TenantGuard: Scalable Runtime Verification of Cloud-Wide VM-Level Network Isolation

Han Song

SJTU

May 24, 2017
Outline

1. Background
2. Architecture and Data Structures
3. Verification
4. Experiments
5. Conclusion
6. Q & A
Isolation Breaches
One of the Biggest Security Concerns in Cloud
Isolation Breaches
One of the Biggest Security Concerns in Cloud

Something went wrong and D is hacked!
Isolation Breaches
One of the Biggest Security Concerns in Cloud
OpenStack real world vulnerabilities

[OSSA 2014-008]
Any tenant is able to create a port on another tenant’s router!
Reported: 22.10.2013
Fixed: 27.03.2014

[OSSA 2015-021]
Security group rules are not effective on instances immediately!
Reported: 02.09.2015
Fixed: 11.09.2015
Isolation Breaches
One of the Biggest Security Concerns in Cloud
One possible solution is: network isolation verification

[OSSA 2014-008]
Any tenant is able to create a port on another tenant's router!
Reported: 22.10.2013
Fixed: 27.03.2014

[OSSA 2015-021]
Security group rules are not effective on instances immediately!
Reported: 02.09.2015
Fixed: 11.09.2015
Network Isolation Verification
Challenges

- Size of virtual networks: 150M+ VM pairs
- Diverse and distributed network functions
- Large data from heterogeneous sources
- Quickly invalidating verification results
Network Isolation Verification

Challenges

- Size of virtual networks: 150M+ VM pairs
- Diverse and distributed network functions
- Large data from heterogeneous sources
- Quickly invalidating verification results
Network Isolation Verification

Challenges

- Size of virtual networks: 150M+ VM pairs
- Diverse and distributed network functions
- Large data from heterogeneous sources
- Quickly invalidating verification results
Network Isolation Verification

Challenges

- Size of virtual networks: 150M+ VM pairs
- Diverse and distributed network functions
- Large data from heterogeneous sources
- Quickly invalidating verification results
Existing Approaches

- Designed for physical networks
  - Not suitable for VM-level pair-wise reachability
- Focus on small to medium virtual infrastructure
  - Not designed for millions of VM pairs
- Can support VM-level reachability
  - Taking minutes to hours for over 100 million pairs
Existing Approaches

- Designed for physical networks
  - Not suitable for VM-level pair-wise reachability
- Focus on small to medium virtual infrastructure
  - Not designed for millions of VM pairs
- Can support VM-level reachability
  - Taking minutes to hours for over 100 million pairs
Existing Approaches

- Designed for physical networks
  - Not suitable for VM-level pair-wise reachability
- Focus on small to medium virtual infrastructure
  - Not designed for millions of VM pairs
- Can support VM-level reachability
  - Taking minutes to hours for over 100 million pairs
Existing Approaches

- Designed for physical networks
  - Not suitable for VM-level pair-wise reachability
- Focus on small to medium virtual infrastructure
  - Not designed for millions of VM pairs
- Can support VM-level reachability
  - Taking minutes to hours for over 100 million pairs
Existing Approaches

- Designed for physical networks
  - Not suitable for VM-level pair-wise reachability
- Focus on small to medium virtual infrastructure
  - Not designed for millions of VM pairs
- Can support VM-level reachability
  - Taking minutes to hours for over 100 million pairs
Existing Approaches

- Designed for physical networks
  - Not suitable for VM-level pair-wise reachability
- Focus on small to medium virtual infrastructure
  - Not designed for millions of VM pairs
- Can support VM-level reachability
  - Taking minutes to hours for over 100 million pairs
Assumptions

- **Focused on:**
  - Verifying security properties specified by cloud tenants
  - Not detecting any specific attack
- **Relies on:**
  - The correctness of input data
  - Existing solutions at other layers
  - No sensitive information in the verification results
Assumptions

- **Focused on:**
  - Verifying security properties specified by cloud tenants
  - Not detecting any specific attack

- **Relies on:**
  - The correctness of input data
  - Existing solutions at other layers
  - No sensitive information in the verification results
### Assumptions

