

Scalability, Fidelity, and Containment in the Potemkin Virtual Honeyfarm

Michael Vrable, et al.
University of California, San Diego
SOSP 2005

Presented by Ramsés Morales

Honeypot

- Information system resource whose value lies in unauthorized or illicit use of that resource.
- Carefully monitored and vulnerable network-connected system.
- Help understand means and methods used to compromise and control Internet hosts.
- Results lead to:
 - Anti virus signatures.
 - Disinfection algorithms.
 - Support criminal investigation and prosecution.

Honeyfarm

- Deployment influenced by trade-off between:
 - Scalability.
 - Fidelity.
 - Containment.
 - Preventing compromised honeypots from attacking.

Low-interaction honeypots

- Low fidelity.
 - Little to no state maintained.
 - Don't execute code from services or OS.
- High scalability.
 - Monitor activity across millions of IP addresses.

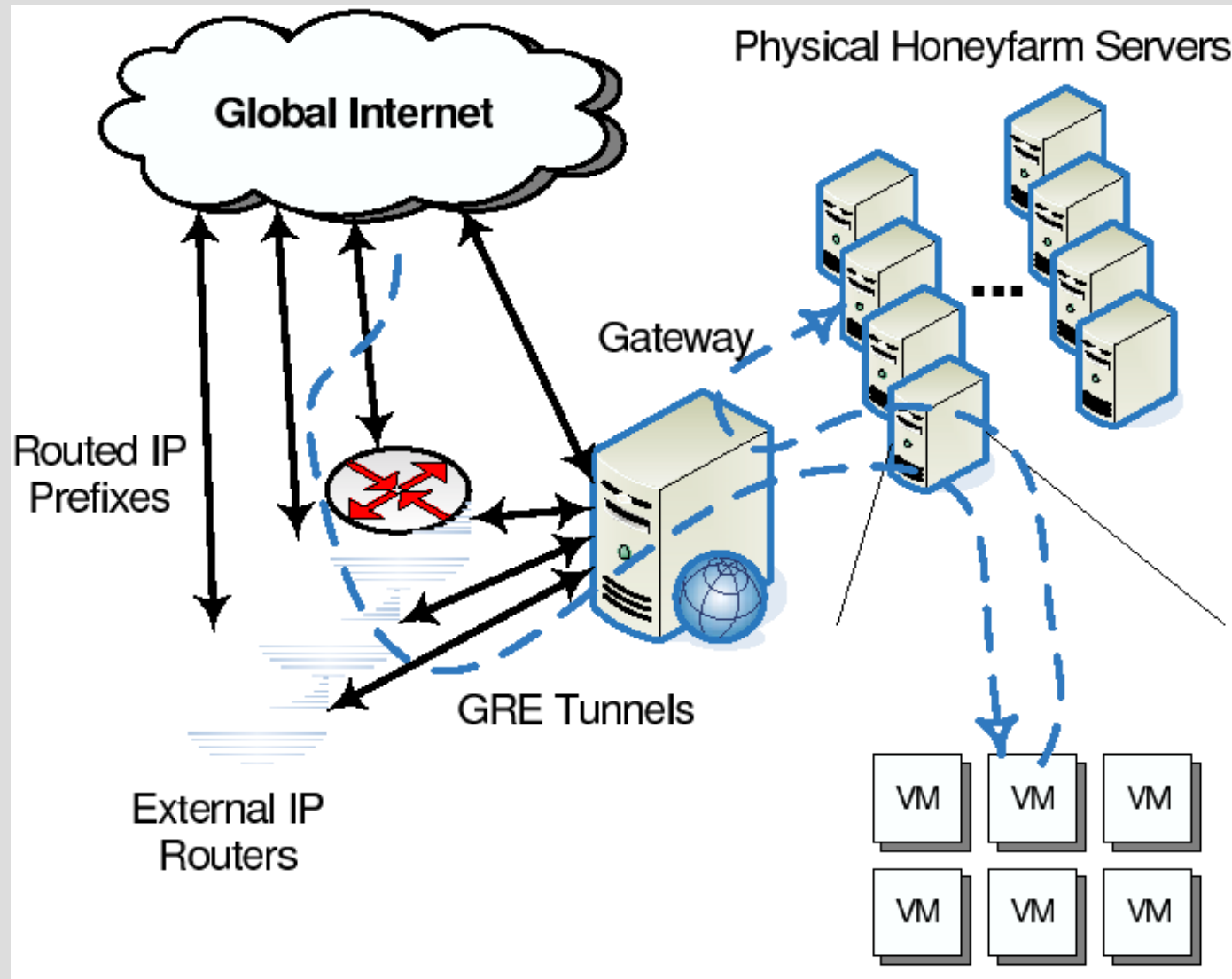
High-interaction honeypots

- High fidelity.
 - Execute OS and service code.
 - Containment is a big issue.
- Low scalability.

Potemkin's Goals

- Scalability close to stateless monitors.
- Fidelity similar to high-interaction honeypots.
- Provide containment policies.
- How?
 - Aggressive memory sharing.
 - Late binding of resources.
 - Exploit network idleness.

Architecture



Architecture: Scalability

- “Only tasks driven by external input have any value”
- Late binding of resources to requests.
- **Flash cloning**: lightweight virtual machine created from reference image.
- **Delta virtualization**: memory allocated copy-on-write.

