Compromise-as-a-Service

Our PleAZURE

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

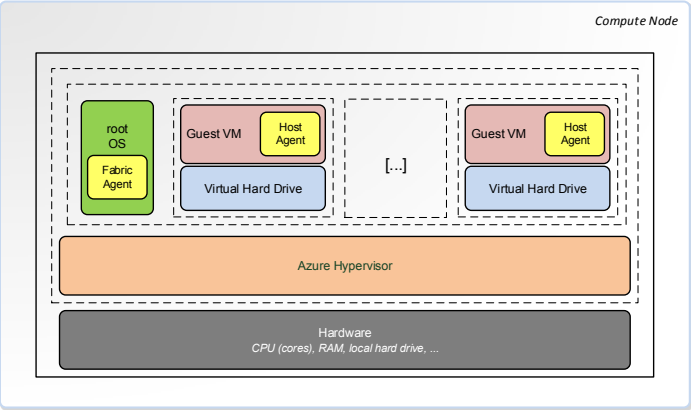
Agenda

- Introduction & Methodology
- Architecture & Attack Surface
- MS13-092

Prelude

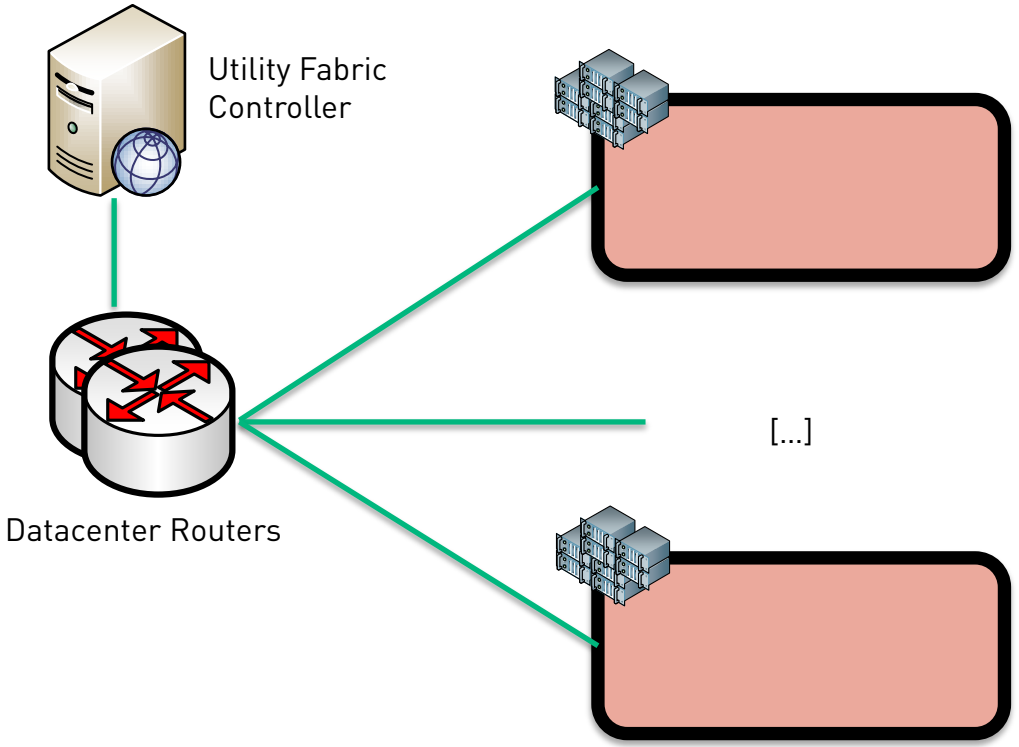
- No complete security evaluation of Hyper-V
  - Still work in progress.
  - We will provide a detailed overview of the attack surface.
- Will talk about stuff that did not work (lots) and stuff that did work (some).
- Goal is to motivate and help other researchers.

Background

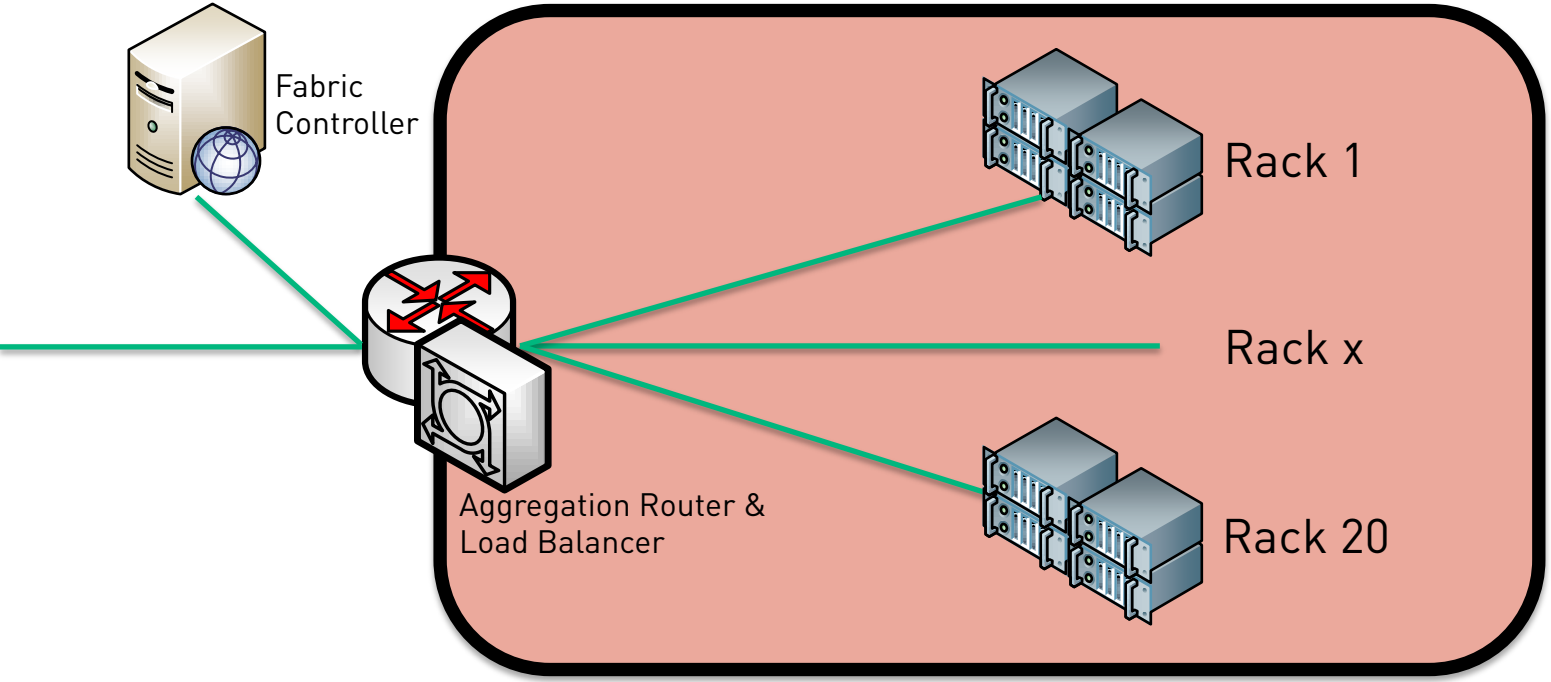


- Original research was part of German government research project.
  - Overall security posture of the Azure Cloud
    - Network Security
    - VM Isolation
    - Management Interfaces
    - ....

