VM Leakage & Orphan Control in Open-Source Clouds

Christopher Dabrowski and **Kevin Mills**

National Institute of Standards and Technology

IEEE CloudCom 2011,

Athens, Greece

Dec. 1, 2011

Presentation Summary

VM leakage

- A type of *resource leakage, in which resources are VMs in cloud systems.*
- Leaked or <u>*orphaned VMs*</u> are VMs allocated within clouds that are unknown to any user and that are not in the process of being terminated
  - Orphaned VMs may persist indefinitely.
  - Orphaned VMs retain assigned computing resources (e.g., virtual cores, memory, disk space), which cannot then be allocated, and <u>*so are lost (or leaked)*</u>.

Potential vulnerabilities exposed VM leakage

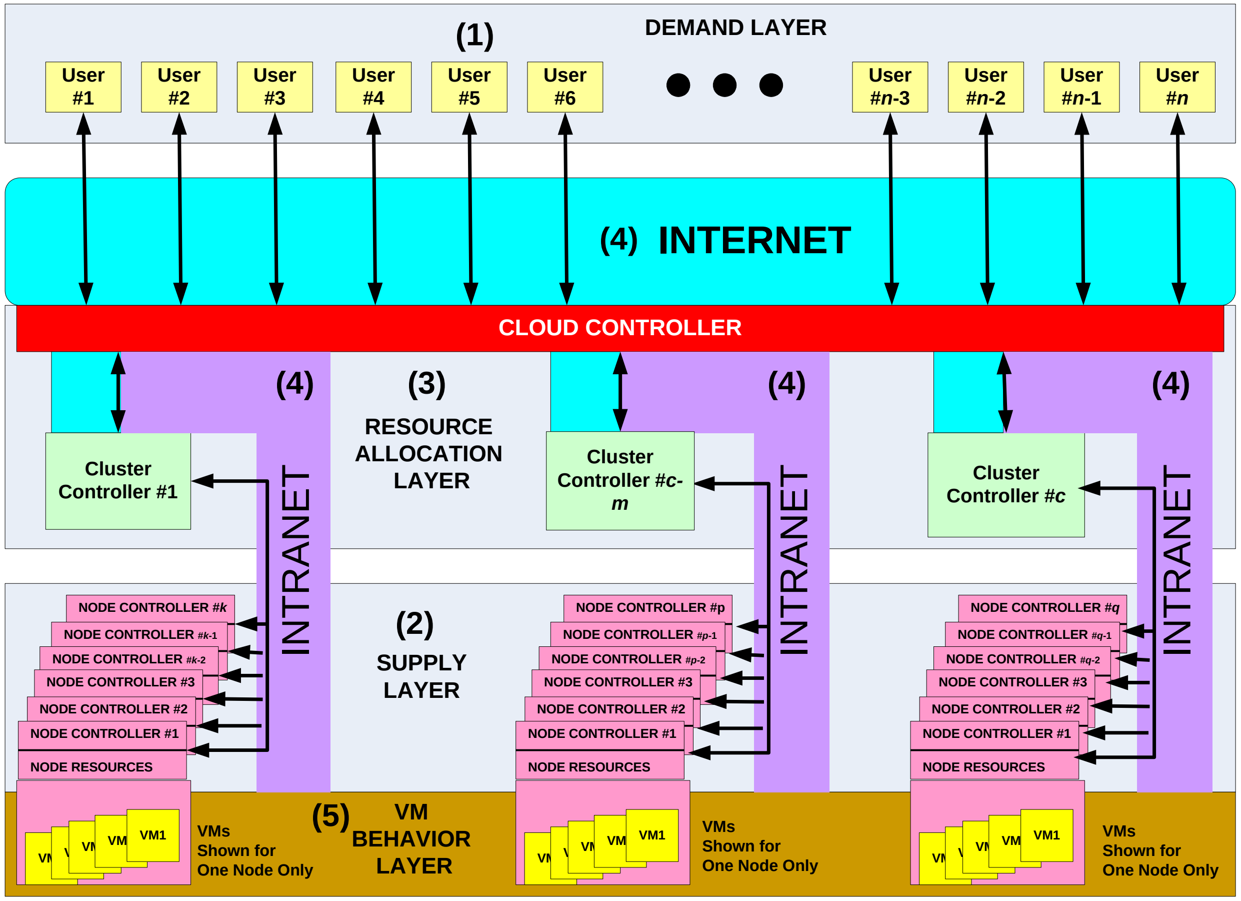
- Using the ***Koala simulator***, we show a Trojan message discard attack in compromised Web-server code leads to VM leakage that exhausts resources in an open-source IaaS cloud.

