Role Based Access Control For Software Defined Networking
Formal Models and Implementation

Dissertation Defense
Abdullah Al-Alaj
Institute for Cyber Security
Department of Computer Science
The University of Texas at San Antonio

Committee:
Prof. Ravi Sandhu, Ph.D. (Advisor)
Dr. Ram Krishnan, Ph.D. (Co-advisor)
Dr. Palden Lama, Ph.D.
Prof. Gregory White, Ph.D.
Dr. Weining Zhang, Ph.D.

Outline

• Introduction
• SDN-RBAC Model
• Parameterized Permissions and Roles
• ParaSDN Model for Fine Grained and Scalable Authorization in SDN
• SDN-RBACa Administrative Model
• Proxy Operations and Custom Permissions
• Conclusion and Future Work
Introduction
Traditional Networks

Management Layer

Infrastructure Layer

Traditional Network

SDN Idea

Decoupling

@ Abdullah Al-Alaj
Introduction
Software Defined Networks (SDN)

Applications
- Routing
- Firewall
- Load Balancing
- Intrusion Prevention
- Network Visualization
- Other

Network Programming APIs
- Network Services
  - Topology Service
    - Entry Pushing
  - Routing Service
    - Device Management
  - Statistics Collection
    - Link Discovery
  - Switch Management
    - Other
- Virtual Network Resources
  - Topology
  - Flow tables
  - Switches
  - Ports
  - Statistics
  - Traffic payloads
  - Configurations
  - VLANs
  - Devices
  - Other

Controller (e.g., Floodlight)
- OpenFlow Protocol
  - Flow table

Infrastructure

Decoupling
- Control plane
  - Manage
  - Other
- Data plane
Features Provided by SDN Architecture

- Logically Centralized Control
- Dynamic Flow Control
- Network Programmability
- Network-Wide Visibility
Flow Table Structure

OpenFlow Table Entry

<table>
<thead>
<tr>
<th>Rule</th>
<th>Priority</th>
<th>Action</th>
<th>Counters</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC src</td>
<td>MAC dst</td>
<td>IP src</td>
<td>IP dst</td>
</tr>
<tr>
<td>..:00:01</td>
<td>..:00:04</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>..:00:04</td>
<td>..:00:01</td>
<td>*</td>
<td>10.0.0.3</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Packet Processing in OpenFlow Switch

1. Packet from network
2. Parse header fields
3. Match Found?
   - No: Send to controller
   - Yes: Update counters
4. Apply actions
Flow Rule Insertion Example

1. Switch receives a network packet.
2. Parse header fields.
3. Match found? (Yes/No)
4. If yes, update counters and apply actions.
5. If no, send to controller.

Packet Processing in OpenFlow Switch

Controller

Flow rule:
Mac-Host-A -> Mac-Host-D: Port-4

Insert flow rule:
Mac-Host-A -> Mac-Host-D: Port-4

Insert flow rule:
Mac-Host-A -> Mac-Host-D: Port-4

Routing app

Routing

Read topology info.
Find shortest path
Read hosts info.
Insert flow rule

Forward packet to network apps

If a matching rule found in table, apply actions; otherwise, forward packet to controller.

Switch receives a network packet

Packet from network

Parse header fields

Match Found?

Yes

Update counters

Apply actions

No

Send to controller
• Control which subjects (network apps) can access which objects (virtual network resources) for performing which actions (SDN operations).
Literature of Access Control for SDN

- Capability-based approaches
  - Direct relation between operations and apps.
  - Well studied and known to have administrative complexities.

### Capability-based approach

- Network apps: App1, App2, App3, App4
- Permissions: P1, P2, P3, P4, P5, P6

*Total associations = 3 x 6 = 18*
- 1 new app requires 6 new associations
- 1 new permission requires 3 new associations

### Role-based approach

- Network apps: App1, App2, App3
- Permissions: P1, P2, P3, P4, P5, P6

*Total associations = 3 + 6 = 9*
- 1 new app requires 1 new association
- 1 new permission requires 1 new association
Problem Statement:

Current Software Defined Networking technology is lacking access control models and enforcement for protecting network resources residing in the SDN controller from unauthorized access by OpenFlow applications.

