CNS 2020

Catching Falling Dominoes: Cloud Management-Level Provenance Analysis with Application to OpenStack

Azadeh Tabiban[1], Yosr Jarraya[2], Mengyuan Zhang[2],

Makan Pourzandi[2], Lingyu Wang[1] and Mourad Debbabi[1]

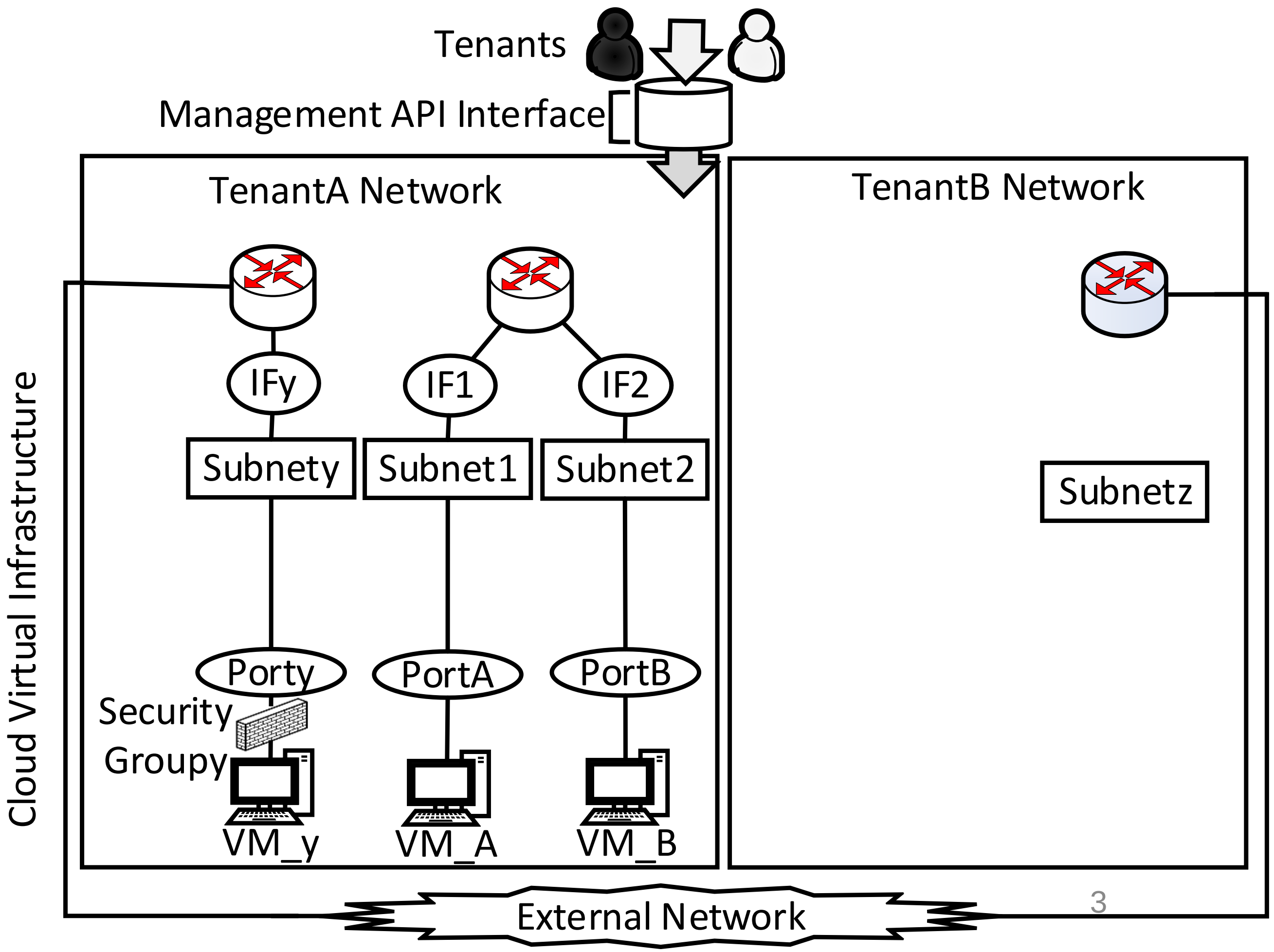
[1] Concordia University, Canada, [2] Ericsson Security Research, Canada

Outline

- Cloud Security Challenge
- Limitation of Exiting Solutions
- Cloud Provenance Model
- Methodology
- Implementation
- Experiment Results
- Conclusion

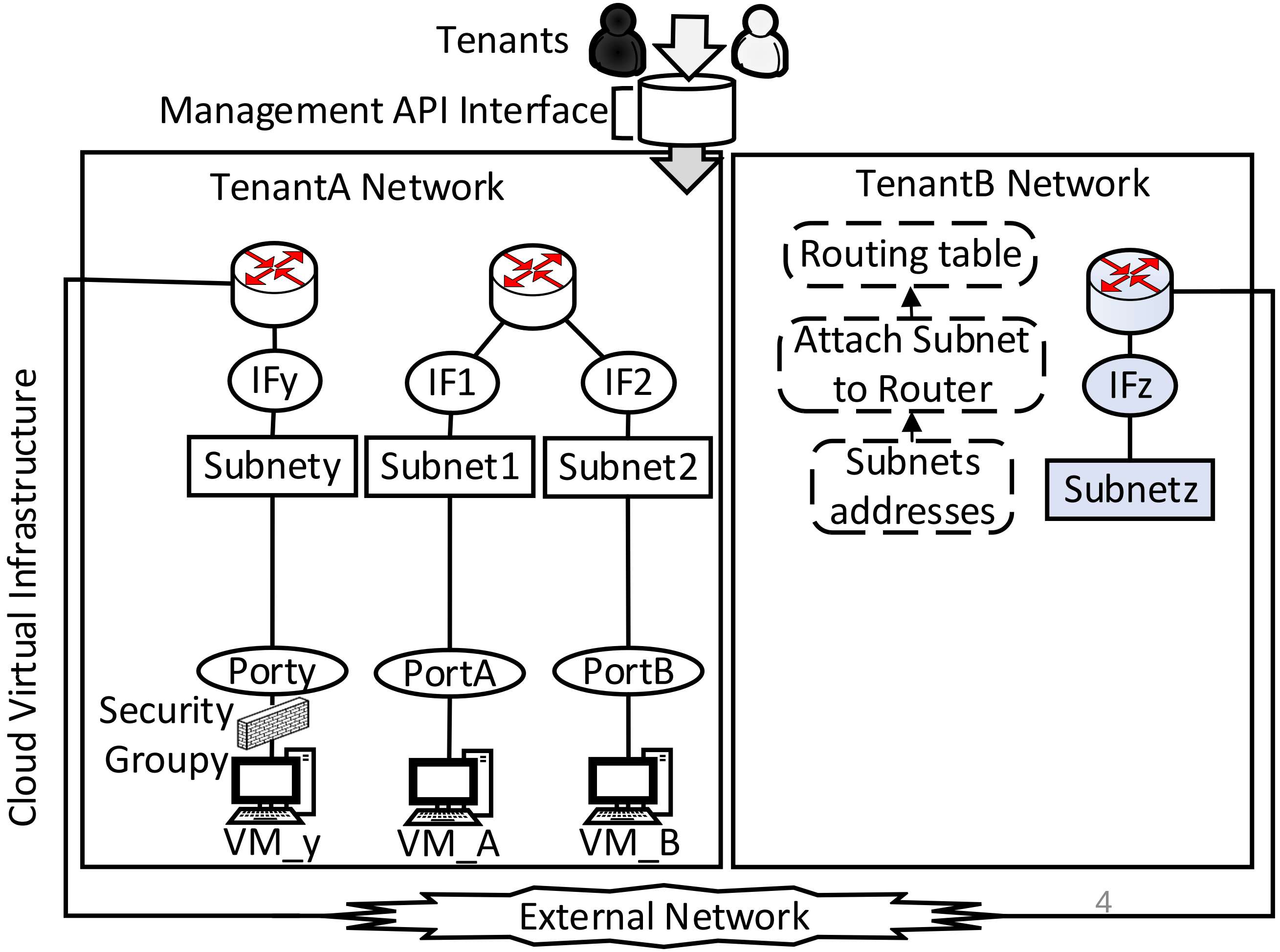
Cloud Virtual Infrastructure

- Cloud computing has been widely adopted to provide users the ability to self-provision resources while optimally sharing the underlying physical infrastructure



