NC STATE UNIVERSITY Computer Science

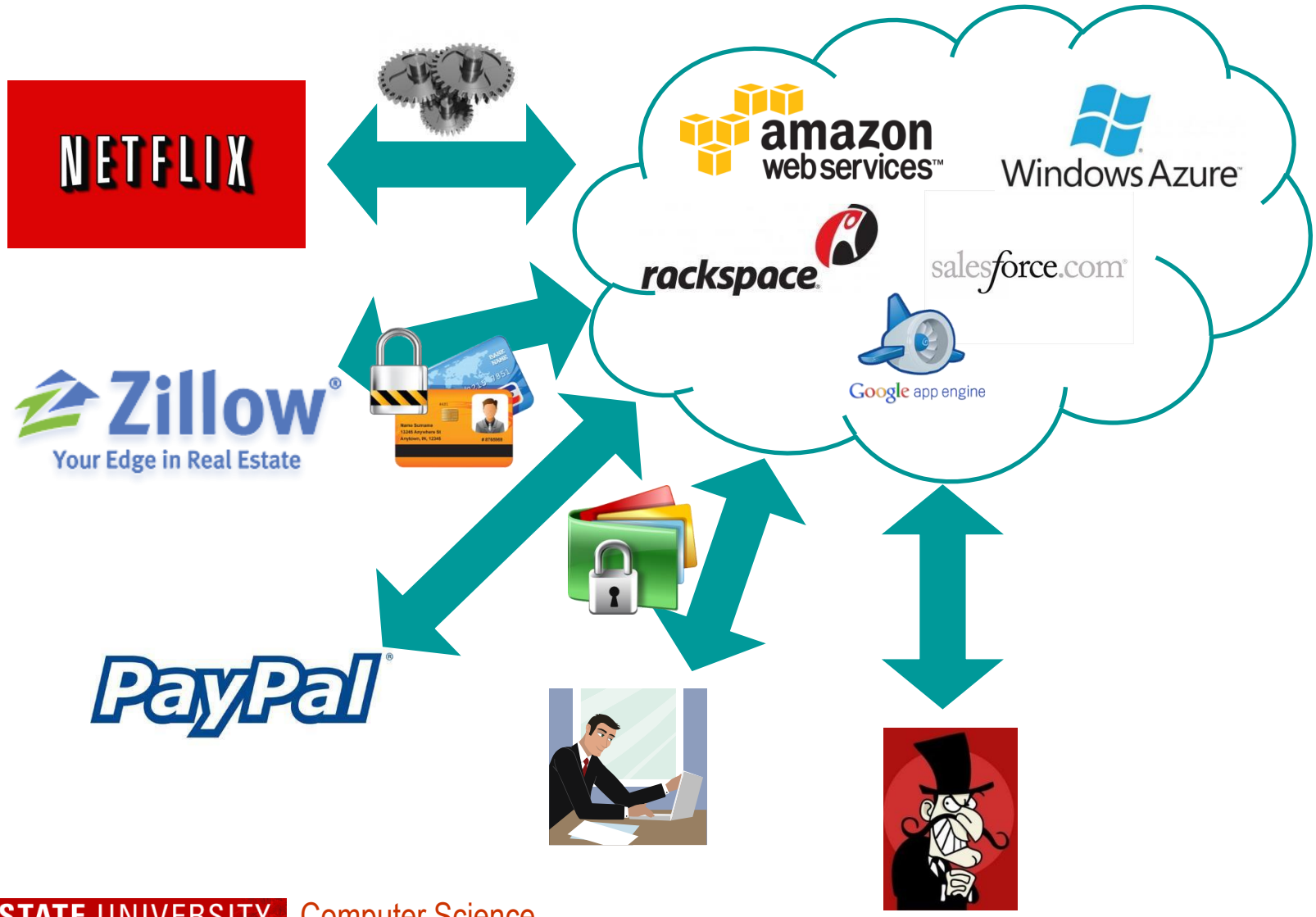
DACSA: A Decoupled Architecture for Cloud Security Analysis