One Possible Solution

- Adding <u>*two orphan control processes*</u>: (1) Creation orphan control and (2) persistent termination, which mitigate the VM Leakage.

Koala Simulator

Models open-source cloud based on Eucalyptus structure and Amazon EC2 interface



→ Demand Layer: users request, hold, & terminate VMs (1)

→ Internet/Intranet Layer: Web services communications (4)

→ Resource Allocation Layer: Cloud and cluster controllers respond to user requests to allocate VMs (3)

→ Supply Layer: Node controllers on clusters host VMs on nodes and execute user operations (2)

→ VM Behavior Layer: N/A (5)

Relevant features of Koala Model

- The VM allocation process
  - User requests (*Runinstances* request)
  - Cloud controller canvass clusters for allocation estimates and chooses
  - Types of Allocations: ***Partial*** and ***Full***
  - ***NERA*** response signifies insufficient resources exist for partial grant
  - Users hold VM allocations for random durations; release and repeat cycle
- Other operations (submitted by user to cloud and forwarded to clusters and nodes)
  - *DescribeInstances* – replies to determine when allocated VMs are ready for logon
  - *TerminateInstances* (intermediate and final)
- Retry regimens
  - Users retry operations (*RunInstances* and T*erminateInstances*), but Eucalyptus does not retry forwarding of operation requests from cloud controller to cluster controller and from cluster to node controller

    → ***This can create problems when messages are lost.***
  - Note: each user retry of *RunInstances* request is treated as a separate and new request by the cloud

Causes of VM Leakage and Orphan Creation

Creation orphans – VMs are created in response to a user request, but any of 3 different confirmation messages are lost:

1. From node to cluster controller confirming VM creation
2. From cluster to cloud indicating successful (full or partial) VM allocation
3. From cloud controller to user indicating a successful result.

In (2) or (3), all VMs in request become orphans. User re-requests are treated as new requests by cloud—*thus increasing potential for creation orphans*

Termination orphans. After user receives requested VMs, subsequent termination operations may fail (*TerminateInstances*) due to lost messages:

- User may receive termination confirmation and think all is OK. If not, usually makes limited number of retries and then stops.
- Eucalyptus makes *no provision for retrying failed termination requests* by cloud or cluster controllers; instead such failures are merely logged.

Thus, *related VMs will remain un-terminated* and may persist on nodes until an administrator scans the log and manually terminates them.

Orphan Control Methods

- **Creation orphan control.** Node controller monitors receipt of *DescribeInstances* requests from users → if not received for a VM by 2h after boot up, VM is terminated and its resources are released.
- **Termination orphan control.** Eucalyptus protocol extended to provide a ***persistent termination*** capability, i.e., resending termination requests until the receiver responds that either (1) the termination request has been received or (2) the termination operation was completed earlier.
  - A *persistent terminator* starts when no response is received to a normal termination request within 90 s.
  - ***Cloud persistent terminator*** resends termination requests to a cluster controller at (90s) intervals. After three retries, doubles retry interval, and persists up to 2 h.
  - ***Cluster persistent terminator*** same but persists for shorter time because on a single VM is affected.

- Note: since persistent termination adds system complexity, we limited it to final termination requests. Therefore, failed intermediate terminations *can result in temporary orphans* until a final termination request succeeds.

Experiment Design and Trojan Attack

Trojan Attack

- Trojan software components are introduced into open-source Web server software distribution at the Internet/Intranet Layer
- Causes arriving and departing messages to be randomly discarded by Web server in users, cloud controllers, cluster controllers, and node controllers.

  → *All system messages, regardless of type or originator, are subject to loss*.

