µ-Xen

Ian Pratt
SVP, Products
Bromium Inc.

Outline

- μ-Xen Goals
- Architecture and implementation
- Evolutionary path
- Use case: Bromium end-point security

The µ-Xen Client Hypervisor

- Derived from the Xen code base
- Designed to address client hypervisor deployment challenges
- Designed to achieve a high-degree of security assurance
- No requirement to support legacy s/w or h/w
- Optimized to support *micro-virtualization*
- Evolution to a new class of hypervisor

Evolution to a new class of hypervisor

- Start with a self-contained type-2
  - Host kernel module containing µ-Xen blob
  - Well-defined interface to/from the host
    - Memory allocation, Schedule VCPUs, Timers, Events
- Similar to hXen project
  - Windows, OS X; easily portable
  - µ-Xen drives VT hardware, leaves IO and other hardware to host OS

µ-Xen streamlines Xen code base

- x64 only
- No support for systems that are not VTx/EPT or AMDV/NPT capable
- No support for PV MMU guests
- Significant code size reductions
- No compat code
- Separate build target
- Future: slim down instruction emulator

µ-Xen virtual IO

- µ-Xen demultiplexes IORecs and events to multiple host processes per VM
  - Able to have separate video, audio, disk, net etc processes
  - Helps scheduling latency, enables better exploitation mitigation lock-down
- Use just the relevant parts of QEMU
- User-space PV backend drivers
- Post-boot virtual IO device revocation

Support for large numbers of similar VMs

- Support for VM fork
  - Simple extension of Populate on Demand code
- Tree of templates/VMs supported
- PV driver support for re-sharing free pages
- IO tracking to re-share swapped and re-read pages

Evolution to a new class of hypervisor

- Goal: host is responsible for resource allocation, but can not interfere with the privacy or integrity of the hypervisor or other guests, e.g.:
  - Hypervisor asks host for memory, receives it, then removes host access to it
  - Host may perform IO on behalf of hypervisor and guests, but must be privacy and integrity protected
  - Host may deny service, crash; IO tamper detectable

De-privilege host into a VT container

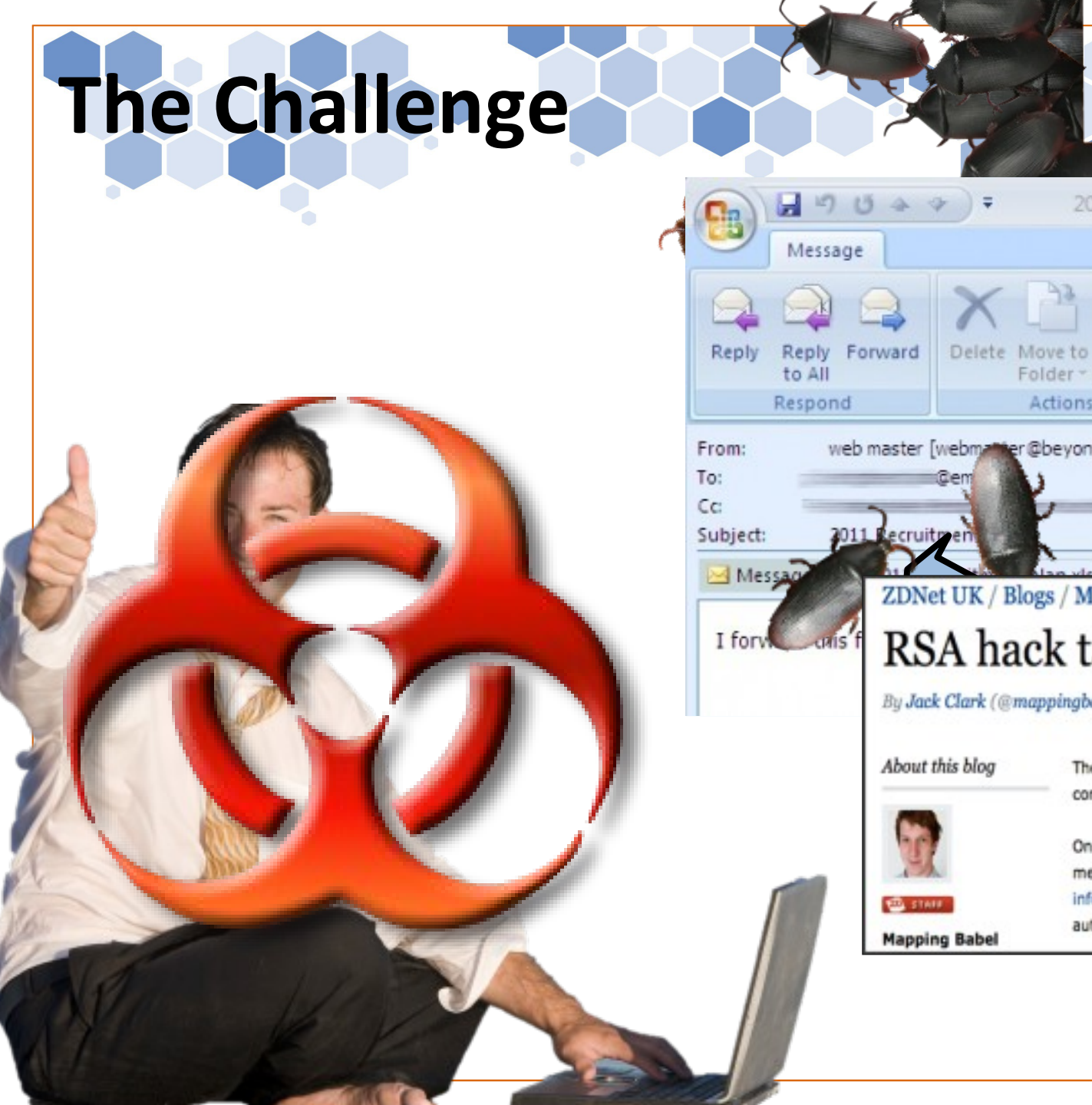
- Remove host access to hypervisor and guest memory in EPT tables
  - h/w devices passed through to host, VTd protected
- Device models can only access Granted memory
  - Require PV network and disk as no virtual DMA
  - Guest IO buffers encrypted/authenticated before grant copy/map
- Save/restore, dom builder must first have guest memory encrypted/authenticated by hypervisor
  - Hypervisor paging requires similar treatment

