

VinciDecoder: Automatically Interpreting Provenance Graphs into Textual Forensic Reports with Application to OpenStack

**Azadeh Tabiban¹, Heyang Zhao¹, Yosr Jarraya²,
Makan Pourzandi² and Lingyu Wang¹**

¹Concordia University, Canada

²Ericsson Security Research, Canada

NordSec 2022

Reykjavik University, Iceland

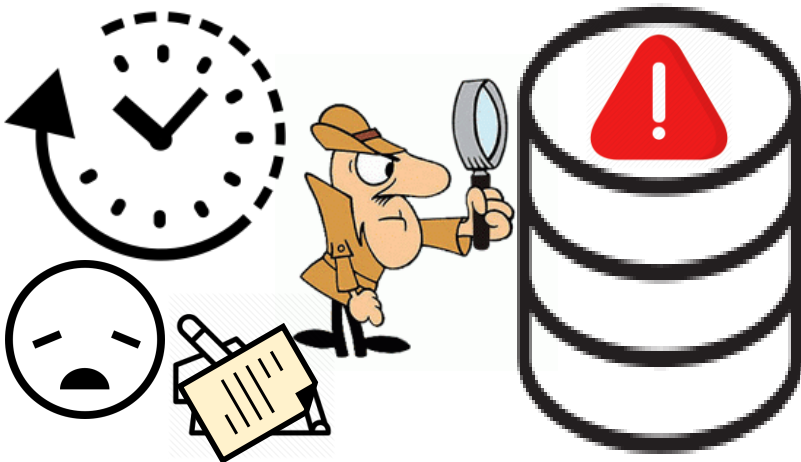


Outline

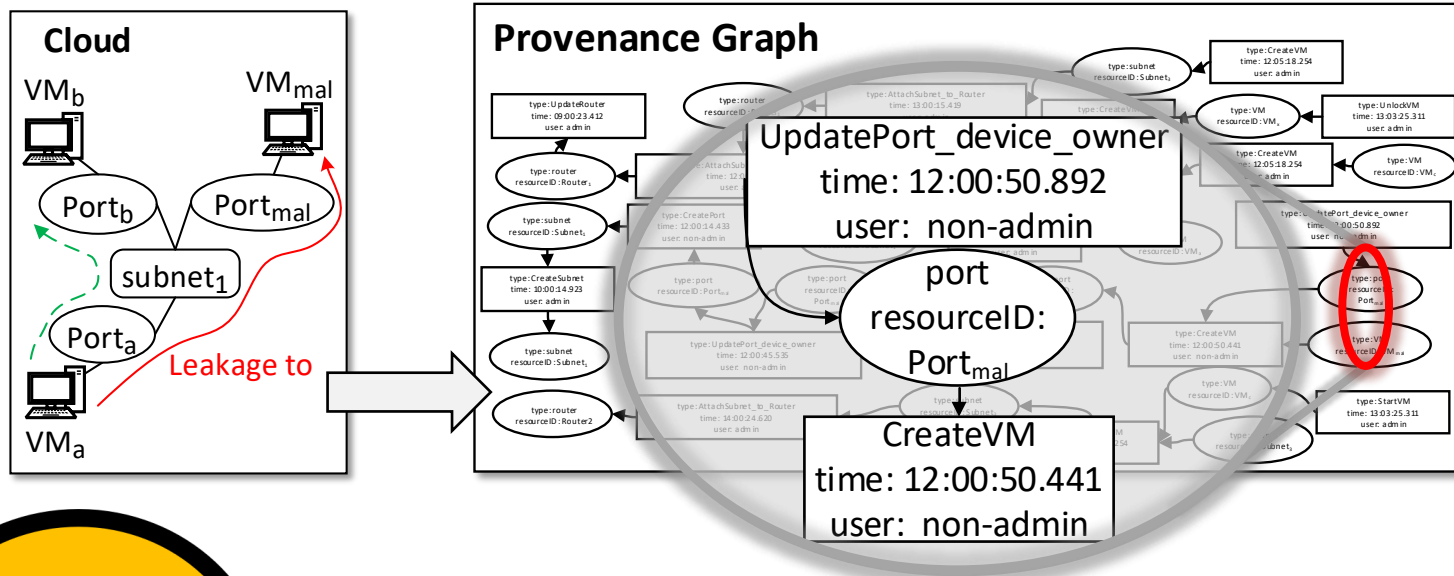
- ① Introduction
- ② Methodology
- ③ Implementation
- ④ Evaluation Results
- ⑤ Concluding Remarks

Introduction

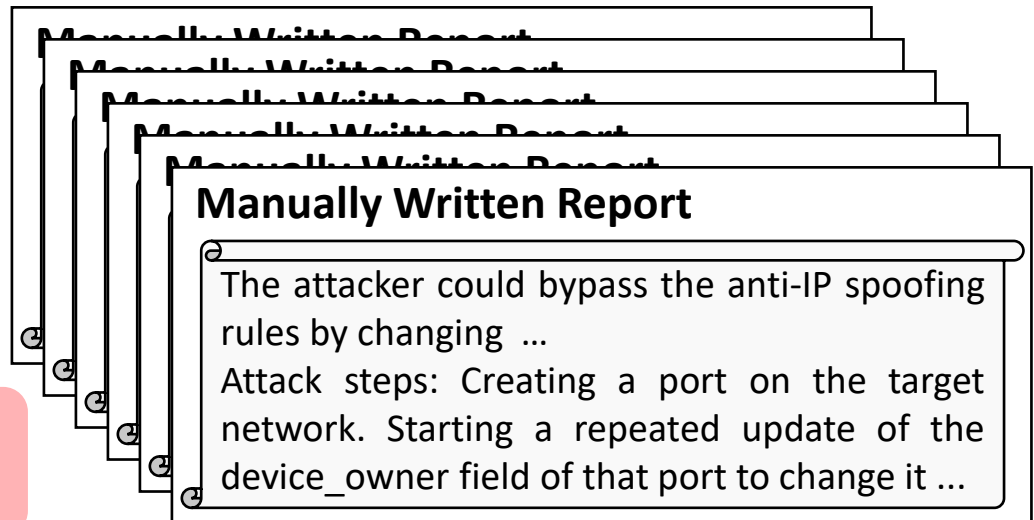
- With the recent surge in adopting clouds, there is an increasing need for explaining the root cause of security incidents
- Sharing forensic reports about the root causes can improve threat detection and attack prevention techniques
- Most existing solutions rely on human analysts to interpret the provenance graphs and document the root causes
- This is prohibitively challenging for a large cloud with tens of thousands of inter-connected virtual resources



