Hypervisors in Your Toolbox

Monitoring and Controlling System Events with HyperPlatform

Satoshi Tanda
Ahmed Samy

Takeaway

- HyperPlatfrom is
  - the simple hypervisor provides you ability to
  - flexibly handle a new class of system events, and
  - write hypervisor-based tools with high compatibility and efficiency
- Actions: revisit unaddressed challenges, review ability of virtualization technology, and develop ideas and solutions

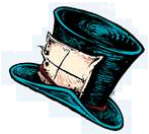
About Us

- Low-level tech enthusiasts
- (Reverse|Software) engineers interested in Windows kernel
- Satoshi Tanda
  - Developer of HyperPlatfrom
  - tanda.sat@gmail.com
  - @standa_t
- Ahmed Samy
  - Developer of KSM – hypervisor derived from HyperPlatform
  - asamy@protonmail.com

Background

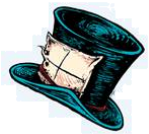
- Virtualization technology (VT) is very handy
  - Dev & QA environment
    - Having multiple OS versions
  - Reverse engineering
    - Malware analysis
    - Vulnerability discovery
  - Additional security
    - Virtualization-based security (a.k.a Hyper Guard)
- Countless of research & implementation

Challenges



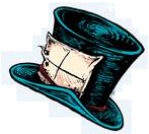
- VT is not accessible to Windows-centric researchers
  - No suitable hypervisor as a platform
- Existing lightweight hypervisors for Windows?
  - e.g., HyperDbg, VirtDbg, MoRE, SimpleVisor
  - No support of x64, multiprocessors, and Win10
  - Made for particular purpose, or overly simple for practical usage
- Comprehensive, consumer-oriented hypervisors?
  - e.g., Xen, VirtualBox, Bochs
  - Significantly complicated code base, lib dependencies, tool sets
  - Excessively slow

Requirements



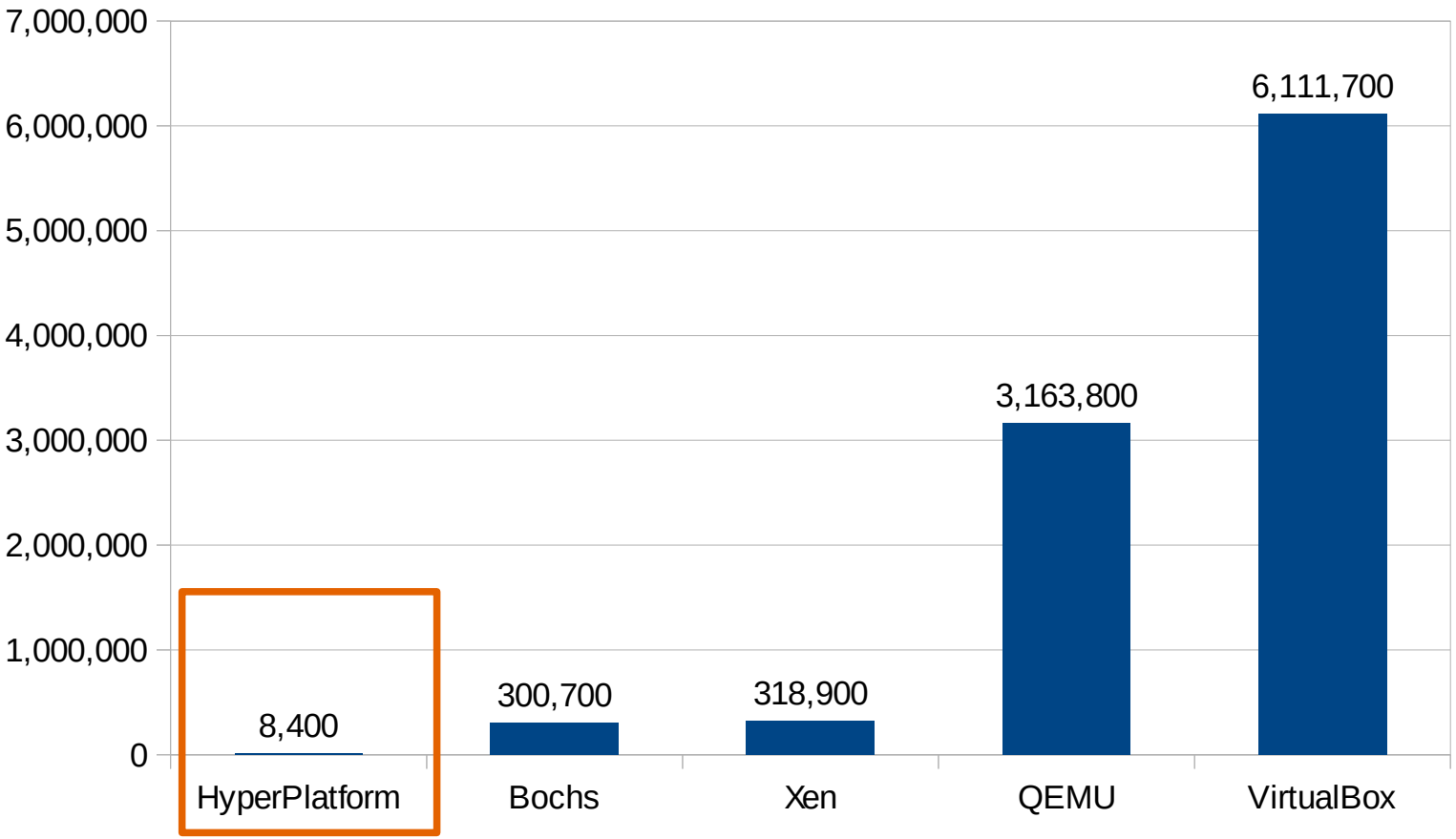
- Compatibility
  - Support all the major Windows w/o no special settings
- Flexibility
  - Provide all key features of VT
  - Be applicable to a wide range of scenarios
- Simplicity
  - Be small, compatible with Windows dev tool sets, free from 3rd party libraries
  - Be documented well
- Efficiency
  - Not introduce excessive negative performance impact

