Acila:
Attaching Identities of Workloads for Efficient Packet Classification in a Cloud Data Center Network

Kentaro Ohnishi, Daisuke Kotani (Kyoto University)
Hirofumi Ichihara, Yohei Kanemaruu (LINE)
Yasuo Okabe (Kyoto University)
Network Control in Cloud

Various software (workload) are running in different styles on cloud Virtual machines, Containers or Pods, Bare metal servers, etc. Enforcing some control on packets between workloads is essential:

- Packet filtering to maintain security
- QoS to keep application performance
Cloud Infrastructure

Clos network topology + IP routing

VMs, containers, etc. (workloads) are elastically deployed on servers where resources are available

- Workloads of multiple systems coexist in one server (hypervisor)
- IP addresses are assigned based on locations

External connections to other networks (legacy systems, enterprise networks, etc)
Rules for Network Control

Traditional Way:
Workloads of each system has IP addresses from the same IP prefix => use IP prefix (+ port) for rules

In Cloud:
Workloads has IP addresses assigned from the IP prefix allocated to the host => use IP address (+ port) for Rules

# of entries and their update frequency will be explosively increased

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>System A 192.0.2.0/26</th>
<th>System B 192.0.2.64/26</th>
</tr>
</thead>
<tbody>
<tr>
<td>System A</td>
<td>Allow</td>
<td>Deny</td>
<td></td>
</tr>
<tr>
<td>System B</td>
<td>Allow</td>
<td>Allow</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>System A 192.0.2.1</th>
<th>System B 192.0.2.2</th>
<th>System A 192.0.2.3</th>
<th>System B 192.0.2.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>System A</td>
<td>Allow</td>
<td>Deny</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System B</td>
<td>Allow</td>
<td>Allow</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

192.0.2.0/26, 192.0.2.64/26, Deny
*, *, Allow
192.0.2.1, 192.0.2.2, Deny
192.0.2.1, 192.0.2.4, Deny
192.0.2.3, 192.0.2.2, Deny
192.0.2.3, 192.0.2.4, Deny
Related Work

Attach a tag to packets to deliver what the source is (Cilium, eZTrust)

Policies requiring both src and dest info can be applied **only at the dest** because packets lack the destination info

Deploy overlay networks and assign IP addresses in the traditional way

Management overhead (incl. interconnection between networks)

Especially for microservices where a system consists of multiple loosely-coupled services

Separation of host identifiers and locators (HIP, Mobile IP, LISP)

Host level granularity are not necessary for the rules
Our Work

Q: Can we design and use identifiers for implementing variety of network control on cloud in a simple and efficient way?

- Supporting VMs, containers, legacy systems via external connections
- Network controls can be applied at both ends and in-network

Our proposal: Add identifiers to packets showing what a client is and a server is (i.e. identity), by Acila

- Design how to generate identifiers and attach them to packets
- Implementation using eBPF as data plane
- Applications and evaluation
Workload, Identity of Workload and Labels

**Workload**: a single piece of software, deployed with a particular configuration for a single purpose (by SPIFFE)

Deployed as VM, container, process in a VM, etc.

**Identity of Workload**: information that characterizes the workload

Represented as a set of pairs of key and value (called Labels)

Cloud controller (or orchestrator) has a database of identity of all workloads in the cloud.

<table>
<thead>
<tr>
<th>Workload</th>
<th>Condition for Identification</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workload A (VM)</td>
<td>IP = IP1, ListenPort = 80</td>
<td>user: alice, app: a</td>
</tr>
<tr>
<td>Workload B (Process)</td>
<td>IP = IP2, UID = 101</td>
<td>user: bob, group: 2</td>
</tr>
<tr>
<td>Workload C (Container)</td>
<td>IP = IP3, ListenPort = *</td>
<td>user: bob, group: 2</td>
</tr>
</tbody>
</table>
**Workload, Policy and Service**

**Service** represents a group of workloads whose packets are processed in the same way, computed from Policy

- Use a subset of labels for describing workload selection criteria
- **SACL ID** is assigned to each service

Each service has workloads whose labels are covered by service’s criteria

<table>
<thead>
<tr>
<th>Workload</th>
<th>IP</th>
<th>User and Group Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workload A</td>
<td>IP = IP1, ListenPort = 80</td>
<td>user: alice, app: a</td>
</tr>
<tr>
<td>Workload B</td>
<td>IP = IP2, UID = 101</td>
<td>user: bob, group: 2</td>
</tr>
<tr>
<td>Workload C</td>
<td>IP = IP3, ListenPort = *</td>
<td>user: bob, group: 2</td>
</tr>
</tbody>
</table>
**Acila at a Glance**

**Workload Information**
(Condition for identification, labels)

**Policy**

**Acila Controller**
- Create Services
- Generate SACL IDs
- Service - Workload mapping
- Translate Policy into Rules using SACL IDs