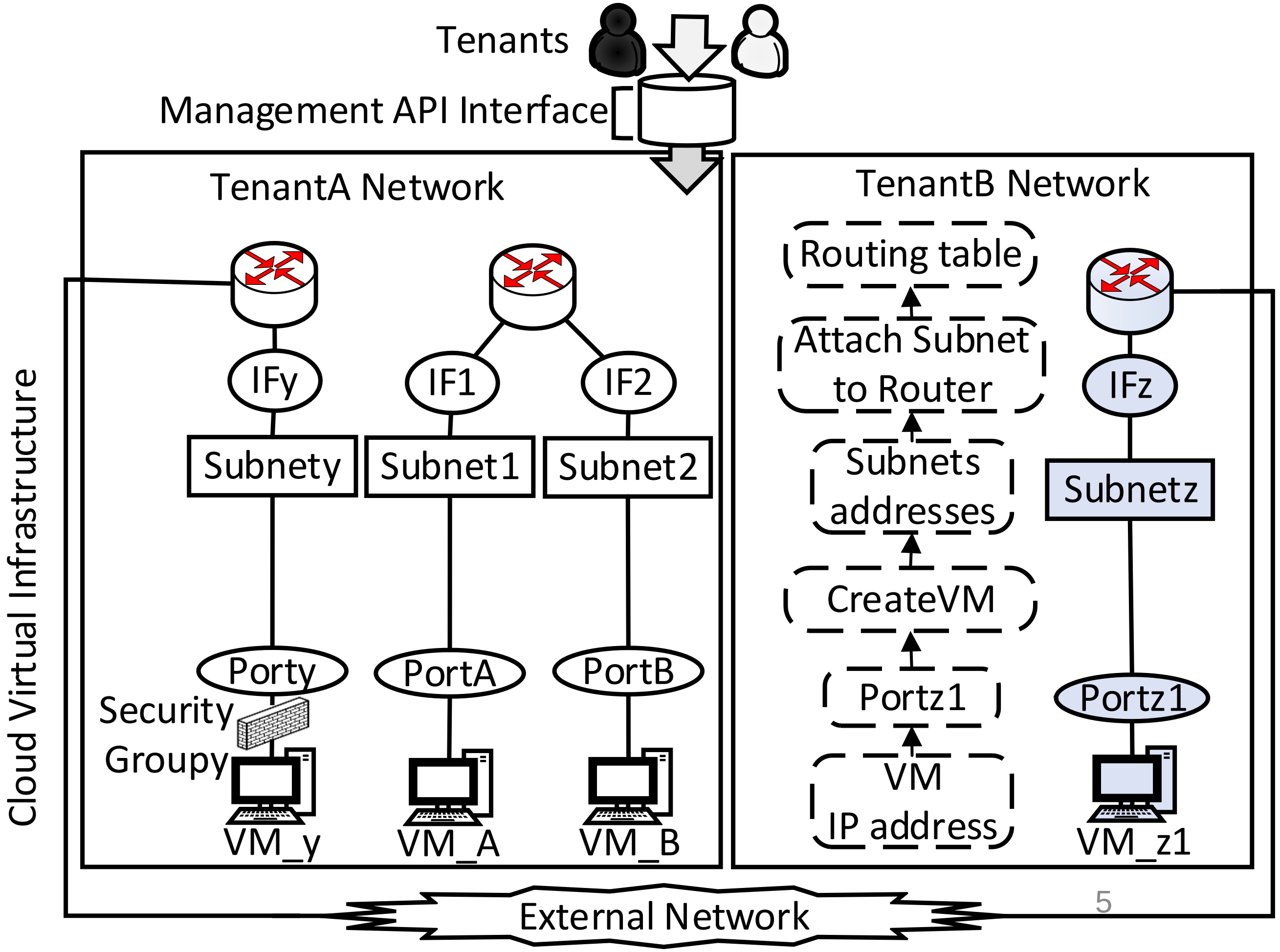
Cloud Virtual Infrastructure

- Cloud computing has been widely adopted to provide users the ability to self-provision resources while optimally sharing the underlying physical infrastructure



Cloud Virtual Infrastructure

- Cloud computing has been widely adopted to provide users the ability to self-provision resources while optimally sharing the underlying physical infrastructure

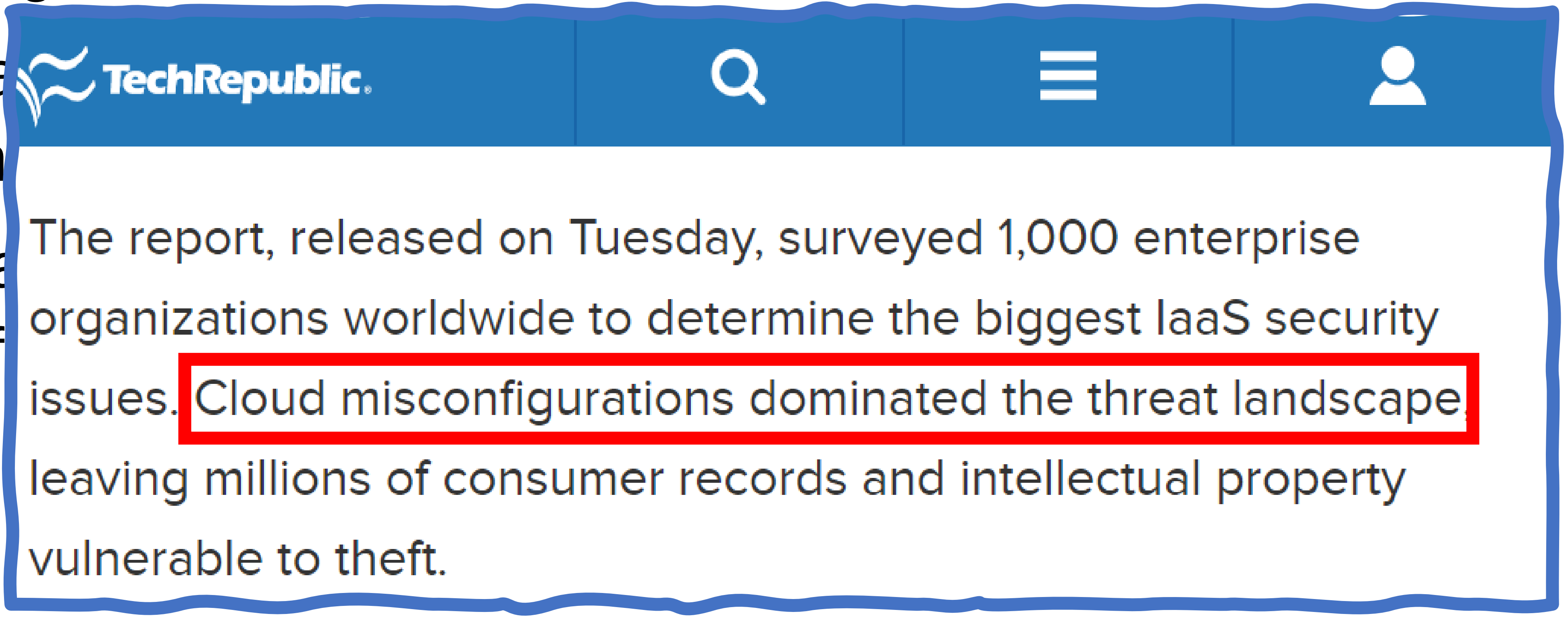


Cloud Virtual Infrastructure

- The self-service and multi-tenancy nature of clouds also leads to a higher complexity and greater chances of misconfigurations
- Adversarial actors can launch their attack by exploiting cloud misconfigurations
- This makes explaining systems behaviour difficult → The way to find the root cause

Cloud Virtual Infrastructure

- The self-service and multi-tenancy nature of clouds also leads to a higher complexity and greater chances of misconfigurations
- Adversa
  cloud m
- This ma
  way to f



TechRepublic.

The report, released on Tuesday, surveyed 1,000 enterprise organizations worldwide to determine the biggest IaaS security issues. Cloud misconfigurations dominated the threat landscape, leaving millions of consumer records and intellectual property vulnerable to theft.

The Need for Root Cause Analysis

- The need for finding the root cause:
  - Forensic analysis
  - Debugging
  - Prevention of recurrent failures
  - Recovery

Outline

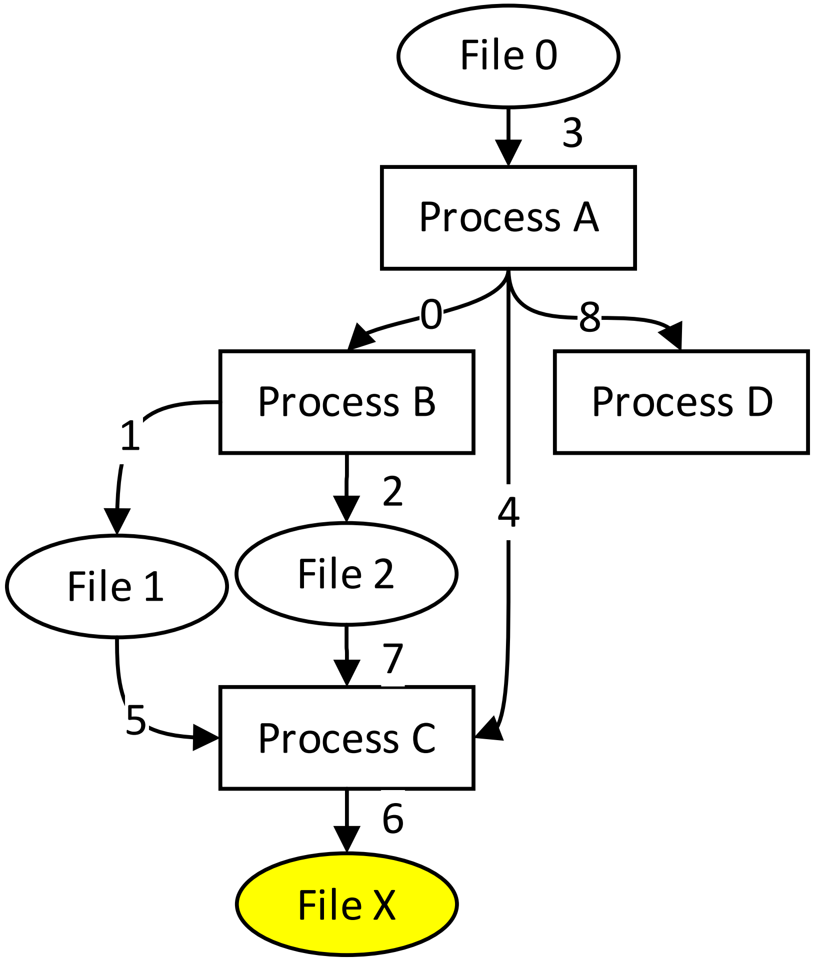
- Cloud Security Challenge
- Limitation of Exiting Solutions
- Cloud Provenance Model
- Methodology
- Implementation
- Experiment Results
- Conclusion

Limitation of Existing Solutions

- **Problem localization solutions**
  - E.g., using alert correlation [1]
  - Not providing root cause operations
- **Cloud logs investigation**
  - No intrinsic central view of changes in different services
  - Log aggregation cannot explain the interdependencies between events
  - Logs are not sufficiently expressive