Jason Gionta, William Enck, Peng Ning – NC State

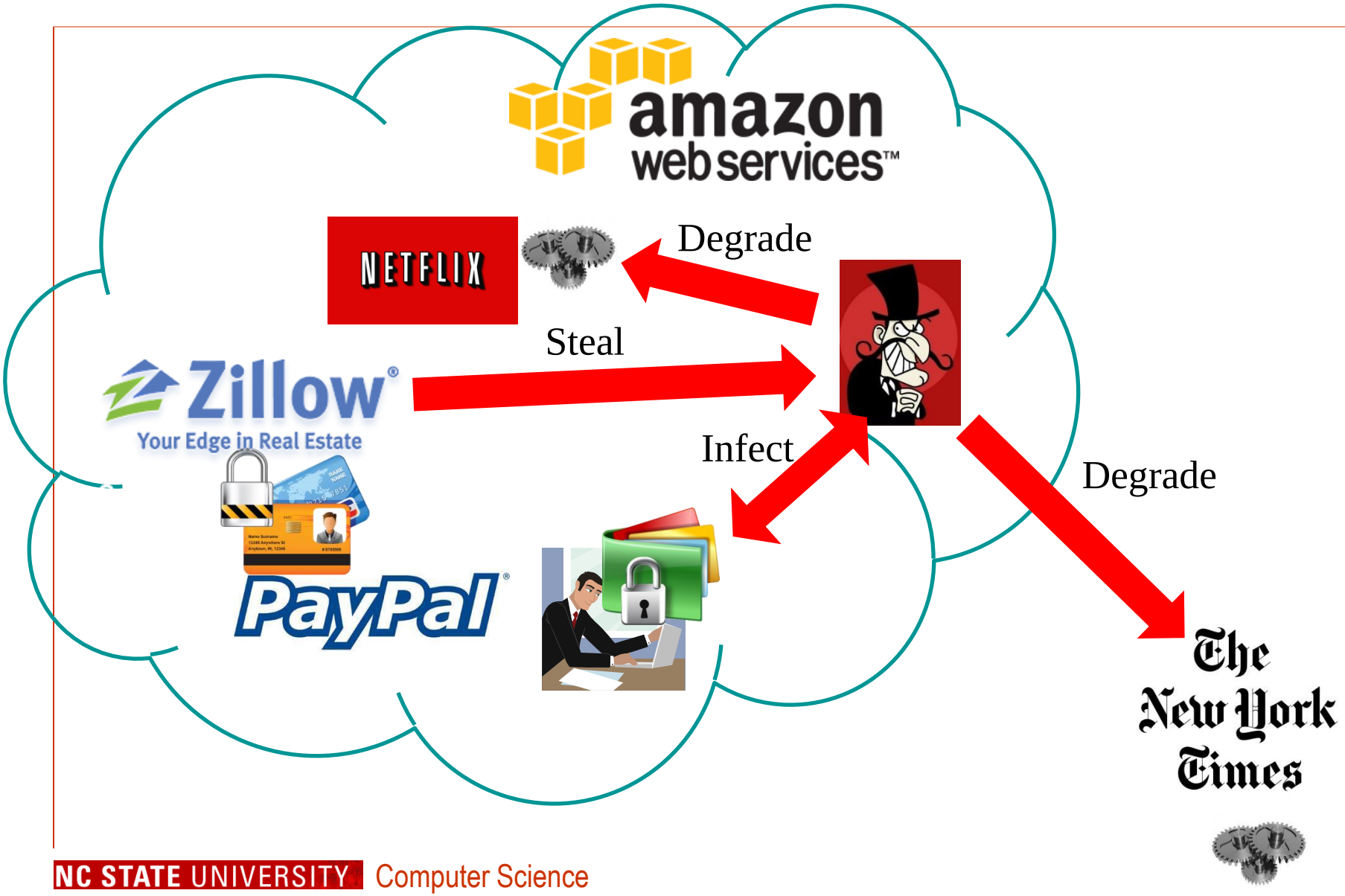
Ahmed Azab – Samsung Ltd

Xialoan Zhang – Google Inc

Cloud Provider Landscape



Cloud Provider Landscape

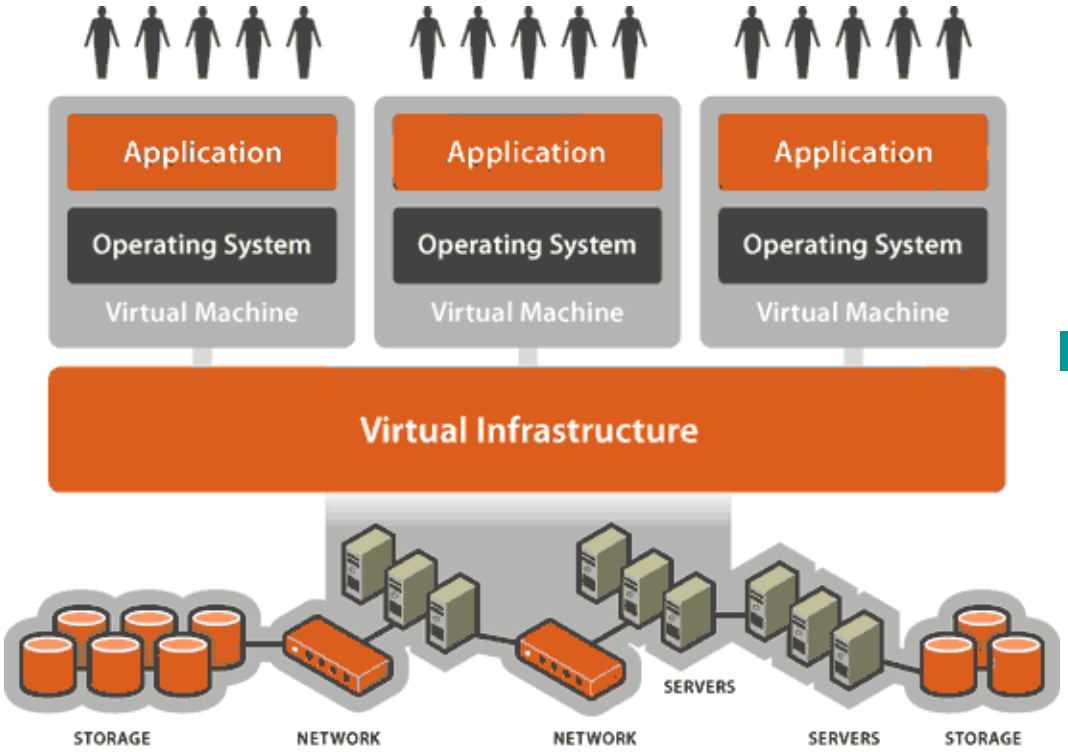