Thesis Statement:

Role-based access control model and its extensions is an effective approach for the specification and administration of dynamic access control for Software Defined Networking.
Summary of Contributions

• Enabling Role Based Authorization for SDN.
  • SDN-RBAC Model and Authorization Framework with Implementation & Enforcement in SDN Controller.

• Fine-Grained and Scalable Access Control for SDN.
  • Access Control Enhanced with Role and Permission Parameters with Authorization Framework Extended with Parameter Engine and Enforcement in SDN Controller.

• Administration of Access Control in SDN.
  • SDN-RBACa Administrative Model for Managing roles, Permissions and Network App Authorizations in SDN.
  • Proxy Operations and Custom Permissions for Enhanced Engineering of Administrative Units in SDN.
Summary of Contributions

• Enabling Role Based Authorization for SDN.
  • SDN-RBAC Model and Authorization Framework with Implementation & Enforcement in SDN Controller.

• Fine-Grained and Scalable Access Control for SDN.
  • Access Control Enhanced with Role and Permission Parameters with Authorization Framework Extended with Parameter Engine and Enforcement in SDN Controller.

• Administration of Access Control in SDN.
  • SDN-RBACa Administrative Model for Managing roles, Permissions and Network App Authorizations in SDN.
  • Proxy Operations and Custom Permissions for Enhanced Engineering of Administrative Units in SDN.
**SDN-RBAC: Conceptual Model**

- **Design goal:** conformance with the standard NIST-RBAC Reference Model.
- **SDN-RBAC** adopts standard RBAC model with evolutionary changes, rather than revolutionary.

**Sample Apps:**
- Load Balancer
- Firewall
- Intrusion Prevention
- Routing app
- etc.

**Sample Sessions:**
- *deep packet inspection session*
- *transmission rate monitoring session*
- *web-traffic filtering session*
- *shortest path recomputation session*
- *traffic redirection session*
- etc.

**Sample Roles:**
- *Flow Mod*
- *Device Handler*
- *Bandwidth Monitoring*
- *Link Handler*
- etc.

**Sample Operations:**
- *addFlow*
- *getAllDevices*
- *getBandwidthConsumption*
- etc.

**Sample Object Types:**
- *FLOW-RULE*
- *DEVICE*
- *PORT-STATS*
- *PORT*
- *LINK*
- etc.
SDN-RBAC Formal Model Definition

1. Model Element Sets:
   - \( APPS, ROLES, OPS, OBS \) and \( OBTS \), a finite set of OpenFlow apps, roles, operations, objects and object types, respectively.
   - \( PRMS \subseteq 2^{OPS \times OBTS} \), the set of permissions.
   - \( SESSIONS \), a finite set of sessions.

2. Assignment Relations:
   - \( PR \subseteq PRMS \times ROLES \), a many-to-many mapping permission-to-role assignment relation.
   - \( AR \subseteq APPS \times ROLES \), a many-to-many mapping app-to-role assignment relation.
   - \( OT \subseteq OBS \times OBTS \), a many-to-one relation mapping an object to its type.

3. Derived Functions
   - \( assigned\_perms(r : ROLES) \rightarrow 2^{PRMS} \), the mapping of role \( r \) into a set of permissions. Formally, \( assigned\_perms(r) \subseteq \{ p \in PRMS | (p, r) \in PR \} \).
   - \( app\_sessions(a : APPS) \rightarrow 2^{SESSIONS} \), the mapping of an app into a set of sessions.
   - \( session\_app(s : SESSIONS) \rightarrow APPS \), the mapping of session into the corresponding app.
   - \( session\_roles(s : SESSIONS) \rightarrow 2^{ROLES} \), the mapping of session into a set of roles. Formally, \( session\_roles(s) \subseteq \{ r \in ROLES | (session\_app(s), r) \in AR \} \).
   - \( type : OBS \rightarrow OBTS \), a function specifying the type of an object, where \( (o, t_1) \in OT \land (o, t_2) \in OT \Rightarrow t_1 = t_2 \).
   - \( avail\_session\_perms(s : SESSIONS) \rightarrow 2^{PRMS} \), the permissions available to an app in a session \( = \bigcup_{r \in session\_roles(s)} assigned\_perms(r) \).
Use-case in SDN-RBAC