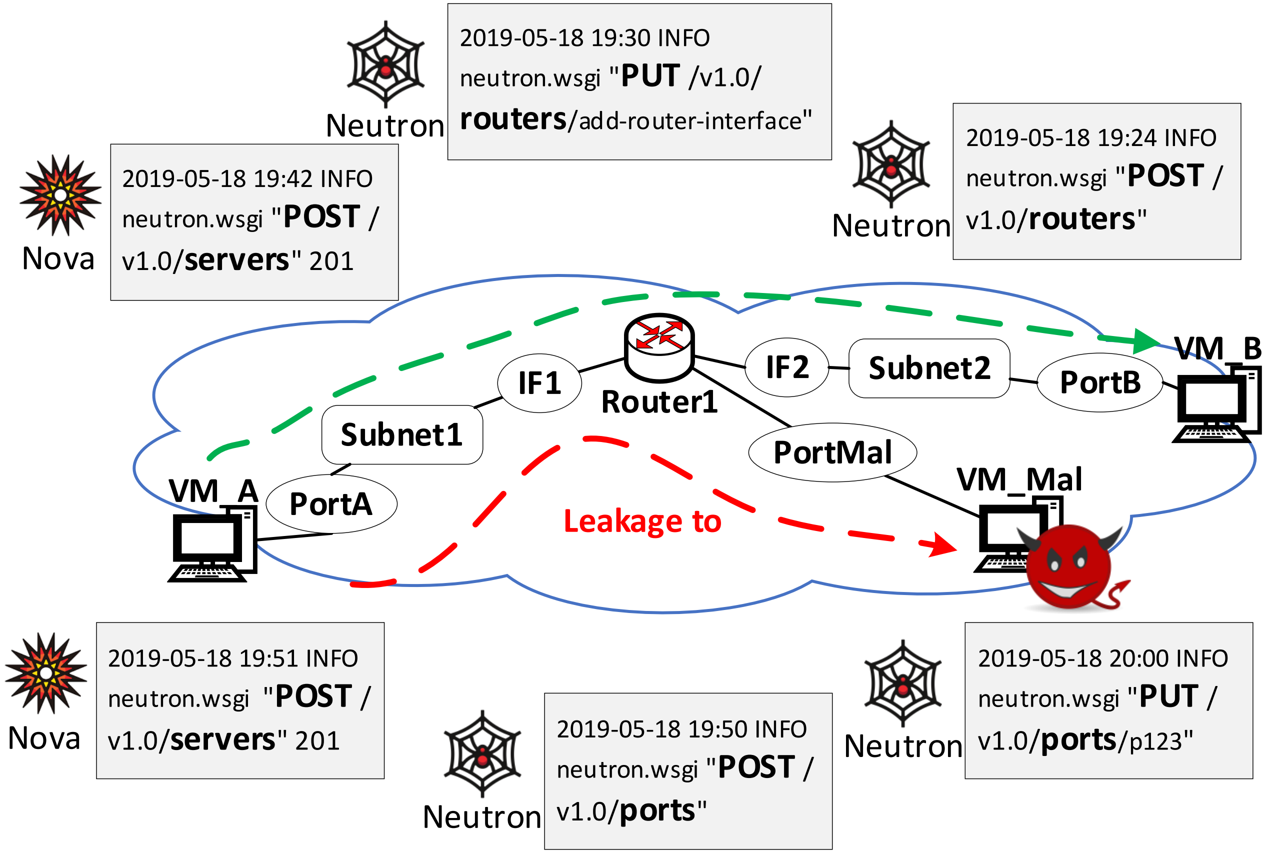
Limitation of Existing Solutions

- **Existing provenance solutions**
  - On Low-level system calls [2, 3] and not sufficient for clouds
    - Big size of generated records
    - Tedious and error-prone analyses
    - Storage and network overhead
  - Impractical interception mechanisms in clouds
    - Requiring system instrumentation



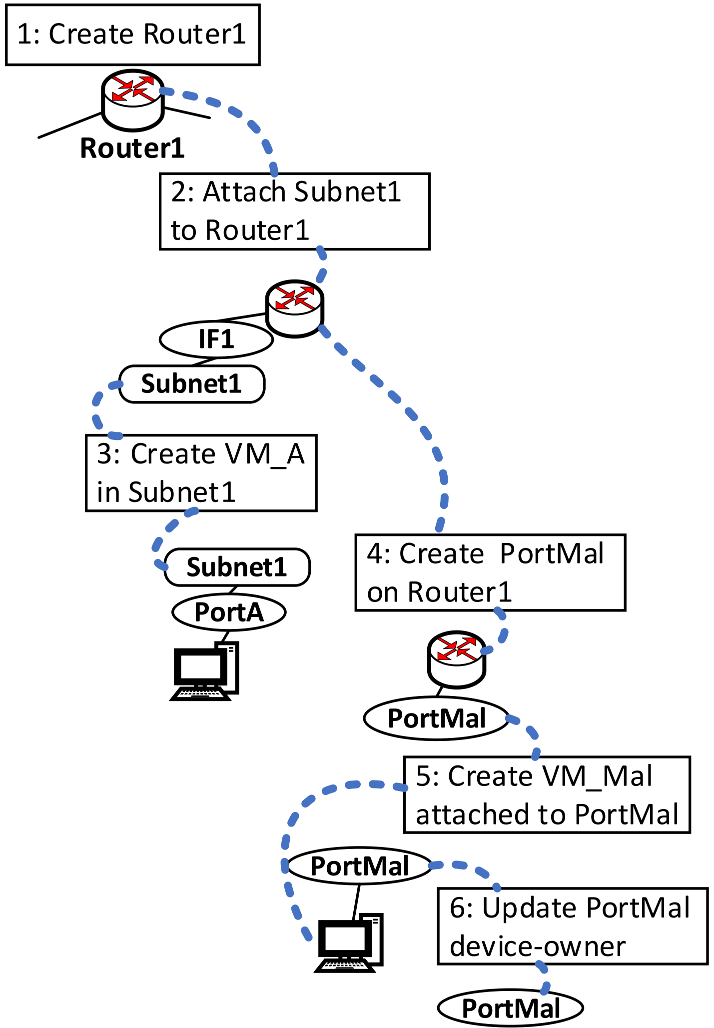
Example Attack Scenario

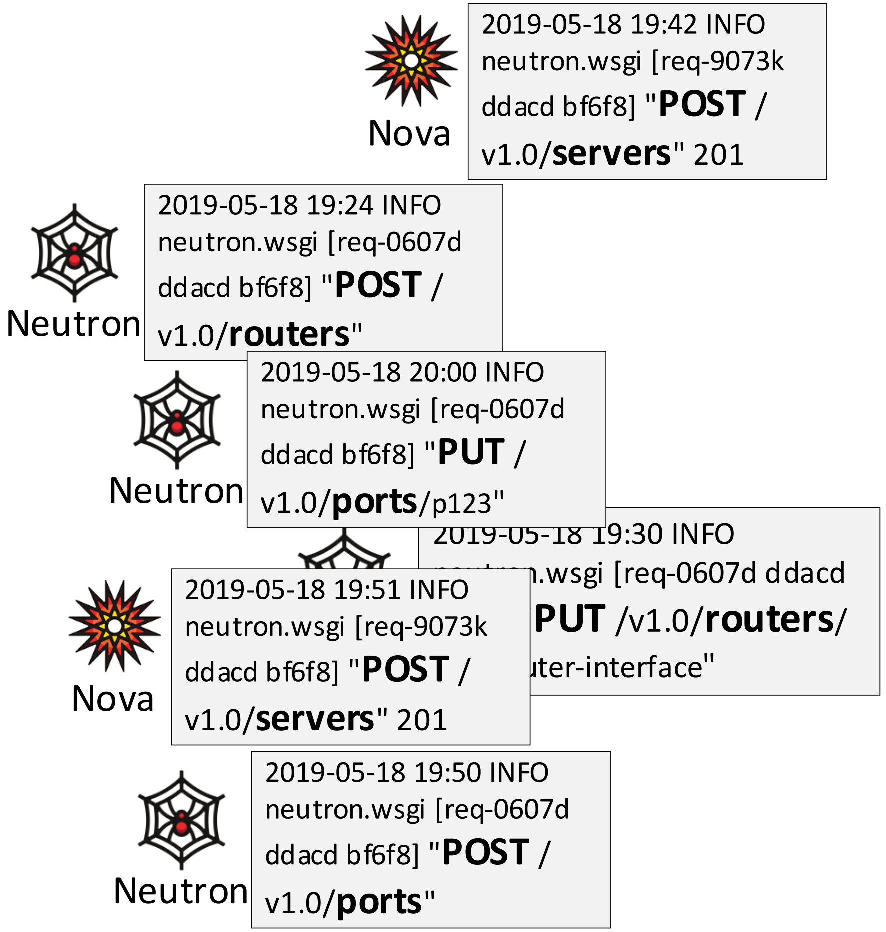
- A data leakage alert is released
- Based on the logs, the cloud admin cannot explain the complex chain of events leading to the attack
  - Interdependency
  - Expressiveness



Example Attack Scenario

- **What if we could find the interdependencies?**





DominoCatcher

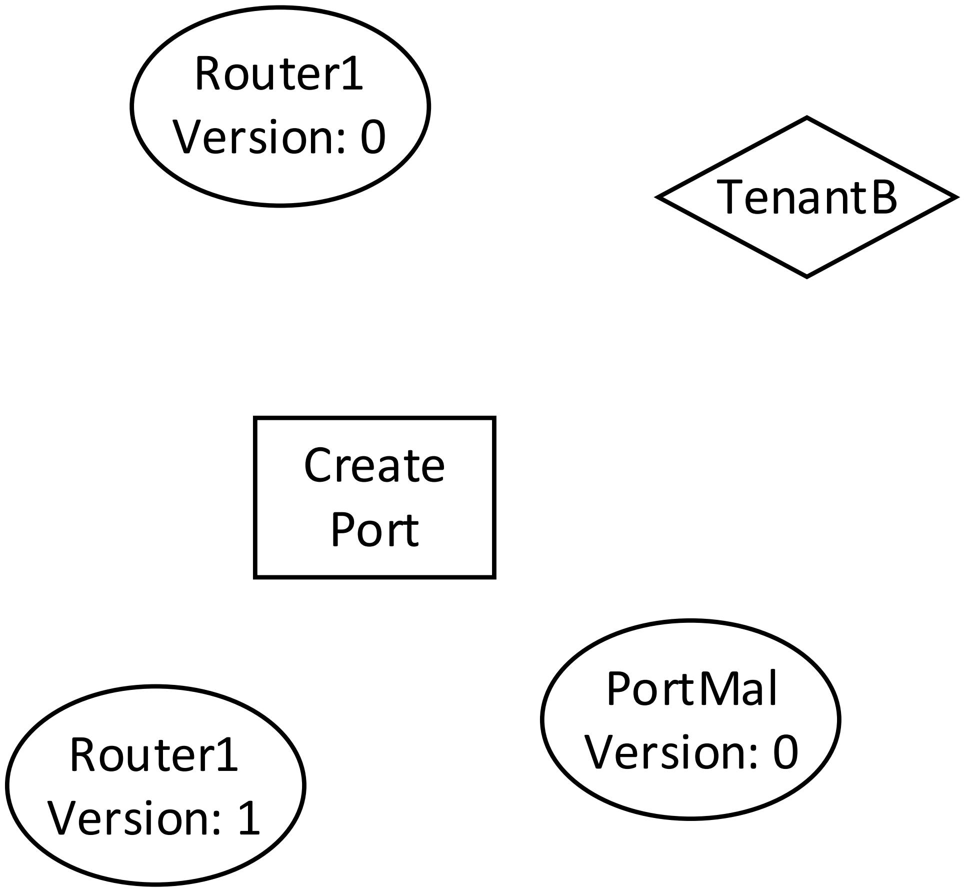
- The first provenance model at cloud management level
- A mechanism to capture the provenance metadata from different services in clouds and construct the provenance graph
  - A less invasive interception mechanism
  - Incremental construction
- Provenance-based forensic approach
  - User preference-based pruning
- Implementation based on OpenStack

Outline

- Cloud Security Challenge
- Limitation of Exiting Solutions
- Cloud Provenance Model
- Methodology
- Implementation
- Experiment Results
- Conclusion

