

# What security features are appropriate for virtualization technology?

John McDermott

NRL

[john.mcdermott@nrl.navy.mil](mailto:john.mcdermott@nrl.navy.mil)

- Scope / meaning of virtualization
- Security features for virtualization technology (VT) per se
- Security features that make sense for complete products that include virtualization but ...
  - should not be implemented by VT
- Security features that you might support with virtualization

# What is virtualization technology?

web-based

open

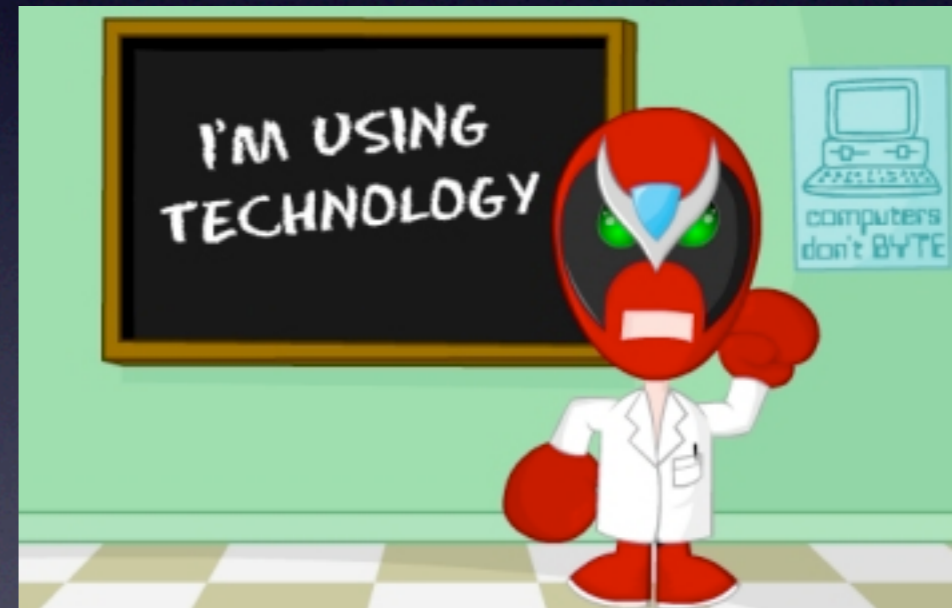
structured

“virtual” is the new “good”

service-oriented

object-oriented

distributed



- Hypervisor/VMM - software that virtualizes complete hardware platforms
  - to run operating systems
  - Type I or Type II (more later)
  - a virtualization product may include a VMM
  - a product may include features not implemented within the VMM per se

Since security is never perfect,  
it should always be described in terms of  
**threat model** and **robustness**.



Understanding these concepts can save you money.

- (A well-formed *threat model* will always contain these components, though the terms used may be different.)
- A *threat* is some adverse action against the services, data, stakeholders, etc., protected by a security solution.
- Each threat is the goal of some *threat actor*.
  - An abstract entity with,
  - *capabilities*,
  - *initial knowledge*, and *initial access*.