Multi session app: Data Usage Cap Manager

Data Cap Analysis Session
- Read Port BW Statistics
- Identify BW Violations
- Get All Devices
- Identify Violating device

Every 5 seconds

Data Cap Enforcing Session
- Read Black List
- Insert Flow Rule

Requires BW Monitoring Role
Every 60 seconds

Requires Flow Mod Role

Requires Device Handler Role

Update

Black Listed Devices
Use-Case Security Configuration in SDN-RBAC

1. Use-Case Sets:
- \( \text{APPS} = \{ \text{DataUsageCapMngr} \} \)
- \( \text{ROLES} = \{ \text{Device Handler, Bandwidth Monitoring, Flow Mod} \} \)
- \( \text{OBS} = D \cup FR \cup PS \), where \( D \) is set of all network devices, \( FR \) is set of all flow rules, and \( PS \) is set of all port statistics in all switches.
- \( \text{OBTS} = \{ \text{DEVICE, PORT-STATS, FLOW-RULE} \} \)
- \( \text{PRMS} = \{ (\text{getAllDevices, DEVICE}), (\text{getBandwidthConsumption, PORT-STATS}), (\text{addFlow, FLOW-RULE}) \} \)
- \( \text{SESSIONS} = \{ \text{DataUsageAnalysisSession, DataCapEnforcingSession} \} \)

2. Assignment Relations:
- \( PR = \{ (\text{getAllDevices, DEVICE}), (\text{Device Handler}), (\text{getBandwidthConsumption, PORT-STATS}), (\text{Bandwidth Monitoring}), (\text{addFlow, FLOW-RULE}), (\text{Flow Mod}) \} \)
- \( AR = \{ (\text{DataUsageCapMngr, Device Handler}), (\text{DataUsageCapMngr, Bandwidth Monitoring}), (\text{DataUsageCapMngr, Flow Mod}) \} \).
- \( OT = \{ (d, \text{DEVICE}) : d \in D \} \cup \{ (ps, \text{PORT-STATS}) : ps \in PS \} \cup \{ (fr, \text{FLOW-RULE}) : fr \in FR \} \).

3. Derived Functions:
- \( \text{assigned_perms(DeviceHandler)} = \{ (\text{getAllDevices, DEVICE}) \} \)
- \( \text{assigned_perms(BandwidthMonitoring)} = \{ (\text{getBandwidthConsumption, PORT-STATS}) \} \)
- \( \text{assigned_perms(FlowMod)} = \{ (\text{addFlow, FLOW-RULE}) \} \)
- \( \text{app_sessions(DataUsageCapMngr)} = \{ \text{DataUsageAnalysisSession, DataCapEnforcingSession} \} \)
- \( \text{session_app(DataUsageAnalysisSession)} = \{ \text{DataUsageCapMngr} \} \)
- \( \text{session_app(DataCapEnforcingSession)} = \{ \text{DataUsageCapMngr} \} \)
- \( \text{session_roles(DataUsageAnalysisSession)} = \{ \text{Device Handler, Bandwidth Monitoring} \} \)
- \( \text{session_roles(DataCapEnforcingSession)} = \{ \text{Flow Mod} \} \)
- \( \text{avail_session_perms(DataUsageAnalysisSession)} = \{ (\text{getAllDevices, DEVICE}), (\text{getBandwidthConsumption, PORT-STATS}) \} \)
- \( \text{avail_session_perms(DataCapEnforcingSession)} = \{ (\text{addFlow, FLOW-RULE}) \} \)
SDN-RBAC Average Authorization Time

On average: 0.0245 ms overhead for 50 operations.