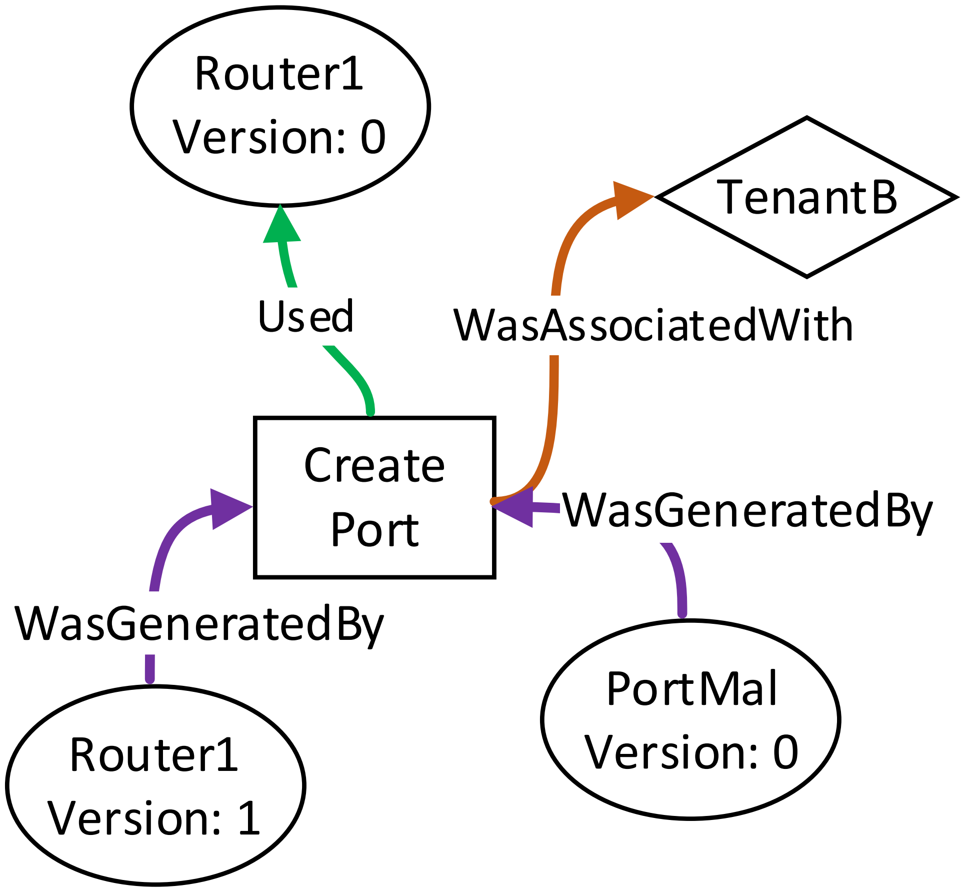
Cloud Management Provenance Model

- Defined based on a standard provenance specification [4] where:
  - **Nodes**:
    - *Entities:* virtual resources
    - *Activities:* cloud management operations
    - *Agents:* cloud tenants or users



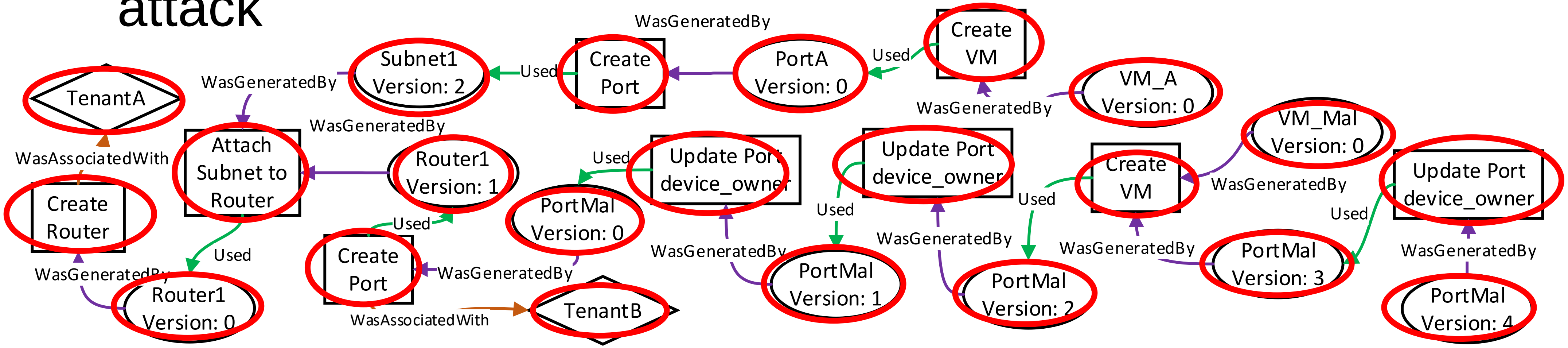
Cloud Management Provenance Model

- Defined based on a standard provenance specification [4] where:
  - **Edges**: The provenance edges encode the dependencies between nodes. E.g.,
    - *Used*, *WasGeneratedBy* (both pointing to the past in time)



Cloud Management Provenance Model

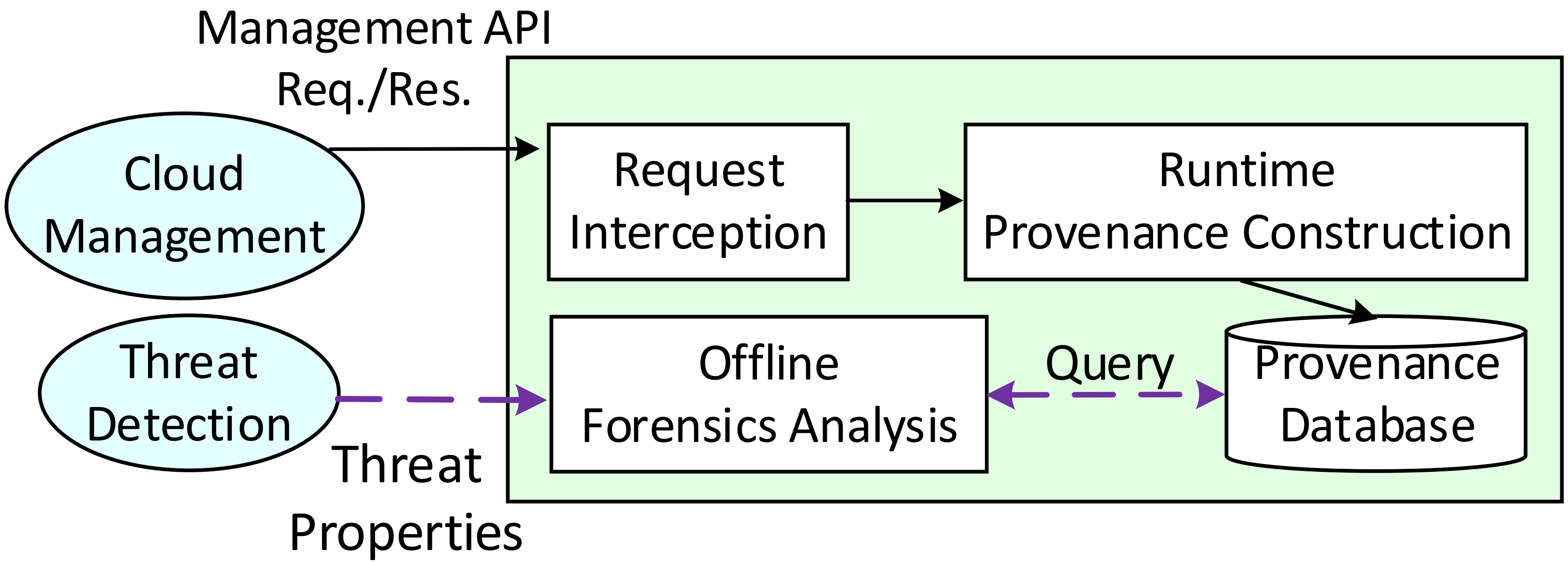
- VMA's network is attached to TenantA's router
- A TenantB's user created a port on that router
  - So, that user could enter TenantA's network
- The port was updated once a new VM was attached to it
  - So, that user could disable anti-spoofing rules to launch the attack



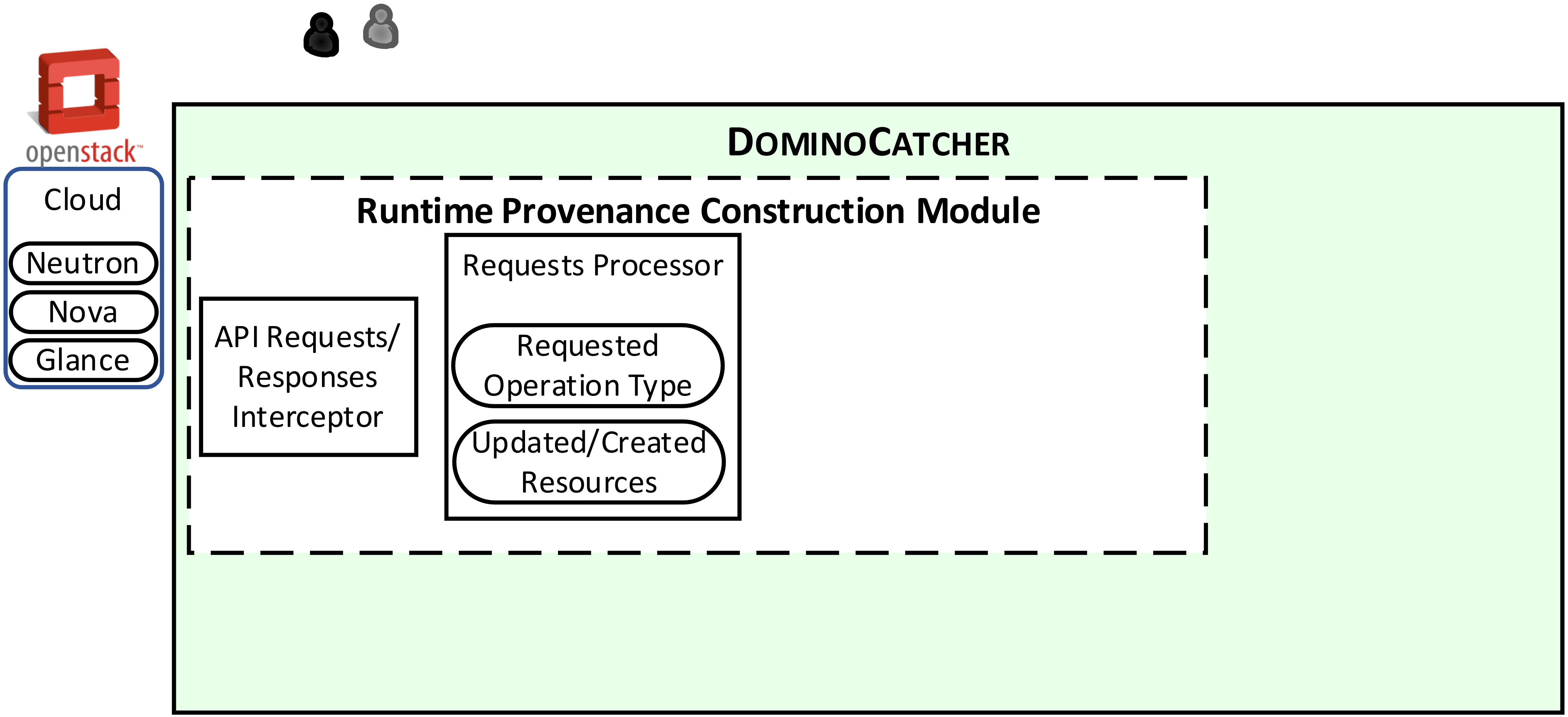
Outline

- Cloud Security Challenge
- Limitation of Exiting Solutions
- Cloud Provenance Model
- Methodology
- Implementation
- Experiment Results
- Conclusion