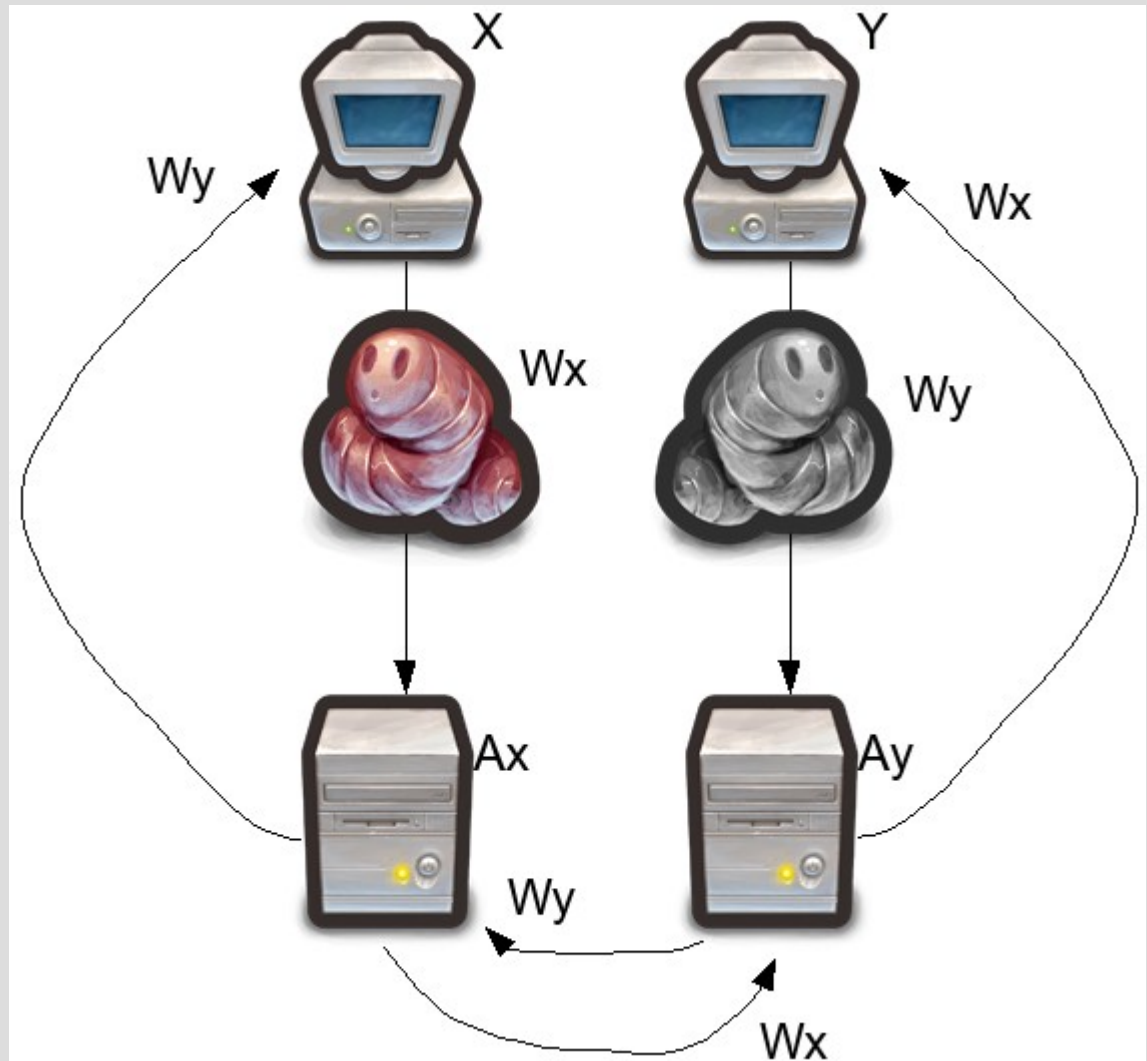
Architecture: Containment

- Can't blindly disallow outbound packets if we want to understand the attack.
 - Botnets “phone home” for commands.
- Policies implemented on the gateway.
 - Track communication patterns.
 - Proxy or forward service requests (DNS).