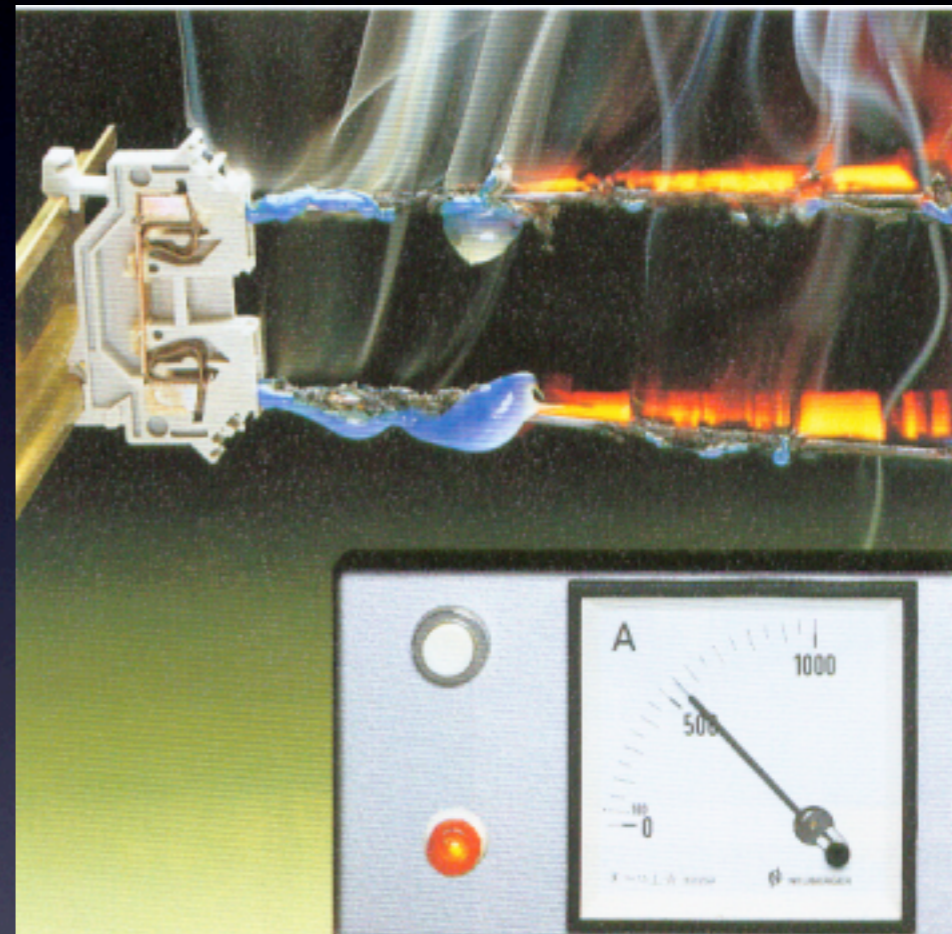
- Our understanding of security flaws that may remain after a product is developed
- Robustness = *strength of mechanism* + *assurance*
  - Strength of mechanism is about conceptual flaws
    - present even in perfect implementations
    - e.g. Caesar cipher
  - Assurance is about development practices and assurance measures taken to address flaws that are not conceptual



- A low-strength / high-assurance solution can be broken through its conceptual flaws
- A high-strength / low-assurance solution can be broken through its implementation flaws
- Low/low can be broken either way
- High/high is expensive to defeat

- One kind of VMM is a “perfectly” strong mechanism
  - no conceptual flaws
- *separation kernel* (SK) - no sharing between guests
- a pure SK cannot be used as a cross-domain solution (David Bell’s computer security intermediate value theorem at ACSAC 21)
- Other kinds of VMM’s can be a less than “perfectly” but otherwise arbitrarily strong mechanisms
  - NPS trusted exemplar, NRL Xenon - allow sharing between guests
  - can be used as a cross-domain solution

- If a VMM is to have all of its guests connected to the same network ...
- ... then the VMM does not need to be any more robust than its guests ...
- ... because the guests can all be attacked via the common network.



- *sensitive instructions* can affect processor mode, memory maps, DMA, interrupt handling ... i.e. VMM control
- *privileged instructions* cause a trap into the VMM code
- sensitive instructions should be a subset of privileged instructions
- guests should execute *innocuous instructions* directly
- define Type I and Type II VMM's: how does the VMM get the trap?

- program counter / flags
- registers
- cache
- descriptor tables
- main memory
- interrupts
- APCI
- boot firmware (e.g. BIOS)
- TPM

- simple network-based abstractions
  - IP packets
  - Ethernet frames
- high-level guest-based abstractions
  - file systems
  - windows
  - disks
  - desktops

# You are using virtualization to ... ?

- Share hardware execution environments
- Remove (hide) undesirable hardware features
- Add (virtualize) desirable hardware features



# Product security features that should not be implemented in the VMM

- most network security
- measurement, including TPM
- cryptographic features, including TPM
- intrusion detection
- identification or authentication
- RBAC
- More research is needed on how to implement these outside of the VMM

# Interesting security features that can be well-supported by virtualization

- Measurement
- Attack space reduction
- Multiple single-level information-flow security