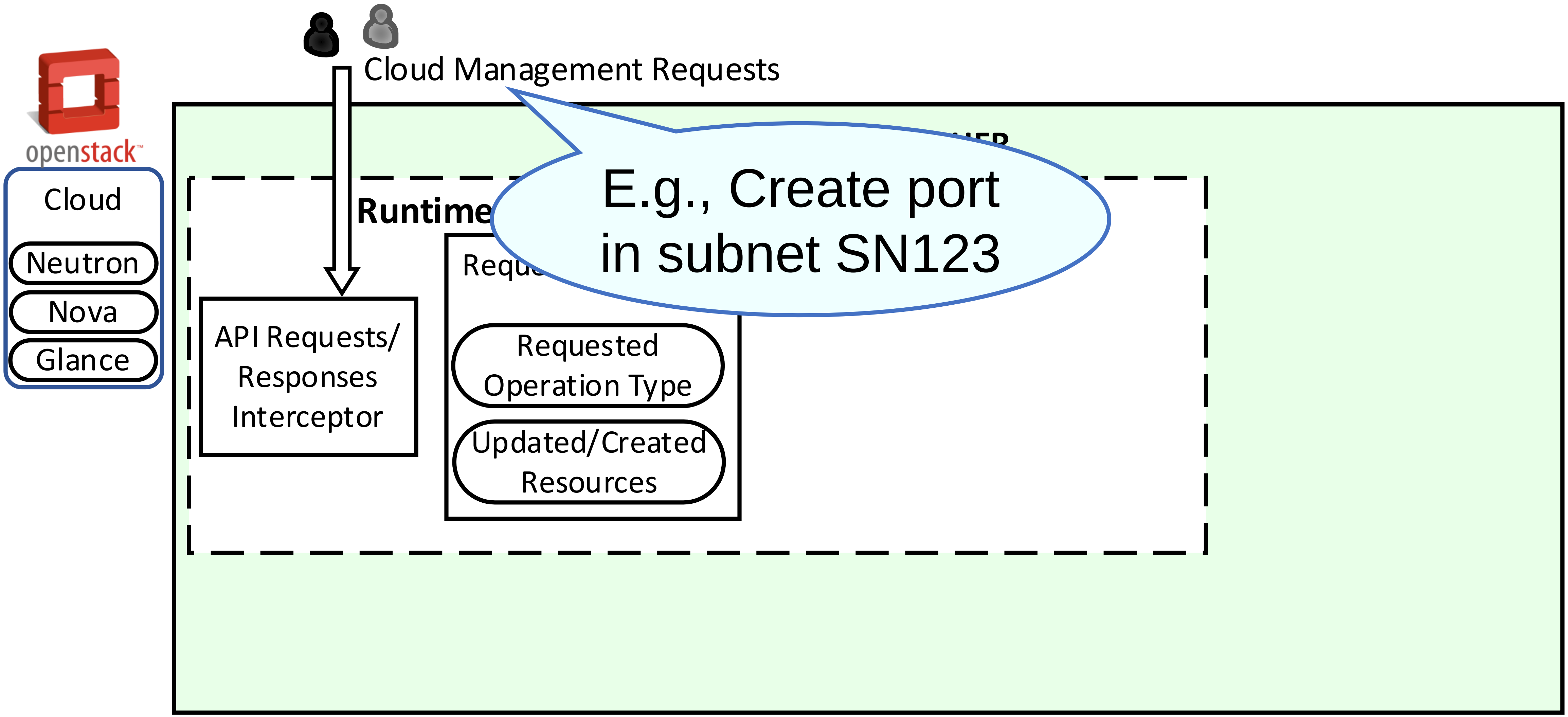
DominoCatcher Methodology



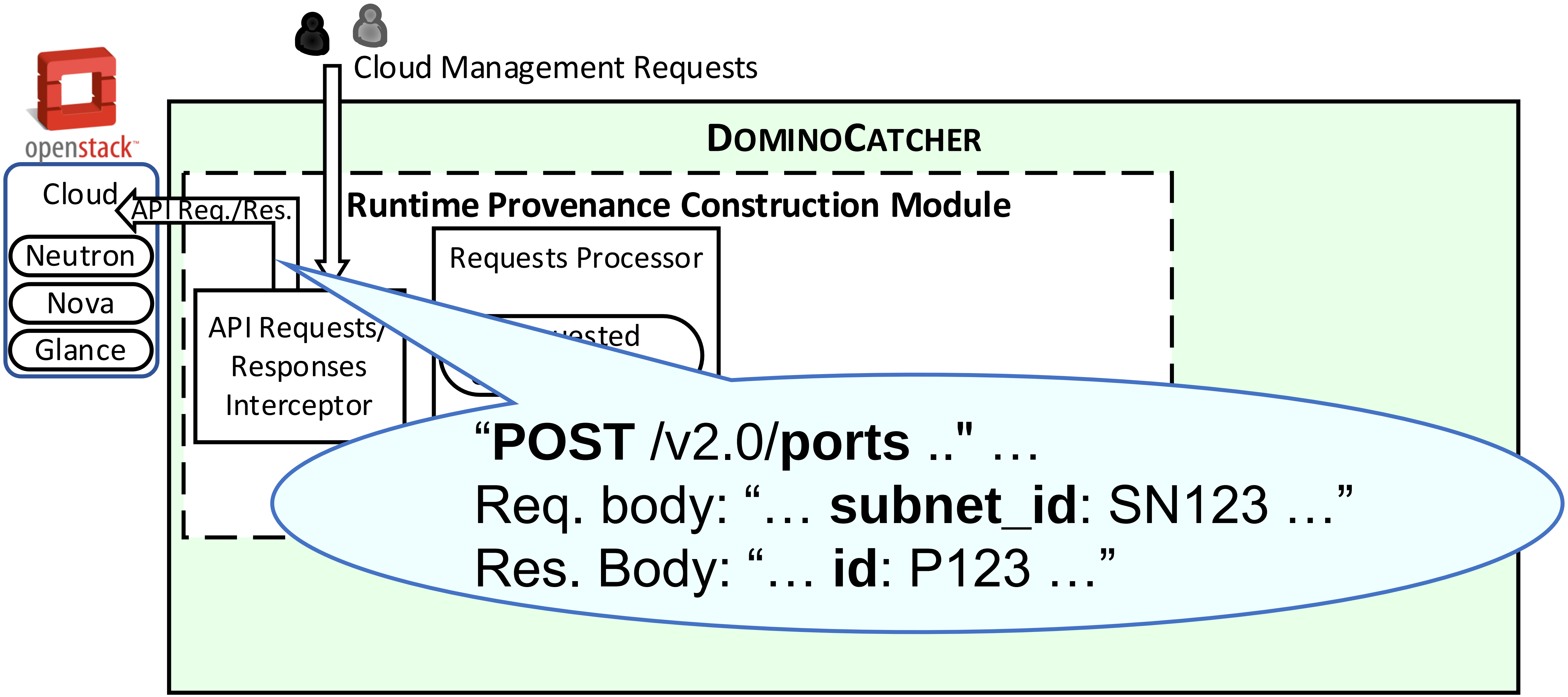
Provenance Construction



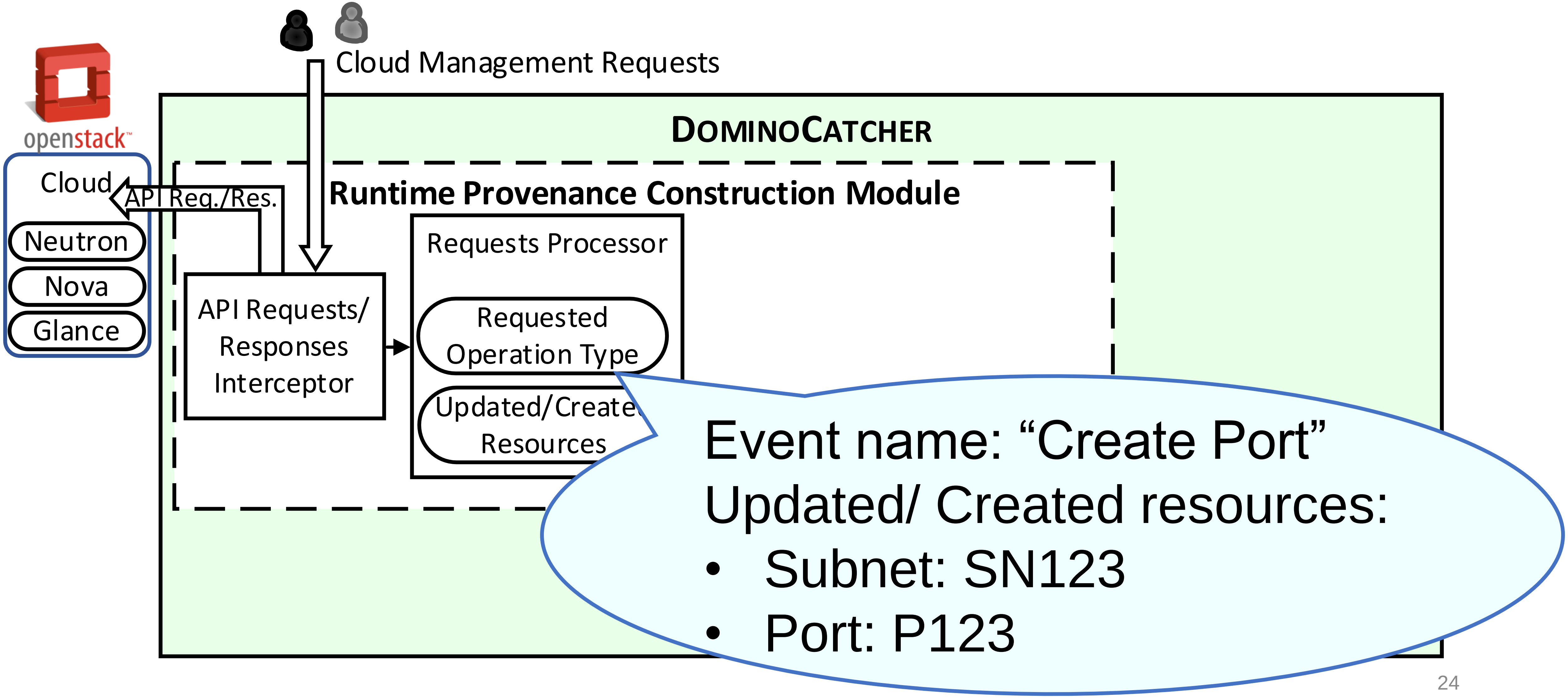
Provenance Construction – Data Collection