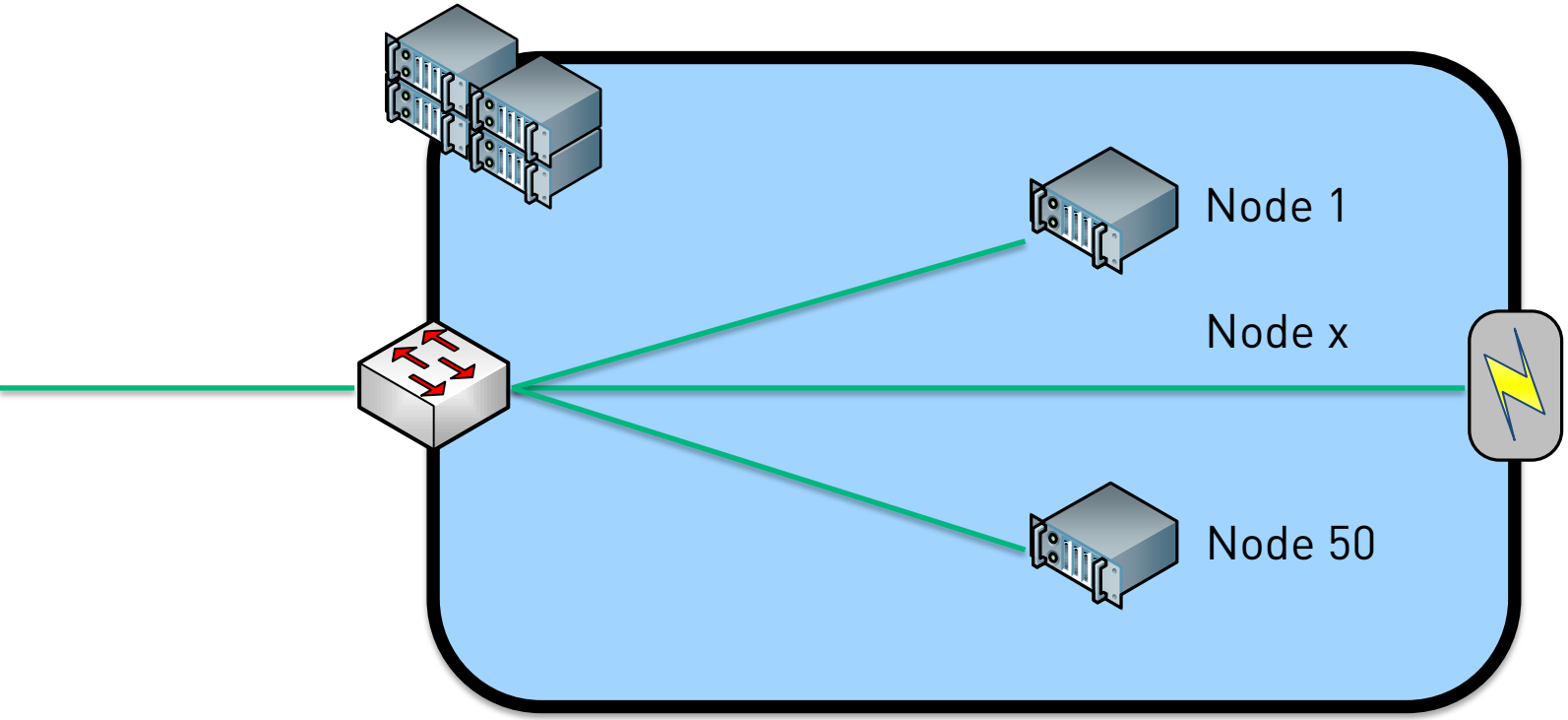
Fabric



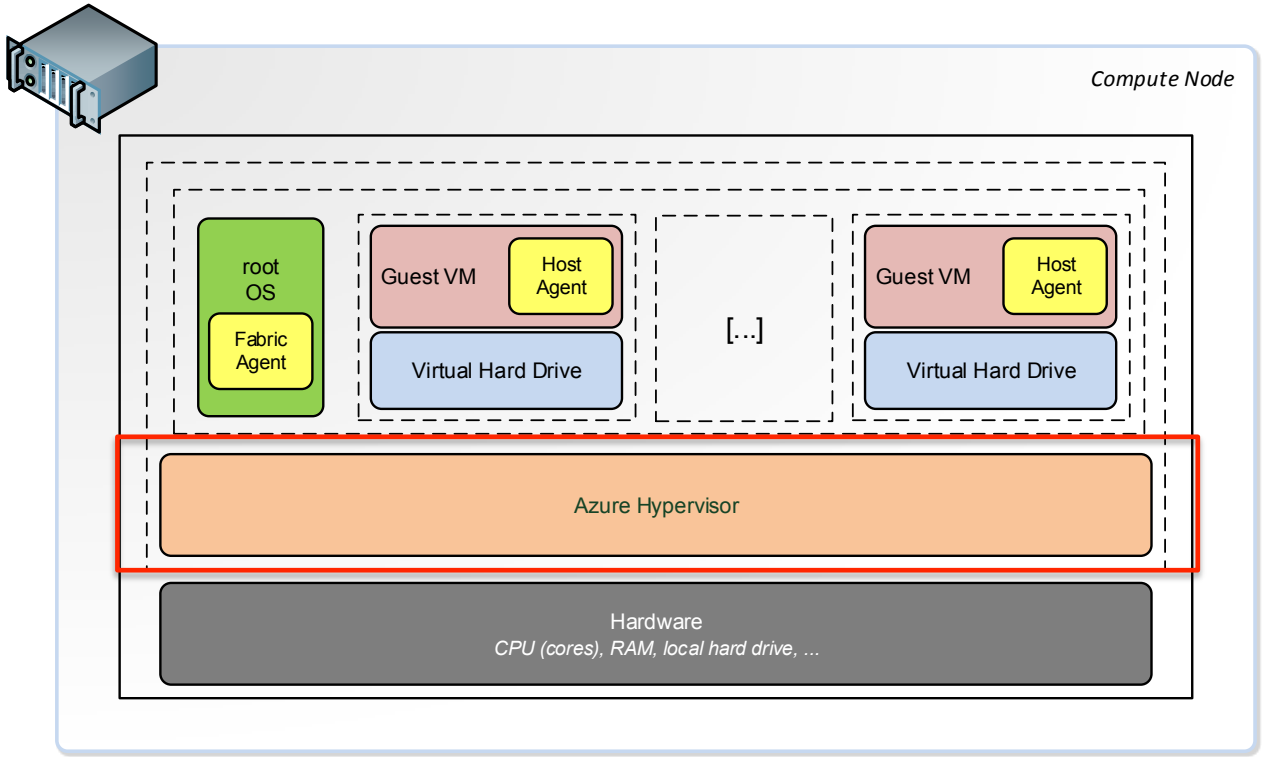
Cluster


Fabric
Controller
Rack 1
Rack x
Rack 20
Aggregation Router &
Load Balancer

Rack



Compute Node

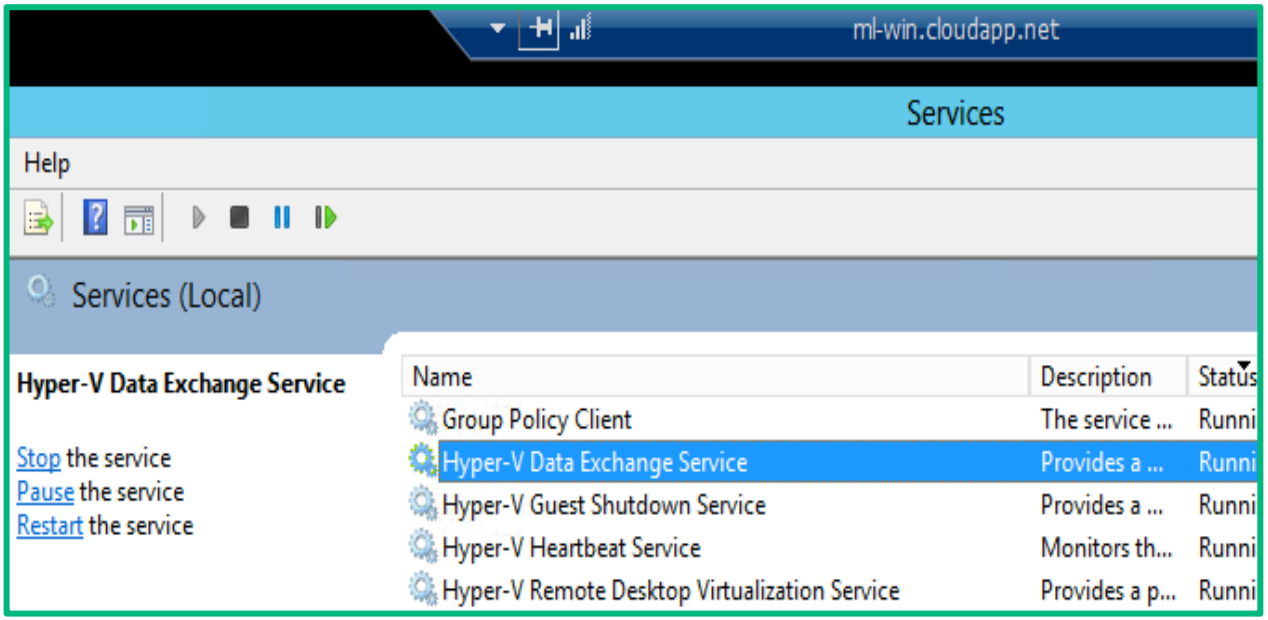


... the Cloud?