Solution: HyperPlatform

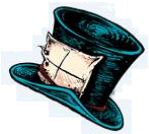


- Compatibility
  - Supports Windows 7-10 on x86/x64 (>= Nehalem)
- Flexibility
  - Designed as a platform for variety of scenarios
- Simplicity
  - Small (8KLOC) + full documents
  - Can be compiled on Visual Studio w/o any 3rd party libs
  - Can be debugged with Windbg + VMware
  - C++ and STL can be used
- Efficiency
  - Fast (about 10% of overhead)

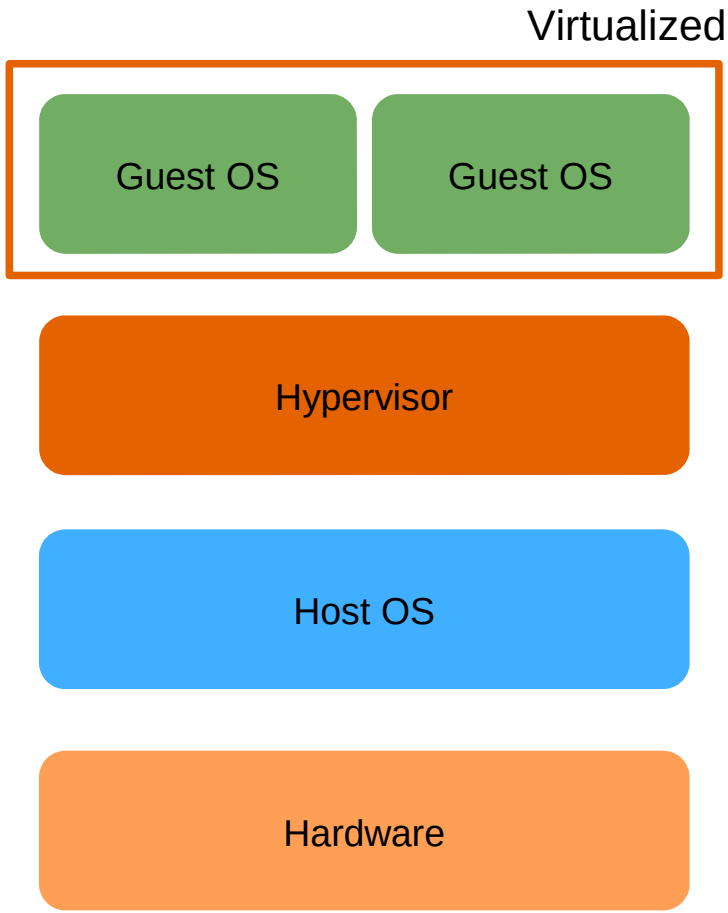
How Small?


7,000,000
6,000,000
5,000,000
4,000,000
3,000,000
2,000,000
1,000,000
0
8,400
300,700
318,900
3,163,800
6,111,700
HyperPlatform
Bochs
Xen
QEMU
VirtualBox