- **Internal Reflection:** redirect an unsafe outbound packet to the honeyfarm.
 - Avoid starvation.

Architecture: Containment Reflection issues

Single virtual
machine per
address.

(Liability)

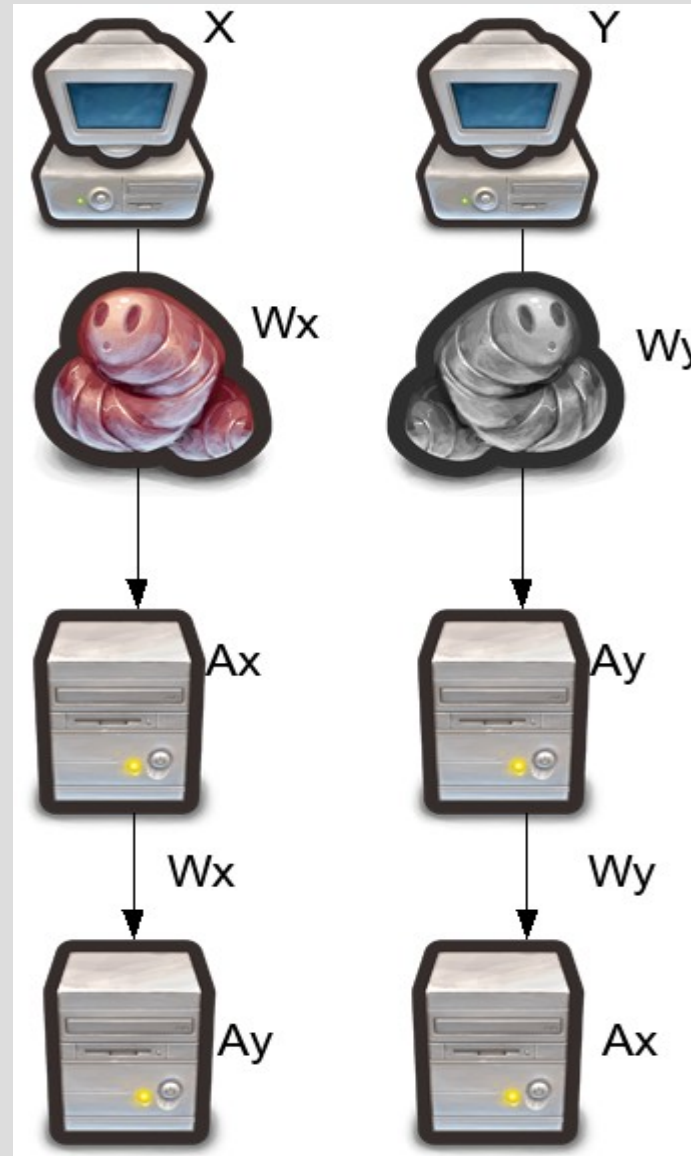


Architecture: Containment

Reflection issues

Unique alias per infection.

