Bitdefender

Zero-Footprint Guest Memory Introspection With Xen

Mihai DONȚU – Technical Project Manager, Bitdefender

CloudOpen, August 19th, 2015

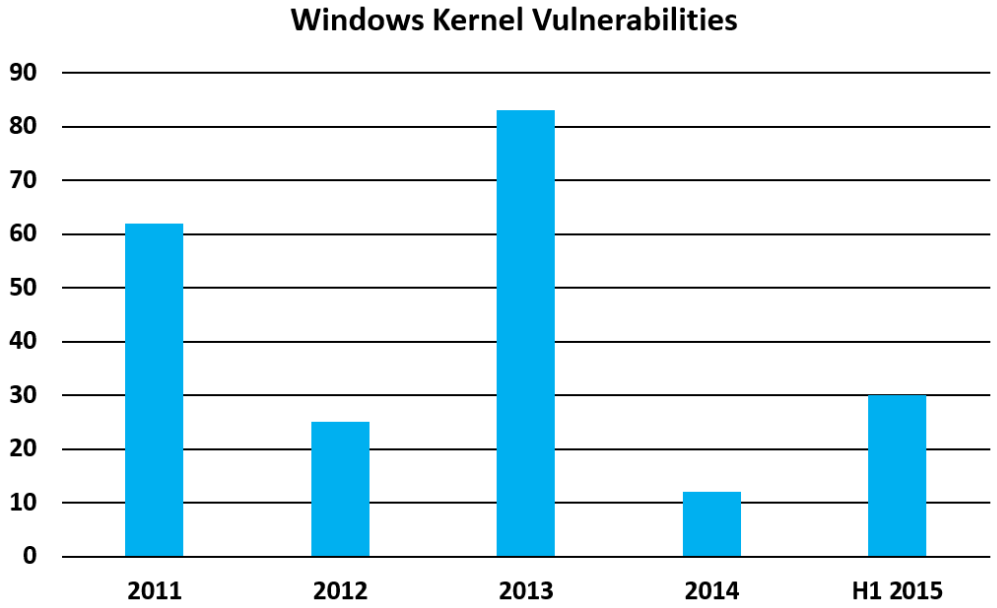
Outline

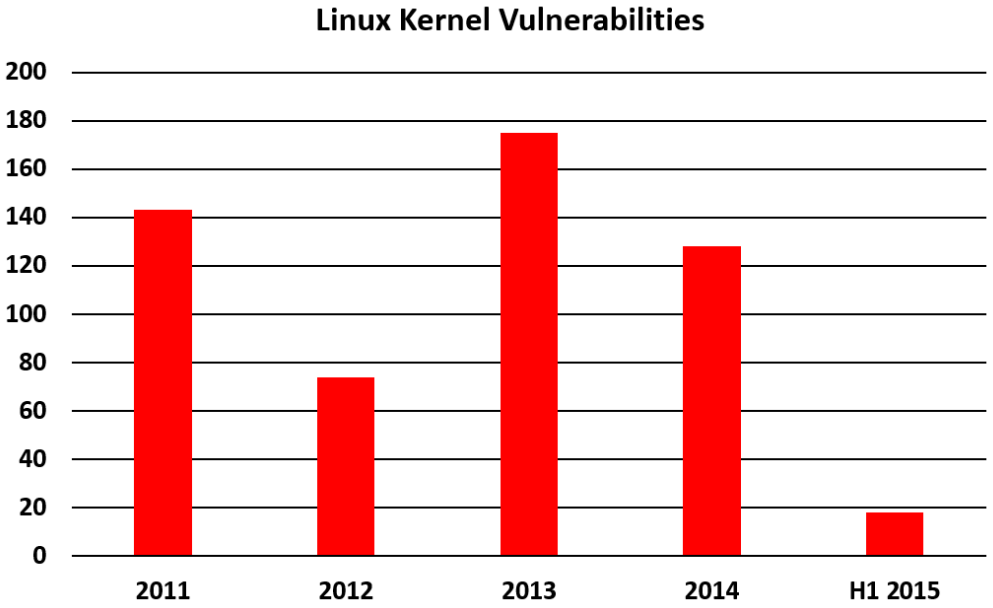
- Some security statistics
- Security issues we are facing today
- Memory introspection
  - Operation
  - Evolution
- XenAccess and mem-events enhancements
- Sample usages
- Hardware Acceleration for memory introspection
- Conclusions

Some statistics ...