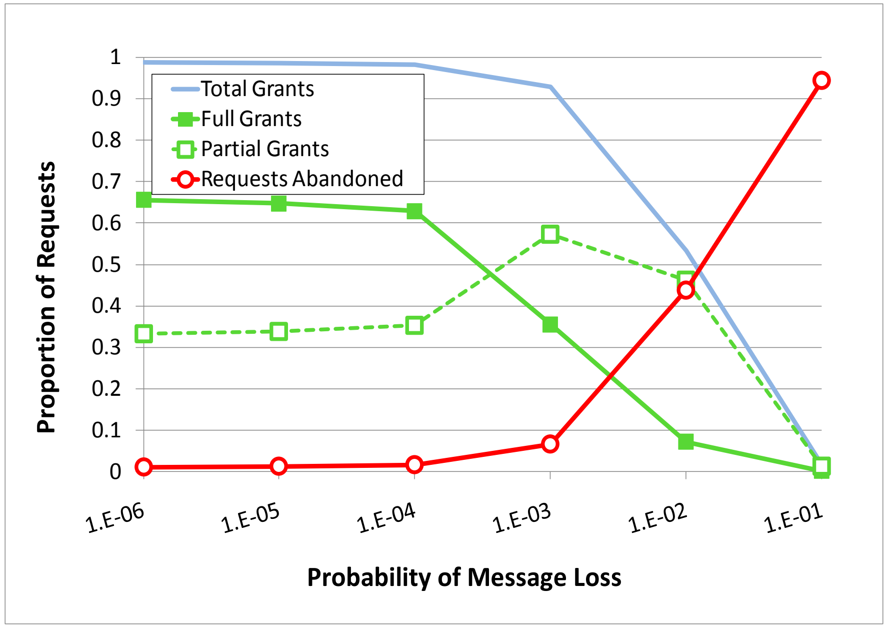
Experiment Design

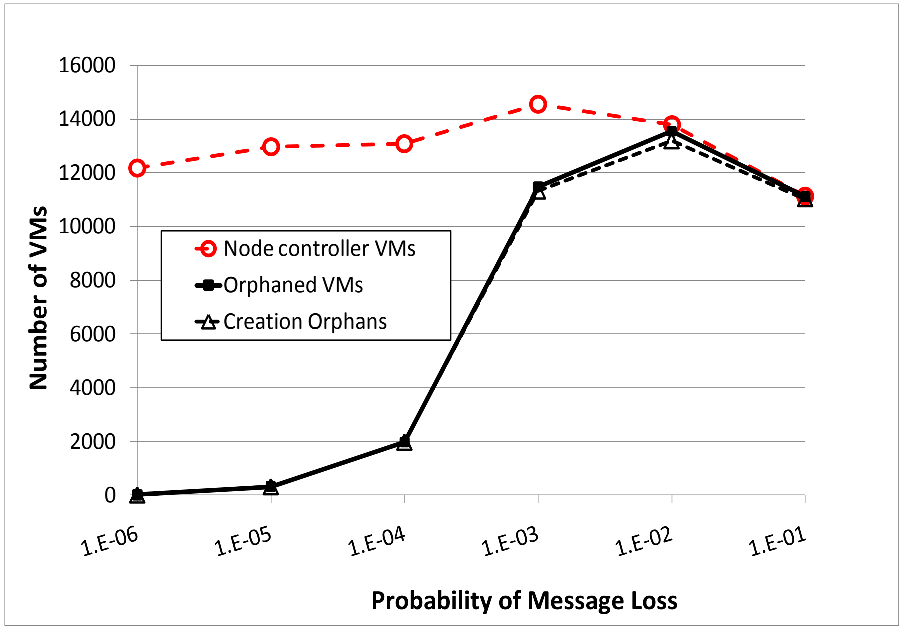
- Executed the model with six order-of-magnitude increases in message loss probability from one in 10^6 messages to one in 10^1
- System operation modeled at each of the six message loss levels, with and without each of the two orphan control methods

  → 24 combinations: 6 message loss rates × 2^2 orphan control on/off values.
- Executed Koala for 1000 simulated hours for each combination and measured system performance at 1 h intervals