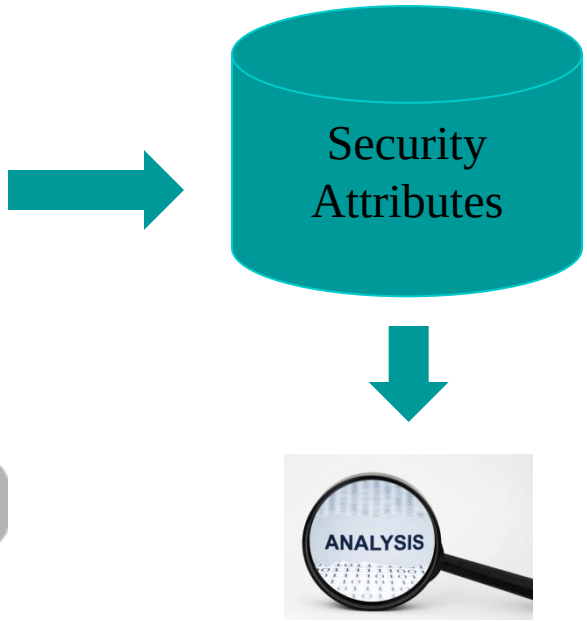
Cloud Provider Landscape

Unique Features of Cloud

- Diverse Components and Applications
- Single Platform Owner
- Geographically Dispersed

Cloud Infrastructure as a Security Testbed





How to Create a Datasource?