ERNW providing security.

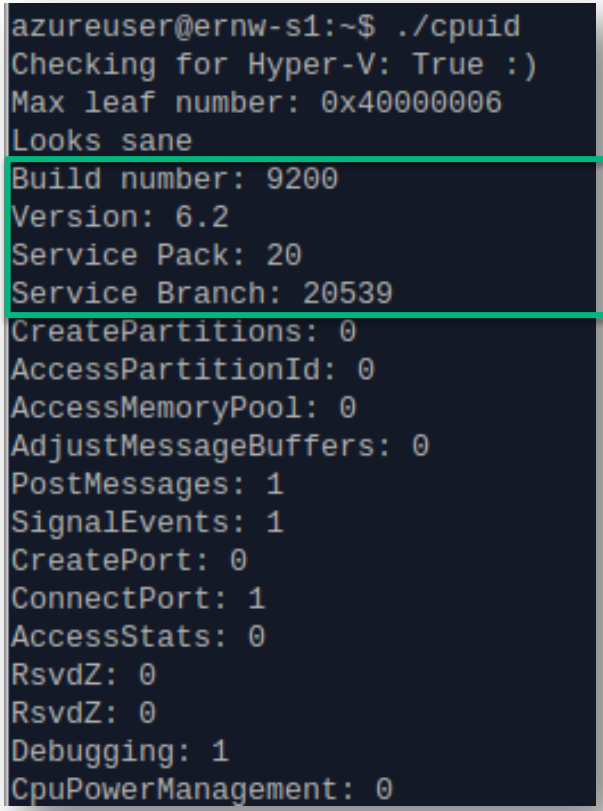
Why Hyper-V?

- Supposedly, very little research so far:
  - Four DoS vulnerabilities at all.
  - Almost no published 3rd party research.
- Used in a variety of corporate environments.
  - Including Azure!
  - Heavily marketed by Microsoft

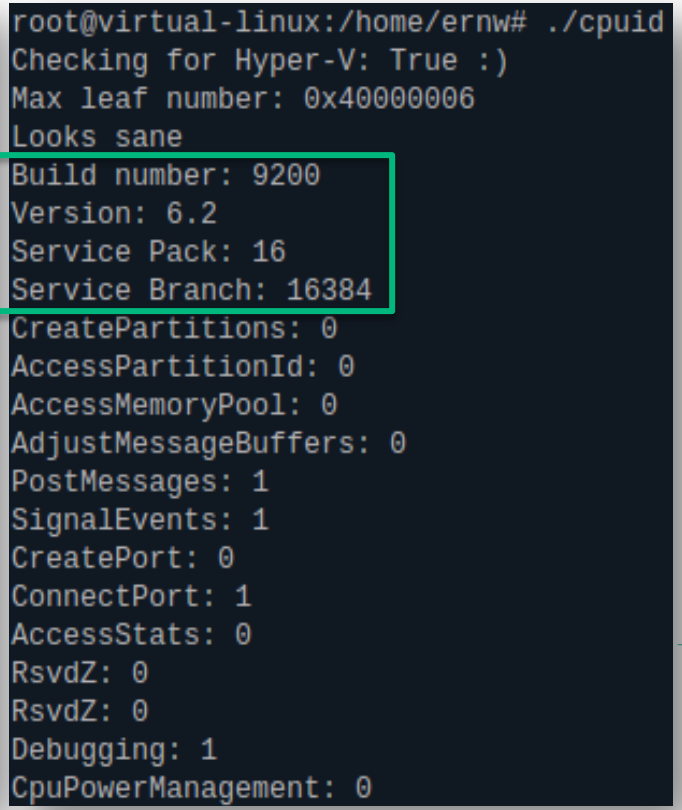


Azure Hypervisor

A.k.a. Hyper-V?



Azure Hypervisor



Hyper-V

Versions

Methodology

Or: We're searching for runtime breakouts, right?

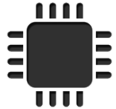
ERNW providing security.

HV Security Objectives

- Main and crucial objective: Isolation!
  - On all layers (network, runtime, storage, ...)!
- Potential attacks violating this objective:
  - Breakout attacks (of course ;-))
  - Side channels & timing attacks
  - Traditional attacks: Account compromises, misconfiguration, ...

ERNW
providing security.

Parts of an Hypervisor

- Device virtualization 
- Memory management & isolation 
- CPU virtualization 
- 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”

Hyper-V

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

Kostya Kortchinsky/Immunity/Cloudburst, 2009

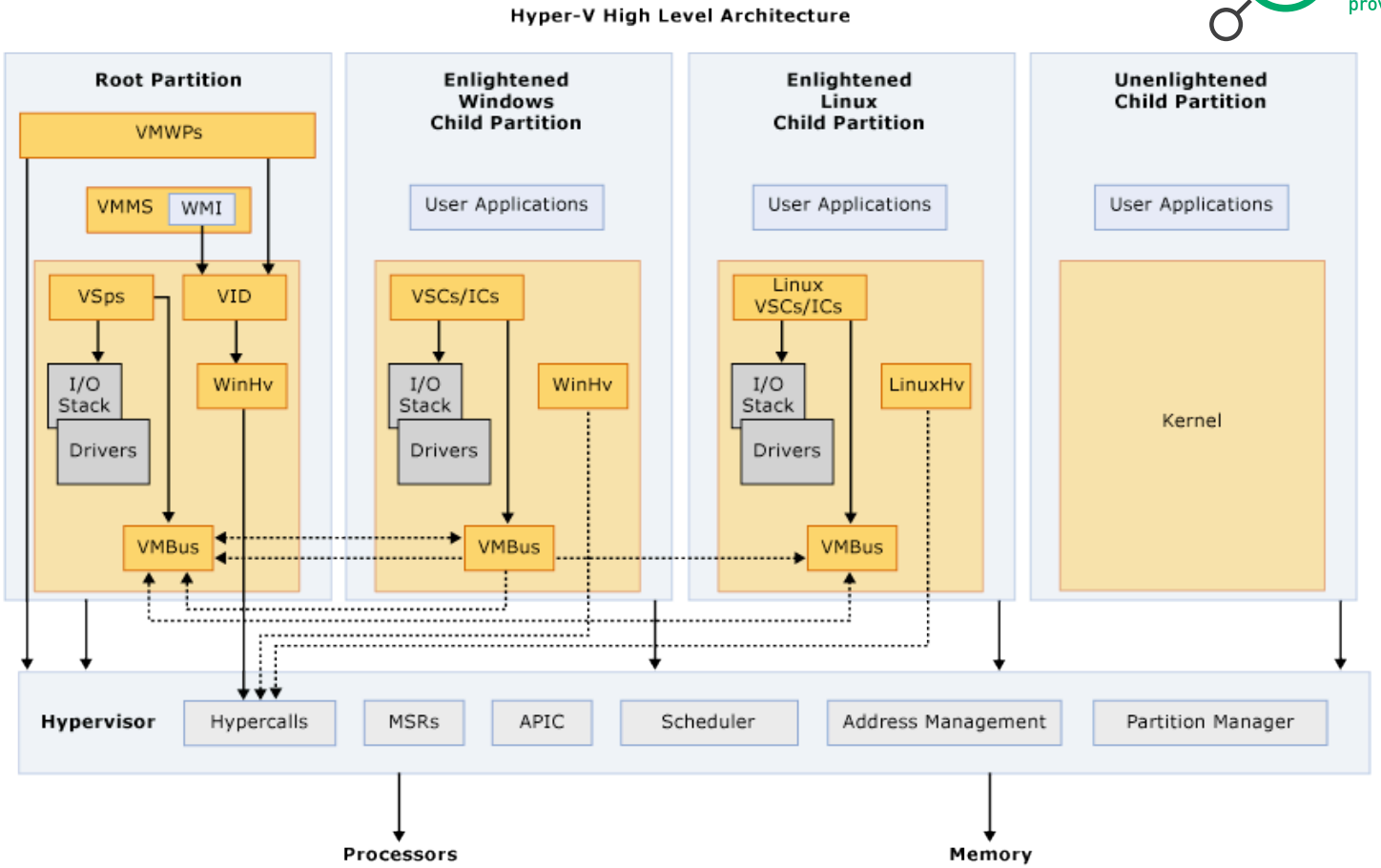
ERNW
providing security.