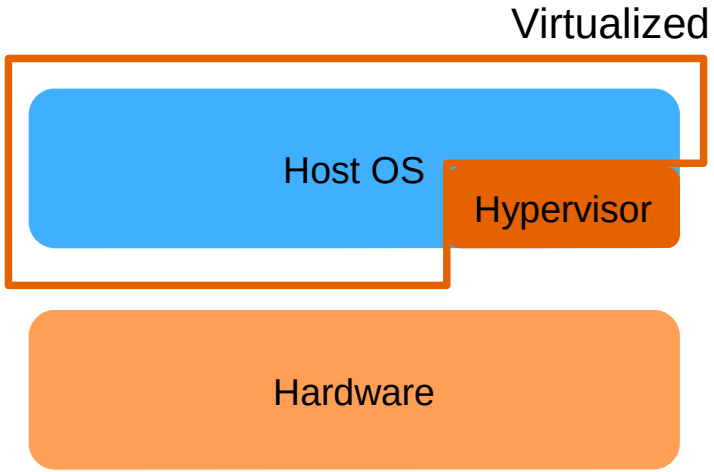
Overview: No Guest Architecture



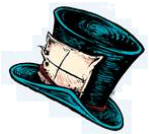
Type 2 Hypervisor

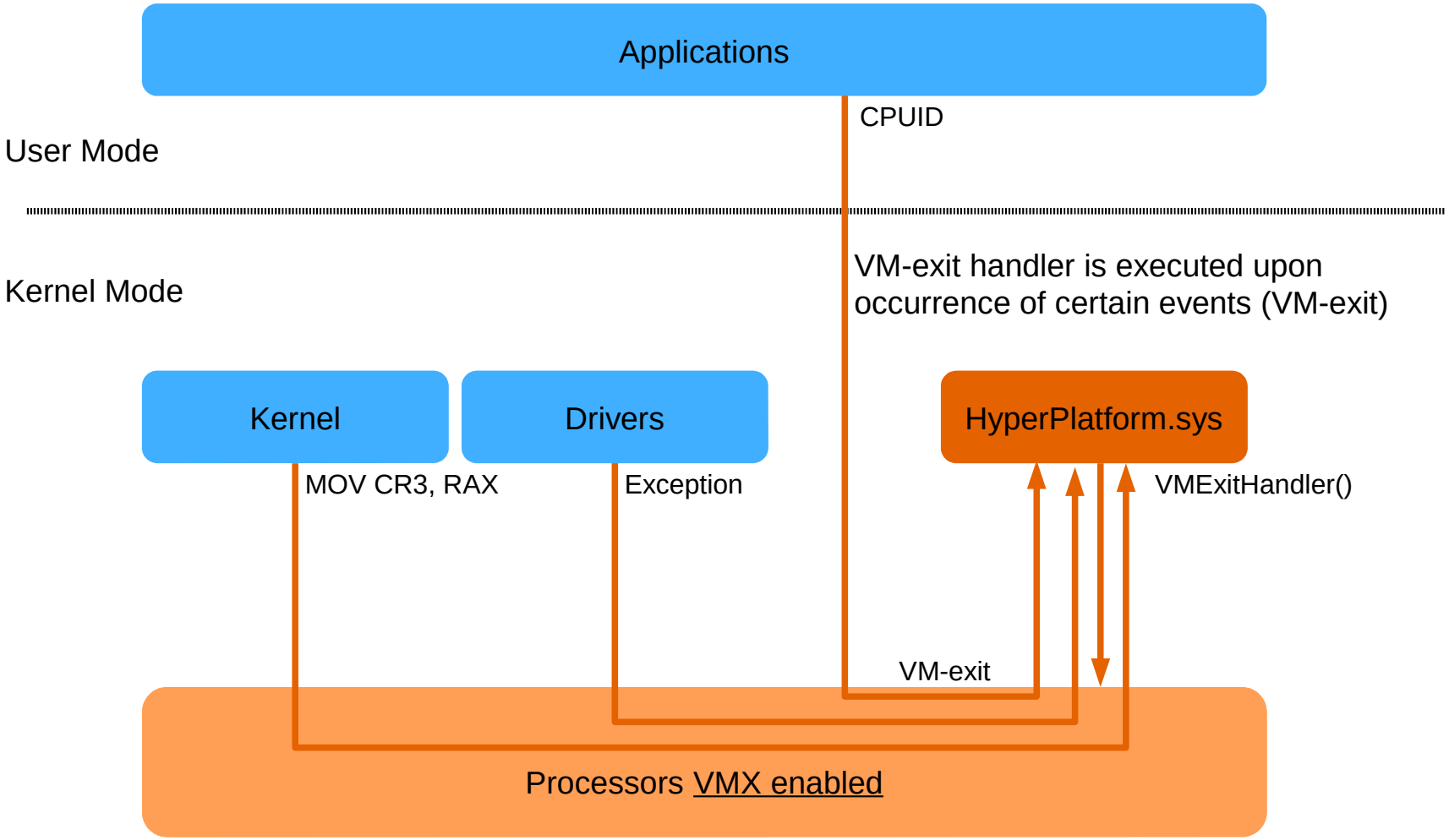


HyperPlatform

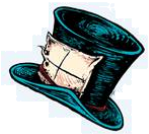


Overview: No Guest Architecture





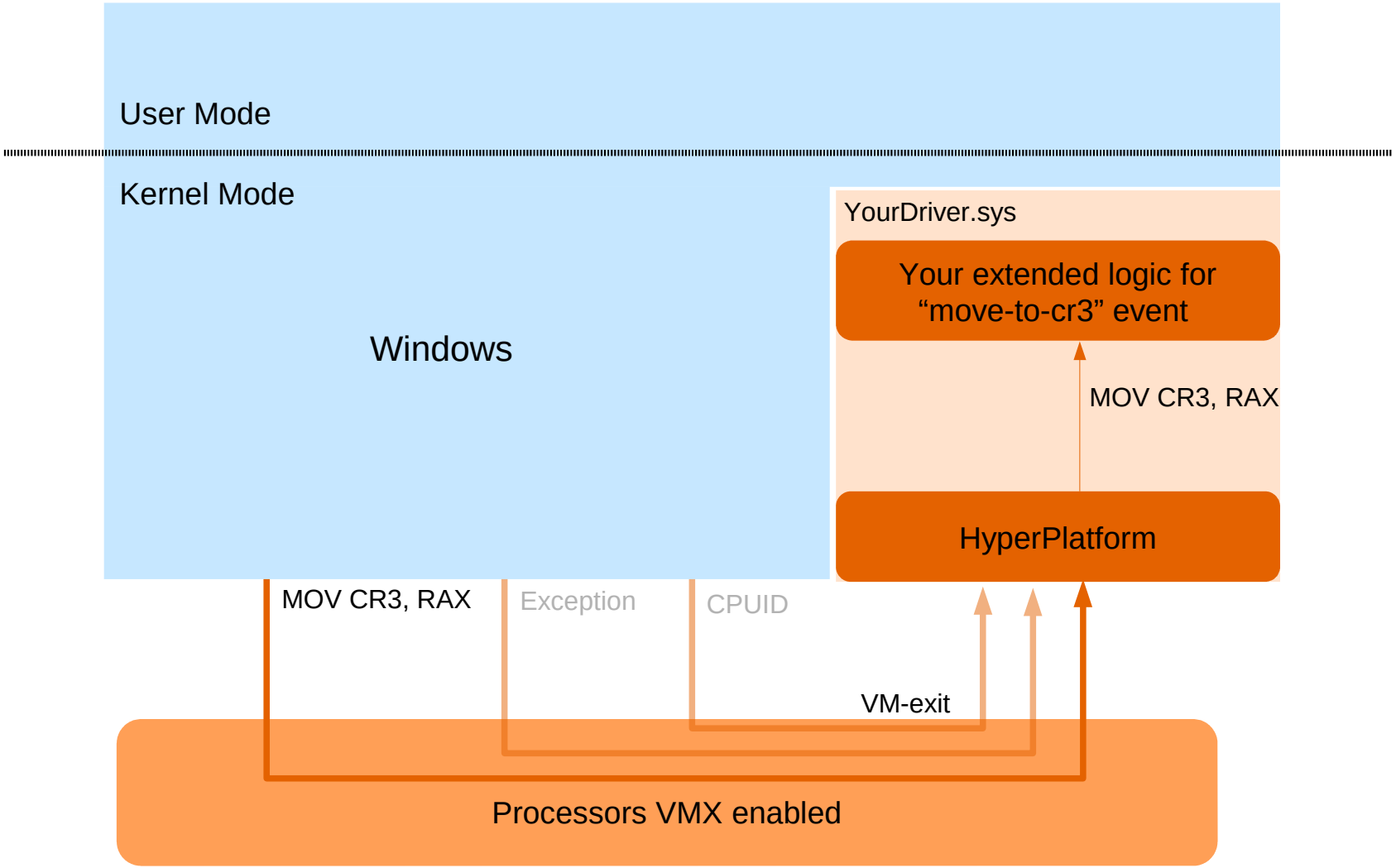
Overview: VM-exit Handler



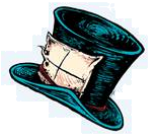
```
void VMExitHandler(
  GuestRegisters* context,
  int exit_reason)
{
  switch (exit_reason)
  {
    case VMEXIT_CPUID:
      CpuidHandler(context); break;
    case VMEXIT_EXCEPTION:
      ExceptionHandler(context); break;
    //...
  }
}
```

- Invoked on VM-exit
- Context of the system and VM-exit reason are given
- Handle an event accordingly