Motivating Example

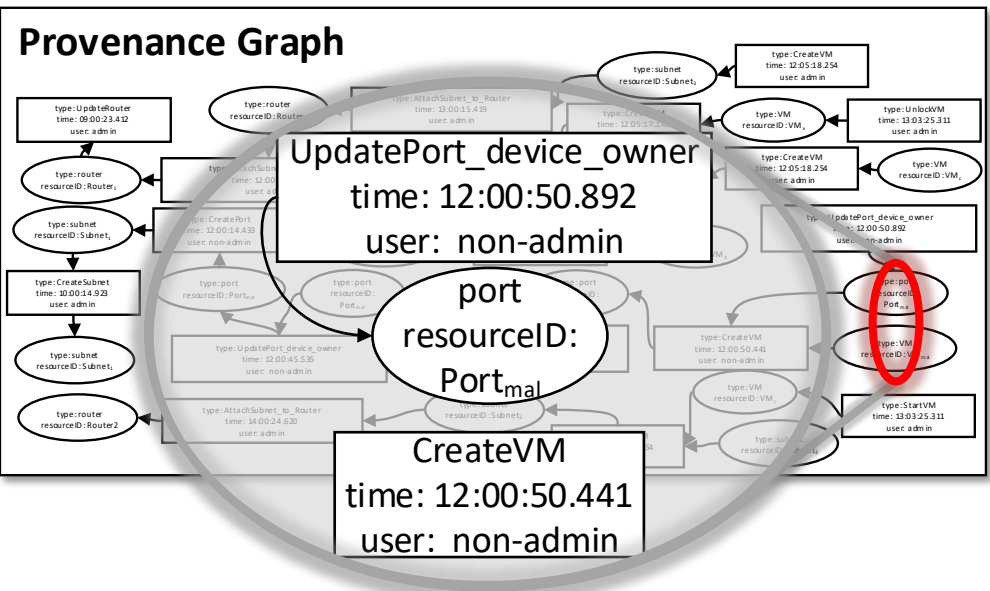


Causing delay, human error and limitation of manual efforts



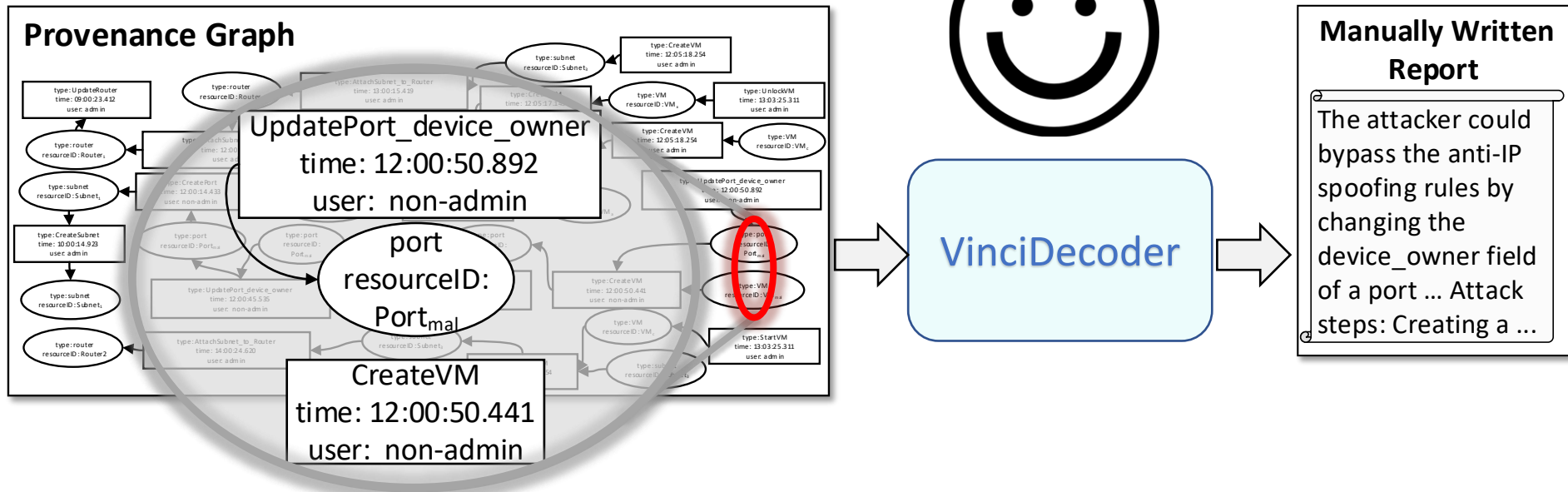
VinciDecoder: Goal

- Bridging the gap between data provenance and human-readable reports



VinciDecoder: Goal

- Bridging the gap between data provenance and human-readable reports

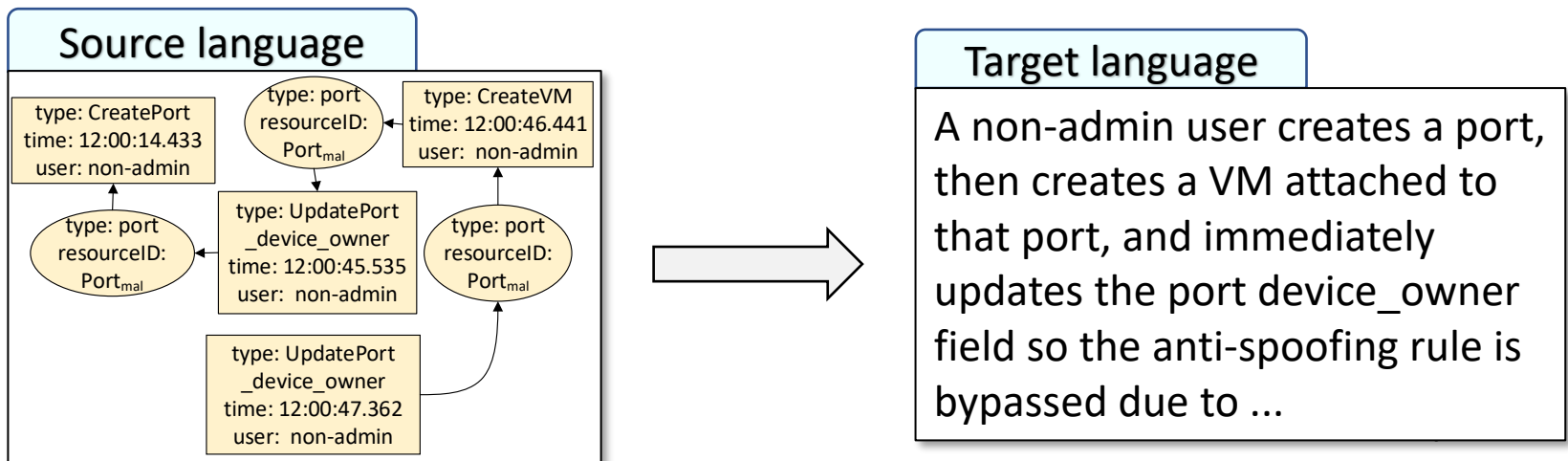


VinciDecoder: Key Ideas

- Nodes compose a path in a similar manner that words compose a sentence in a (*source*) natural language

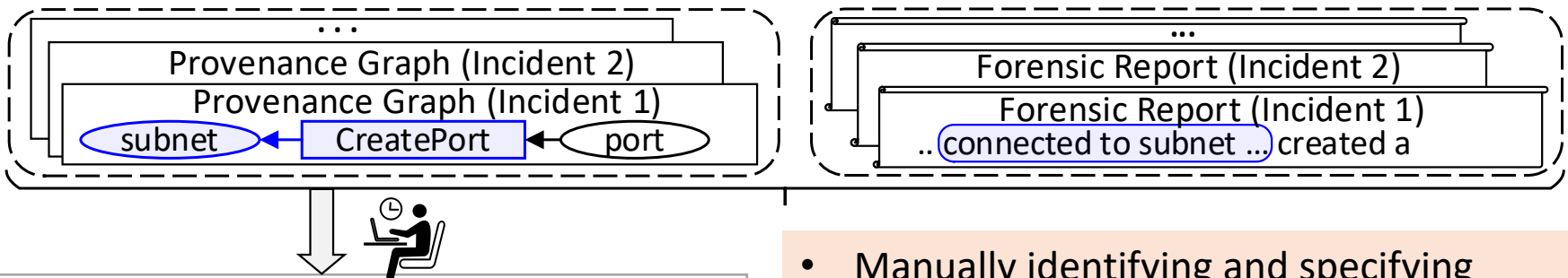


- However, applying this idea to translate provenance graphs into forensic reports (in a *target* language) has unique challenges which requires a more advanced solution