- **Focused on:**
  - Verifying security properties specified by cloud tenants
  - Not detecting any specific attack

- **Relies on:**
  - The correctness of input data
  - Existing solutions at other layers
  - No sensitive information in the verification results
Assumptions

- **Focused on:**
  - Verifying security properties specified by cloud tenants
  - Not detecting any specific attack

- **Relies on:**
  - The correctness of input data
  - Existing solutions at other layers
  - No sensitive information in the verification results
Assumptions

- **Focused on:**
  - Verifying security properties specified by cloud tenants
  - Not detecting any specific attack

- **Relies on:**
  - The correctness of input data
  - Existing solutions at other layers
  - No sensitive information in the verification results
Assumptions

- Focused on:
  - Verifying security properties specified by cloud tenants
  - Not detecting any specific attack

- Relies on:
  - The correctness of input data
  - Existing solutions at other layers
  - No sensitive information in the verification results
Assumptions

- Focused on:
  - Verifying security properties specified by cloud tenants
  - Not detecting any specific attack

- Relies on:
  - The correctness of input data
  - Existing solutions at other layers
  - No sensitive information in the verification results
Highlights

TenantGuard, a VM-level network isolation verification system

- Pairwise reachability for over 25K VMs in 13s
- Built on OpenStack, a popular cloud management platform
- Based on a hierarchical model for virtual networks
- Leveraging efficient data structures, incremental verification and parallel computation
Highlights

TenantGuard, a VM-level network isolation verification system

- Pairwise reachability for over 25K VMs in 13s
- Built on OpenStack, a popular cloud management platform
  - Based on a hierarchical model for virtual networks
  - Leveraging efficient data structures, incremental verification and parallel computation
Highlights

TenantGuard, a VM-level network isolation verification system

- Pairwise reachability for over 25K VMs in 13s
- Built on OpenStack, a popular cloud management platform
- Based on a hierarchical model for virtual networks
- Leveraging efficient data structures, incremental verification and parallel computation
Highlights

TenantGuard, a VM-level network isolation verification system
- Pairwise reachability for over 25K VMs in 13s
- Built on OpenStack, a popular cloud management platform
- Based on a hierarchical model for virtual networks
- Leveraging efficient data structures, incremental verification and parallel computation
TenantGuard: Architecture
Key Ideas

- Hierarchical virtual network model (Router, subnet, VM)
- Top-down verification approach (from prefix-level to IP-level)
- Efficient data structures (Radix Trie and X-fast Binary Trie)
Key Ideas

- Hierarchical virtual network model (Router, subnet, VM)
- Top-down verification approach (from prefix-level to IP-level)
- Efficient data structures (Radix Trie and X-fast Binary Trie)
Key Ideas

- Hierarchical virtual network model (Router, subnet, VM)
- Top-down verification approach (from prefix-level to IP-level)
- Efficient data structures (Radix Trie and X-fast Binary Trie)
Hierarchical Virtual Network Model
Hierarchical Virtual Network Model
Baseline Approach

Verifying every possible VM pair (e.g., over 150 million pairs!!)
Top-Down Verification

- Step 1: Check isolation between subnets within the same tenant environment
- Step 2: Check isolation between different tenant environments
- Step 3: Check VM-isolation only for subnets found to be reachable
Top-Down Verification

- Step 1: Check isolation between subnets within the same tenant environment
- Step 2: Check isolation between different tenant environments
- Step 3: Check VM-isolation only for subnets found to be reachable
Top-Down Verification

- Step 1: Check isolation between subnets within the same tenant environment
- Step 2: Check isolation between different tenant environments
- Step 3: Check VM-isolation only for subnets found to be reachable
**Top-Down Verification**

- Step 1: Check isolation between subnets within the same tenant environment
- Step 2: Check isolation between different tenant environments
- Step 3: Check VM-isolation only for subnets found to be reachable
Top-Down Verification

- Step 1: Check isolation between subnets within the same tenant environment
- Step 2: Check isolation between different tenant environments
- Step 3: Check VM-isolation only for subnets found to be reachable
Efficient Data Structure
Radix Trie: Capturing Routing Rules