Hyper-V

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

Hyper-V

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



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

ERNW
providing security.

Hyper-V vs. VMware ESXi

	Hyper-V	ESXi
Architecture	Micro Kernel	Monolithic
Most interesting attack surface	Parent Partition	Hypervisor
Platform to exploit	Windows	Proprietary Unix-like environment with a lot of custom libraries and characteristics

The micro kernel improves the security posture of the hypervisor, but moves attention to the parent partition – which simplifies post-exploitation!

Hyper-V Information Sources

(19) **United States**
(12) **Patent Application Publicati**
Post et al.

(54) **SHARED MEMORY BETWEEN CHILD AND PARENT PARTITIONS**

(75) Inventors: **Bradley Stephen Post**, San Francisco, CA (US); **Ed Cox**, Morgan Hill, CA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(21) Appl. No.: **12/894,896**

(22) Filed: **Sep. 30, 2010**

- Hypervisor Top Level Functional Specification
  - Detailed documentation of Hypercall API
- Linux Integration Services
  - Open Source Hypercall / VMBus / VSC implementation
- Patent Applications
  - See picture
- Singularity Header Files
  - https://singularity.svn.codeplex.com/svn/base/Windows/Inc/
- Common Criteria Certification Documents
  - http://www.commoncriteriaportal.org/files/epfiles/0570b_pdf.pdf
- Binaries
  - ...

Reverse Engineering - Hypervisor

- Hypervisor itself is implemented in 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!

Reverse Engineering – VMX

- No documented system libraries
- Almost no strings
- Intel VMX instructions and VMCS are biggest help
  - Semi-automated approach: Locate vmx instructions and access to VMCS properties
  - Translate VMCS property values into corresponding name
  - Implemented using IDAPython script

Intel® 64 and IA-32 Architectures
Software Developer's Manual

Volume 3 (3A, 3B & 3C):
System Programming Guide

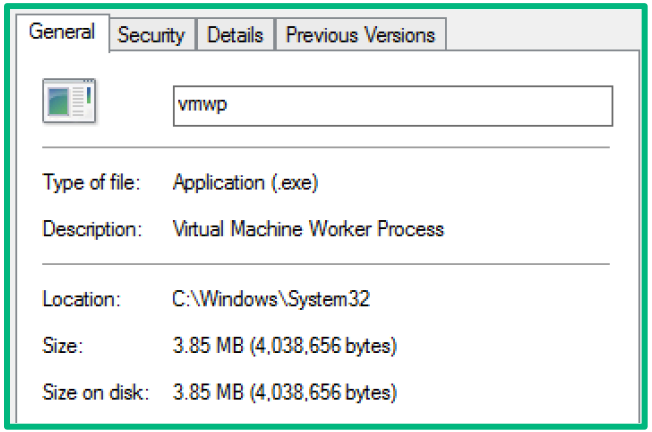
```
FFFFF8000064EA8D mov     eax, VMCS_Guest_SS_access_rights
FFFFF8000064EA92 vmread  rax, rax
FFFFF8000064EA95 mov     [rsp+78h+arg_8], rax
FFFFF8000064EA9D shr     al, 5
FFFFF8000064EAA0 and     al, 3
FFFFF8000064EAA2 cmp     al, 3
FFFFF8000064EAA4 jz      loc_FFFFF8000064EB72
```

ERNW
providing security.

Reverse Engineering - 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 ☺

Reverse Engineering - VSP



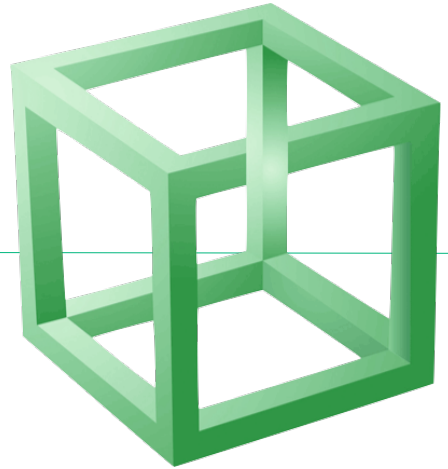
- Virtualization Service Provider
  - Implemented in Kernel (Storage, Networking) or Userspace(Framebuffer, HID)
- Public debugging Symbols are “sometimes” available
  - vmswitch.sys ☺
  - storvsp.sys ☹
  - vmwp.exe ☹ (Many debugging strings)

Attack Surface

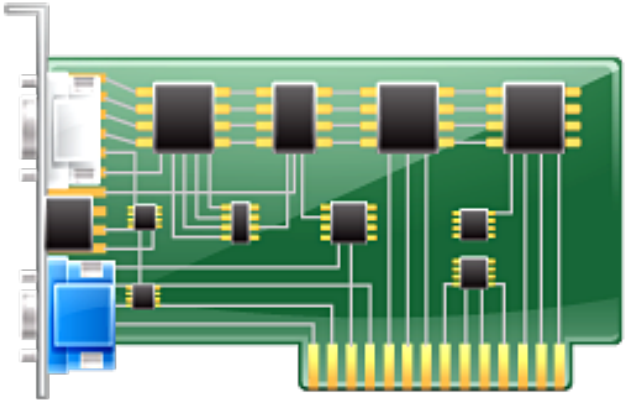
Attack Surface

- Based on the taxonomy presented earlier:
  - CPU: VM Exits
  - Device Virtualization: VMBus, Emulated Devices, RemoteFX
  - APIs: VMBus, Hypercalls

Emulated Devices

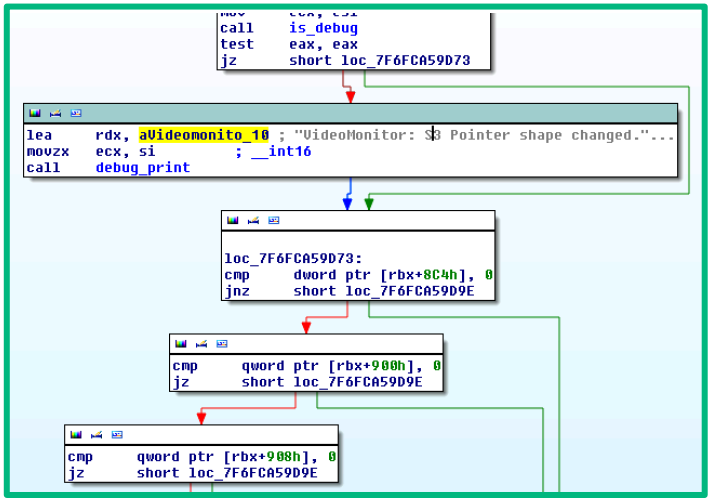


Emulated Devices



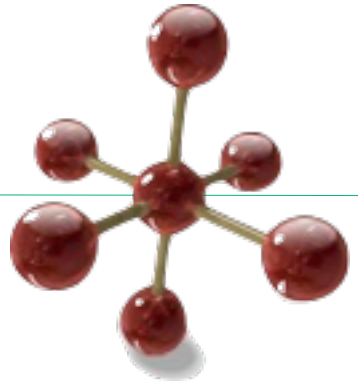
- Emulation of standard devices to allow unenlightened guest partitions
- Includes
  - Network adapter
  - S3 Trio graphic card
  - Keyboard / Mouse
  - IDE Controller
- Implemented in vmwp.exe