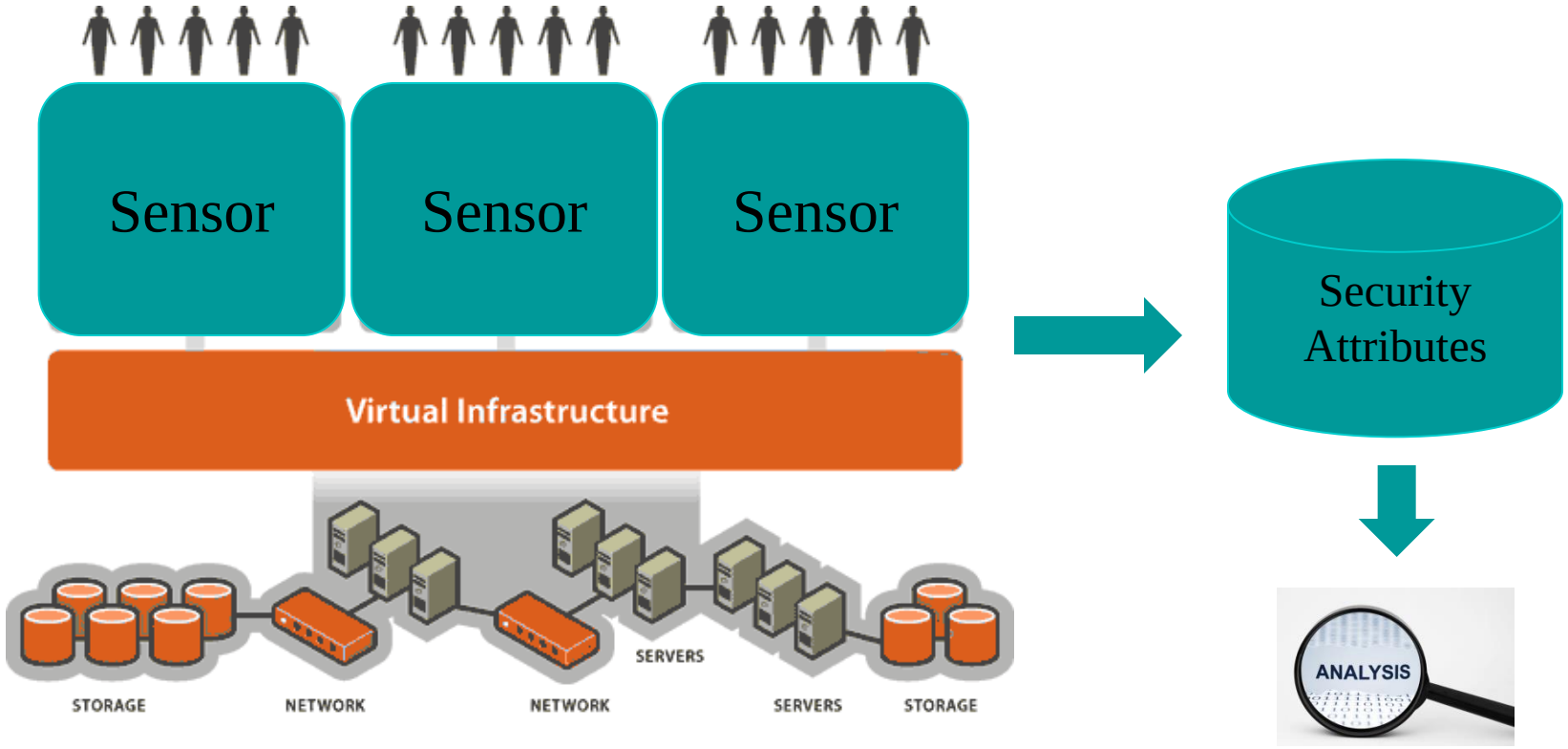
- Network Monitoring
  - Flow analysis
  - Encrypted network data
- In-Guest VM Monitoring
  - Virus Scanners / Security Software
  - Application Firewalls
  - Resource Intensive
  - Software Management
- Host Based “Out of VM” Monitoring
  - Peer into VM - VMWatcher
  - Record and Replay - Revirt
  - Scalability