Advanced persistent threats (APT-s), botnets, cyber-espionage etc., rely heavily on:

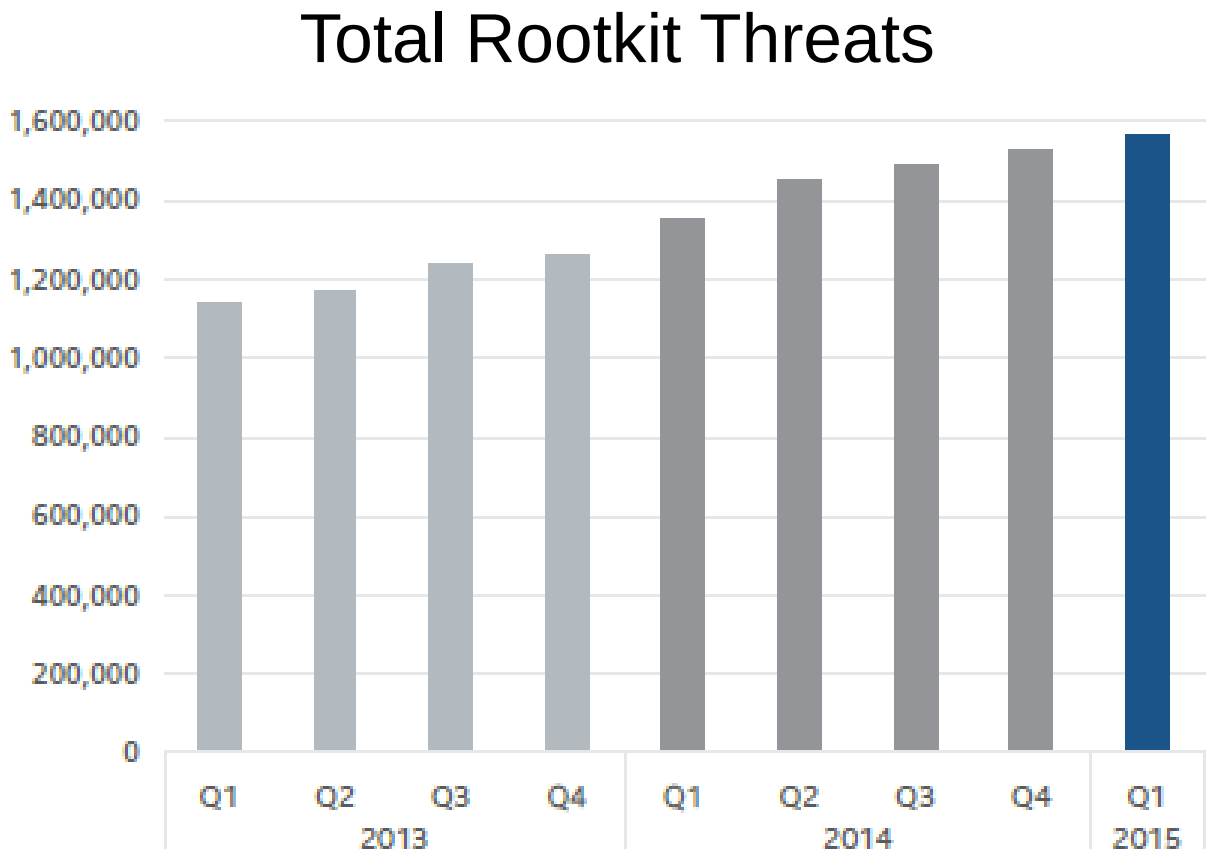
- Rootkits
- Kernel exploits
- 0-days





source: nvd.nist.gov

Some statistics …



source: Intel® Security – McAfee Labs Threats Report, May 2015

Advanced Persistent Threats (APTs) Action-flow

- Spear phishing
- Drive-by downloads
- Trojans

- Code injection (Energetic Bear, Epic Turla, Zeus etc.)
- API hooking (Dyreza, GameOver etc.)