**Control Plane of other systems**

**Control Plane**

**Data Plane**

**SACL Gateway**
- Attach/Detach SACL IDs in packets

**Distribute rules, entries for Workload identification -> SACL IDs, other data**

**Workload A**

**Workload B**

**Original Packet**

**Workload C**

**Workload D**

**Original Packet**

**Original Packet**

**Original Packet**

**Client SACL ID**

**Server SACL ID**
Data Plane - Where to Install SACL Gateway

Close to the workloads as much as possible where trusted by cloud admins

- Rich information is available to identify the workload Interface, IP address before translated (e.g. Virtual IP), etc
- Avoid forgery of SACL IDs in packets
Data Plane - Client Side and Server Side

Translate Workloads into SACL IDs, and attach them to packets as IPv6 Hop-by-hop Option designed for Acila

SACL Gateway for Client Workload

- Identify Client Workload
  - VMs and Containers by IP address
  - Process by LID Marker inside the VM
- Identify Server Workload by IP address and port

SACL Gateway for Server Workload

- Use Conntrack to identify client workloads
Each policy consists of:

- **Client Selectors**: a set of (key, operator, values)
- **Server Selectors**: a set of (key, operator, values)
- **Value**: (optional, used for applications)

**key**: a key of labels of workloads to be searched for

**operator**: whether any value in “values” should exist (or not)

**values**: a set of values that should or should not exist
Acila Controller

SAACL ID generation

Currently labels of workloads and SAACL ID have one-to-one mapping (just for simplicity, should improve in the future)

Workload identification and SAACL ID mapping rule

IP address and listen port (+ process UID via LID Marker)

=> SAACL ID

Distribute rules for SAACL Gateways and switches
Acila Applications

Packet Filtering:
- Use Client/Server SACL IDs for matching condition
- Can filter out packets anywhere in the network

Priority Packet Scheduling in The Network
- Use Client/Server SACL IDs for matching condition
- Deploy rules at all switches in the network

Acila is effective because there is no need to update rules according to elastic changes of workloads
  c.f. IP address based rules should be updated once a workload is created or deleted
Implementation for Packet Filtering

Control Plane: Policies are interpreted as allowlist

Data Plane

![Diagram of packet filtering process]

- **Hypervisor**
  - **VM**
    - **[Client] Workload**
  - **Process**
    - **[Client] Workload**
    - **iptables LID Marker**
    - **Add LID by Linux uid**
  - **tc (eBPF)**
    - **SACL Gateway**
    - **Add src / dst SACL ID**

- **Switches / Routers**
  - **Packet Filtering by SACL ID**
  - **Add src / dst SACL ID via conntrack**
  - **Allow established, related connection via conntrack**

- **VM**
  - **[Server] Workload**
  - **tc (eBPF)**
    - **SACL Gateway**
    - **Packet with src / dst SACL ID by conntrack**
    - **Packet with src LID**
    - **Normal packet**
Evaluation: Estimated # of Entries

Spine switches under priority packet scheduling in a clos network

<table>
<thead>
<tr>
<th>Entries</th>
<th>Update Frequency by Workload Creation and Deletion</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP address based</td>
<td>Acila</td>
</tr>
<tr>
<td>$\sum_{w_i \in W} \sum_{s_k \in SS_{s_j}}</td>
<td>W_{s_k}</td>
</tr>
</tbody>
</table>

W: All Workloads, W_{s_j}: Workloads belonging to Service s_j, S: All Services SS_{s_j}: Server Services accessed by Client Service s_j with priority CS_{s_j}: Client Services that access Server Service s_j with priority

Clearly, Acila requires less entries and less entry updates

More on our paper and preprint (arxiv:2109.08343)
Evaluation: Performance of SACL Gateway

SACL Gateway will have many entries for mapping IP addresses, ports, and LIDs to Client/Server SACL IDs as well as Conntrack.

Measure impact on packet forwarding performance by entries

Assumption of entries about # of VMs, Workloads, Connections between Services:
1.3 \times 10^2\alpha for Client SACL IDs,
3.8 \times 10^3\alpha for Server SACL IDs,
3.8 \times 10^3\alpha for Conntrack

Environment:
Two machines connected by 100GbE, AMD EPYC 7282, 128GB RAM
Concluding Remarks and Future Work

In cloud, IP address is useless for identifying endpoint in network control. Acila provides a new identifiers, SACL IDs, attached to packets for identifying endpoints in network control. SACL IDs are generated from identity of workloads and policies. SACL IDs are useful for various applications at least for packet filtering and priority packet scheduling.

Future work includes:

- Improve Workload identification and SACL IDs generation process.
- Support/implement for other cases, such as legacy systems and NIC offload.