- Test app with 50 ops covered by 10 different roles.
- Report authorization time for all 50 requests.
- Different security policies.
- Test repeated 100 times for each security policy.
- Average authorization time is calculated.
- Floodlight’s boot-up time is ignored.
Summary of Contributions

- Enabling Role Based Authorization for SDN.
  - SDN-RBAC Model and Authorization Framework with Implementation & Enforcement in SDN Controller.

- Fine-Grained and Scalable Access Control for SDN.
  - Access Control Enhanced with Role and Permission Parameters with Authorization Framework Extended with Parameter Engine and Enforcement in SDN Controller.

- Administration of Access Control in SDN.
  - SDN-RBACa Administrative Model for Managing roles, Permissions and Network App Authorizations in SDN.
  - Proxy Operations and Custom Permissions for Enhanced Engineering of Administrative Units in SDN.
Limitations of SDN-RBAC

- Apps are authorized on object types (e.g., (addFlow, FLOW RULE)) → Fine grained access control is required.

- Multiple very closely related roles are defined to achieve fine-grained access control.
- Roles are limited in membership.

Role explosion

Permission explosion
Introducing Parameterized Roles and Permissions in SDN

Role: Flow Mod
Assigned Perms:
- (addFlow, FLOW RULE)
- (deleteFlow, FLOW RULE)
- (updateFlow, FLOW RULE)
- (readFlow, FLOW RULE)

dept = CS
depth = CIS
depth = CE

Requires restriction
Parameterized Permissions and Roles

- **Parameters**
  - name:value pairs.
  - Add restrictions on access to network resources.

- **Parameterized Roles:**
  \[(r_i, \{(\text{par}_1, \text{val}_1), (\text{par}_2, \text{val}_2), \ldots\})\]

  **Example:**
  \[(\text{Flow Mod}, \{(\text{dept}, \bot), (\text{traffic}, \bot)\})\]

- **Parameterized Permissions:**
  \[((\text{op}_i, \text{ot}_i), \{(\text{par}_1, \text{val}_1), (\text{par}_2, \text{val}_2), \ldots\})\]

  **Example:**
  \[((\text{addFlow, FLOW-RULE}), \{(\text{dept}, \bot), (\text{traffic}, \bot)\})\]

\(\bot = \text{Unknown.}\)
ParaSDN Conceptual Model

many-to-many

one-to-many

APPs

VALUES (VAL)

PARAMETERS (PAR)

PVPAIRS

ROLES

PROLES

PARAMETERS (PAR)

VALUES (VAL)

OPERATIONS (OPS)

OBJECT TYPES (OBTs)

OBJECTS (OBS)

PPA

AA

app_sessions

session_roles

PRMS

PPRMS

PVPAIRS

VALUES (VAL)

OT

PRMS

PARAMETERS (PAR)
ParaSDN Formal Model Definition

1. Basic Sets:
   - APPS, ROLES, OPS, OBS, OBTS, PAR, and VAL: set of apps, roles, operations, objects, object types, parameters, and parameter values.
   - For each \( par \in PAR \), Range(\( par \)) represents the parameter’s range, a finite set of atomic values. We assume VAL includes a special value “\( \bot \)” to indicate that the value of a parameter is unknown.
   - \( \text{parType} : PAR \rightarrow \{ \text{set, atomic} \} \) specifies parameter type as set of atomic valued.
   - \( \text{PRMS} \subseteq \text{OPS} \times \text{OBTS} \), set of ordinary permissions.
   - \( \text{SESSIONS} \), set of sessions.