- Espionage & data exfiltration
- Identity theft
- Sabotage

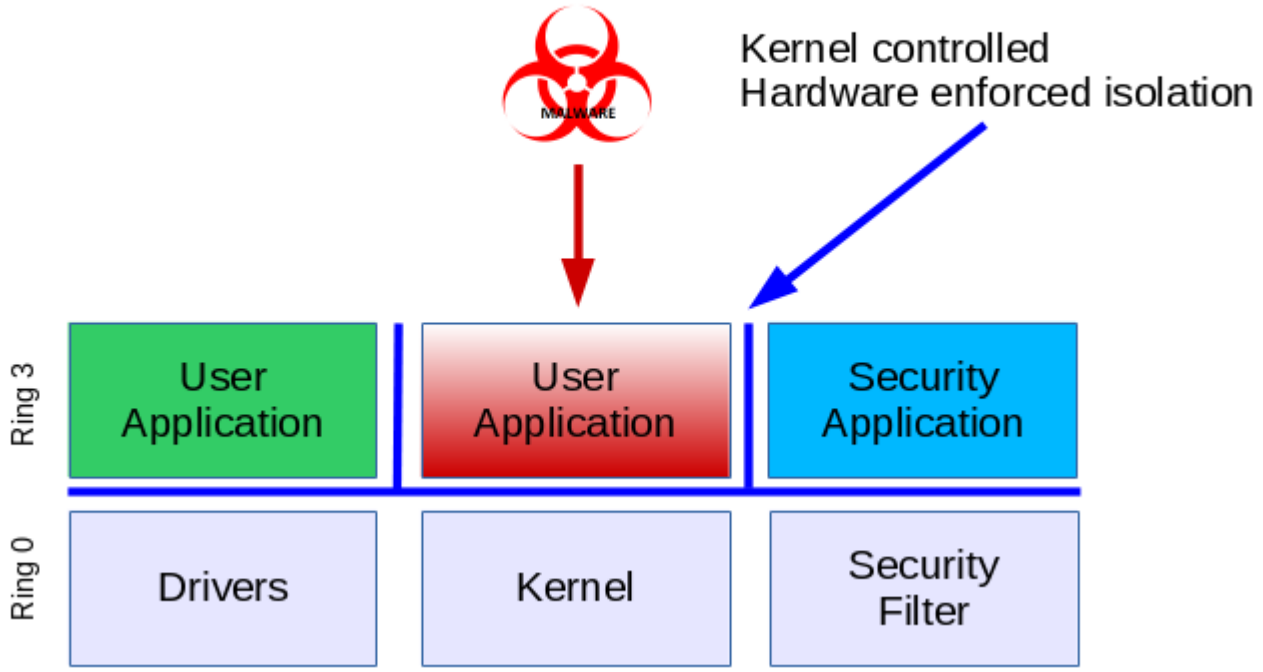
- CVE-2012-0158 ◊ APT28
- CVE-2013-1347 ◊ Energetic Bear APT
- CVE-2014-0497 ◊ DarkHotel APT
- …

- Stealthiness & persistence ◊ kernel rootkits (Necurs, TDL), bootkits, …

Security issues we are facing today

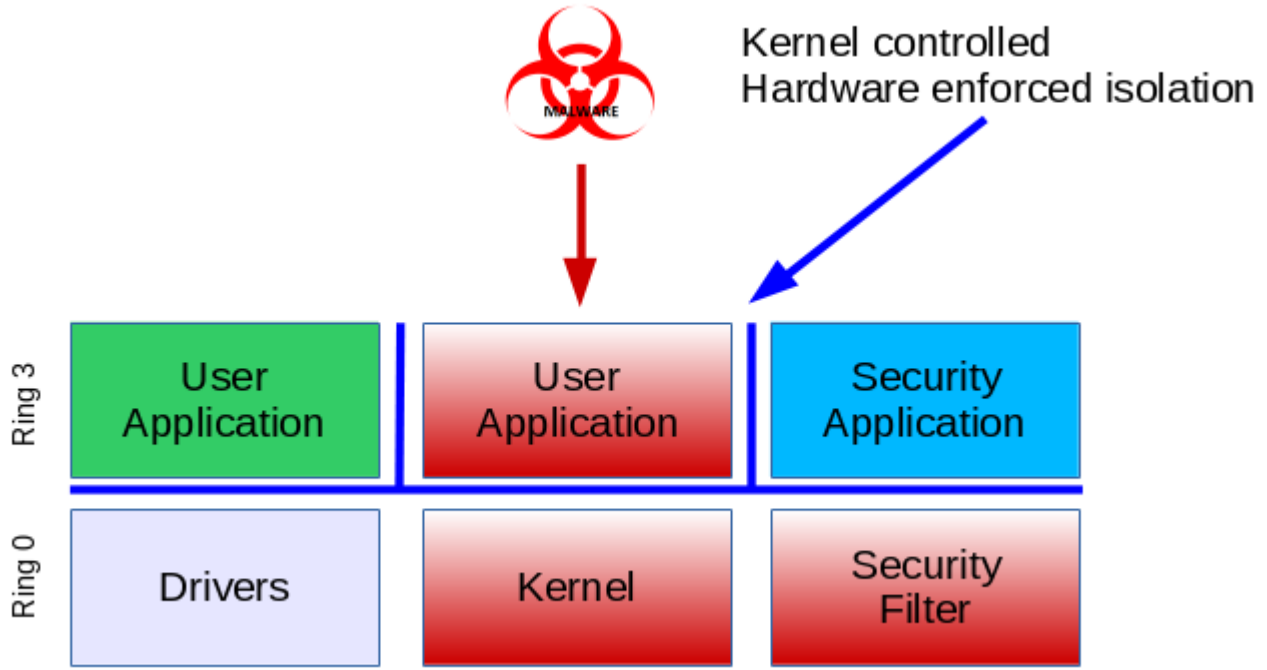
Bitdefender

Advanced threats begin by taking over a common user application (browser, document editor etc.)