Matching rule is $O(L)$, here $L$ is max. 32

<table>
<thead>
<tr>
<th>Rule</th>
<th>Prefix</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0</td>
<td>10.0.1.0/24</td>
<td>IF_A12</td>
</tr>
<tr>
<td>r1</td>
<td>1.10.0.0/22</td>
<td>RG_A1</td>
</tr>
<tr>
<td>r2</td>
<td>1.10.0.0/24</td>
<td>IF_A22</td>
</tr>
<tr>
<td>r3</td>
<td>1.10.0.0/28</td>
<td>IF_A31</td>
</tr>
</tbody>
</table>
Efficient Data Structure

BTries: Storing Intermediary Results

- Storing results of matching routing rules against IP ranges
- Searching is $O(\log L)$, here $L$ is max. 32
Efficient Data Structure

BTries: Storing Intermediary Results

- Storing results of matching routing rules against IP ranges
- Searching is $O(\log L)$, here $L$ is max. 32
Efficient Data Structure

BTries: Storing Intermediary Results

- Storing results of matching routing rules against IP ranges
- Searching is $O(\log L)$, here $L$ is max. 32
Prefix-to-prefix Algorithm

**Algorithm 1** prefix-to-prefix(btrie)

1: Input/Output: btrie
2: counter=0
3: for each range [L, H] in btrie.leafs with RLB = 00 do
4:    router = get(HR, r_id)
5:    dst = getroot(btrie)
6:    if searchTries(dst, router) = false then
7:        TempBTrie = Match(RadixTrie(router), dst)
8:    else
9:        TempBTrie = getBTrie(dst, router)
10:   Copy(btrie, TempBTrie, [L, H])
11:   counter = counter + 1
12: if counter ≠ 0 then
13:   prefix-to-prefix(btrie)
VM-to-VM Algorithm

Algorithm 2 VM-to-VM($VM_{src}$, $VM_{dest}$)

1: Triepub = getBTree($VM_{dst}.publicIP.CIDR, VM_{src}.subnet_id$)
2: Triepriv = getBTree($VM_{dst}.privateIP.CIDR, router_id$)
3: routable = Route-Lookup(Triepub, Triepriv)
4: if routable = true then
5: VerifySecGroups($VM_{src}, VM_{dest}$)
Incremental Verification

Graph update
Radix trie creation/deletion
Radix trie update
X-fast trie creation/deletion
X-fast trie update
VM-level isolation verification
Security group verification

Tenant alpha
Tenant beta

NEW!
Incremental Verification

Adding a Security Group
Application to OpenStack

- OpenStack Kilo with one controller and 80 compute nodes
- Parallelization of reachability verification with Apache Ignite
- Integration to OpenStack Congress
Experimental Settings

- Test Environment
  - Two series of datasets
    - SNET (represents small to medium networks)
    - LNET (represents large networks)
  - NoD (NSDI15) and a baseline algorithm

- Real Cloud
  - Ericsson research cloud
  - Mainly to evaluate the real world applicability of TenantGuard
  - Only observed a minor incompatibility issue due to version mismatch
Performance Evaluation

Data collection and processing time vary from 1.5 to 2 seconds

TenantGuard performs 82% faster than the baseline
Further Performance Improvement

![Graphs showing performance improvement with different VM counts and worker node numbers.]

- **Reachability**: Between 168 millions VM pairs in 13 seconds
- **Relationship**: Between cluster size and speedup gain
Identifying Performance Factors

Number of VMs and hops have less effects due to the reduced complexity and design

Number of routing rules has almost no effect due to the use of Radix and X-fast tries
Conclusion

Future Work
- Integrating existing tools at other layers (physical, L2)
- Ensuring integrity of input data
- Addressing privacy issues from the verification results

Summary
- TenantGuard, a VM-level network isolation verification system
- Integrated our approach to OpenStack
- Reachability for over 150 million VM pairs in 13 seconds
Thank You!

Q & A?