Impact of the Trojan Attack without Orphan Control

- With increased intensity of attack and without orphan control, nearly all allocated VMs can become orphans, and up to 95% of users are un-served.
- Performance crashes as incidence of message loss reaches 1.E-01: very few grants and many NERAs. All VMs are orphans and no resources remain:

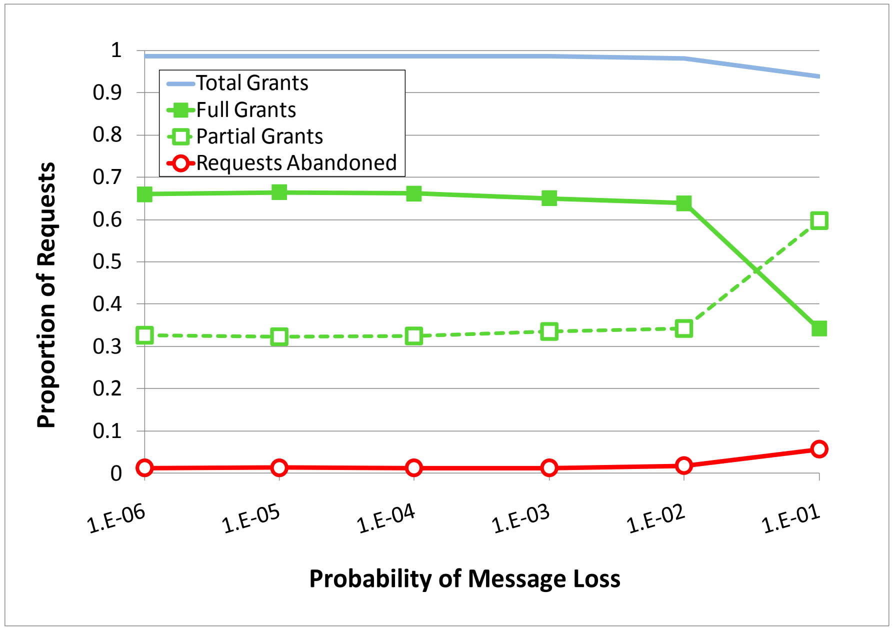


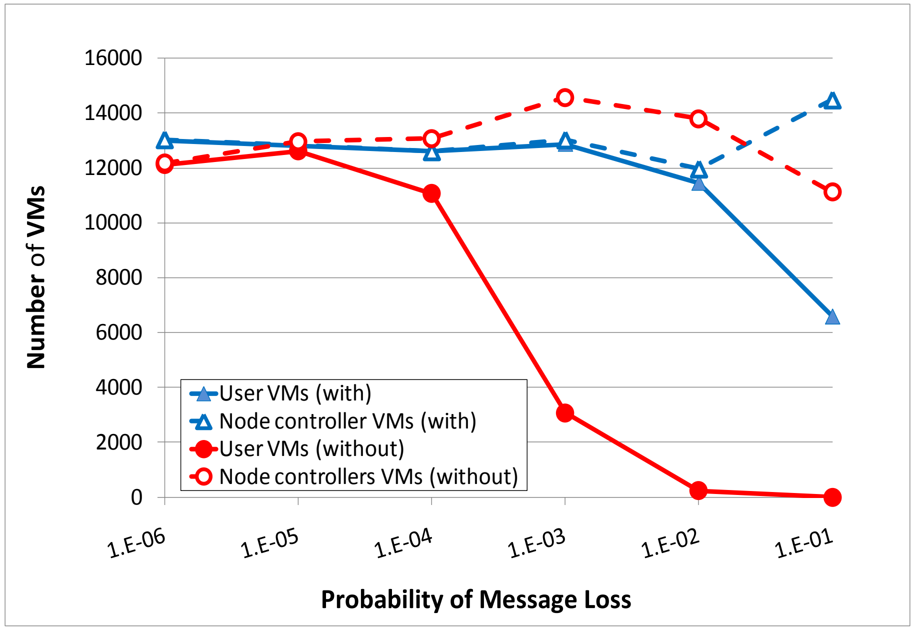


- Creation orphans dominate, because opportunities to occur happen earlier in the request cycle and more often (but see below).