Security issues we are facing today

… and ends up controlling the entire operating system

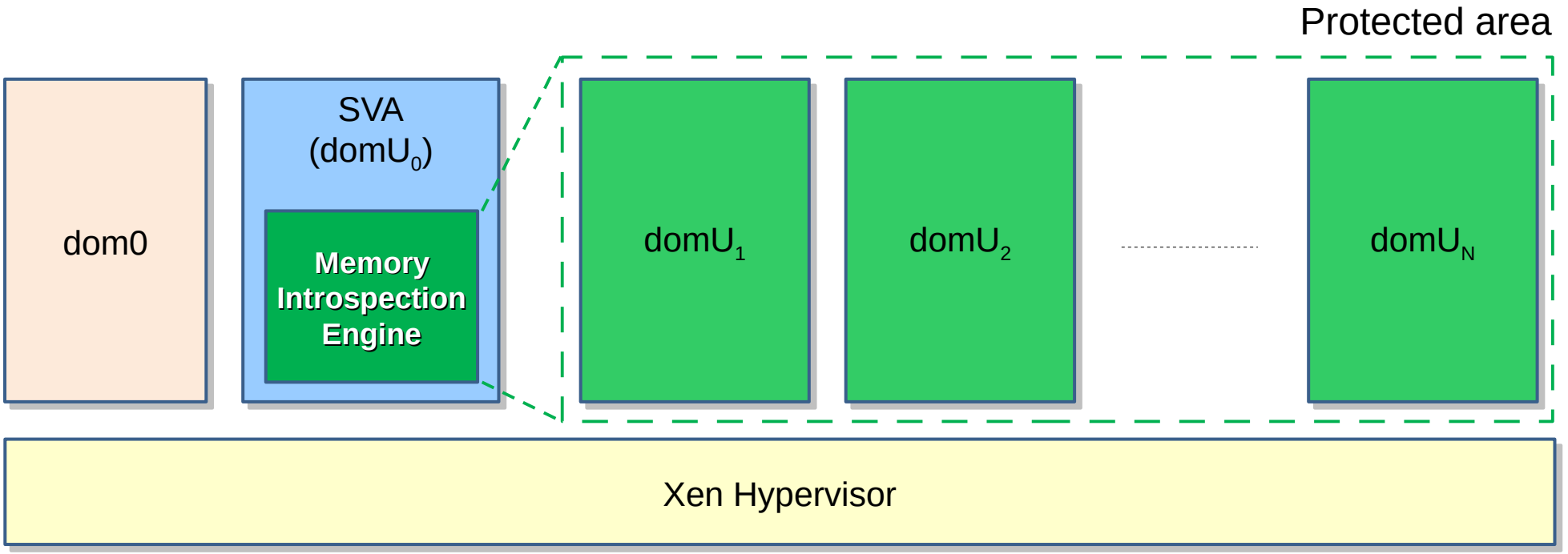


Malicious code executes in the same context and with the same privileges as security software → **lack of proper isolation**