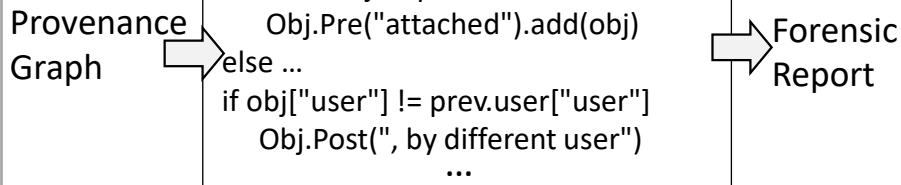


VinciDecoder: Key Ideas

- Our rule-based approach generates customized forensic reports based on lexicons and grammar rules specified by the analyst
- To enable handling more dynamic use cases, we also propose a learning-based approach



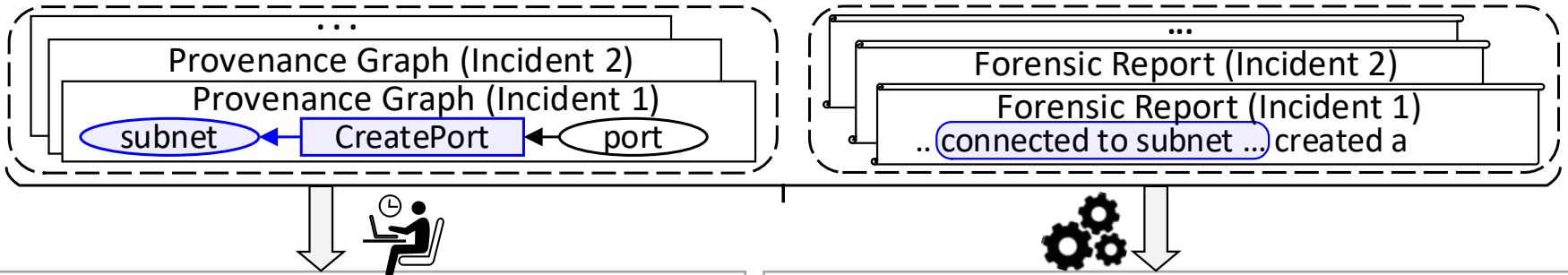
Rule-based generation of forensic reports



- Manually identifying and specifying detailed rules by investigating existing reports
- Updating rules with newly identified exploits or introduced systems

VinciDecoder: Key Ideas

- Our rule-based approach generates customized forensic reports based on lexicons and grammar rules specified by the analyst
- To enable handling more dynamic use cases, we also propose a learning-based approach