(No symbiosis)



Architecture: Containment Reflection issues

- **Universe Identifiers** to accommodate both scenarios.
- One universe per transaction.
- Packets may be reflected to hosts within the same universe.
- Mix-in universes to capture symbiotic behavior.

Architecture: Gateway Router

- Directs inbound traffic to individual honeyfarm servers.
- Containment management.
- Long-term resource management.
- Interface for detection, analysis, and UI components.

Architecture: Gateway Router Inbound Traffic

- Can use BGP or,
- tunneling from external router.
- Packets to inactive IPs sent to server with enough resources.
- Packets to active IPs sent to server running associated VM.
- Servers are addressed using datalink layer.

Architecture: Gateway Reclaiming a VM

- Ideally, when an attack is unsuccessful.
- Should interpret input from network/host based detectors.
- Compromised VMs can persist for further analysis.

Architecture: Virtual Machine Monitor (VMM)

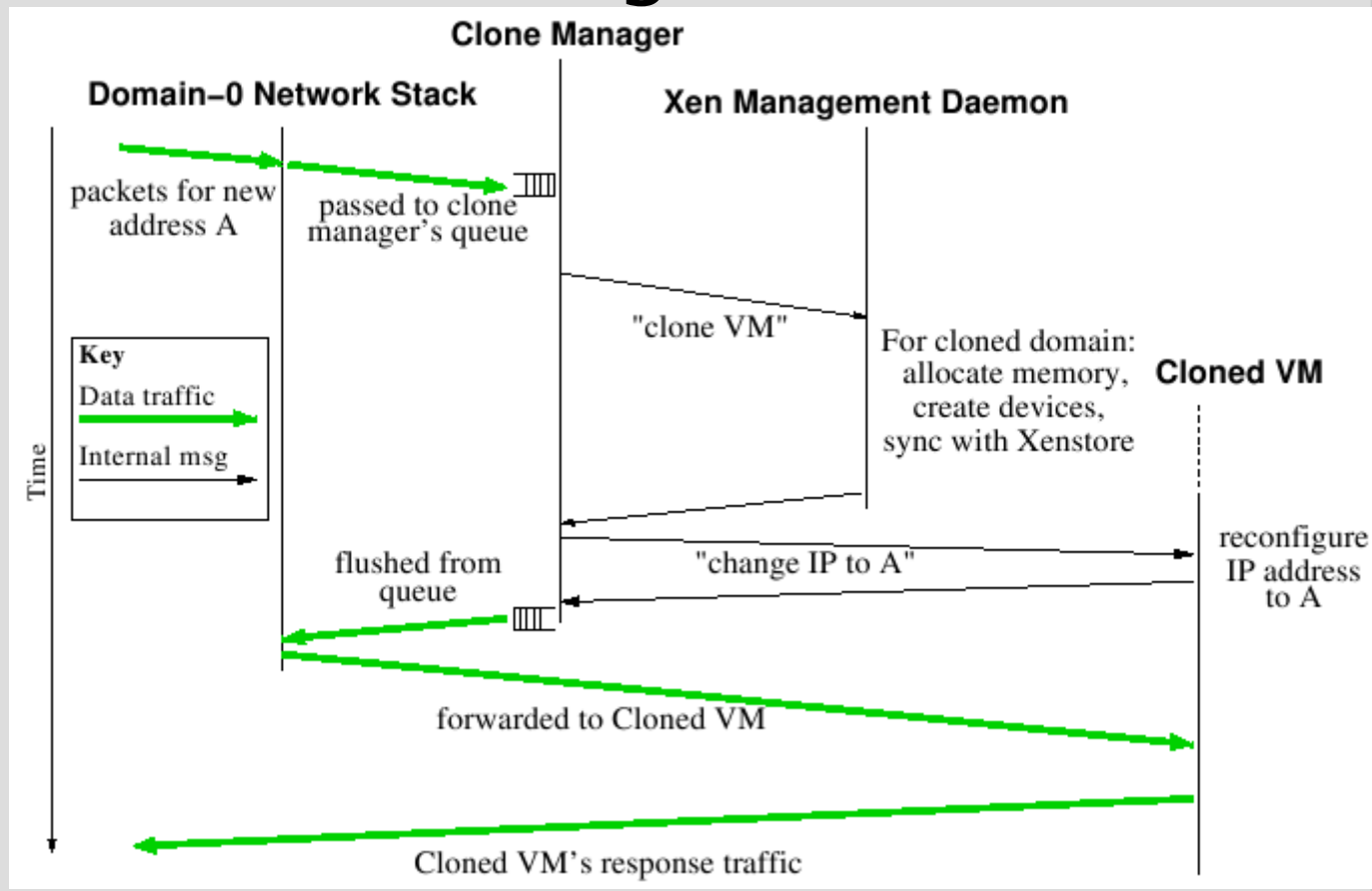
- One VM per IP address served.
 - Isolation is required to avoid an attack from polluting the monitoring of another attack.
- VMM reclaims resources of a VM when instructed by the gateway.
- Memory snapshot of pre-initialized OS and application environment used by *flash cloning*.
- *Delta-virtualization* shares VM pages copy-on-write.