Emulated Devices

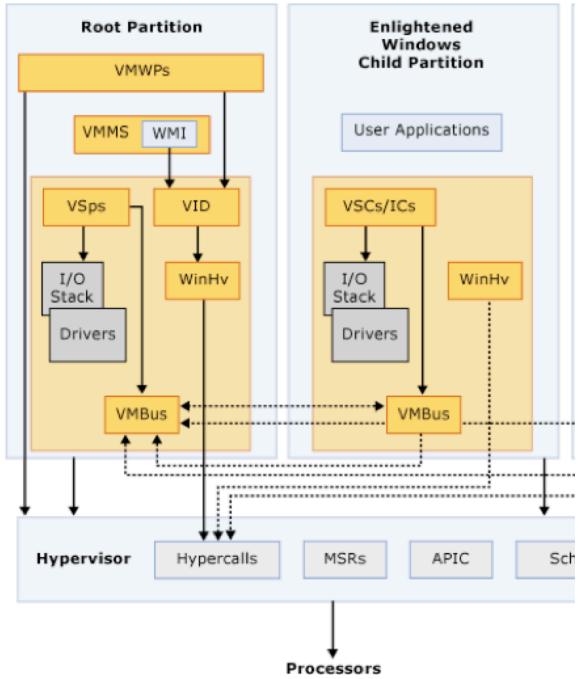


- We performed basic fuzzing (think IOFUZZ)
  - Permanent reboots
  - Worker process eats 100% CPU and requires hard kill
  - … no interesting results besides that.
- Basic static analysis
  - No symbols, but debug strings.
  - Lots of sanity checks.

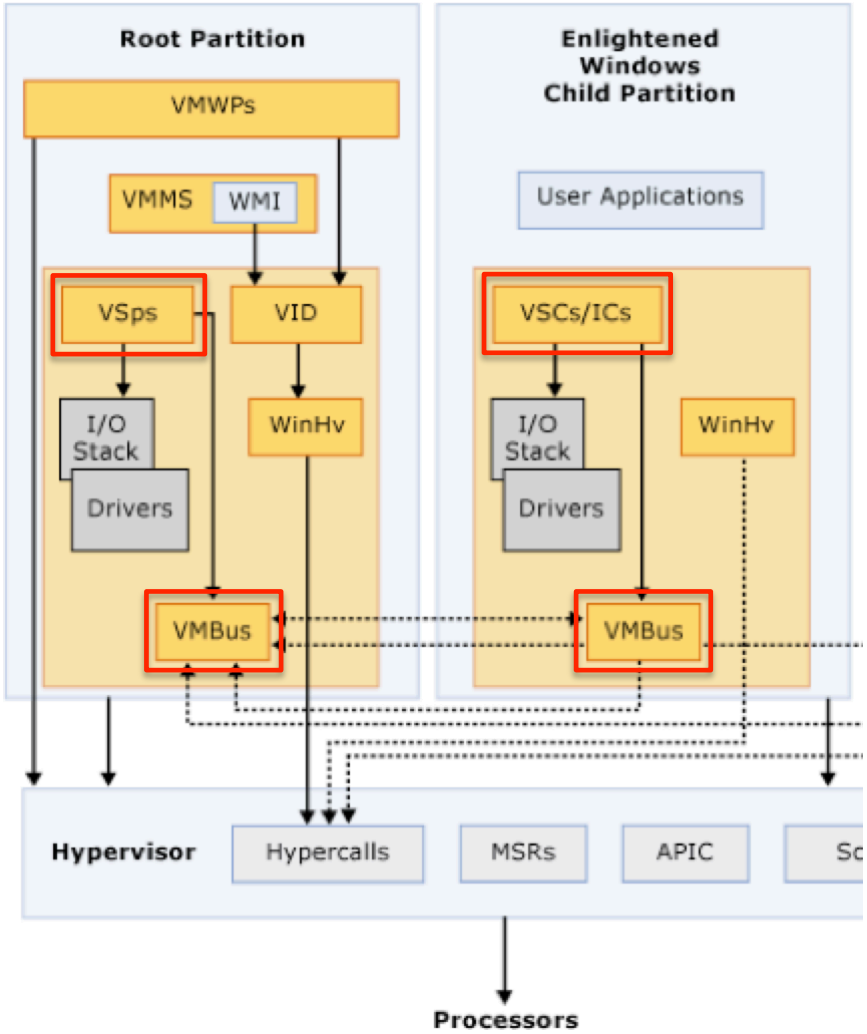
VMBus



VMBus

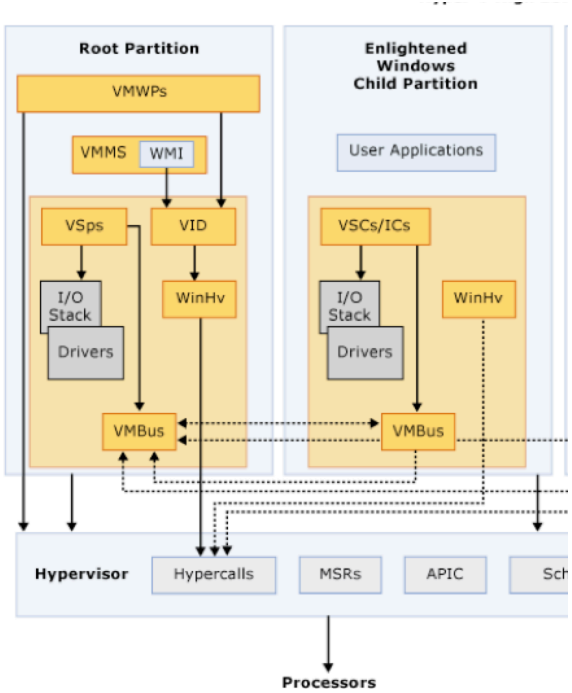


- Message bus for communication between partitions
- Default configuration:
  - Only communication to and from root partition allowed
- Memory pages that are mapped for multiple partitions
  - Heavily used for performance critical tasks in enlightened partitions
- Large attack surface



VMBus Components

VMBus Architecture



- Initialization:
  - PostMessage HyperCall
    - Is used to create VMBus between Partitions
  - Negotiates/sets up shared memory sections
- Communication:
  - One or more channels are used to isolate communication between different functionality
    - Think of IP transport and TCP connections
  - Each channel has two ring buffers, one for inbound and one for outbound communication
  - Large transfers are performed via GPADLs