2. Assignment Relations:
   - \( \text{OT} \subseteq \text{OBS} \times \text{OBTS} \), a many-to-one relation mapping an object to its type, where
     \[(a, ot_1) \in \text{OT} \land (a, ot_2) \in \text{OT} \Rightarrow ot_1 = ot_2.\]
   - \( \text{PVPAIRS} \subseteq \text{PAR} \times \text{VAL} \), a many-to-many mapping parameter to value assignment relation.
     - For convenience, for every \( \text{pppair} = (par_i, val_i) \in \text{PVPAIRS} \), let \( \text{pppair.par} = par_i \) and \( \text{pppair.val} = val_i \).
   - \( \text{PPRMS} \subseteq \text{PRMS} \times 2^{\text{PVPAIRS}} \), a relation mapping a permission role to subset of (parameters \( , \) value) combinations.
     - For convenience, for every \( \text{pp} = ((\text{op}_i, ot_i), \text{PVPAIRS}_i) \in \text{PPRMS} \), let \( \text{pp.op} = \text{op}_i \), \( \text{pp.ot} = \text{ot}_i \), and \( \text{pp.PVPAIRS} = \text{PVPAIRS}_i \).
   - \( \text{PROLES} \subseteq \text{ROLES} \times 2^{\text{PVPAIRS}} \), a relation mapping a role to subset of combinations of parameters and their values.
     - For convenience, for every \( \text{pr} = (r_i, \text{PVPAIRS}_i) \in \text{PROLES} \), let \( \text{pr.r} = r_i \) and \( \text{pr.PVPAIRS} = \text{PVPAIRS}_i \).
   - \( \text{PPA} \subseteq \text{PRMS} \times \text{PROLES} \), a many-to-many mapping parameterized permission to parameterized role assignment relation.
   - \( \text{AA} \subseteq \text{APPS} \times \text{PROLES} \), a many-to-many mapping app to parameterized role assignment relation.

3. Derived Functions:
   - assigned_\( p \text{perms} \) : \( \text{PROLES} \rightarrow 2^{\text{PRMS}} \), the mapping of parameterized role into a set of parameterized permissions.
     - Formally, \( \text{assigned.\( p \text{perms}(pr) = \{ pp \in \text{PPRMS} \mid (pp, pr) \in \text{PPA} \}. \)} \)
   - app_sessions : \( \text{APPS} \rightarrow 2^{\text{SESSIONS}} \), the mapping of an app into a set of sessions.
   - session_app : \( \text{SESSIONS} \rightarrow 2^{\text{APPS}} \), the mapping of session into the corresponding app.
   - session_roles : \( \text{SESSIONS} \rightarrow 2^{\text{PROLES}} \), the mapping of session into a set of parameterized roles.
     - Formally, \( \text{session_roles(s)} = \{ pr \in \text{PROLES} \mid (\text{session_app(s), pr}) \in \text{AA} \}. \)
   - type : \( \text{OBS} \rightarrow \text{OBTS} \), a function specifying the type of an object defined as
     \( \text{type}(o) = \{ t \in \text{OBTS} \mid (o, t) \in \text{OT} \}. \)
   - avail_session_\( p \text{perms} \) : \( \text{SESSIONS} \rightarrow 2^{\text{PRMS}} \), the parameterized permissions available to an app in a session.
     - Formally, \( \text{avail_session_\( p \text{perms}(s) = \bigcup_{pr \in \text{session_roles(s)}} \text{assigned_\( p \text{perms}(pr) \).} \}) \)

4. Parameter Verification Functions:
   - \( \text{VERIFIERS} = \{ V_1, V_2, \ldots, V_n \} \) a finite set of Boolean functions.
     - For each \( V_i \in \text{VERIFIERS} \), \( V_i : \text{SESSIONS} \times \text{OPS} \times \text{OBS} \times \text{PVPAIRS} \rightarrow \{ \text{True, False} \}. \)
   - \( \text{param_verifier} : \text{OBTS} \times \text{PAR} \rightarrow \text{VERIFIERS} \), a function that maps a combination of object type and parameter to the corresponding verification function needs to be evaluated.
Use-Case Security Configuration in SDN-RBAC