As a VM-exit Filtering Platform

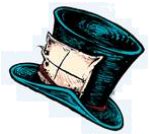


Recap – Advantages of VT



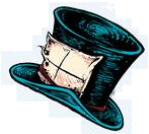
- VM-exit is a new class of events
  - access to system registers
  - occurrence of exceptions and interruptions
  - execution of certain instructions
  - access to memory
- VM-exit handler is flexible
  - Can return different register values and memory contents
- None of them is easy to achieve without VT

Demo

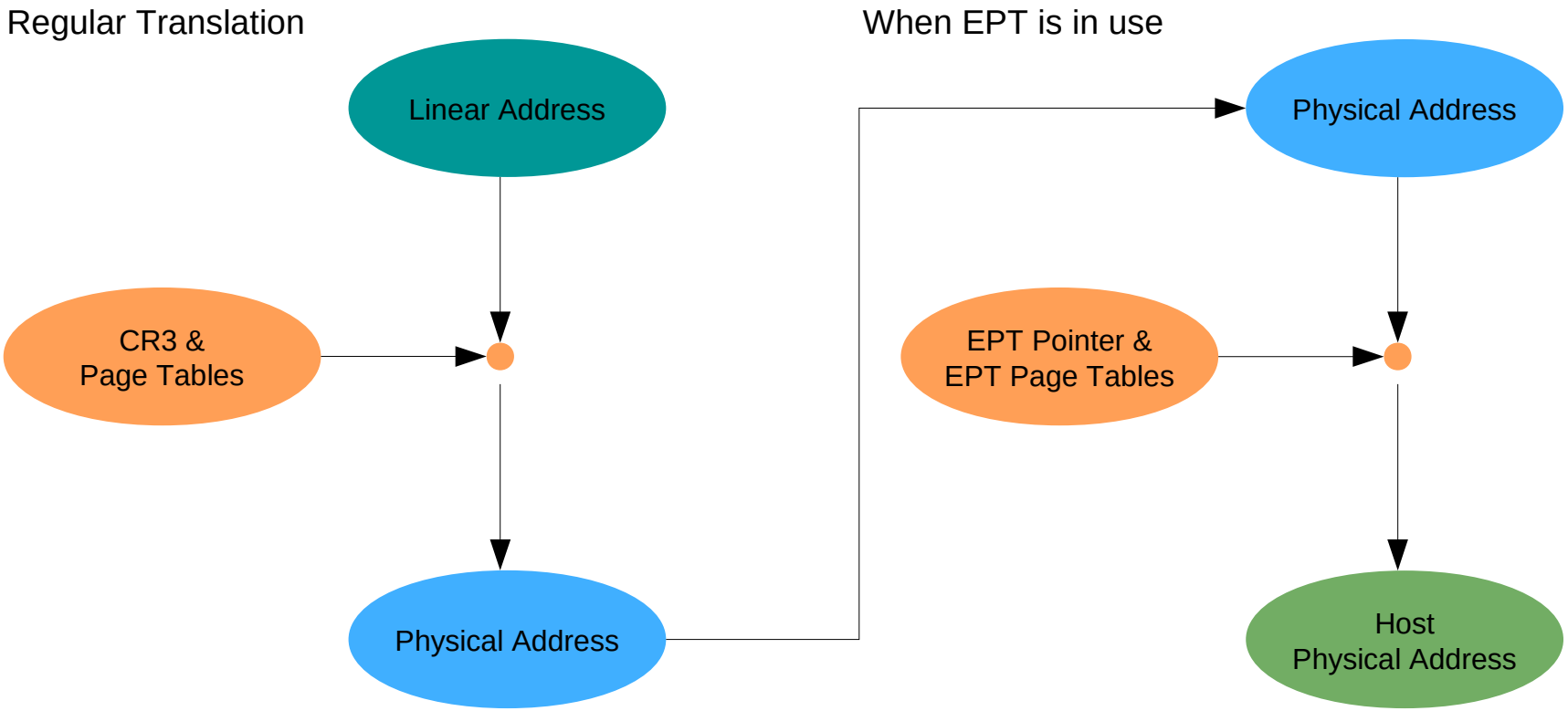


- Straightforward development process
- Example application to security
  - Detection of system resource modification (CR4.SMEP)
  - vs. Capcom.sys
    - See: twitter.com/TheWack0lian/status/779397840762245124