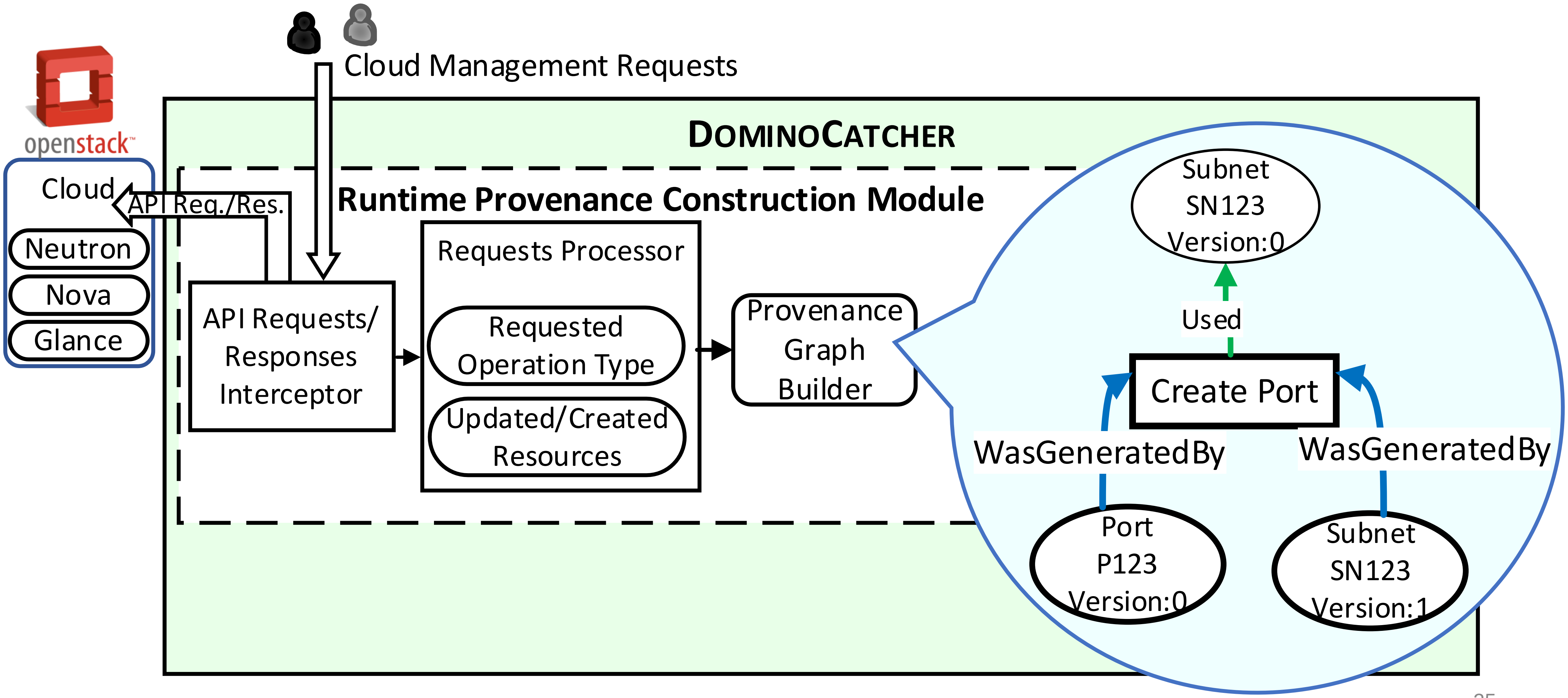
Provenance Construction – Data Collection



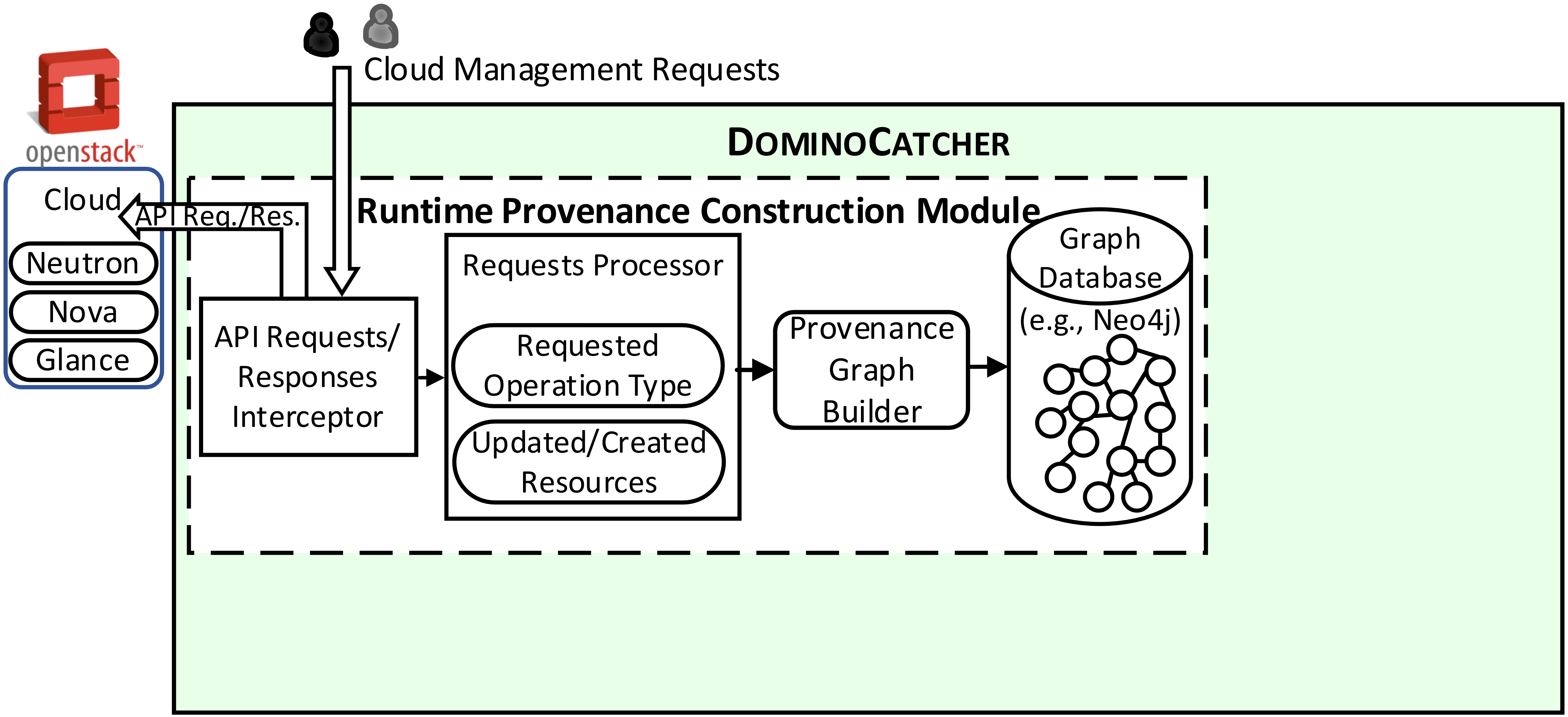
Provenance Construction – Data Collection



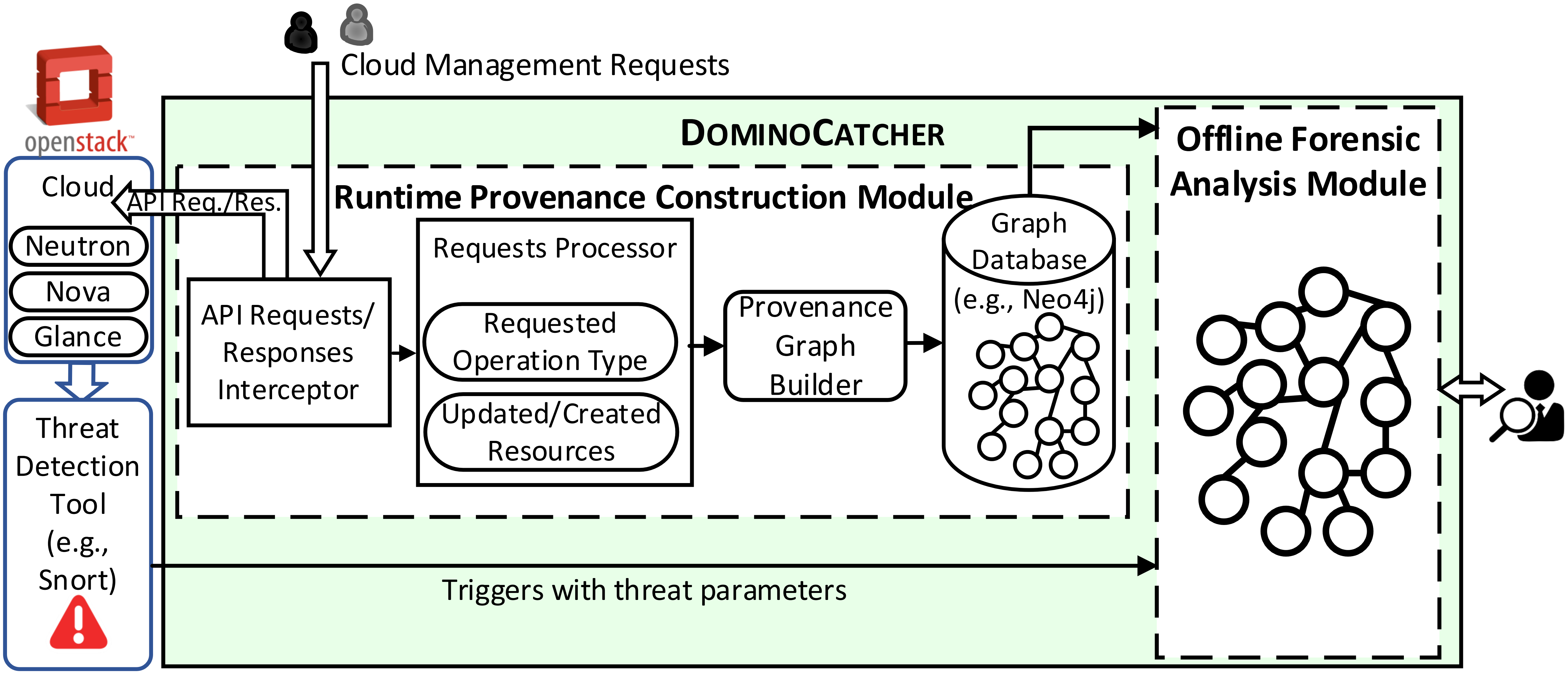
Provenance Construction – Graph Generation