Conclusion

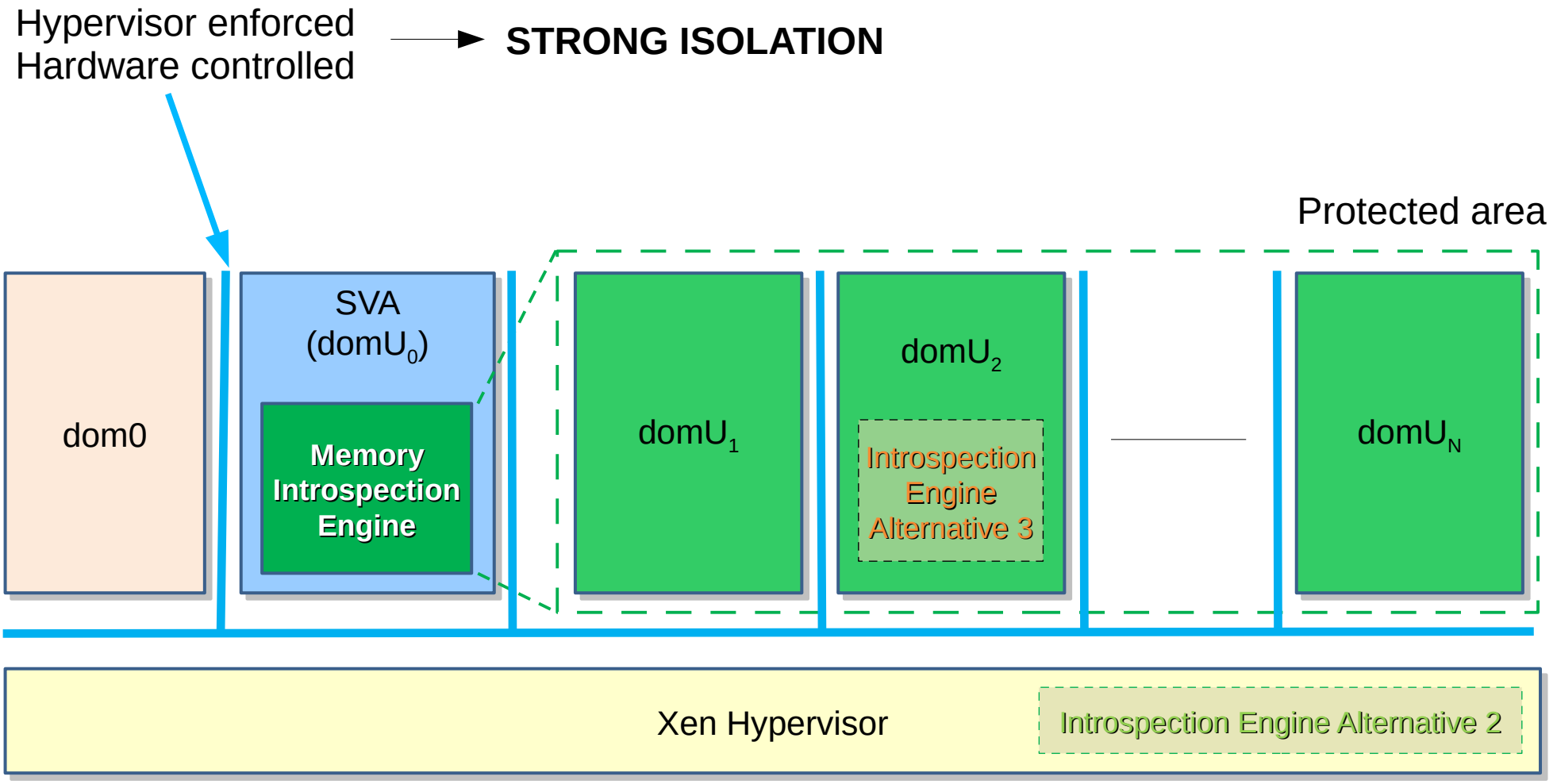
Advanced Attacks Can Evade Traditional Security Solutions

Envision the big picture



so… what's the big difference?...

Memory introspection



What is memory introspection?

Bitdefender

- Address a number of security issues from outside the guest OS without relying on functionality