Extended Page Tables (EPT): Translation



- Additional address translation



Extended Page Tables (EPT): Logic & Data

- Resembles to x64 address translation
  - Use physical address
    - instead of linear address
  - Use EPT pointer
    - instead of CR3
  - Use EPT page tables
    - instead of page tables

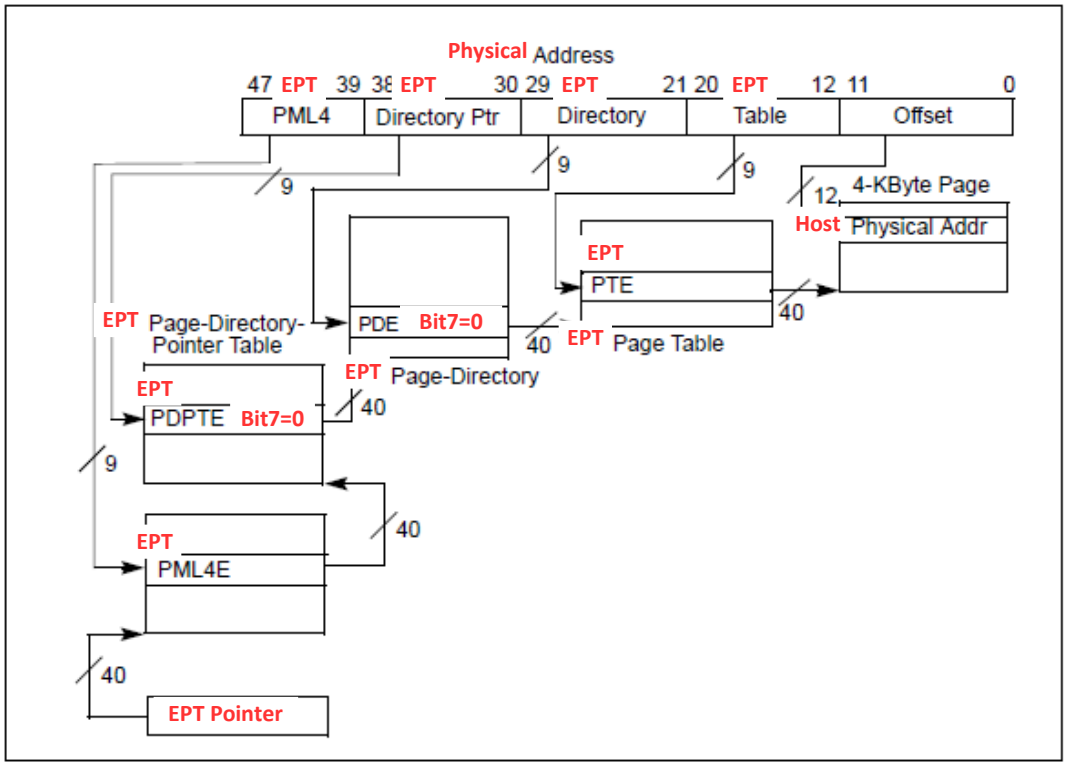


Figure 4-8. Linear-Address Translation to a 4-KByte Page using IA-32e Paging

Extended Page Tables (EPT): Protection

- Defines protection of each page along with translation
  - Page Table Entry

Table 4-19. Format of an IA-32e Page-Table Entry that Maps a 4-KByte Page

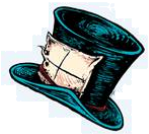
Bit Position(s)	Contents
1 (R/W)	Read/write; if 0, writes may not be allowed to the 4-KByte page referenced by this entry (see Section 4.6)
(M–1):12	Physical address of the 4-KByte page referenced by this entry
63 (XD)	If IA32_EFER.NXE = 1, execute-disable (if 1, instruction fetches are not allowed from the 4-KByte page controlled by this entry; see Section 4.6); otherwise, reserved (must be 0)

  - EPT Page Table Entry

Table 28-6. Format of an EPT Page-Table Entry that Maps a 4-KByte Page

Bit Position(s)	Contents
0	Read access; indicates whether reads are allowed from the 4-KByte page referenced by this entry
1	Write access; indicates whether writes are allowed to the 4-KByte page referenced by this entry
2	Execute access; indicates whether instruction fetches are allowed from the 4-KByte page referenced by this entry
(N–1):12	Physical address of the 4-KByte page referenced by this entry[1]

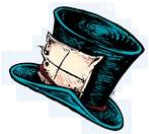
Application #1: Memory Access Monitoring



- Protection violation causes VM-exit (instead of #PF)
- Any access to arbitrary memory can be monitored
  - `ept_pte.writable = false;`