Gateway Implementation

- Processing modules filter inbound and outbound traffic.
- Important inbound filter: *scan filter*.
- *Flow cache* used to bind an attack with a VM.
- *History table* maintains long-term source/destination state.
- *Host type map* determines emulation targets.
- Containment policies:
 - History: forward if recent entry in history table.
 - Internal reflect: reflect filtered-back packets.
 - Protocol proxy: forwards service request.

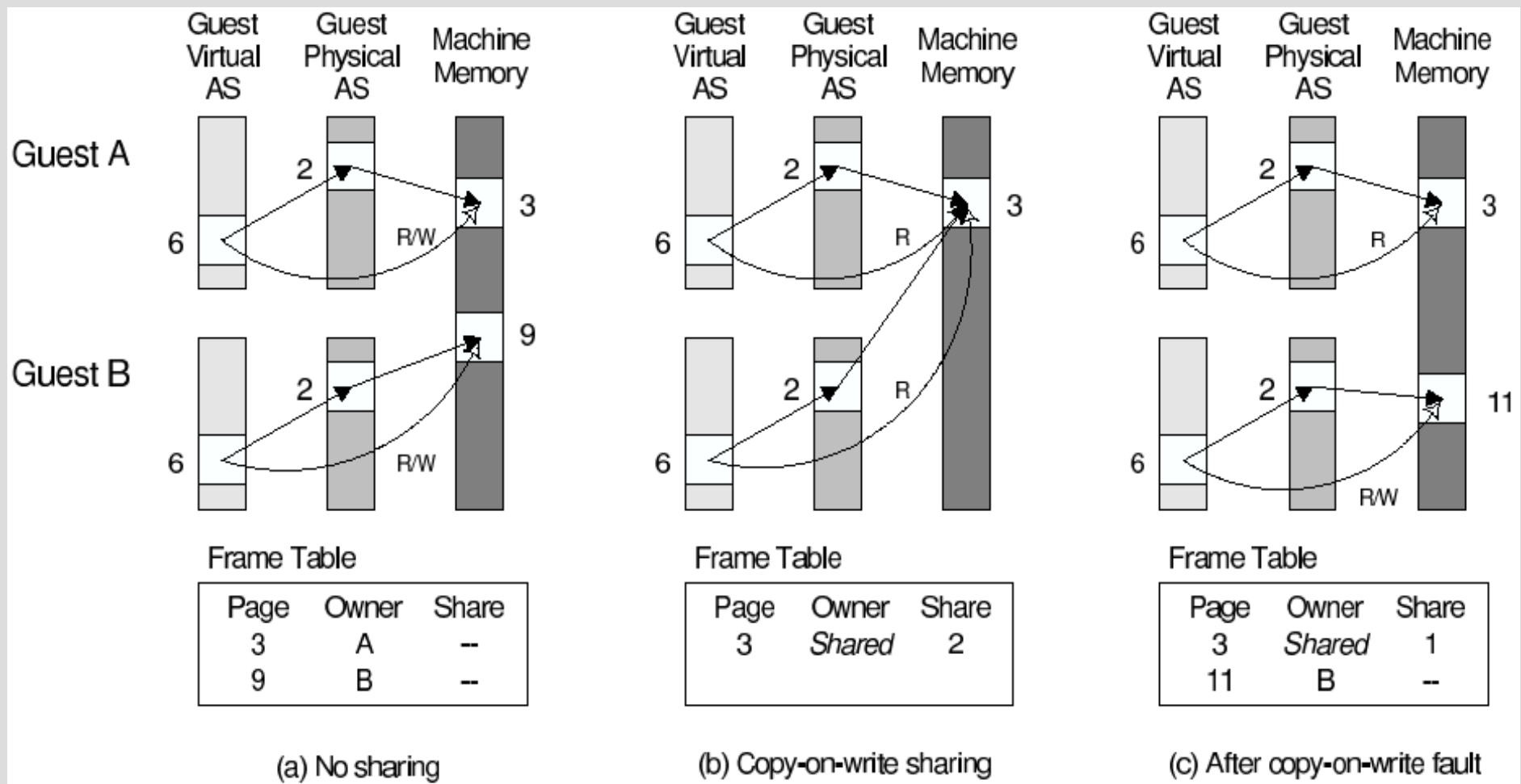
VMM Implementation

- Linux and Xen.
- VMM boots a new (reference) VM and its snapshot is used in flash cloning.



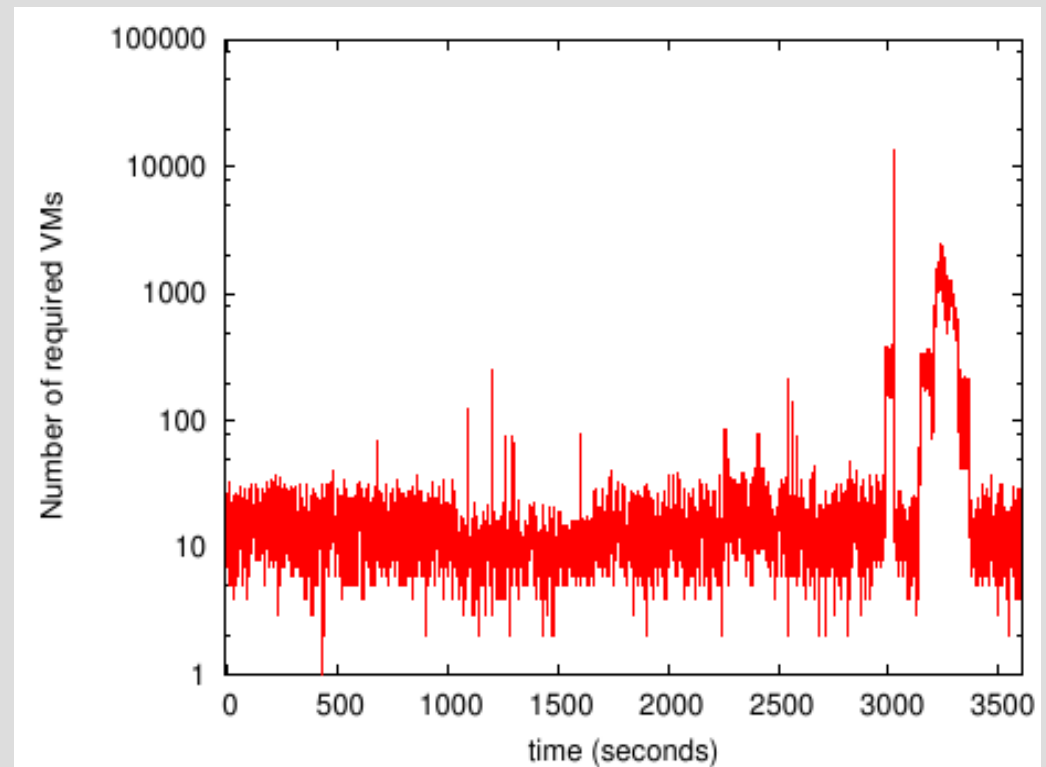
VMM implementation

- Reference VM used in delta virtualization.