Impact of the Trojan Attack *with Orphan Control*

- Orphan control allows recovery of VMs for allocation to incoming requests; restores performance of system at high attack intensity: P(loss) = 1.E-01.
- Both forms of orphan control needed: when creation orphan control omitted, collapse is nearly total; when persistent termination omitted performance decline is still severe (48.1% of users un-served).



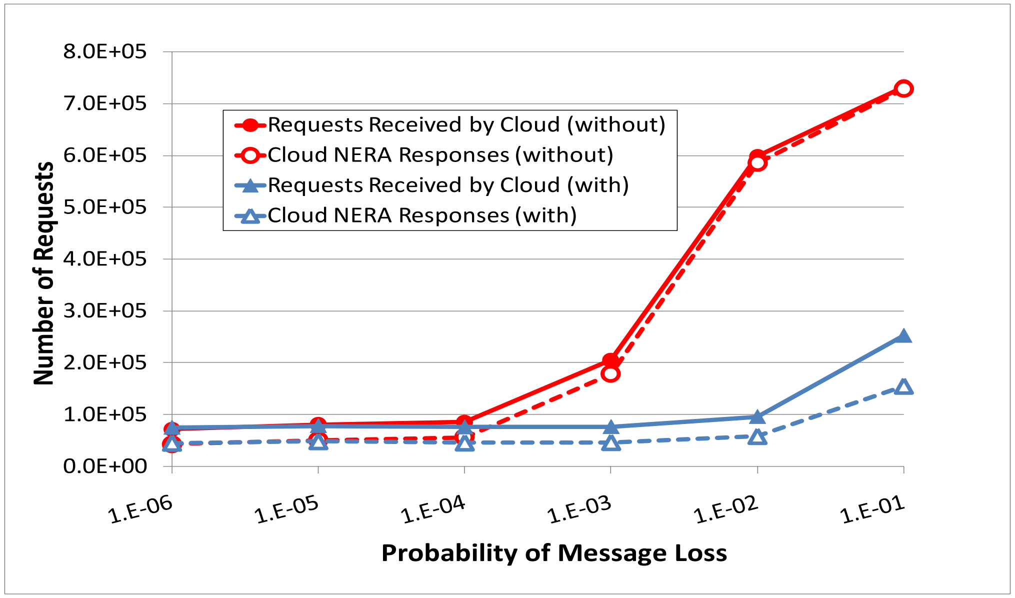


- However, even with orphan control, a larger proportion of grants are partial when attack intensity is high: P(loss) = 1.E-01.

Reason: temporary orphans still reduce VM availability!

Operation of Cloud with and without orphan control

- Without orphan control (red), as attack intensity increases:
  - number of requests received increase (due to user retries)
  - Overall number of messages increase from 2.6E+07 to 6.4E+07 as P(loss) goes from 1.E-06 to 1.E-01
  - NERAs increase (because orphan VMs unavailable)
- With orphan control (blue), the problem is mitigated:
  - Overall number of messages increase to 4.5E+07 as P(loss) goes to 1.E-01 (Note that only 0.44% of messages relate to orphan control)



Conclusions

- VM leakage is a type of resource leakage specific to clouds
  - → represents a potential vulnerability that can lead to resource exhaustion and performance degradations (as shown in simulated operation)
  - → can be exploited by Trojan attack that introduces malicious code modifications to an open-source cloud implementation.
- Orphan control methods are needed to detect and eliminate VM orphans
  - → allows cloud to sustain a higher level of resource availability during malicious attacks.
- VM leakage is a potential problem that must be considered in the design of cloud systems, if these systems are to be reliable.
  - → Scale of the problem precludes manual discovery and removal of VM orphans by system administrators
- In the future, it will be necessary to investigate other orphan control methods and extend this work to obtain a more general understanding the potential for VM leakage in different kinds of cloud systems operating under a wide range of conditions.

Questions?

Contact information about studying Complex Information Systems:
{kmills, jfilliben, cdrabowski@nist.gov}

Contact information about Cloud Information Visualization:
sressler@nist.gov

Contact information about NIST Cloud Computing Program:
dawn.leaf@nist.gov

For more information see: http://www.nist.gov/itl/antd/emergent_behavior.cfm
and/or http://www.nist.gov/itl/cloud/index.cfm