Rule-based generation of forensic reports

Provenance Graph

```

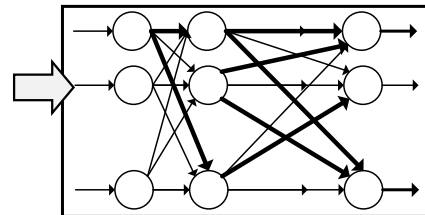
if verb != "delete" then
  if Obj in ["subnet", "router"] then
    Obj.Pre("connected to a").add(obj)
  else if Obj = "port" then
    Obj.Pre("attached").add(obj)
  else ...
  if obj["user"] != prev.user["user"]
    Obj.Post(", by different user")
  ...
    
```

Forensic Report

Learning-based generation of forensic reports



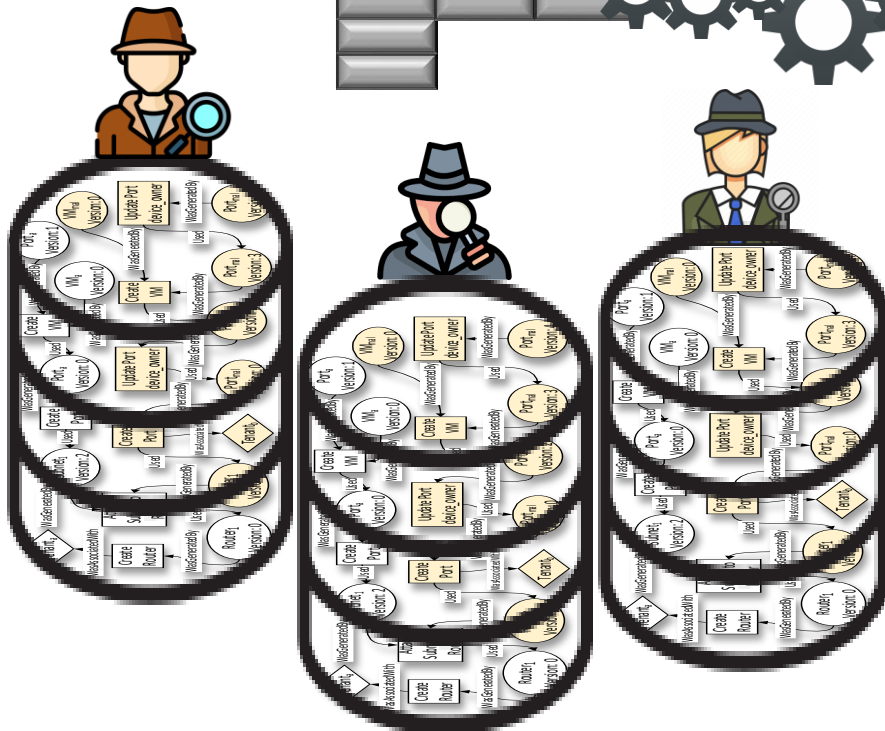
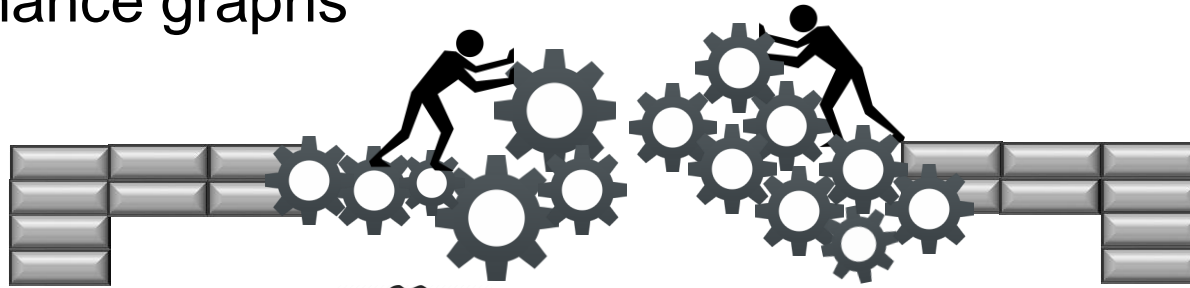
Provenance Graph



Forensic Report

Benefit of Leveraging NMT

- Neural Machine Translation (NMT) automatically learns the associations between existing reports and their corresponding provenance graphs



[OSSA 2015-018] IP, MAC, and DHCP

LODEINFO v0.5.6: multiple encryption for C2 communication with ancient crypto algorithm

This LODEINFO v0.5.6 shellcode extracted from a loader module demonstrates several enhanced evasion techniques for certain security products, as well as three new backdoor commands implemented by the developer.

After infecting the target machine, the LODEINFO backdoor beacons out machine information to the C2, such as current time, ANSI code page (ACP) identifier, MAC address and hostname. The beacon also contains a hardcoded key (NV4HDOeOVyL) used later by [the age-old Vigenere cipher](#). Furthermore, randomly generated junk data is appended to the end of the data, possibly to evade beaconing detection based on packet size.

524

stone

on 7.0.0

ty"