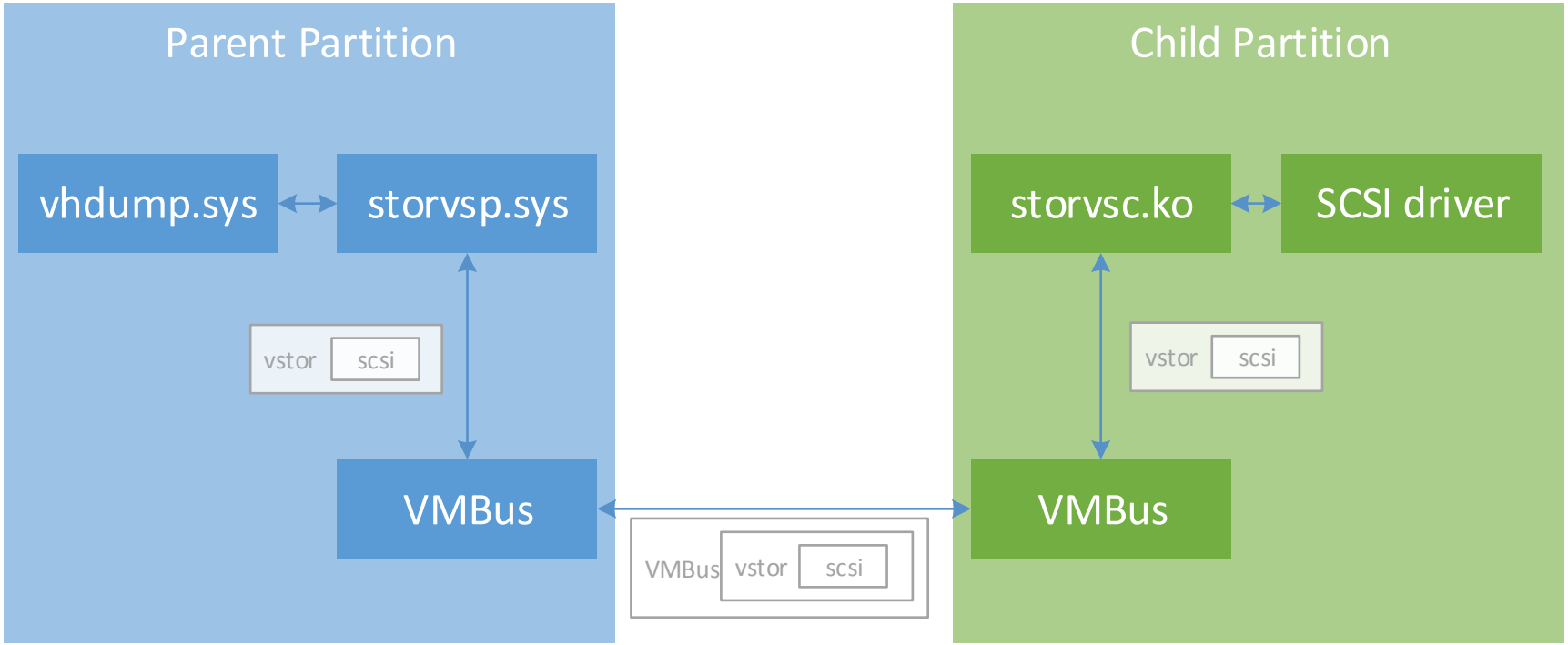
VMBus-based Services

ERNW providing security.

Term	Description	Driver
Networking	VM traffic is sent via VMBus to the Hyper-V Virtual Switch running in the parent partition.	Child: hv_netvsc.ko Parent: vmswitch.sys
Storage	Child partitions send (mainly) SCSI commands encapsulated in so-called *vstor* packets.	Child: hv_storvsc.ko Parent: storvsp.sys, vhdump.sys
Utilities	Different minor functionality such as heartbeats, time synchronization, or shutdown is available via SyncIC messages	Child: hv_util.ko Parent: vmwp.exe
Framebuffer	Virtual child console in parent Hyper-V Manager tool	Child: hyperv_fb Parent: vmwp.exe

- Network and Storage: Interesting attack surface
  - Parent endpoints in kernel space
  - In contrast to VMWP.exe running under non-admin user
- Potential vulnerabilities:
  - Protocol/logical flaws
  - Memory corruption in parent partition drivers/processes

Storage


Parent Partition
vhdump.sys
storvsp.sys
vstor
scsi
VMBus
VMBus
vstor
scsi
Child Partition
storvsc.ko
SCSI driver
vstor
scsi
VMBus

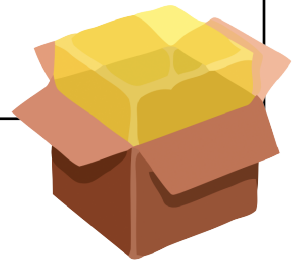
ERNW providing security.

vstor_dmp

```
# ./vstor_dmp
0x000000: 03 00 00 00 01 00 00 00 ........
0x000008: 00 00 00 00 34 00 00 00 ....4…
0x000010: 02 00 00 00 0a 14 00 00 ........
0x000018: 00 20 00 00 2a 00 07 c5 . ..*…
0x000020: ee 70 00 00 10 00 00 00 .p......
0x000028: 00 00 00 00 00 00 ......
Operation:        03 00 00 00(VSTOR_OPERATION_EXECUTE_SRB)
Flags:            01 00 00 00
Server Status:    00 00 00 00
```

vstor_packet

0	32
Operation ENUM (e.g. RESET_LUN)	Flags (currently one, RequestCompletion)
Server Status	
Payload, up to 52 Bytes e.g. SRB, Channel Properties, or WWN Packet	



VMBus Packet

0 32

Type	DataOffset
Length	Flags
Transaction ID	
Reserved	
Rangecount	
Page Buffer Range (array of PFNs, multiples of 16 bytes)	
Payload e.g. vstor_packet	

Analysis

- Performed so far:
  - Dumb fuzzing
    - →Guest VM crashes due to VSC failures
  - Manual protocol analysis
    - No results, but difficult to do due to high amount of VMBus packets
    - → Debugging is PITA
  - Basic static analysis
    - No interesting results
    - Checked for banned functions (strcpy, sprintf, you name it)
      - Not too surprising, SDL works here

ERNW providing security.

Basic Fuzzing

- Implemented using kprobes
  - https://www.kernel.org/doc/Documentation/kprobes.txt
- Allows to hook (almost) arbitrary kernel functions and tamper with existing data.
- Approach:
  - Hook `packet_send` functions and tamper with packet arguments.