How to Create a Datasource?

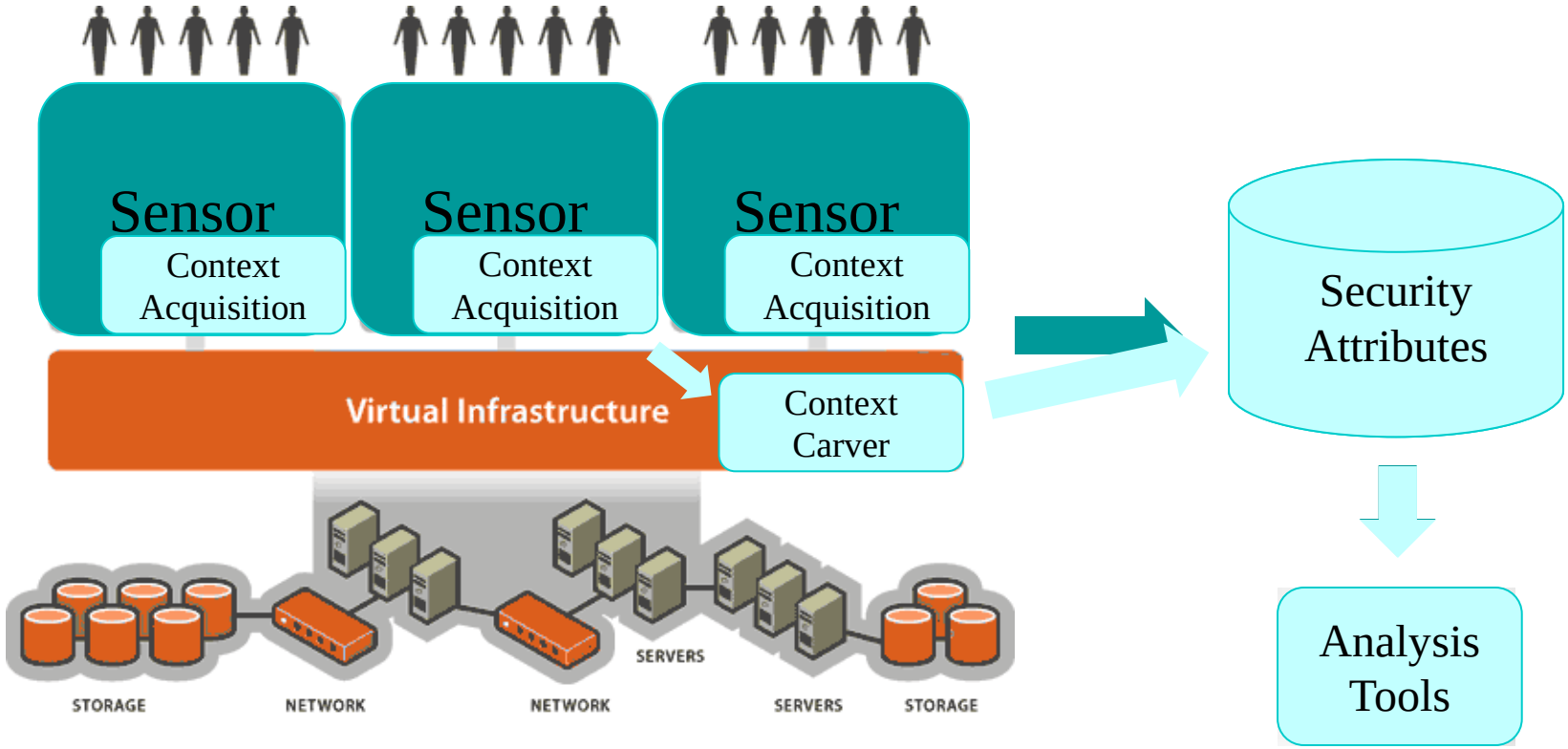
- Network Monitoring
  - Flow analysis
  - Encrypted network data
- In-G
  -
  -
  -
  -
- Host Based "Out of VM" Monitoring
  - Peer into VM - VMWatcher
  - Record and Replay - Revirt
  - Scalability