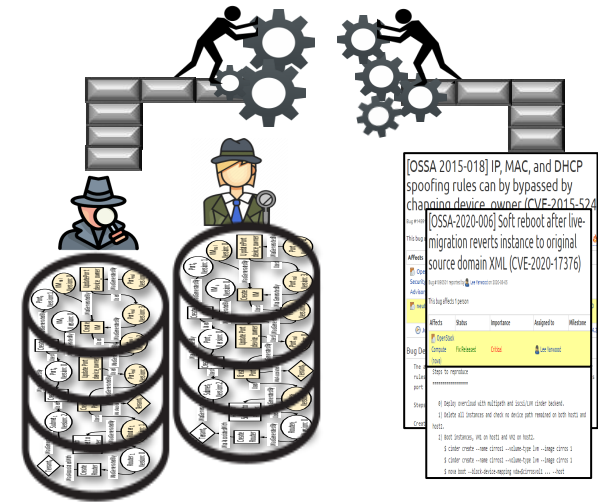
on 2014.2

roofing

node's

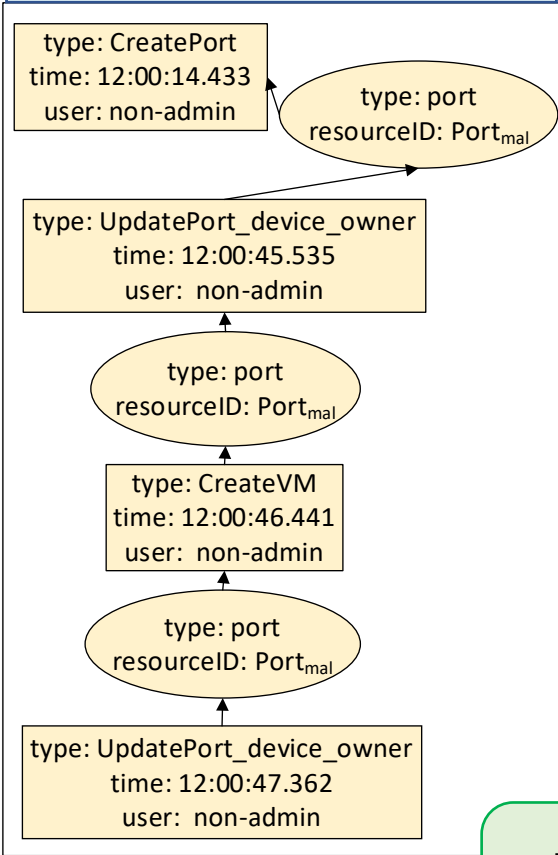
VinciDecoder: Contributions

- The first solution for generating forensic reports based on provenance analysis results using both rule-based and learning-based techniques
 - Avoiding the human-error, delay and limitation of manual effort
- A mechanism to convert provenance graph paths into primitive sentences with properties of nodes stored as words
 - Enabling the application of NMT to provenance graph paths
- Implementation and evaluation based on real OpenStack cloud testbed



VinciDecoder: Contributions

Excerpt of the provenance graph



Our automatically generated report

By the detection time, there are 4 operations performed in 1 minute corresponding to the resource VMmal.

A non-admin user created a port with the ID Portmal connected to subnet subnet1. Next, this user modified that port device_owner after less than a minute. He/She also created a VM named VMmal attached to that port after less than a second. Once done, he/she modified that port device_owner after less than a second again.

More details can be found in the provenance graph following this node path [416 - 419 - 422 - 425].

Right order



Manually written report

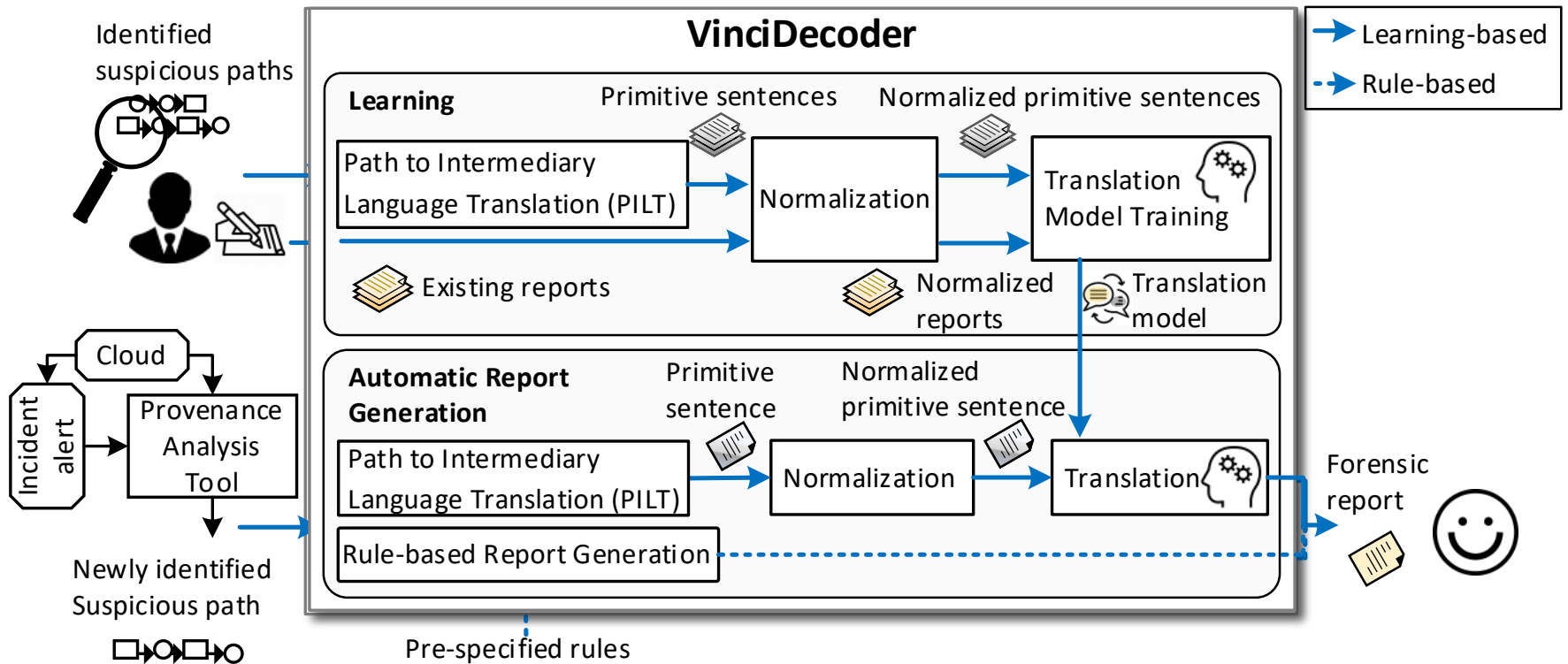
The anti-IP spoofing rules and anti-DHCP spoofing rules can be bypassed by changing the device_owner field of a port attached to a VM to a value starting with 'network:...'

Create a port on the target network. Start a repeated update of the device_owner field of the port so you can set it to 'network:...' right after OpenStack compute service sets this field to 'compute:<whatever>' upon attaching the VM to it. This has to be done quickly after creating the VM and before the OpenStack networking service L2 agent wires up the security group rules to the port. Create a new VM attached to the updated port.