Hypercalls

Hypercall Interface

5.6.1 HvCreatePartition

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

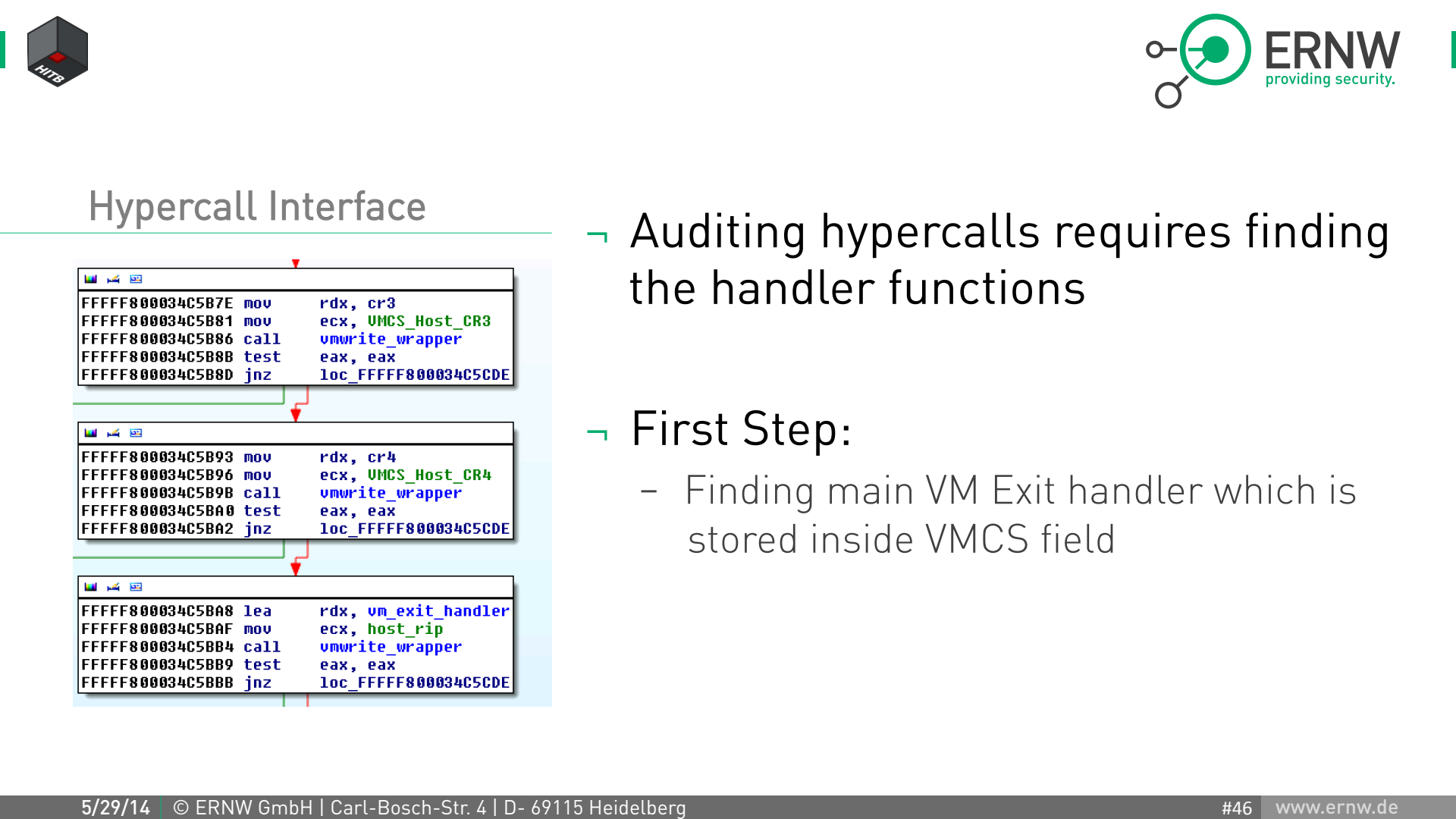
Wrapper Interface

```
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)

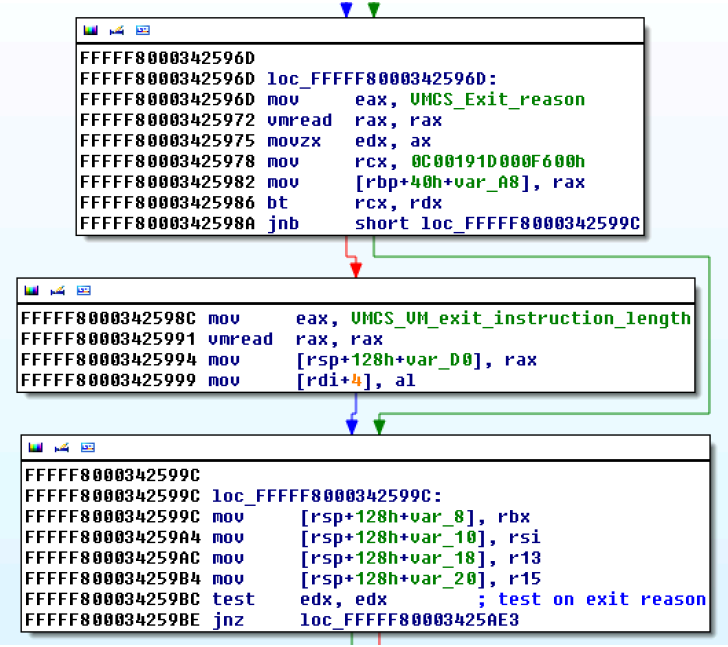
- Like system calls but for communication between VM kernel and hypervisor
- Documented interface
  - And known security boundary
  - RCX = Call Number
  - RDX = Input GPA
  - R8 = Output GPA
- Used by enlightened partitions to improve performance
- Root partition manages other partition using hypercalls



Hypercall Interface

- Auditing hypercalls requires finding the handler functions
- First Step:
  - Finding main VM Exit handler which is stored inside VMCS field

Hypercall Interface



- VM Exit handler looks similar in all VMM implementations:
  - Store Guest State
  - Perform action depending on VMCS Exit Reason
  - Restore Guest State
- Hypercalls are executed using “vmcall” instruction.

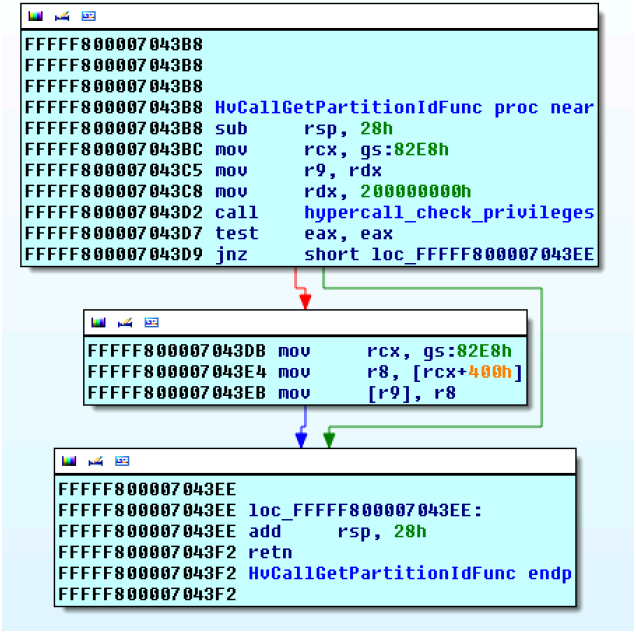
ERNW providing security.

Hypercall Interface

```
dq offset HvCallSwitchVirtualAddressSpaceFunc
dw HvCallSwitchVirtualAddressSpace      ; cal
dw 0                                    ; rep
dw 8                                    ; inp
dw 0                                    ; unk
dw 0                                    ; out
db 2 dup(0), 43h, 3 dup(0)              ; unk
dq offset HvCallFlushVirtualAddressSpaceFunc;
dw HvCallFlushVirtualAddressSpace       ; cal
dw 0                                    ; rep
dw 18h                                  ; inp
dw 0                                    ; unk
dw 0                                    ; out
db 2 dup(0), 43h, 3 dup(0)              ; unk
dq offset HvCallFlushVirtualAddressListFunc; 
dw HvCallFlushVirtualAddressList        ; cal
dw 1                                    ; rep
dw 18h                                  ; inp
dw 8                                    ; unk
dw 0                                    ; out
db 2 dup(0), 43h, 3 dup(0)              ; unk
dq offset HvCallGetLogicalProcessorRunTimeFun
dw HvCallGetLogicalProcessorRunTime     ; cal
dw 0                                    ; rep
dw 8                                    ; inp
dw 0                                    ; unk
dw 20h                                  ; out
```

- RCX register stores hypercall number
- Is used as index into handler table
- Handler Table allows quick identification of all handler functions

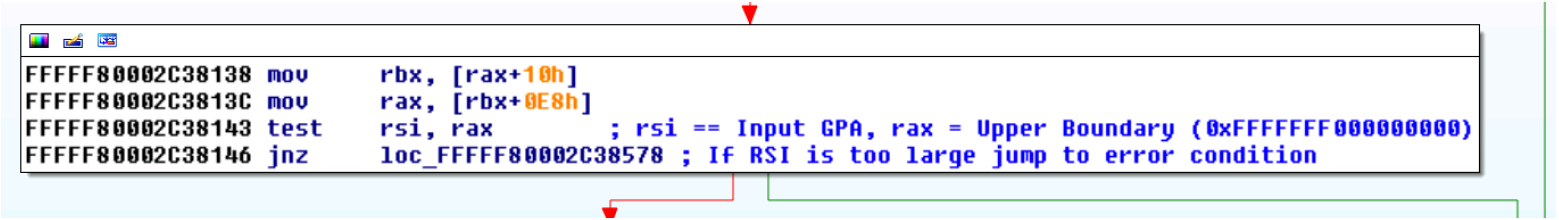
Hypercall Interface



- Hypercall handler are quite simple
- RCX = Input / RDX = Output
  - Copied from original input pages
- Early permission checks guard most complicated handler functions
- Performed fuzz testing but no interesting results.

However…

- Sanity checks are executed before calling the actual hypercall handler:
  - Hypercall comes from Ring 0?
  - Input and Output GPA are aligned?
  - Input and Output GPA look sane?



The Bug

```
mov     r8, [rcx+ept_object.pte_table]
mov     rax, rdx
shr     rax, 12
mov     rbp, rdx
mov     rsi, rcx
mov     r8, [r8+rax*8]  ; rax = rdx = r
                        ; r15 = Hyperca
mov     rax, 8000000000000000h
test    rax, r8
jz      return_loc
```

- Guest GPAs >= 0x1000000000 and < 0x1000000000000 set special error condition
- Error function tries to map GPA to system PTE
  - r8 points to table
  - rax = input_gpa >> 12;
- Classic out-of-bound access because GPA can be almost unlimited large.
- The trigger?
  - `hypercall(0xXY, input_address = 0x4141414141414141...)` ☺

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)

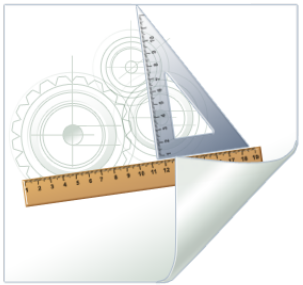
ERNW
providing security.

Some thoughts on exploitation...

- DoS trivially possible by accessing invalid memory.
- Privilege Escalation possible?

