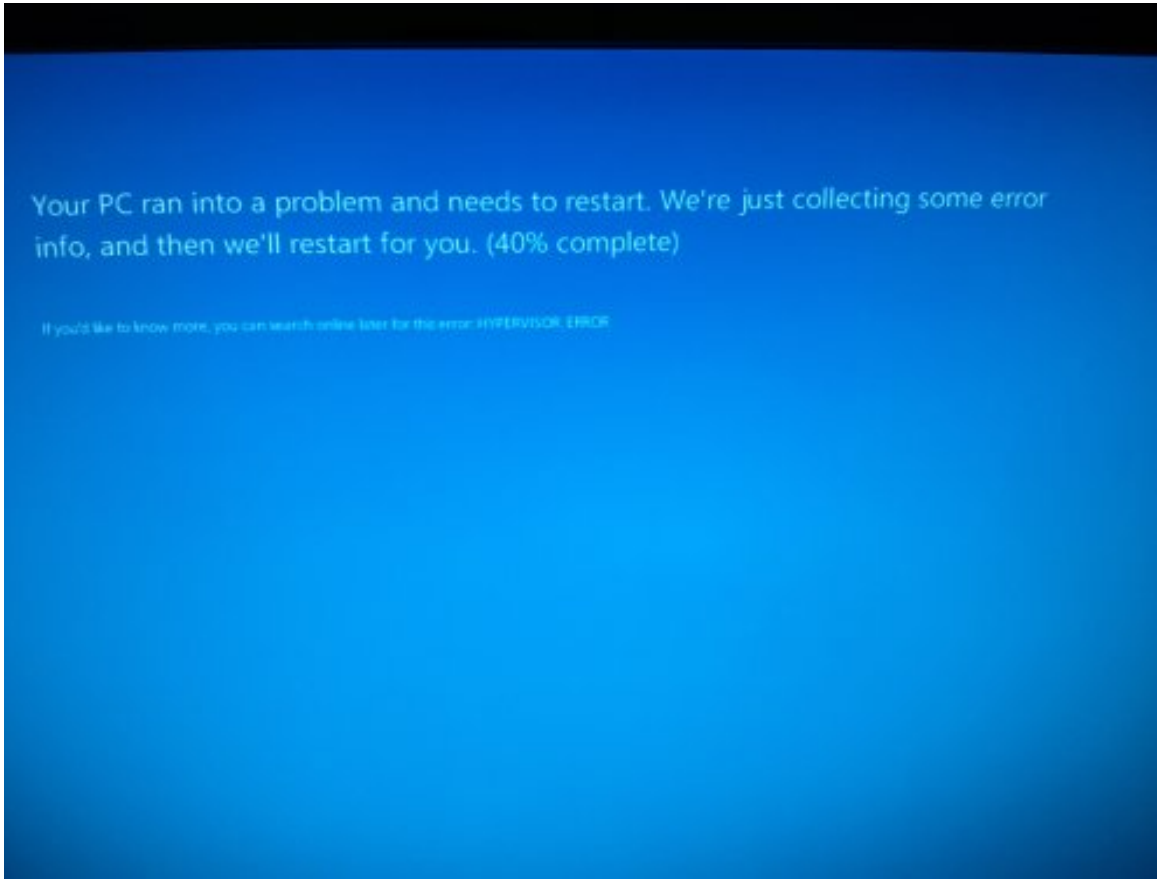
This paper gives a brief overview on the Hypervisor with a special focus on verification related challenges this kind of low-level software poses. It discusses how the design of VCC addresses these challenges, and highlights some specific issues of the Hypervisor verification and how they can be solved with VCC.

Formal verification...



Microsoft SDL...