Wrong order

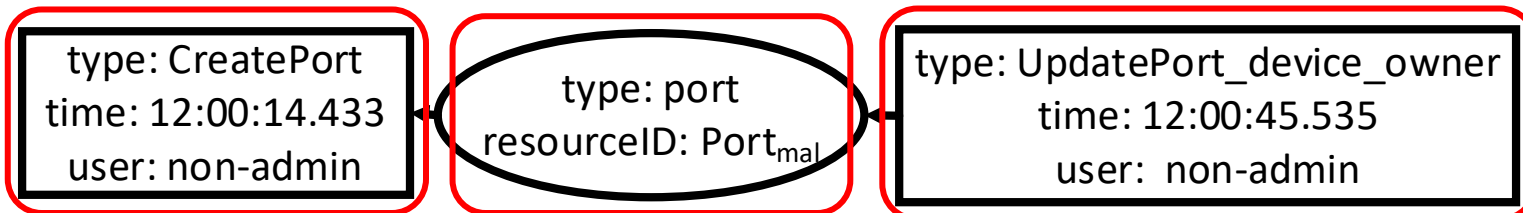
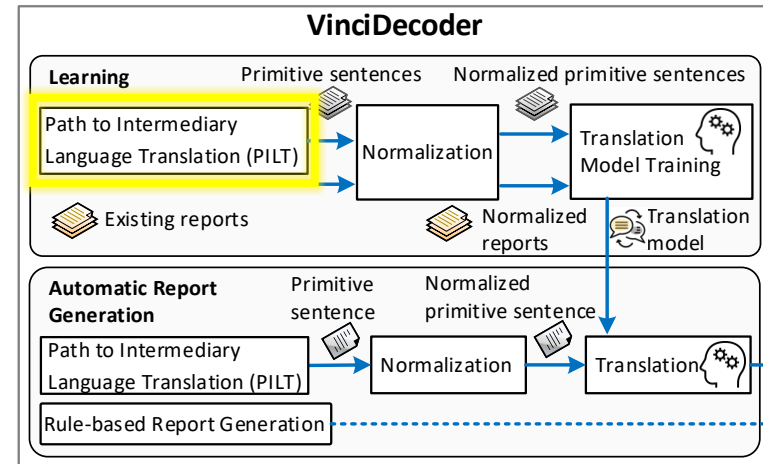


VinciDecoder: Overview



1. Path to Intermediary Language Translation

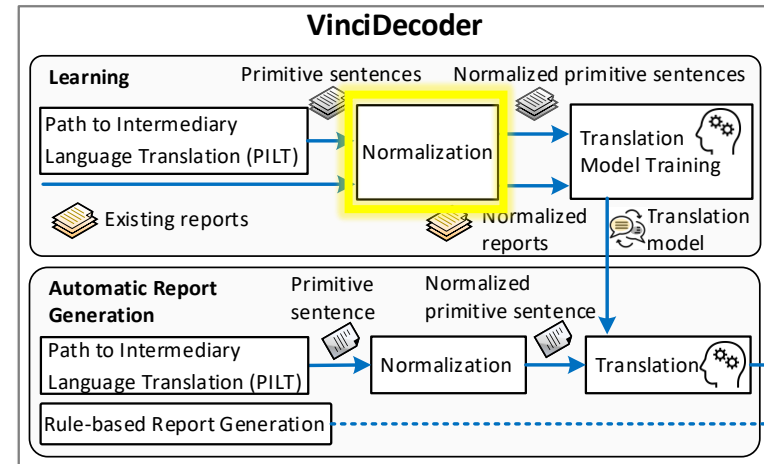
- **Objective:** enabling the adoption of NMT by providing suspicious paths as sentences



➔ "type:CreatePort,user:non-admin"
"type:Port,resourceID:Portmal" "type:
UpdatePort_device owner,user:non-
admin,ElapsedTime:31-seconds" ...

2. Normalization

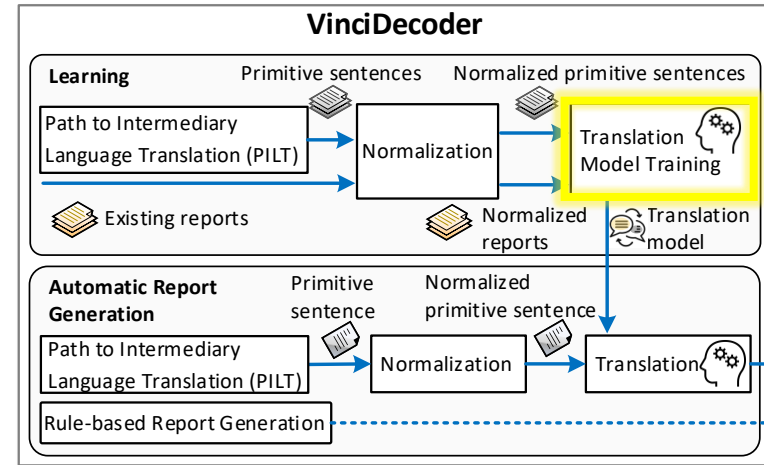
- **Objective:** allowing NMT to focus on generic words instead of application-specific ones
- Normalization module removes instance-specific information from the dataset
 - E.g., the number preceding the string “-milliseconds”) with a placeholder (i.e., \0) based on our specified rules



```
“type:CreatePort,ElapsedTime:\0-seconds,user:non-admin,ID:
Portmal” “type:port,user:non-admin” “type:CreateVM, Elapsed
Time:\0-seconds,user:non-admin” “type:port, user:non-admin”
“type:UpdatePortDeviceOwner, ElapsedTime: \0-milliseconds”
```

3. Translation Model Training

- Objective:** building a translation model to profile the correspondence between the existing forensic reports and their associated suspicious paths using NMT