that can be rendered unreliable by advanced malware

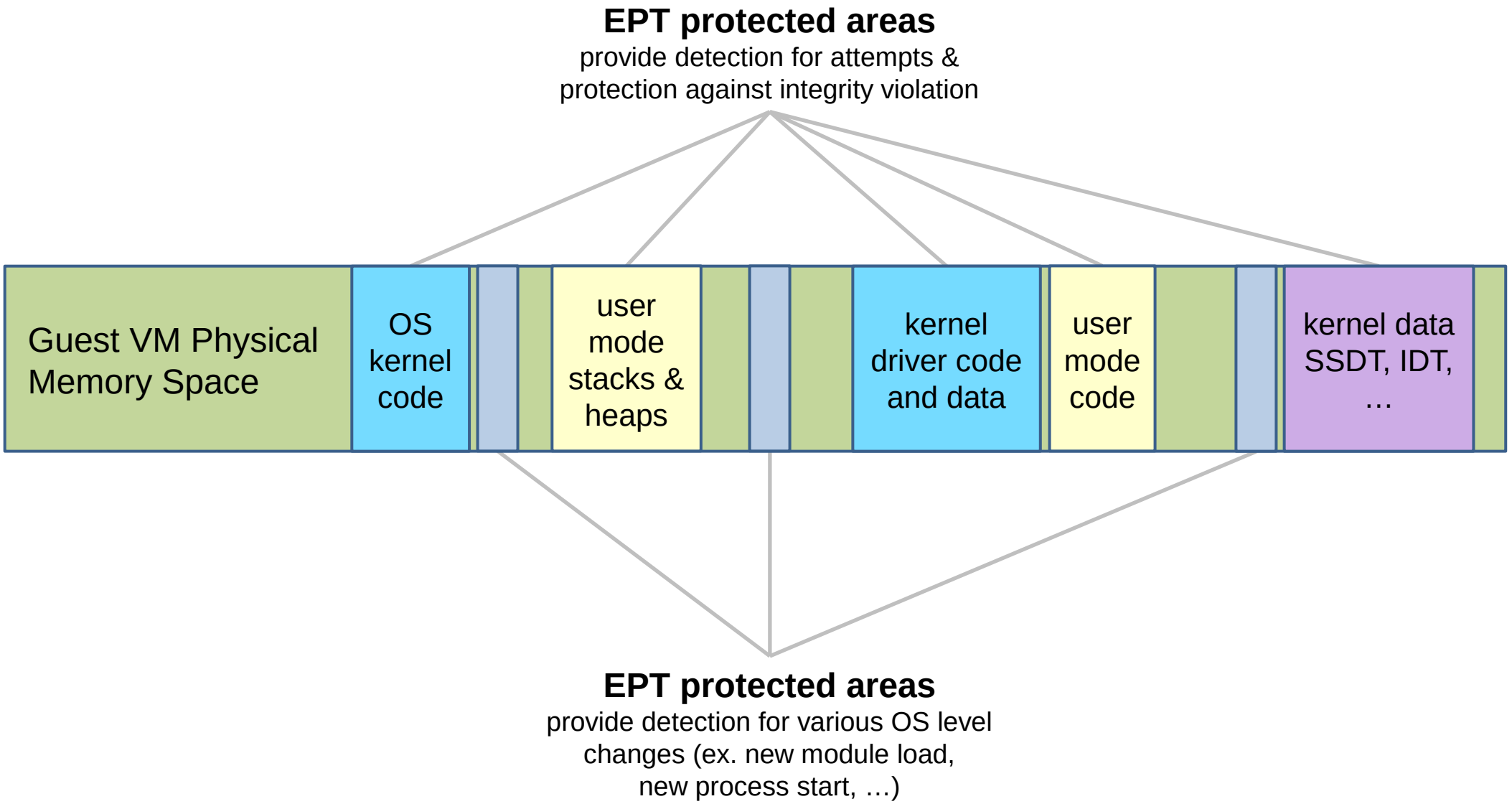
- Analyze raw memory image of guest OS, services and user mode applications, then identify:
  - kernel memory areas
  - driver objects, driver code, IDT, etc.
  - user memory areas
  - process code, process stack, process heap, etc.