→ Low Expectations

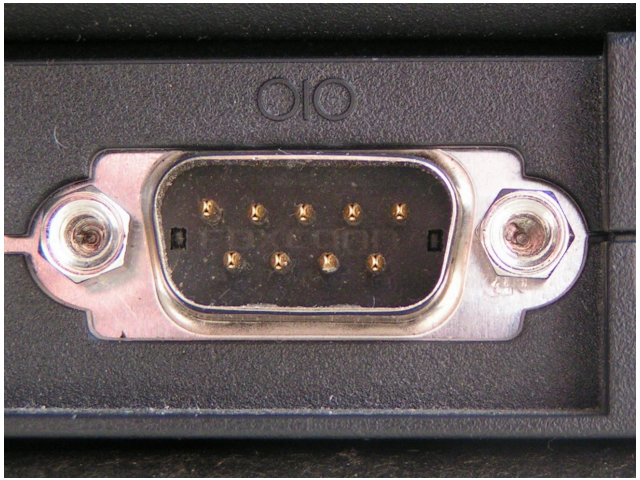


.. two days later ☺

Root cause analysis

- Crashing was easy
  - Understanding the bug is the hard part
- Hyper-V is mainly implemented Hvix64.exe (Intel) and Hvax64.exe (AMD)
- Unfortunately no public debugging symbols available ☹
- However some symbols can be ported from winload.exe and hvloader.exe
  - Networking Code, USB Stack, Debugger implementation
  - Full credit to Gerhart from securitylab.ru for this idea!

Debugging



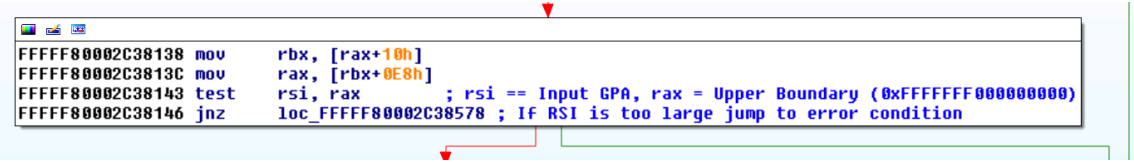
- Debugging via WinDBG is supported
  - Serial Port, Firewire, Ethernet…
- Fortunately no dedicated hardware is needed
  - Vmware supports nested virtualization
- Hyper-V specific functionality requires hvexts.dll
  - Not publicly available
  - Have a copy? Drop us a mail ☺

The Bug

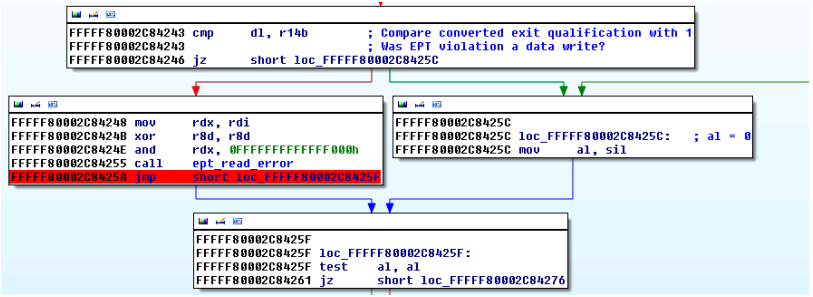
- Hypercall interface:
  - VM executes “vmcall” instruction
  - Arguments are passed in registers (for slowcalls):
    - RCX = Hypercall number
    - RDX = Guest Physical Address of Input Data
    - R8 = Guest Physical Address of Output Data
- Multiple checks before hypercall handler is called
  - Valid hypercall number
  - Alignment of GPA addresses
  - Hypercall comes from Ring0
  - ....

The Bug

- Check upper boundary of Input GPA



- Error condition will result in execution of EPT error handler



The Bug

- Input address is used as index into the extended page table.

```
FFFFF80002CD62F0 mov     r9b, r8b
FFFFF80002CD62F3 mov     r8, [rcx+ept_object.ept_address]
FFFFF80002CD62FA mov     rax, rdx
FFFFF80002CD62FD shr     rax, 12
FFFFF80002CD6301 mov     rbp, rdx
FFFFF80002CD6304 mov     rsi, rcx
FFFFF80002CD6307 mov     r8, [r8+rax*8]  ; rax = rdx = rdi = rbp = Input Memory
FFFFF80002CD6307                         ; r15 = Hypercall Number
FFFFF80002CD630B mov     rax, 8000000000000000h
FFFFF80002CD6315 test    rax, r8         ; Secret Value 1541-5205-8933-5815-1341
FFFFF80002CD6318 jz      return_loc
```

- Classic Out-of-Boundary access
- The trigger?
  - `hypercall(0xXY, input_address = 0x4141414141414141...)`

ERNW
providing security.

Fixed with MS13-092

- Denial of Service of complete hypervisor
- Potentially privilege escalation to other virtual machines
- Azure affected !