Provenance Construction – Graph Generation



Forensic Analysis

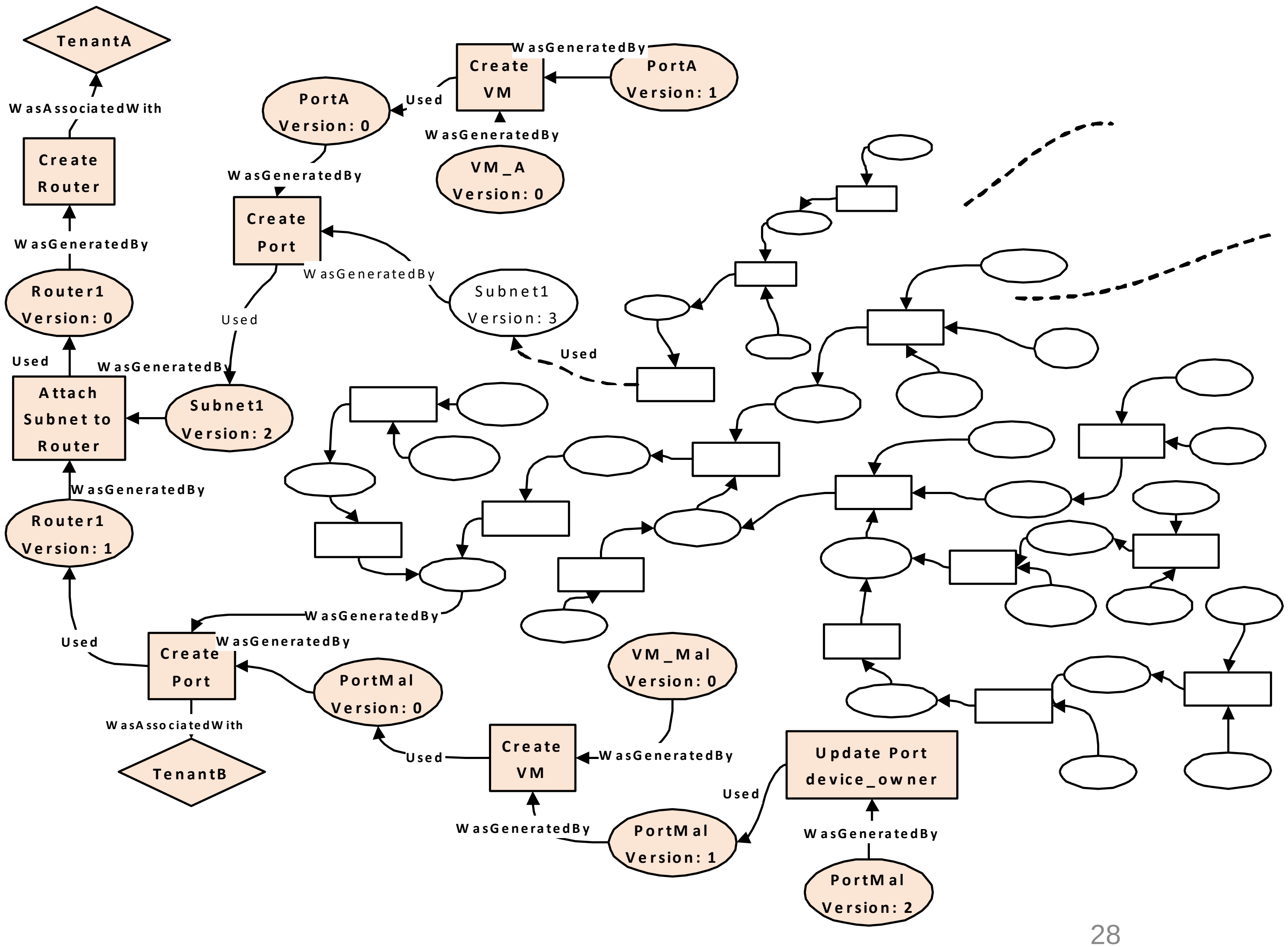


Forensic Analysis – Challenges

- Large size of the provenance graph
- Different tenants' analysis requirements

Forensic analysis only on this tenant users' behaviour

Forensic analysis on all tenants with whom there was a traffic exchange



Forensic Analysis – Pruning Schemes

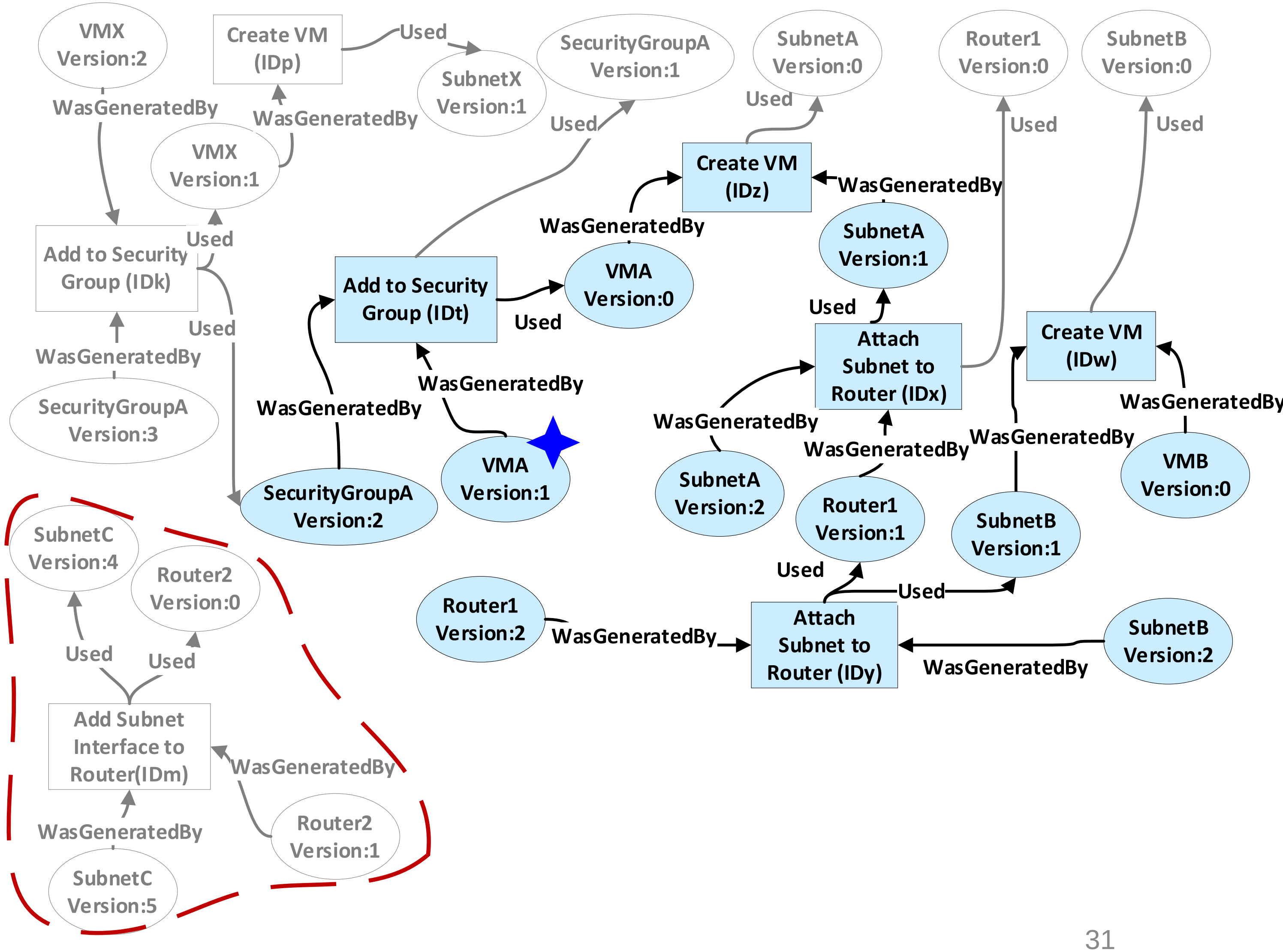
1. Disjoint subgraph
2. Context-based

Forensic Analysis – Pruning Schemes

1. Disjoint subgraph
   - Discards the subgraphs to which there is no path from the target resource node

Forensic Analysis – Pruning Schemes

1. Disjoint subgraph



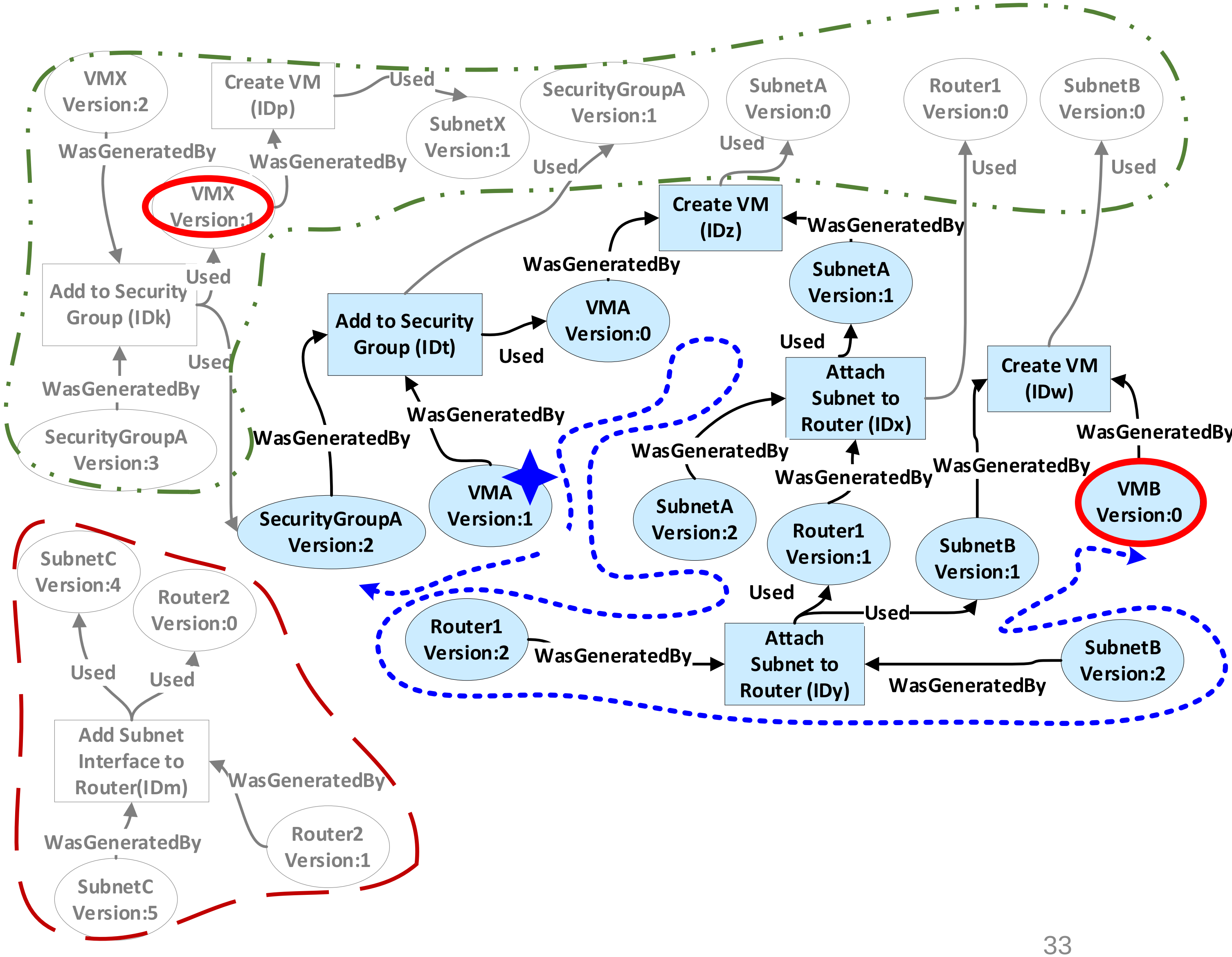
Forensic Analysis – Pruning Schemes

2. Context-base
   - Traverses paths while checking the specified constraints to identify a subgraph of resources and operations interdependent with the victim virtual resource

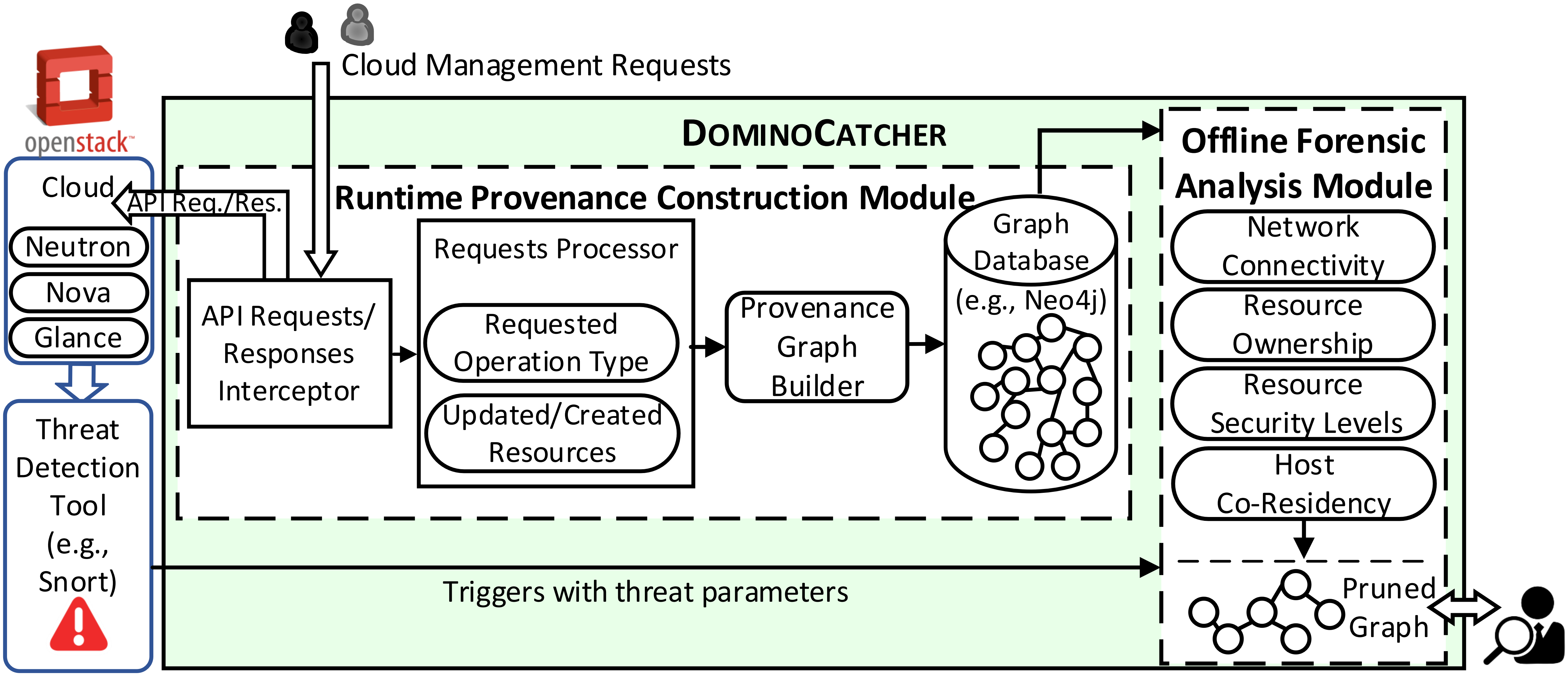
Forensic Analysis – Pruning Schemes

2. Context-base

[Create-VM, Attach-Subnet-to-Router]



Forensic Analysis



Outline

- Cloud Security Challenge
- Limitation of Exiting Solutions
- Cloud Provenance Model
- Methodology
- Implementation
- Experiment Results
- Conclusion

Implementation

- Implemented on *OpenStack (Rocky)*
- *Neo4j* as the graph database and *Cypher* language to query
- Deployed as *WSGI* middlewares on OpenStack services
  - Configuration changes are performed by these services → Capturing all configuration changes through API calls
  - As an attached interface requires less customization → Less invasiveness
- *Py2neo* library to translate python queries into Cypher

Outline

- Cloud Security Challenge
- Limitation of Exiting Solutions
- Cloud Provenance Model
- Methodology
- Implementation
- Experiment Results
- Conclusion