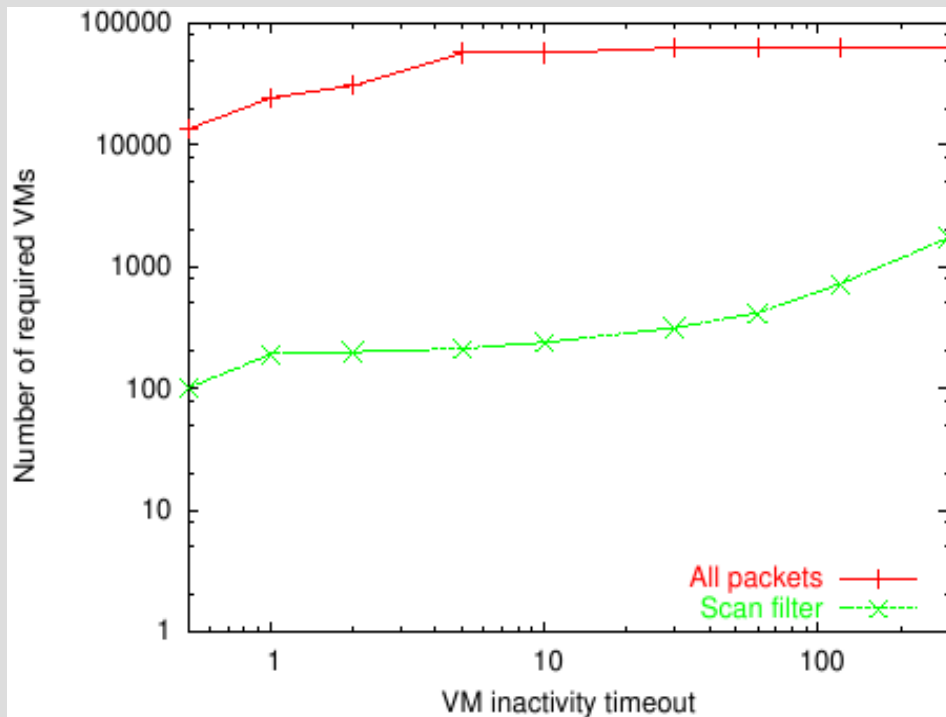
Evaluation: Multiplexing address space

- /16 network (64k IPs)
- Live traffic.
- VMs recycled after 500ms of inactivity.
- Average active VMs: 58.
- Peak active VMs: 13614.

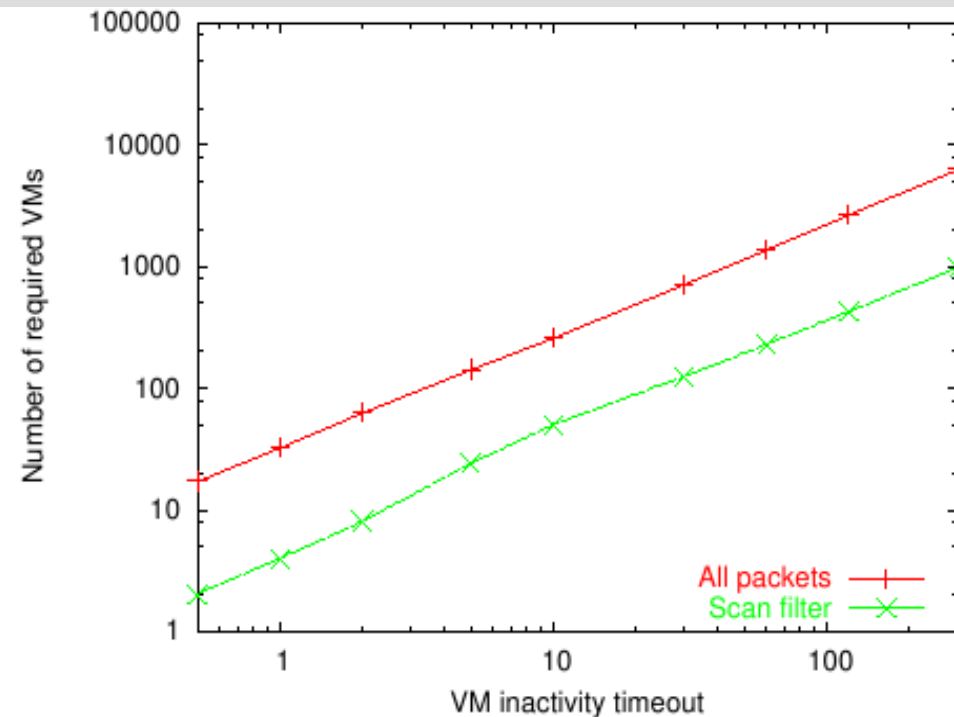


Evaluation: Multiplexing address space

- Max with filter and 5min timeout: 1745 VMs.
- Max without filter and 5min timeout: 62960 VMs.