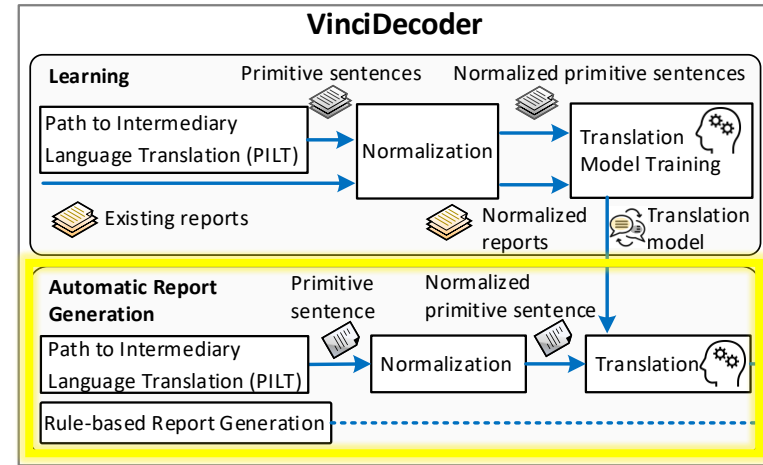


“type:CreatePort,ElapsedTime:\0-seconds,user:non-admin,ID:Portmal” “type:port,user:non-admin” “type:CreateVM, Elapsed Time: \0-seconds,user:non-admin” “type:port, user:non-admin” “type:UpdatePortDeviceOwner, ElapsedTime: \0-milliseconds”
 ...

A non-admin user creates a port, then creates a VM attached to that port, and immediately updates the port device_owner field so the anti-spoofing rule is bypassed due to the vulnerability exploit.
 ...

4. Automatic Report Generation

- **Objective:** generating forensic reports based on the identified suspicious paths corresponding to a new incident by applying the translation model



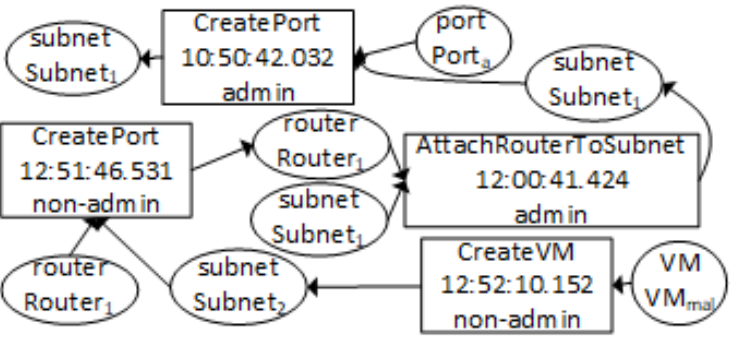
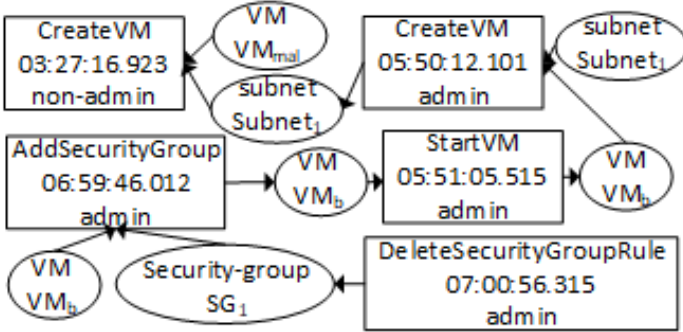
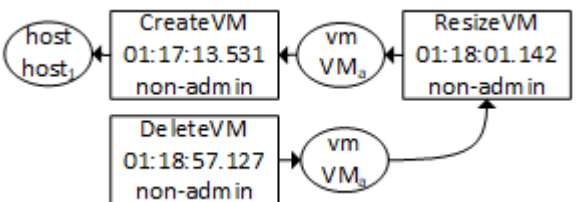
By the detection time, there are 4 operations performed in 1 minute corresponding to the resource VMmal.

A non-admin user created a port with the ID Portmal connected to subnet subnet1. Next, this user modified that port device_owner after less than a minute. He/She also created a VM named VMmal attached to that port after less than a second. Once done, he/she modified that port device_owner after less than a second again.

Implementation

- Implemented in a cloud testbed based on OpenStack
 - Only our PILT module and some of our rules are platform-specific
 - Our modular design makes VinciDecoder easily portable to other platforms or provenance models (e.g., OS-level provenance)
- Neo4j as the graph database
- Open-Source Toolkit for Neural Machine Translation (ONMT) for translation
- Embedded each path into a 500 dimensional vector
- Applied the default batch size and the dropout rate of 64 and 0.3, respectively