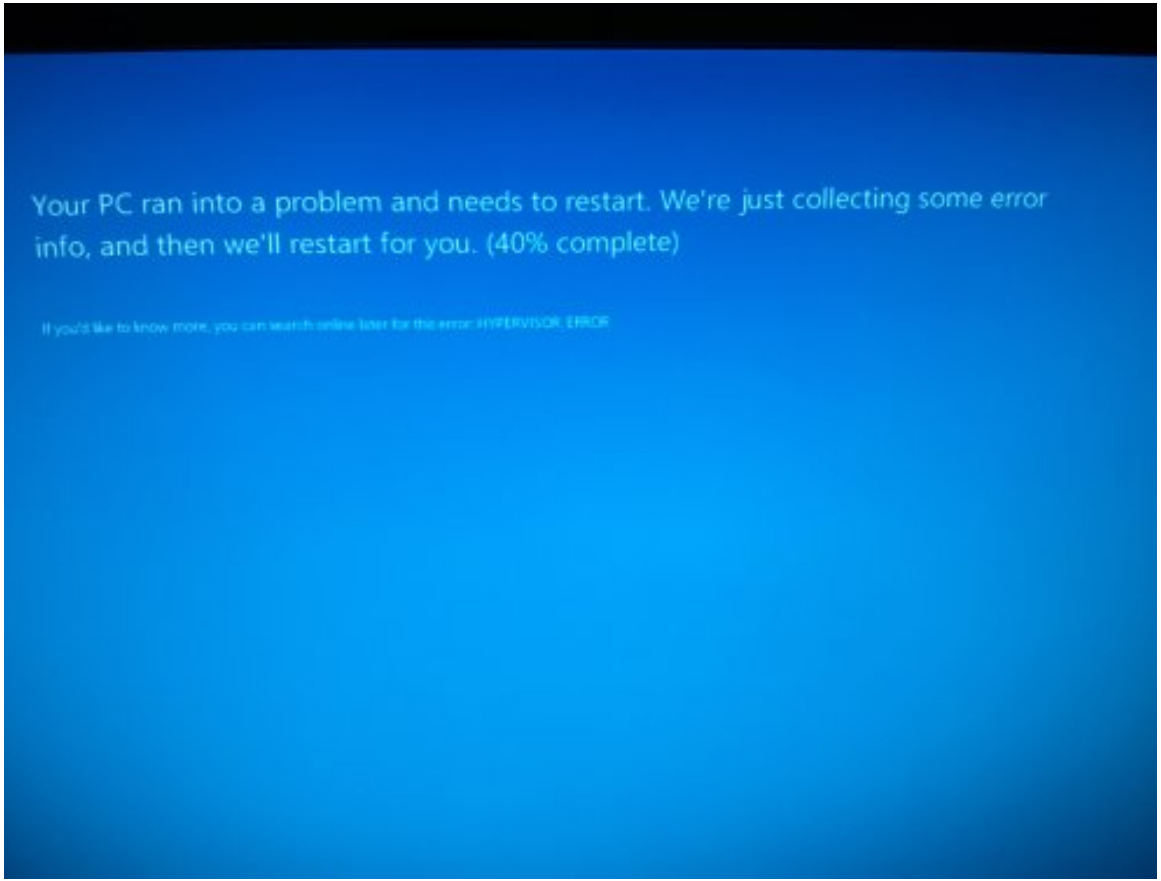
Acknowledgments

Microsoft thanks the following for working with us to help protect customers:

- thinktecture (www.thinktecture.com) & ERNW (www.ernw.de; Felix Wilhelm) on behalf of Bundesamt für Sicherheit in der Informationstechnik (BSI, German Federal Office for Information Security) for reporting the Address Corruption Vulnerability (CVE-2013-3898)

Privilege Escalation / Breakout ?

- Can not be used to get memory write corruption
  - We tried ☹
- However, exploitation of logic errors seems to be possible
  - Unfortunately nothing presentable (yet)


Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you. (40% complete)

Demo

Conclusions

- Hypervisor security & research is not hard.
- Hypervisor security & research is hard.
- Even verified code contains bugs which can easily be exploited.
- Still traversing the layers… there is lot of fun to be explored.
- Virtualization is an additional layer to exploit – hypervisors gain features, but not security!
- *Make the theoretical practical.*

There's never enough time...

THANK YOU...

Felix Wilhelm
fwilhelm@ernw.de
@_fel1x

...for yours!

Matthias Luft
mluft@ernw.de
@uchi_mata

ERNW
providing security.

Sources & Mentions

- Gal Diskin, Virtually Impossible
- Gerhart, http://www.securitylab.ru/contest/444112.php
- Hypervisor Functional Specification, Microsoft