@ Abdullah Al-Alaj

1. Model Basic Sets:

- **APPS** = {Data Usage Cap Mngr, Intrusion Prevention App}
- **ROLES** = {Device Handler, Bandwidth Monitoring, Flow Mod, Packet-In Handler}
- **OPS** = {queryDevice, getBandwidthConsumption, addFlow, readPacketInPayload}
- **OBS** = D ∪ PS ∪ FR ∪ PIP, where D = set of all network devices, PS = set of all port statistics in all switches, FR = set of all flow rules, and PIP = set of all packet-in messages.
- **OBTS** = {DEVICE, PORT-STATS, FLOW-RULE, PI-PAYLOAD}
- **PAR** = \{vlan_id, attachment_point, dept, traffic\}
  - Range(vlan_id) = {1, 2}, Range(attachment_point) = {0x1:1, 0x1:2, 0x2:2, 0x3:1}, Range(dept) = {CS, CE}, Range(traffic) = {web}
- **parType** = atomic
- **PRMS** = {queryDevice, DEVICE, getBandwidthConsumption, PORT-STATS, addFlow, FLOW-RULE, readPacketInPayload, PI-PAYLOAD}
- **SESSIONS** = {DataUsageAnalysisSession, DataCapEnforcingSession, IntrusionPreventionSession}

2. Assignment Relations:

- **OT** = \{d, DEVICE\} : d ∈ D \} \cup \{ps, PORT-STATS : ps ∈ PS} \cup \{fr, FLOW-RULE : fr ∈ FR} \cup \{pip, PI-PAYLOAD : pip ∈ PIP\}
- **PPRMS** = ((queryDevice, DEVICE), ((vlan_id, ⊥)), ((getBandwidthConsumption, PORT-STATS), (attachment_point, ⊥))), ((addFlow, FLOW-RULE), (dept, ⊥), (traffic, ⊥)), (readPacketInPayload, PI-PAYLOAD), (attachment_point, ⊥))
- **PROLES** = ((Device Handler, (vlan_id, ⊥)), (Bandwidth Monitoring, (attachment_point, ⊥))), ((Flow Mod, (dept, ⊥), (traffic, ⊥)), (Packet-In Handler, (attachment_point, ⊥))), ((PBP), (dept, ⊥), (traffic, ⊥)), (Bandwidth Monitoring, (attachment_point, ⊥)), ((addFlow, FLOW-RULE), (dept, ⊥), (traffic, ⊥)), (Flow Mod, (dept, ⊥), (traffic, ⊥)), ((readPacketInPayload, PI-PAYLOAD), (attachment_point, ⊥)), (Packet-In Handler, (attachment_point, ⊥))
- **AA** = {Data Usage Cap Mngr, (vlan_id, 1)}, (Data Usage Cap Mngr, (Bandwidth Monitoring, (attachment_point, (0x1:1, 0x1:2, 0x2:2, 0x3:1)))), (Data Usage Cap Mngr, (Flow Mod, (dept, (CS), (traffic, web)))), (Intrusion Prevention App, (Device Handler, (vlan_id, 2)), (Intrusion Prevention App, (Packet-In Handler, (attachment_point, (0x3:1)))), (Intrusion Prevention App, (Flow Mod, (dept, (CE), (traffic, web))))