Further Enhanced Security

- Migrate certain host h/w devices to ownership by hypervisor or privileged guest(s), replace with virtual device
  - E.g. Keyboard controller to avoid key logging
  - Possibly WiFi/NIC too
- Measured launch of hypervisor
  - TXT/TPM to establish DRoT
  - Remote attestation of hypervisor configuration

Task-based Micro-virtualization

The Challenge



What Went Wrong?

- Poor Isolation leads to Vulnerability
  - Multi-tenancy of code (& data) from a huge set of trust sources
  - Code & Data are executed in the context of a few, weakly organized and poorly isolated “bins” representing levels of trust - e.g. UID, GID, PID
  - Overloaded, complex specs & buggy implementations of porous interfaces
- Thus malware can:
  - Access data and documents in the same “bin”
  - Subvert other good code that is placed into the same “bin”

Why Legacy Security Solutions Fail

Black-listing: **Detection is not protection**

- Doesn’t protect users from zero day attacks
- Unable to detect targeted & advanced malware

White-listing: **Restriction is not protection**

- Users want to be productive
- Attackers can compromise trusted applications and web sites

The Challenge

How do we plan for the unknown threat?

What do we do about human mistakes?

The Micro-Virtualization Approach

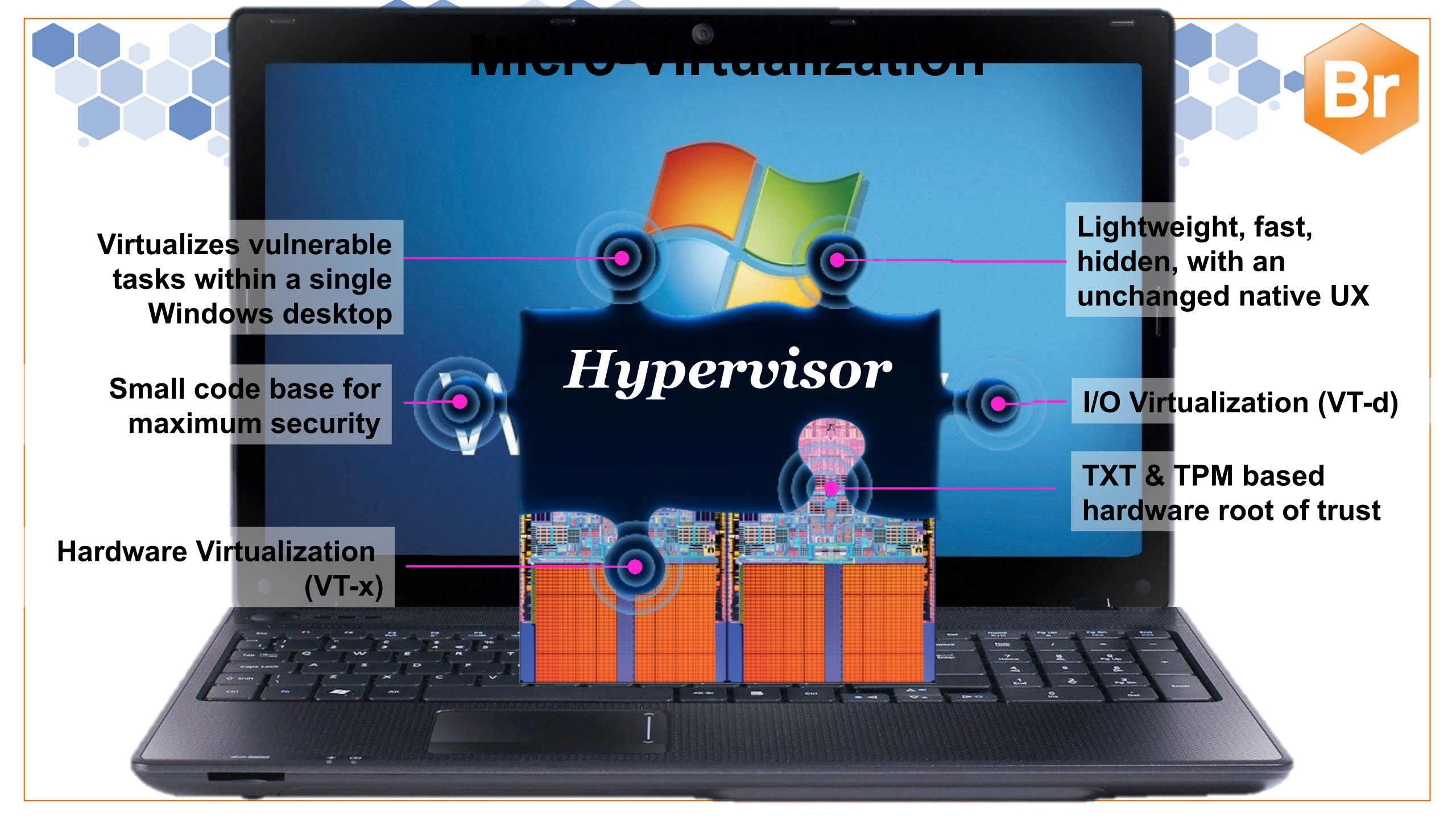
Next generation endpoint security

- A pragmatic approach to trusted computing

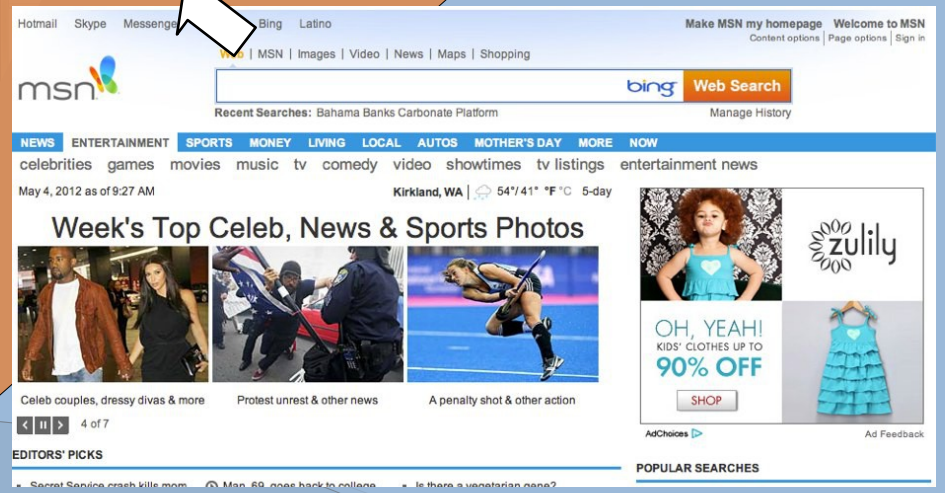
Based on hardware based isolation and not detection

- Contain risky activity from corporate data and network
- Block zero day and polymorphic threats

Provides isolation & trust through hardware virtualization

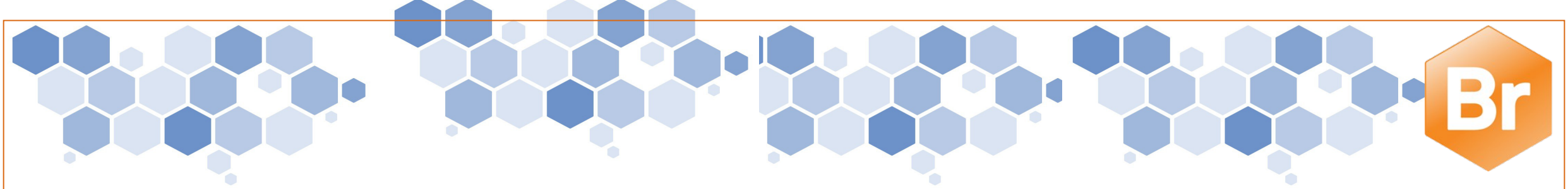

Micro-Virtualization
Hypervisor
Virtualizes vulnerable tasks within a single Windows desktop
Small code base for maximum security
Hardware Virtualization (VT-x)
Lightweight, fast, hidden, with an unchanged native UX
I/O Virtualization (VT-d)
TXT & TPM based hardware root of trust

The Microvisor isolates vulnerable tasks from Windows, each other & key system resources



Micro-VMs have "least privilege" access to files, networks & devices, and execute CoW

Each vulnerable task is instantly isolated in a micro-VM, invisible to the user



- Demo

Key Threat Vectors Are Isolated

Web Browsing
External Documents

E-mail Attachments
USB files

Conclusions

- µ-Xen is targeted at client systems
- Combines ease of deployment with useful security properties
- Enables task-based micro-virtualization, allowing mandatory access control to be retrofitted to commodity client OSes

- **We're hiring! Please email ian@bromium.com**