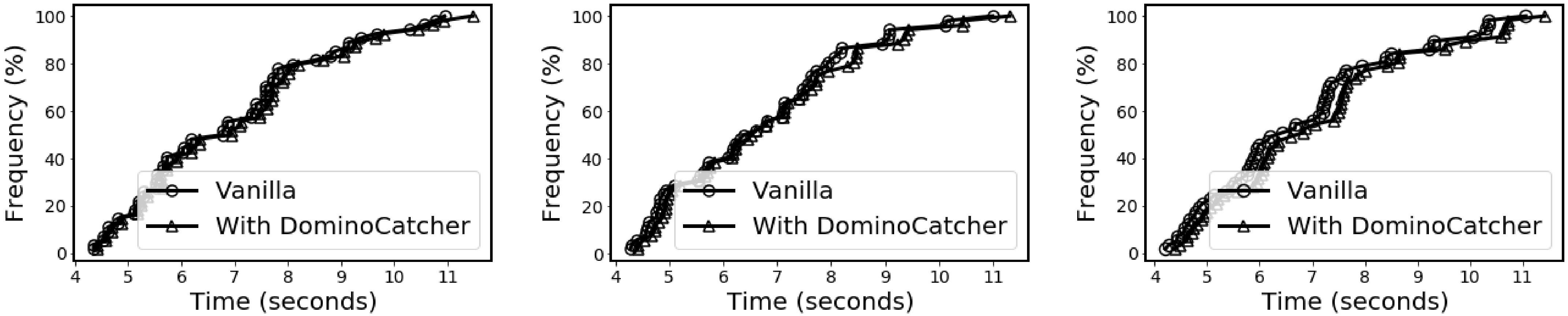
Experiment Results

- Measured the ratio between the added latency and OpenStack management operations execution time in various cloud sizes
  - Total overhead remains under %4.17

Cloud Size	# of Provenance Graph Nodes	Data Collection	Graph Generation	Total Overhead
600 VMs	43069	%.21	%1.89	%2.10
1800 VMs	64689	%.23	%3.32	%3.56
3000 VMs	107936	%.23	%3.94	%4.17

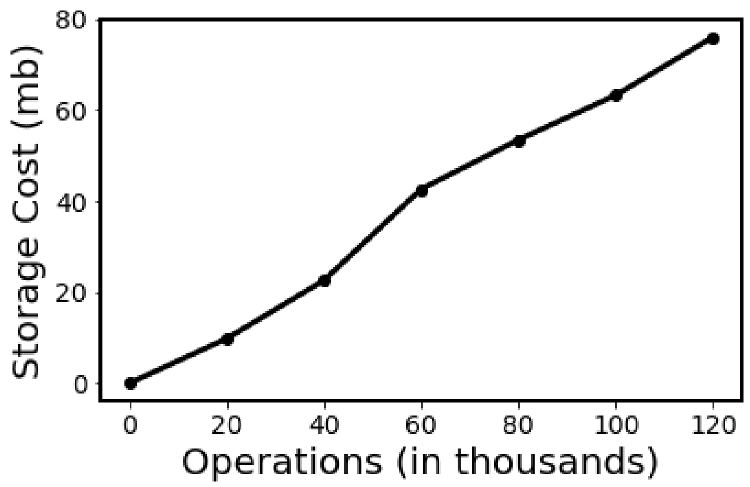
Experiment Results

- Measured the ratio between the added latency and OpenStack management operations execution time in various cloud sizes
  - Total overhead remains under %4.17



Experiment Results

- For a provenance graph constructed with 120,000 operations, only 80-megabyte storage is required
  - Higher than the number of configuration API calls issued in one day in a real enterprise cloud



Outline

- Cloud Security Challenge
- Limitation of Exiting Solutions
- Cloud Provenance Model
- Methodology
- Implementation
- Experiment Results
- Conclusion

Concluding Remarks

- Defined a provenance model on cloud management level
- Provided an interception mechanism deployed in different cloud services
  - Less invasiveness
- Proposed a provenance-based forensic analysis approach for clouds
- Implemented and evaluated in OpenStack

References

1. Vitrage (rca (root cause analysis) service). https://governance.openstack.org/tc/reference/projects/vitrage.html
2. KING, S. T., AND CHEN, P. M. Backtracking intrusions. (SOSP’03)
3. HASSAN, W. U., AGUSE, L., AGUSE, N., BATES, A., AND MOYER, T. Towards scalable cluster auditing through grammatical inference over provenance graphs. (NDSS’18)
4. Prov-dm: The prov data model. W3C Recommendation. http://www. w3. org/TR/prov-dm (2013)

Thanks & Questions

Project webpage: arc.encs.concordia.ca

Corresponding author: Azadeh Tabiban (a_tabiba@encs.concordia.ca)