How does it work?

Bitdefender

- Use existent hardware virtualization extensions (Intel EPT / AMD RVI)
- Set hooks on guest OS memory
  - mark 4K pages as non-execute or non-writable
  - hooking & notification must be supported efficiently by HV & CPU
- Set hooks on special registers (CRx) and MSR-s
  - determine when page tables change
  - determine when the OS has initialized
- Audit access of those areas by the code running in VM (OS or apps)
  - write attempts (driver objects, fast I/O tables, page tables)
  - execution attempts
- Allow or deny attempts – decision provided by security logic

How does it work?

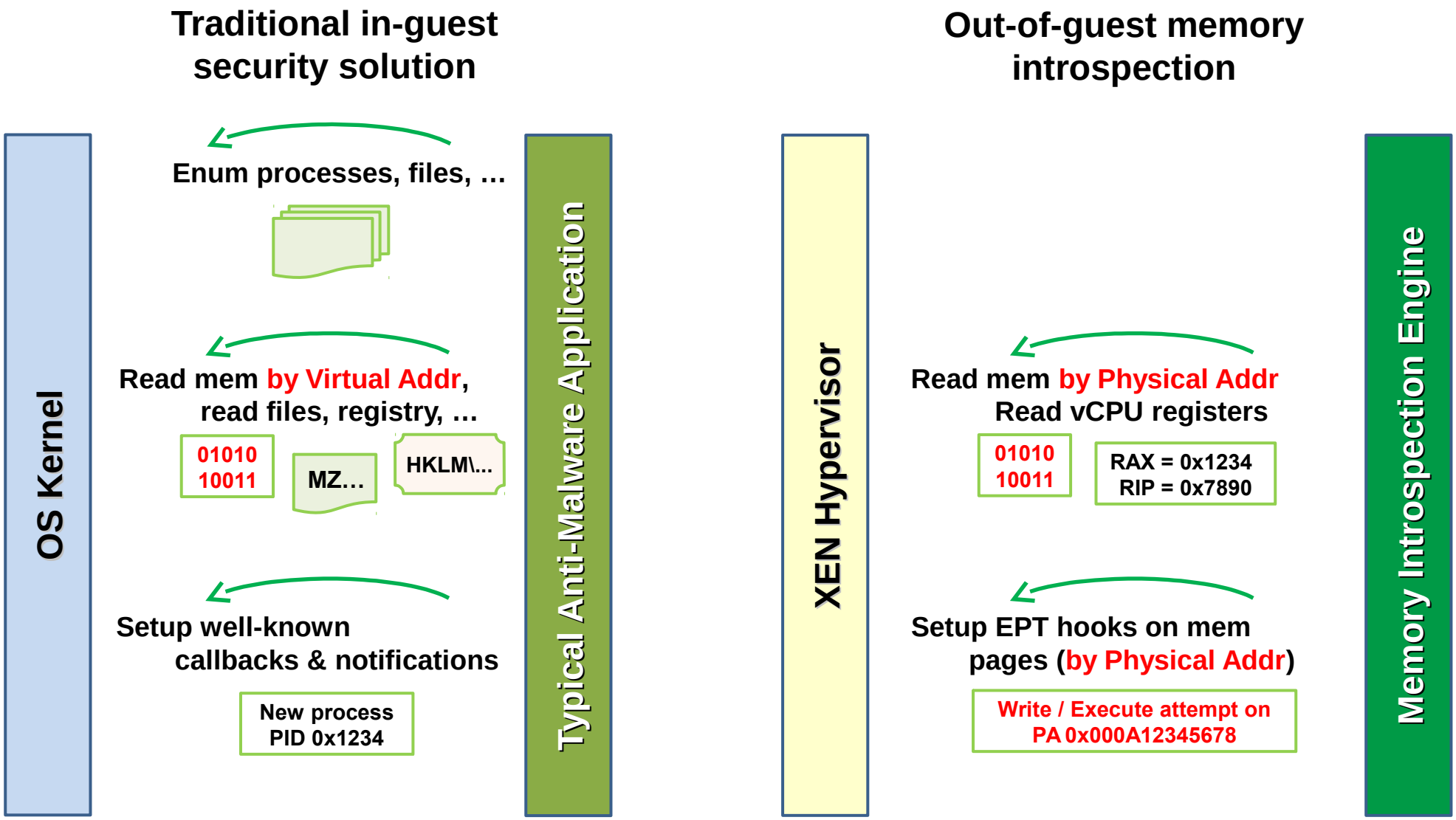


How does it work?

- Building a reliable image of the guest OS state
  - what objects are inside a guest VM?
  - what operations are being performed inside a guest VM?
  - object and event identification and correlation is done by the *introspection engine* – to decide event and object maliciousness

- Using hooks we can detect numerous events, including
  - a driver / kernel module is loaded or unloaded
  - a new user process or thread is created
  - user stack / heap is allocated
  - memory is being paged in / out

How does it work?



Two big challenges

Bitdefender

- bridging the semantic gap – obtain rich semantics from only raw physical memory pages and virtual CPU registers
  - how do we correlate 4K memory pages with semantically rich and meaningful OS specific data structures?
  - to be solved by security solution vendors
- forward lots of memory events with low overhead to the introspection engine
  - to be solved by hypervisor and CPU vendors

Memory introspection evolution

- 2003 – Garfinkel & Rosenblum: “A Virtual Machine Introspection Based Architecture for Intrusion Detection”
  - the starting point for a considerable amount of academic research
- 2006 – Jiang & Wang: “’Out-of-the-box’ Monitoring of VM-based High-Interaction Honeypots”
- 2008 – Dinaburg et al.: “Ether: Malware Analysis via Hardware Virtualization Extensions”
  - Built on top of Xen 3.1
- 2008 – VMsafe API announced by VMware, which provides access to a guest’s:
  - CPU, memory, disk, I/O devices etc.
  - supported memory introspection for vSphere / ESXi
- 2010 – VMware vShield Endpoint (as a replacement for VMsafe API)
  - in-guest agent based
  - file introspection only
- 2012 – VMware deprecates VMsafe

Why Xen?

Bitdefender

- Open source
- Relatively easy to hack
- Mature code base
- Very active developer community
- A good part of the needed support is already there
- Sits at the core of several commercial products: XenServer, Amazon EC2, Rackspace, Oracle Cloud
- Used in CloudStack, OpenStack

