

# ***Security & Availabilty***



***Steven Hand***

***University of Cambridge***

***& XenSource Inc.***



UNIVERSITY OF  
CAMBRIDGE

Computer Laboratory

9/18/06



# Security Requirements



- Standard model built around:
  - a small 'separation kernel',
  - mandatory access control, and
  - a set of validated security policies
- Modern technology also enables *trust*
  - SRT (TPMs), DRT (SMX & Presidio)
- But experience shows we also need:
  - *User Experience*
    - Incremental benefit, high performance, convenience
  - *Quality of Service*
    - Predictable partitioning and performance
    - Defense against DOS attacks

# Availability



- Can capture quality of service and (to an extent) user experience as **availability**
- No good if system is 'secure' if you can't use it 😊
- Key things are:
  - predictability (temporal scheduling)
  - fairness (spatial scheduling)
  - robustness (non-crashability)

# Predictability



- Xen designed to provide “secure isolation”
  - soft real-time scheduling in from the start
  - applied to all temporal resources
- Unfortunately SRT doesn't necessarily imply “principle of least surprise”
  - DWIM versus ‘Just Do It’
- Hence Xen 3.x has *less* SRT than Xen 1.0
  - Opens door for intentional or inadvertent DoS
  - Risk not really quantified right now

# Fairness



- Xen designed for secure isolation
  - 'hard' allocation of memory, #CPUs
  - no inter-domain interaction
- But new DD model + load balancing mean that we now have
  - split devices (+ grant tables)
  - dynamic memory & CPU adjustment
- More possibilities for interference
- Risk not yet quantified

# Robustness



- Xen designed for secure isolation
  - unpriv hypercalls 'safe' by design
  - priv hypercalls 'safe' by fiat
- But most testing is *positive* only
  - if it works, it's fine
  - defensive programming a guideline, but not validated/checked
- Working with minios or OS ports shows we're not always as robust as we'd like 😊

# A call to arms



- Need to augment existing testing with extensive stress/negative testing
- HMP ('hurt me plenty'):
  - set of stress tests to use, abuse and confuse hypercall apis (random + det inputs)
  - set of malicious workloads driving PV I/O path (netflood, disk seeker) and [later] emu path.
  - (inspired by crashme tool)
- Working on prototype
  - based on minios so no hvm yet
  - suggestions (or code!) for HMP tests welcome