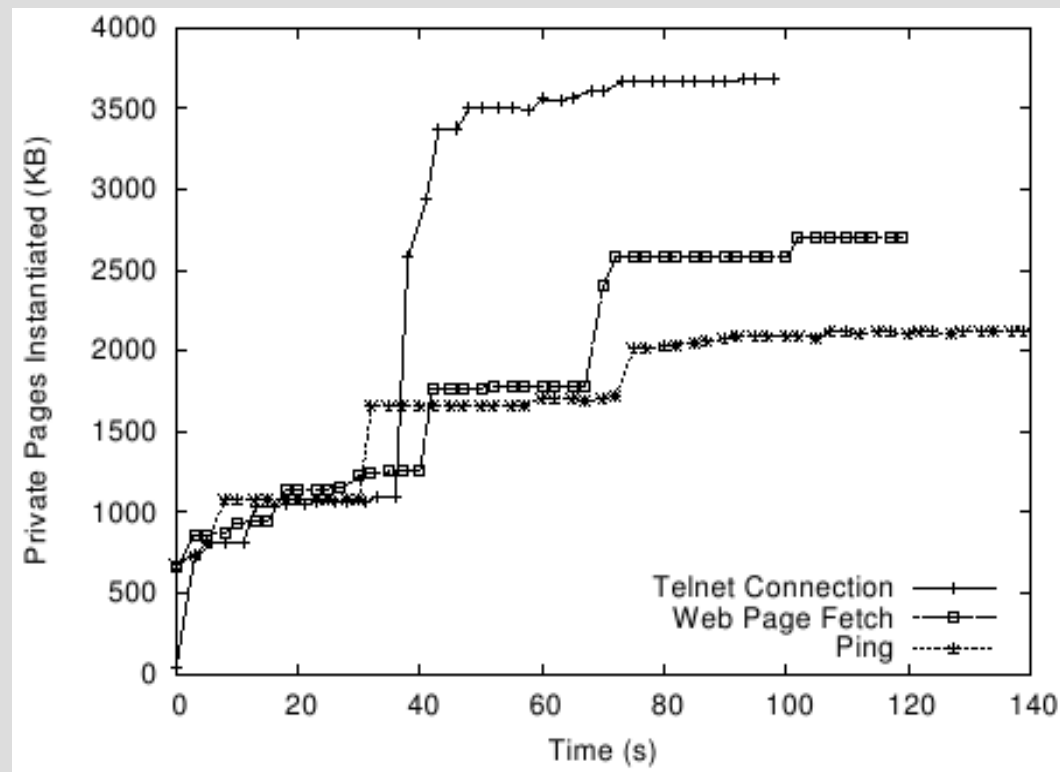
(a) Maximum number of simultaneous VMs



(b) Average number of simultaneous VMs

Evaluation: Delta Virtualization

- Reference VM: 128 MB Linux image.
- Xen heap limitations allows only 116 VMs, and 1310MB remain unused in a 2GB machine.
- 1500 VMs per server in theory.



Evaluation

- Flash cloning: 521.2 ms
- VM teardown: 315.5 ms
- VM CPU utilization (Apache HTTP request response): < 0.01%.

Evaluation: Gateway

- 160,000 packets per second (when hitting flow cache).
- 28,000 packets per second (random traffic).
- 256MB to support 1M flows.

Issues

- Potemkin is far from production (crashes, etc.).
- Can a honeypot attract traffic of interest?
- Some attacks require application specific topologies (email, p2p).
- Some malware requires human input (spyware).
- Without 100% virtualization bots detect honeypots.
- Even with 100% virtualization honeypot detection can occur by profiling.
- Virtualized honeyfarms have extra DoS vulnerabilities.

Thanks.