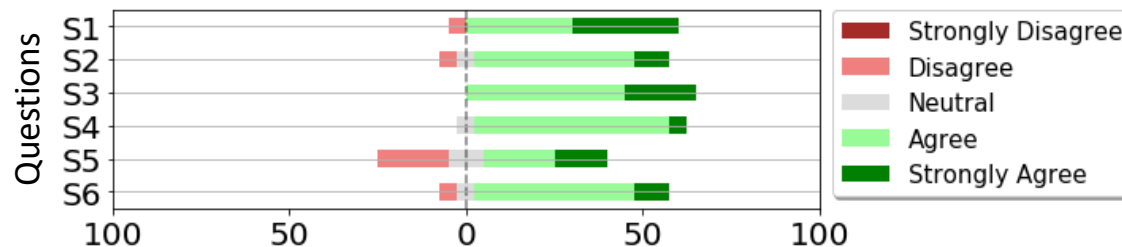
Case Studies

Provenance graph path	Automatically generated report
 <p>The diagram shows a provenance graph path for Case 1. It starts with a 'router Router₁' node. From this router, two paths emerge: one to 'CreatePort 12:51:46.531 non-admin' and another to 'subnet Subnet₂'. The 'CreatePort 12:51:46.531 non-admin' node leads to 'router Router₁' and 'subnet Subnet₁'. The 'subnet Subnet₁' node leads to 'CreatePort 10:50:42.032 admin' and 'AttachRouterToSubnet 12:00:41.424 admin'. The 'CreatePort 10:50:42.032 admin' node leads to 'port Port_a' and 'subnet Subnet₁'. The 'AttachRouterToSubnet 12:00:41.424 admin' node leads to 'subnet Subnet₁' and 'CreateVM 12:52:10.152 non-admin'. The 'port Port_a' node leads to 'subnet Subnet₁'. The 'CreateVM 12:52:10.152 non-admin' node leads to 'VM VM_{mal}'.</p>	<p>An admin user created a port named porta on a subnet. This admin user attached that subnet to a router after around 1 hours. A nonadmin user created a port on that subnet and on that router, previously affected by a different user. After that, (s)he created a vm named vmmal, which is associated to the alert.</p>
 <p>The diagram shows a provenance graph path for Case 2. It starts with 'VM VM_a' and 'Security-group SG₁'. From 'VM VM_a', two paths emerge: one to 'CreateVM 03:27:16.923 non-admin' and another to 'AddSecurityGroup 06:59:46.012 admin'. From 'Security-group SG₁', one path leads to 'AddSecurityGroup 06:59:46.012 admin' and another to 'DeleteSecurityGroupRule 07:00:56.315 admin'. The 'CreateVM 03:27:16.923 non-admin' node leads to 'VM VM_{mal}' and 'subnet Subnet₁'. The 'AddSecurityGroup 06:59:46.012 admin' node leads to 'VM VM_b'. The 'DeleteSecurityGroupRule 07:00:56.315 admin' node leads to 'VM VM_b'. The 'VM VM_{mal}' node leads to 'CreateVM 05:50:12.101 admin'. The 'subnet Subnet₁' node leads to 'CreateVM 05:50:12.101 admin'. The 'CreateVM 05:50:12.101 admin' node leads to 'subnet Subnet₁' and 'StartVM 05:51:05.515 admin'. The 'StartVM 05:51:05.515 admin' node leads to 'VM VM_b'.</p>	<p>A nonadmin user created vm named vmmal on a subnet. An admin user created a vm named vmb on that subnet. Once done, (s)he started that vm vmb. Then attached a securitygroup named SG1 on that vm. The administrator deleted securitygroup rule from that SecurityGroup after around 1 minutes.</p>
 <p>The diagram shows a provenance graph path for Case 3. It starts with 'host host₁' and 'vm VM_a'. From 'host host₁', one path leads to 'CreateVM 01:17:13.531 non-admin' and another to 'DeleteVM 01:18:57.127 non-admin'. From 'vm VM_a', one path leads to 'CreateVM 01:17:13.531 non-admin' and another to 'ResizeVM 01:18:01.142 non-admin'. The 'CreateVM 01:17:13.531 non-admin' node leads to 'vm VM_a'. The 'DeleteVM 01:18:57.127 non-admin' node leads to 'vm VM_b'. The 'vm VM_b' node leads to 'ResizeVM 01:18:01.142 non-admin'.</p>	<p>A nonadmin user created a vm named vma on a host. Later, (s)he resized that vm after less than a minutes. Next, (s)he deleted that vm after less than a minute.</p>

Evaluation – User Study

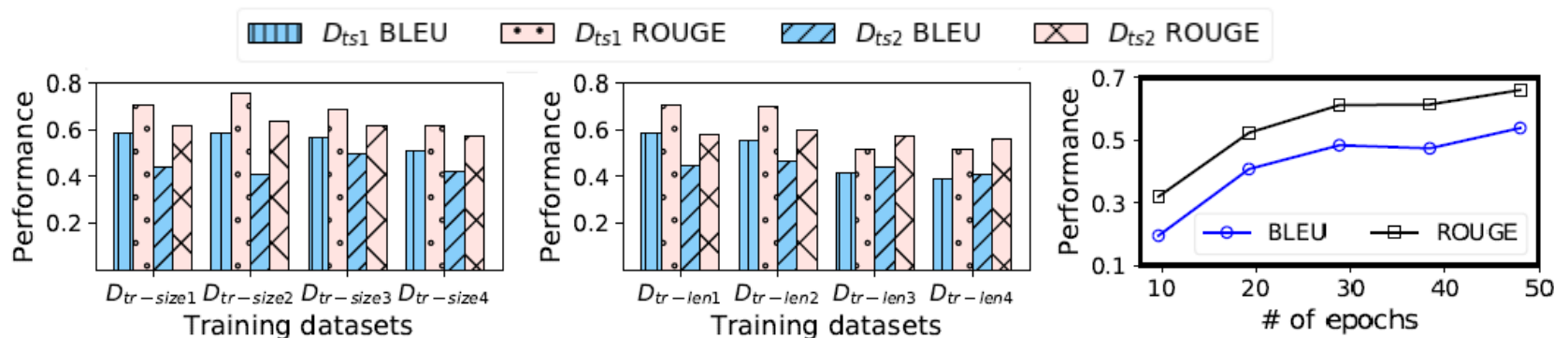
- Participants
 - From a telecommunication industrial organization
 - Graduate students working in cybersecurity
- Our user study shows that the generated reports provide sufficient readability for human analysts
 - E.g., 92% of participants agree that understanding attack steps is much easier using our approach

Questions
Understanding the attack steps using the generated text is easier than using the path.
The generated text is consistent with the explained attack scenario.
The generated text is consistent with the path regarding the relationships of operations.
The generated text captures all the information of the suspicious path.
The generated text is sufficiently fluent compared with the manually written report.
The generated text is consistent with the manually written report regarding attack steps.



Evaluation – Numerical Results

- Our numerical results demonstrate that VinciDecoder generates high-quality reports
 - E.g., up to 0.56 BLEU score for precision
- The performance improves with the number of epochs and reach almost constant values after training over 50 epochs



Concluding Remarks

Summary

- Proposed a rule-based and a learning-based solution for automatically translating provenance analysis results into human-readable forensic reports
- Represented provenance graphs in an intermediary language, which can then be translated into a human readable language using NMT
- Implemented and evaluated based on real OpenStack cloud setup

Future Directions

- Integrating with other (e.g., OS-level) provenance analysis tools
- Exploring other translation techniques and hyperparameters, which may further improve the effectiveness of our approach

Thanks & Questions

Azadeh Tabiban: a_tabiba@encs.concordia.ca

NSERC/Ericsson Industrial Research Chair in SDN/NFV Security: <https://arc.encs.concordia.ca/>