3. Derived Functions:

- **assigned_pperms** (Device Handler, (vlan_id, ⊥)) = (((queryDevice, DEVICE), (vlan_id, ⊥)),)
  - assigned_pperms (Bandwidth Monitoring, (attachment_point, ⊥)) = (((getBandwidthConsumption, PORT-STATS), (attachment_point, ⊥)),)
  - assigned_pperms (Flow Mod, (dept, ⊥), (traffic, ⊥)) = (((addFlow, FLOW-RULE), (dept, ⊥), (traffic, ⊥)),)
  - assigned_pperms (Packet-In Handler, (attachment_point, ⊥)) = (((readPacketInPayload, PI-PAYLOAD), (attachment_point, ⊥)),)
- **app_sessions** (Data Usage Cap Mngr) = {DataUsageAnalysisSession, DataCapEnforcingSession, IntrusionPreventionSession}
- **session_roles** (DataUsageAnalysisSession) = {(Device Handler, (vlan_id, 1)), (Bandwidth Monitoring, (attachment_point, (0x1:1, 0x1:2, 0x2:2)))}
- **session_roles** (DataCapEnforcingSession) = {(Flow Mod, (dept, (CS)), (traffic, web))}
- **session_roles** (IntrusionPreventionSession) = {(Device Handler, (vlan_id, 2)), (Packet-In Handler, (attachment_point, (0x3:1)))}
- **avail_session_pperms** (DataUsageAnalysisSession) = (((queryDevice, DEVICE), (vlan_id, 1)),)
- **avail_session_pperms** (DataCapEnforcingSession) = (((addFlow, FLOW-RULE), (dept, (CS)), (traffic, web))),
- **avail_session_pperms** (IntrusionPreventionSession) = (((queryDevice, DEVICE), (vlan_id, 2)), (readPacketInPayload, PI-PAYLOAD), (attachment_point, 0x3:1))
- **avail_session_pperms** (DataUsageCapEnforcingSession) = (((addFlow, FLOW-RULE), (dept, (CS)), (traffic, web))),
- **avail_session_pperms** (DataCapEnforcingSession) = (((queryDevice, DEVICE), (vlan_id, 2)), (readPacketInPayload, PI-PAYLOAD), (attachment_point, (0x3:1)))

4. Parameter Verification Functions:

- **VERIFIERS** = {DeviceVlan, VStatsAttachpoint, VRuleSwitch, VRuleTraffic, VPInAttachpoint}
- **param_verifier** (DEVICE, vlan_id) = DeviceVlan
- **param_verifier** (PORT-STATS, attachment_point) = VStatsAttachpoint
- **param_verifier** (FLOW-RULE, dept) = VRuleSwitch
- **param_verifier** (FLOW-RULE, traffic) = VRuleTraffic
- **param_verifier** (PI-PAYLOAD, attachment_point) = VPInAttachpoint

25
ParaSDN Framework Implementation in Floodlight
ParaSDN Implementation & Evaluation

Timer
- Started
- Ended

Application Plane
- Session
- Network App
- Session

Control Plane (Floodlight)
- SDN-RBAC Authorization Framework
- Interception Component (PEP)
- Request Evaluation & Decision (PDP)
- Parameter Engine
- ParaSDN Policy (PIP)
- Verifiers Retrieval Point (VRP)
- Verifiers Map

Floodlight Services
- Topology Service
- Statistics Collection
- Device Management
- Routing Service
- Link Discovery
- Other

Floodlight Objects
- Topology
- Flow Tables
- Statistics
- Links
- Devices
- Other
ParaSDN Evaluation - 1

- Test app with 50 ops covered by 10 different roles.
- Report authorization time for all 50 requests.
- Different security policies (parameters and roles).
- Test repeated 100 times for each security policy.
- Average authorization time is calculated.
- Floodlight’s boot-up time is ignored.

On average: ParaSDN adds 0.031 ms overhead compared to 0.025 for SDN-RBAC.