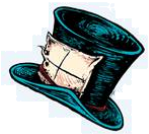
    will cause VM-exit on any write access
- We can write hypervisor-based protection for sensitive regions
  - ```
    pa = MmGetPhysicalAddress(HalDispatchTable);
    EptGetEptEntry(pa)->writable = false;
    ```
    ```

Application #1: Demo



- Example application to security
  - Prevention of LPE exploit for ngs64.sys (CVE-2014-0816)

Application #2: Stealth API Hook



- Can force system to read from fake pages, while hooks are executed
  - Arrange two EPT settings for PA to fake, e.g., 0x1000(PA):

Settings	PA	Host PA	Protection
ForExec	0x1000	0x1000	--E
ForRW	0x1000	0x2000	RW-

0x1000(HPA) = Hooked code

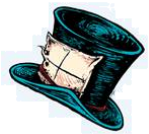
```
0x1000:
    jmp trampoline
    push rbx
    push rbp
```

0x2000(HPA) = Fake contents to show

```
0x2000:
    mov [rsp+18h], r8
    push rbx
    push rbp
```

  - Use ForExec as default => hooked code is executed normally

Application #2: Stealth API Hook



- Hiding hook from read

```
...
...
mov rax, [0x1000(PA)]
...
...
```

EPT violation
VM-exit

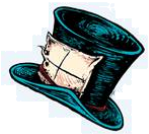
MTF VM-exit

- Switch to ForRW settings
- Set Monitor Trap Flag
- System refers to 0x2000 in HPA
- Clear Monitor Trap Flag
- Switch to ForExec settings

Settings	PA	Host PA	Protection
ForExec	0x1000	0x1000	--E
ForRW	0x1000	0x2000	RW-

- System reads fake contents

Application #2: Demo



- Example application to reverse engineering
  - Monitoring pool-allocation with stealth API hook
  - Invisible from the system

Limitation

- Requires hardware VT support (i.e., not run inside VirtualBox)
- No AMD-V support (Issue #2)
- Not run with other hypervisors simultaneously (Issue #14)

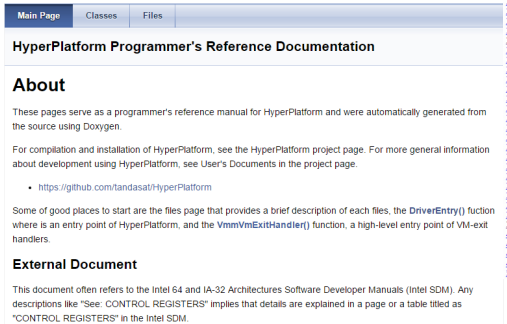
Future Work

- Memory access tracing for MMIO monitoring
- Addressing the limitation #14
- Stealth API hook for user-mode

- Looking for more ideas and feedback

Conclusion

- HyperPlatfrom is
  - the simple hypervisor provides you ability to
  - flexibly handle a new class of system events, and
  - write hypervisor-based tools with high compatibility and efficiency
- Actions: revisit unaddressed challenges, review ability of virtualization technology, and develop ideas and solutions
- Learn more at GitHub and talk to us with your ideas
  - github.com/tandasat/HyperPlatform


Main Page | Classes | Files

HyperPlatform Programmer's Reference Documentation

About

These pages serve as a programmer's reference manual for HyperPlatform and were automatically generated from the source using Doxygen.