Decouple analysis from attribute acquisition

Decoupled Architecture for Analysis



DACSA



DACSA Components

DACSA Goals

- Limit impact to client VMs and hosts
  - Enable analysis on supporting infrastructure
- Transparent to clients
- Test for security violations
  - ToS
  - Bots/C&C
  - Malicious software development

Context Acquisition

- Fast Memory Snapshots
  - Logical memory copy of guest memory
    - COW
  - Limit impact to guest and host
- Reliable copies
  - Pause guest
  - Flush Asynchronous I/O

Carving Memory

- Apply memory forensic techniques
  - Extract security centric information
    - Open ports, registry keys, processes, API hooks
    - Hashes of executable pages
- Forensic tools
  - Volatility
- Work directly on memory
  - Interpose file I/O

Analysis

- Clustering of system features
  - Blacksheep – Bianchi et al.
- Memory based virus scanning
  - Memory only malware
- Security Audit
  - PCI Requirements

Implementation

- Host – Ubuntu 12.04 64-bit
- Guests – Windows 7 SP1 64-bit
- KVM/QEMU
  - Fork QEMU process
- Shared library for interposing File I/O
  - Volatility
  - Custom tool for parsing memory
    - Window 7 GS register to walk internal data structures
- Analysis
  - Scan viruses in memory

Evaluation

- Platform
  - IBM System X server Xeon E5450 Quad-core
  - Guests 1GB Ram
- Impact to Guest
- Impact to Host
- Correctly identify infected processes

Impact to Guest

- 1-15 VMs, snapshot VMs, carve process list
- Pause time
  - Flush Async I/O, Fork QEMU Process, Resume VM
  - ~.2112 seconds / standard deviation .07359 sec
- Reduction in system performance
  - Run Novabench in snapshotted VM
    - Measure CPU Ops/Sec and Memory Ops/Sec
  - 0-6% CPU, 0-3% Memory

Impact to Host

- 1-15 VMs, snapshot VMs, carve process list
- Increased CPU and Memory Utilization
  - ~3% CPU
  - Negligible Memory overhead
- Write Working Set
  - 100-300 MB per minute

Carving Process Memory

- Infected VM with Cerberus RAT
  - iexplore.exe host process
- Carved process memory
- Scanned memory with ClamAV
- Identified infected process

Related Work

- Live VM Migration (Clark et al.)
  - Migrations takes upwards of 90 seconds
  - Performance degradation upto 20%
- Fast VM Cloning (Sun et al.)
  - COW based by write protecting pages
  - Technical challenges of cloning

Conclusion

- DACSA turns clouds into a platform for security analysis
  - VMs lightweight sensors
  - Minimal impact to VM and host operations
- Apply large scale analysis
- Future Work
  - Deploy to Virtual Computing Lab at NC State
  - Memory Scanning as a Service

Questions?

- Thanks
  - Reviewers insightful comments
  - Eric Eide for shepherding

jjgionta@ncsu.edu