Memory introspection in Xen

Bitdefender

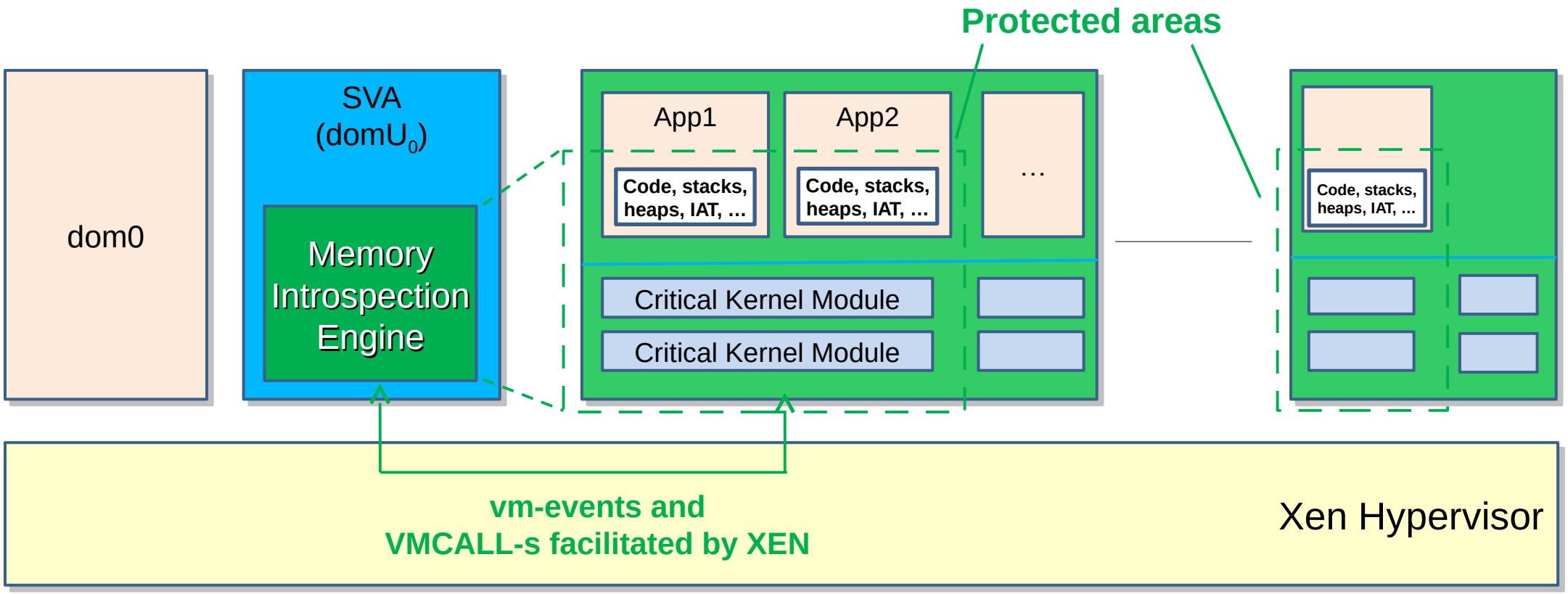
- 2007 – XenAccess, XenProbes
- 2008 – Lares
- 2009 – first patches for the mem-events API
- 2010 – libVMI – uses XenAccess and XenStore
- 2014 – API extending patches
- 2015 – API cleanup and extending patches (mem-events → vm-events)
- 2015 – libbdvmi – a lightweight LibVMI

XenAccess and vm-events enhancements

Bitdefender

- 2014, 2015 – published patches that:
  - add support for discarding writes (via emulator)
  - optimally access the guest state (vCPU registers)
  - force the monitoring of specific MSR-s
  - cleanup the CRx notification support
  - add support for content hiding
  - add support for denying certain register writes
  - mem-events are renamed vm-events (Tamas K Lengyel)
  - add ARM support (Tamas K Lengyel)
- Identified and helped address three XSA-s, in the process

Example use of the extended API



Example use of the extended API

- Bitdefender's own introspection engine can:
  - protect the kernel from known rootkit hooking techniques
  - protect user processes (e.g. browsers, MS Office, Adobe Reader, …) from:
    - code injection
    - function detouring
    - code execution from stack / heap
    - unpacked malicious code
  - inject remediation tools into the guest on-the-fly (no help from 'within' needed)
- Runs in userspace in a user domain (SVA – Security Virtual Appliance)
- Introspection logic has very small overhead
  - bulk of the overhead is given by sending / receiving events and calls between protected guest VMs and SVA

Hardware acceleration for memory introspection

- Latest Intel CPUs offer:
  - Virtualization Exceptions (#VE)
  - VMFUNC
  - Multiple EPT Views

- Patches for xen 4.6
- Emulated where hardware support is missing

Virtualization Exceptions (#VE)

Bitdefender

- Allows the conversion of EPT violations into guest exceptions (IDT)
- An in-guest component can:
  - handle the exception itself
  - defer to VMM

- This allows the embedding of the introspection logic into the guest itself

VMFUNC / multiple EPT views

- VMFUNC: access VMM functionality without a VMEXIT
- VMFUNC leaf 0 → EPT switch (per vCPU)
- Fast single stepping

Future research directions

Bitdefender

- Improve vm-event overhead
- Nested VMM
  - 1-to-1 introspection
  - very simple guest VMM
  - can increase virtualization overhead
  - but faster introspection
- Add support for guest introspection to other open source VMM-s (KVM, bhyve etc.)
- Unified introspection API across VMM-s

Conclusions

- Today Xen can be the base for providing a much improved layer of security – serves as a model for other HV vendors
  - Truly agentless security (zero in guest footprint)
  - IT Admins can deploy introspection based solutions seamlessly, without changing a single line of config inside the guest VMs
- Hardware enforced isolation (against kernel exploits, zero days etc.)
- Hardware extensions enable intra-VM isolation to enable protected agent based introspection for high frequency access monitoring and agent isolation
- Both models require straight-forward Xen infrastructure changes (multiple-EPT views, hardware acceleration capabilities)

Bitdefender

Thank you!

enterprise.bitdefender.com

Bitdefender®