For compilation and installation of HyperPlatform, see the HyperPlatform project page. For more general information about development using HyperPlatform, see User's Documents in the project page.

- https://github.com/tandasat/HyperPlatform

Some of good places to start are the files page that provides a brief description of each files, the DriverEntry() fuction where is an entry point of HyperPlatform, and the VmmVmExitHandler() function, a high-level entry point of VM-exit handlers.

External Document

This document often refers to the Intel 64 and IA-32 Architectures Software Developer Manuals (Intel SDM). Any descriptions like "See: CONTROL REGISTERS" implies that details are explained in a page or a table titled as "CONTROL REGISTERS" in the Intel SDM.

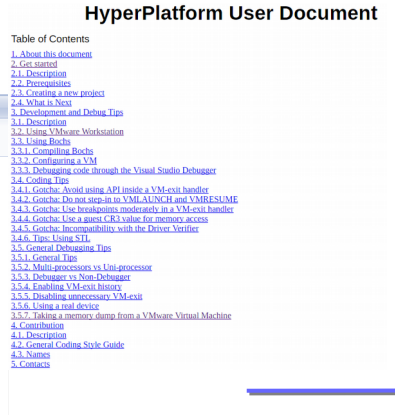

HyperPlatform User Document

Table of Contents

One more in your toolbox…

KSM

https://github.com/asamy/ksm
Ahmed Samy <asamy@protonmail.com>

A (rather) really simple x64 hypervisor with a unique feature set aimed at sandboxing, real-time virtualization.

Facts

- Written by an Ancient Egyptian Pharaoh
- Lightweight
- **Supports #VE, VMFUNC, EPTP-Switching, IDT Shadowing, Page-shadowing, and others.**
- Super fast, pure simple C code (with a bit of Assembly.
- Supports Intel processors only (>= Haswell)
- Random 3-letter name with K(ernel) in it.
- Aims real-time virtualization (sandboxing, malware detection, …).

Lost and f0wnd

- **Lost: VM exit** costs a lot of cycles! Not to mention reading/writing to VMCS is another story.
- **f0wnd: Virtualization Exceptions** (currently only **EPT violations** are supported).

#VE via Guest's IDT

- Uses IDT index 20
- Requires a 4-KByte page
- Can be caught via **exception bitmap**
- Same **severity** as **Page Faults**
- **Rare case**: it may be diverted as VM-Exit. E.g. when delivering another exception.
- "**exception mask**" to control delivery (take the VM-exit road instead if set to FFFFFFFFh)
- Very fast as it's delivered through guest's IDT in this case we must trust the running kernel.

EPTP Switching VMFUNC

Apart from **#VE** (one does not imply the other)

- **Intel** allows up to 512 entries, thus you have to allocate a page for the “EPTP list” (512 * 8).
- Easily switch EPT pointers from within guest on demand.
- Causes a vm-exit when function specified is not supported or EPTP index is too high (>= 512)
- **No CPL Checks**! (Sometimes an advantage)

IDT Hooking

PatchGuard protects the IDT, we cannot simply just modify the entry and get away with it, eventually we end up with a bug-check. The solution is quite simple:

- By enabling the desc-table (GDT/IDT/LDT/TR) exiting bit in secondary processor control, we can easily establish a “shadow IDT”.