```
1  long pte = gpa_to_spa[guest_physical_addr >> 12]
2  if (!(pte & 0x8000000000000000)||(pte & 0x1E00000000000000)
3          return 0;
4  if (pfn(pte) != SPECIAL_ADDR)
5          return 0;
6  if (last_three_bits(pte) & 0b111)
7          return 1;
8  acquire_locks();
9  func_x(pte | 5, guest_physical_addr)
10 return 1;
```

Some thoughts on exploitation...



- Patch adds additional range check in front and force-returns 0
  - Code flow called after vuln function returns 0 is uninteresting
- Can't read attacker controlled value
  - Upper limit + 64bit address space
- However, offset to "next" virtual machine PTE table is constant
  - Possible to access PTEs of other VMs

ERNW providing security.

Some thoughts on exploitation…

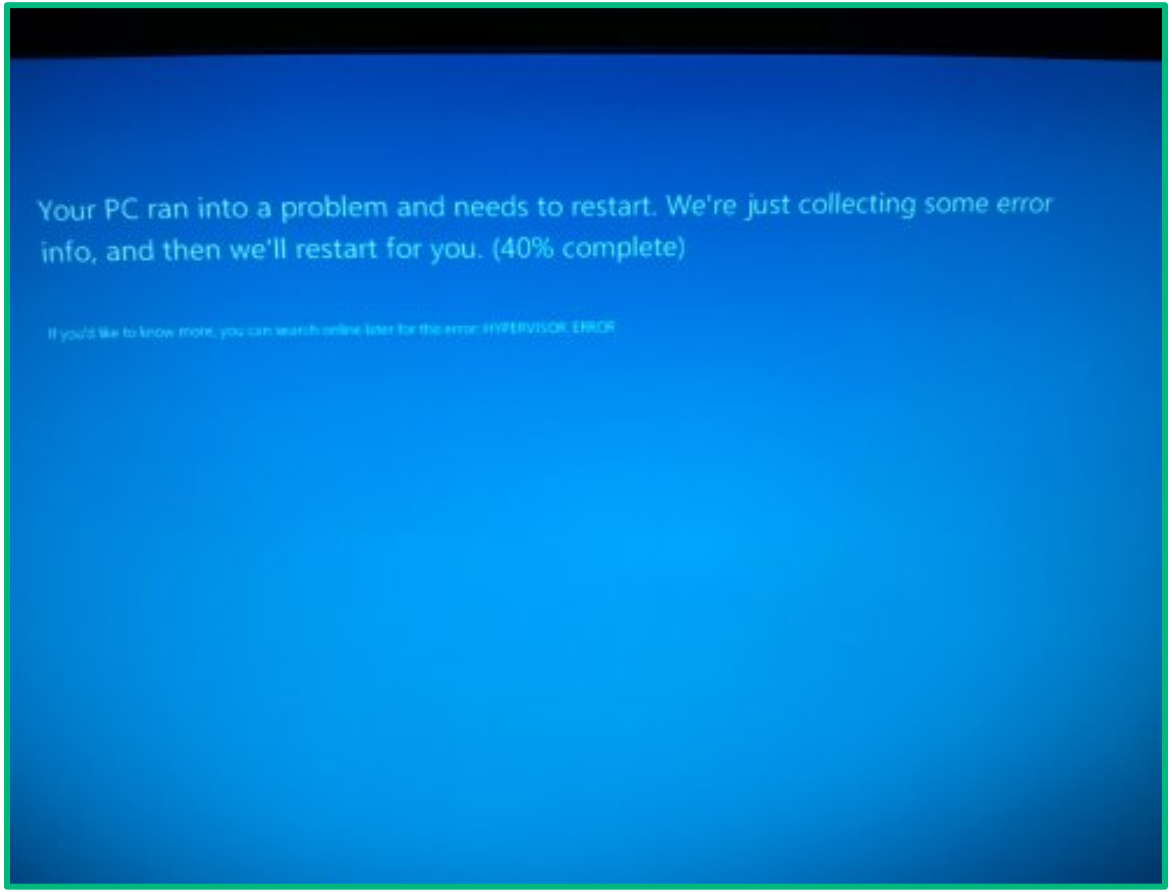
```
if (pfn(pte) != SPECIAL_ADDR)
        return 0;
```

- Limited to a “special” PFN number
  - Could not clearly identify its use case + not used by VMs in our lab :/
- Code flow after successful checks is interesting
  - But can only be minimally influenced by attacker
  - Two values used: PTE and INPUT_GPA (both are more or less fixed)
- Classic Memory corruption not possible
  - But triggering a PTE map from/to a different partition would be game over as well.

ERNW
providing security.

Some thoughts on exploitation...

- Patch is noisy
  - *Pretty sure*™ that we did not miss any additional security relevant modifications
- Might have missed something completely obvious..


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

Future Work

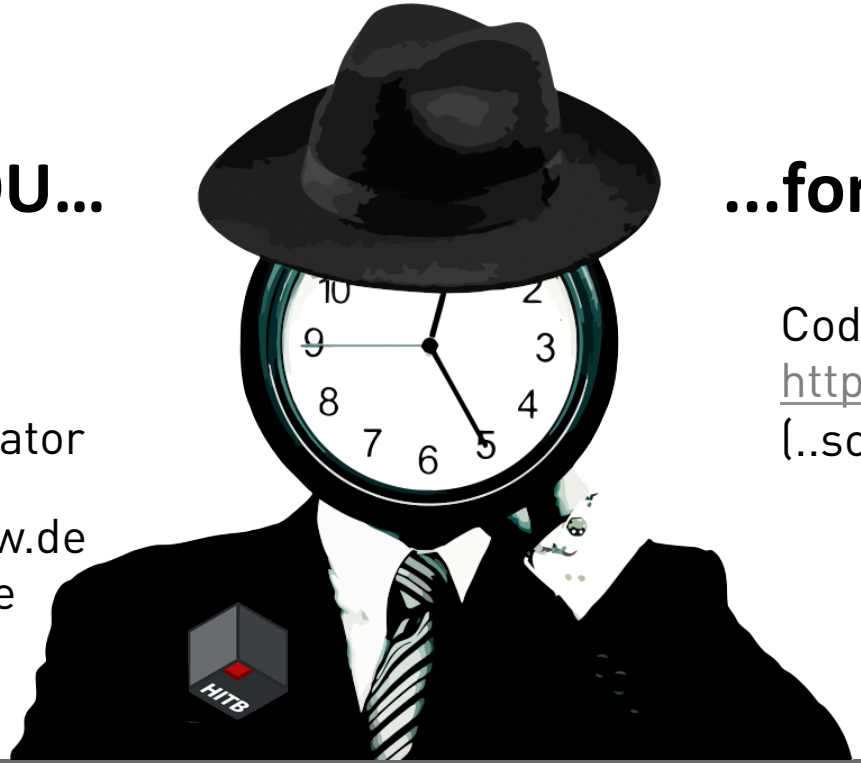
- Improve fuzzing
  - Think of analysis approach that allows high coverage and automation without constantly crashing the guest.
- Look at new stuff
  - RemoteFX
  - Gen2 VMs
  - Xbox One
- Go deeper
  - VM exit handling

Conclusions

- Hypervisor security & research is not hard.
  - Just software and big attack surface.
- Hypervisor security & research is hard.
  - .. but you learn a lot!
- Hyper-V is solid software
  - .. but still has critical security bugs.
- Hypervisors are getting faster/more features but not more secure.
  - .. still not enough public research!
- *Make the theoretical practical.*

There's never enough time...

THANK YOU...



...for yours!

@_fel1x
@uchi_mata
@Enno_Insinuator

fwilhelm@ernw.de
mluft@ernw.de
erey@ernw.de

Code & Slides:
https://www.insinuator.net
(..soon)

Sources & Mentions

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

Disclaimer

All products, company names, brand names, trademarks and logos are the property of their respective owners!