- We also allocate a separate “shadow” IDT for obvious reasons.
- Viola.

An example

Note: It's simpler to do this, but other ideas are too wide to implement as an example.

- **If we wanted to hook a kernel function** (we use 3 **EPTP**: **EXEC**, **RW**, **RWX** normal):

1. **EXEC**: **eXe-cute-only**, redirects to our shadow PFN.
2. **RW**: **rw-**, and redirects to **normal page in case of read/write fault (PG for example)**
3. **RWX-normal**: **redirects to normal page with all access-writes so we can call "original" function and switch back when done.**

An example (contd.)

- You may have guessed: By using **VMFUNC**, we can simply just switch to appropriate **EPTP** on **#VE**
- We simply switch to the RWX EPTP (“Normal”) when we need to call an original function from within a hooked function and switch back to EXHOOK.
- Still saves a nice amount of time. Violations like this are likely to occur. Standard interrupt handling.
- Same approach can also be applied to virtualizing user processes / kernel device drivers, etc.

Cons

- **Xen supports nesting for this, but difficult to get running.**
- KVM struggles with Windows VM nesting and does not support it.
- Other good VMs are either not open source or just do not support it at all.
- Easy to make mistakes with but end result is very nice and fast.

Thank You

https://github.com/asamy/ksm

Ahmed Samy

asamy@protonmail.com

KSM Alum

Open for opportunities

Satoshi Tanda

tandasat@gmail.com

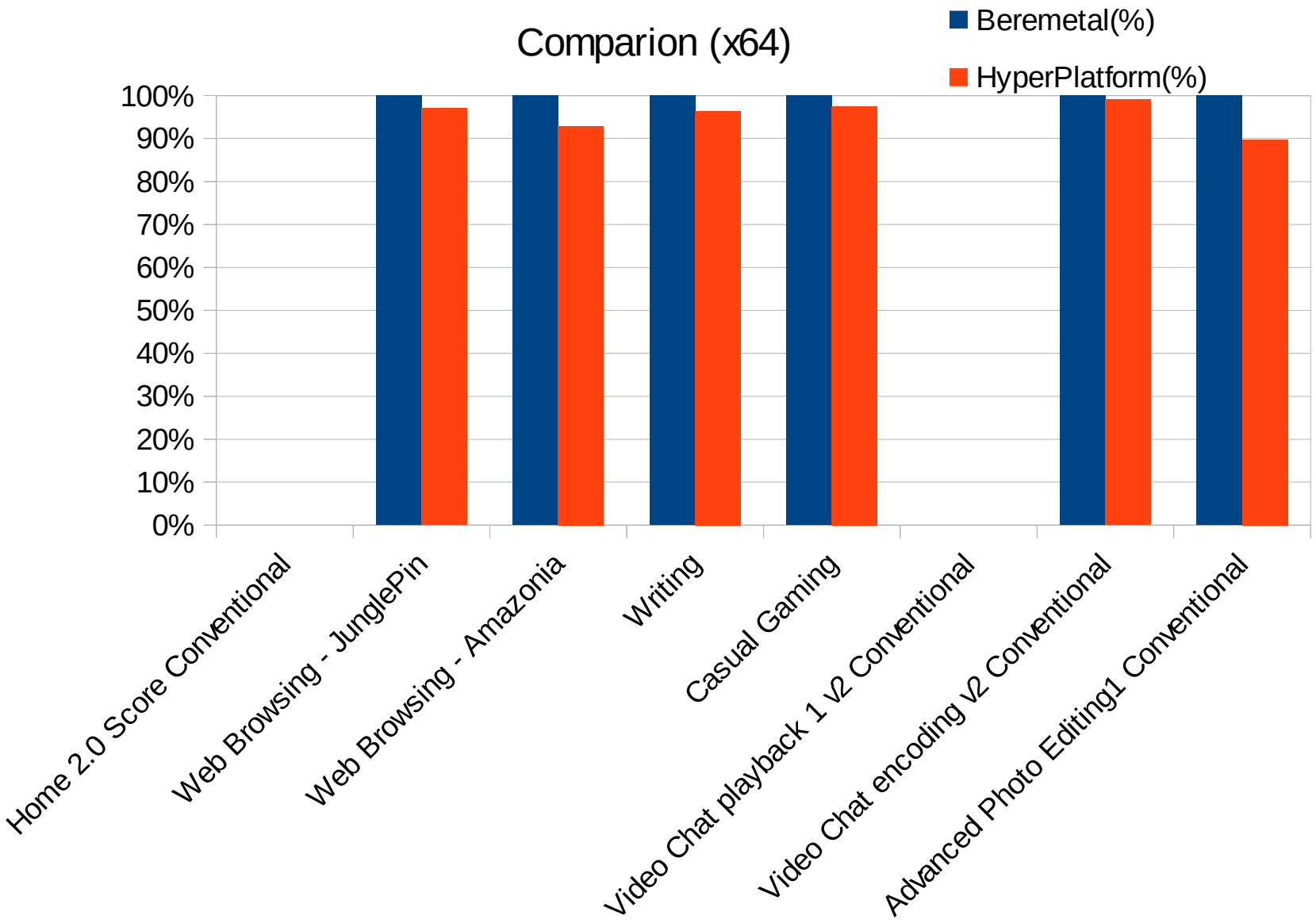
HyperPlatform developer

Questions?

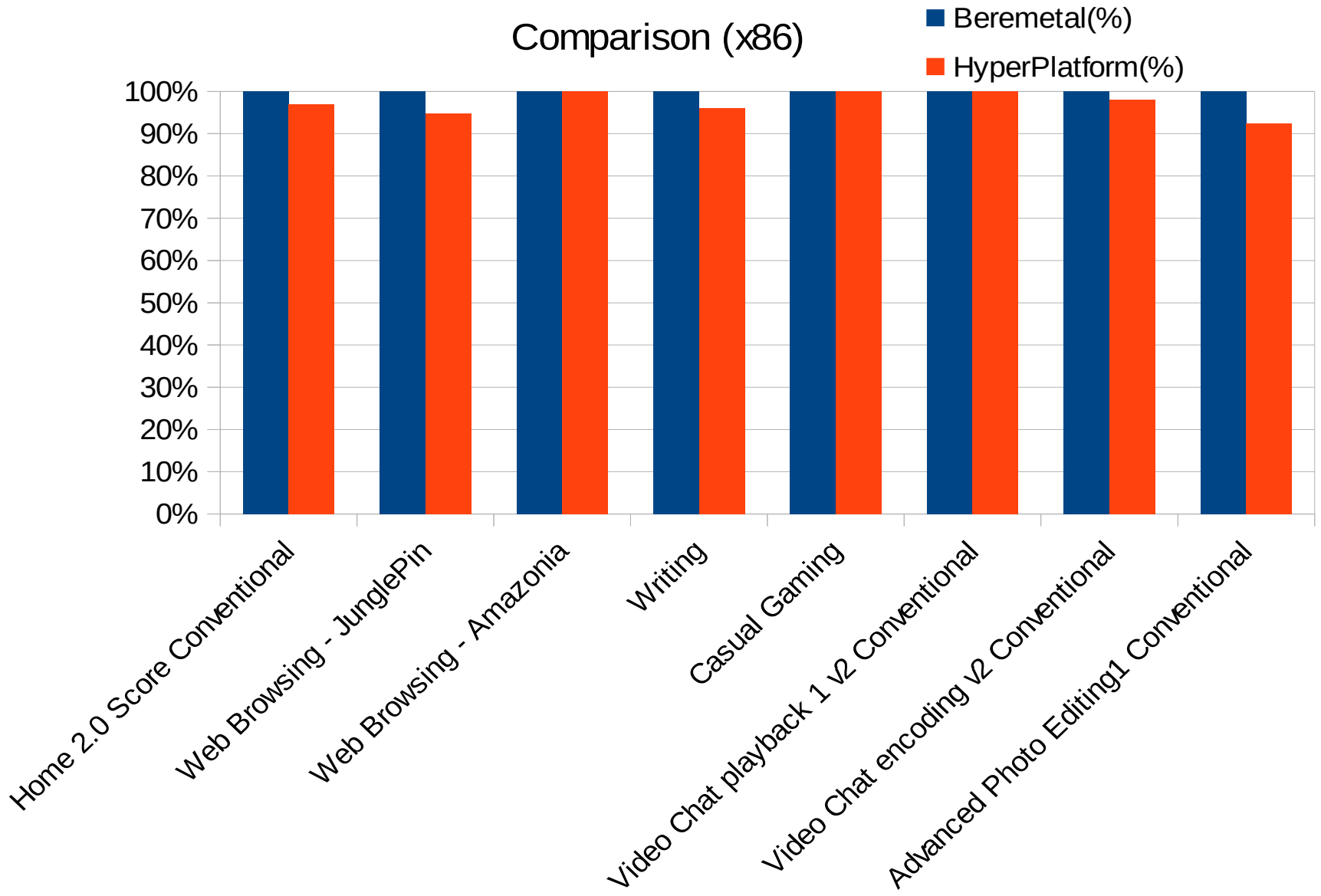
This page was intentionally left blank.

Resources

- Demo2: MemoryMonRWE
  - https://github.com/tandasat/MemoryMon/tree/rwe_bh
- Demo3: DdiMon
  - https://github.com/tandasat/DdiMon/tree/demo_bh
- SimpleVisor – excellent resource for learning hypervisor programming
  - https://github.com/ionescu007/SimpleVisor
- Benchmark with PCMark8 (bare-metal vs HyperPlatform)
  - On Win10 x64 (Haswell)
  - http://www.3dmark.com/compare/pcm8hm3/264434/pcm8hm3/264440
  - On Win10 x86 (Westmere)
  - http://www.3dmark.com/compare/pcm8hm3/264436/pcm8hm3/264442



* For some reasons, PCMark8 never generated scores for the Video Chat playback test on the system regardless of whether HyperPlatform was installed or not


Comparison (x86)
Beremetal(%)
HyperPlatform(%)
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Home 2.0 Score Conventional
Web Browsing - JunglePin
Web Browsing - Amazonia
Writing
Casual Gaming
Video Chat playback 1 v2 Conventional
Video Chat encoding v2 Conventional
Advanced Photo Editing1 Conventional