- 1st parameter in all roles is:
  activePeriod = “08:00-17:00”.
- Any request submitted outside active period, will be denied.
- Test 8 is conducted outside active period.
ParaSDN Evaluation - 2

@ Abdullah Al-Alaj
Summary of Contributions

- Enabling Role Based Authorization for SDN.
  - SDN-RBAC Model and Authorization Framework with Implementation & Enforcement in SDN Controller.
- Fine-Grained and Scalable Access Control for SDN.
  - Access Control Enhanced with Role and Permission Parameters with Authorization Framework Extended with Parameter Engine and Enforcement in SDN Controller.
- Administration of Access Control in SDN.
  - SDN-RBACa Administrative Model for Managing roles, Permissions and Network App Authorizations in SDN.
  - Proxy Operations and Custom Permissions for Enhanced Engineering of Administrative Units in SDN.
Access Control Administration in SDN

- App-role and permission-role relations need management.
- In SDN-RBACa administrative model (inspired by Uni-ARBAC):
  - Indirect permission-role assignment.
  - Permissions are grouped into permission-pools (tasks).
  - Tasks: units of network functions.
  - Apps are grouped into app-pools.
  - Administrative Units for administering app-role and task-role relations.
SDN-RBACa Administrative Model

9. Administrative Actions:
- `assign_task_to_role(u: USERS, t: TASKS, r: ROLES)`
  - Authorization condition: can_manage_task_role(u, t, r) = True
  - Effect: TA' = TA ∪ {(t, r)}.
- `revoke_task_from_role(u: USERS, t: TASKS, r: ROLES)`
  - Authorization condition: can_manage_task_role(u, t, r) = True
  - Effect: TA' = TA \ {(t, r)}.
- `assign_app_to_role(u: USERS, a: APPS, r: ROLES)`
  - Authorization condition: can_manage_app_role(u, a, r) = True
  - Effect: AA' = AA ∪ {(a, r)}.
- `revoke_app_from_role(u: USERS, a: APPS, r: ROLES)`
  - Authorization condition: can_manage_app_role(u, a, r) = True
  - Effect: AA' = AA \ {(a, r)}.

8. Administrative User Authorization Functions:
- `can_manage_task_role(u: USERS, t: TASKS, r : ROLES) = \exists au\in AU : (u, au) \in TA_{admin} \land r \in roles(au) \land t \in tasks(au).`
- `can_manage_app_role(u: USERS, a : APPS, r : ROLES) = \exists au\in AU : ((u, au) \in AA_{admin} \land r \in roles(au)) \land \exists ap\in AP : ((a, ap) \in APPA \land ap \in app_pools(au)).`

6. Administrative Units and Partitioned Assignment:
- `roles(au : AU) \rightarrow 2^{ROLES}`, assignment of roles, where \( r \in \text{roles}(au_{1}) \land r \in \text{roles}(au_{2}) \Rightarrow au_{1} = au_{2}. \)
- `tasks(au : AU) \rightarrow 2^{TASKS}`, assignment of tasks, where \( t \in \text{tasks}(au_{1}) \land t \in \text{tasks}(au_{2}) \Rightarrow au_{1} = au_{2}. \)
- `app_pools(au : AU) \rightarrow 2^{AP}`, assignment of app-pool, where \( ap \in \text{app_pools}(au_{1}) \land ap \in \text{app_pools}(au_{2}) \Rightarrow au_{1} = au_{2}. \)

7. Administrative User Assignment:
- `TA_{admin} \subseteq USERS \times AU`.
- `AA_{admin} \subseteq USERS \times AU`.

@ Abdullah Al-Alaj
SDN-RBACa Administrative Model Definition

1. Basic Sets
   - APPS is a finite set of SDN apps.
   - OPS is a finite set of operations.
   - OBS is a finite set of objects.
   - OBTS is a finite set of object types.
   - PRMS ⊆ OPS × OBTS, set of permissions.
   - ROLES is a finite set of roles.
   - TASKS is a finite set of tasks.
   - AP is a finite set of app-pools.
   - USERS is a finite set of administrative users.
   - AU is a finite set of administrative units.

2. Assignment Relations (operational):
   - PA ⊆ PRMS × TASKS, permission-task assignment relation.
   - TA ⊆ TASKS × ROLES, task-role assignment relation.
   - AA ⊆ APPS × ROLES, app-role assignment relation.
   - OT ⊆ OBS × OBTS, a many-to-one mapping an object to its type, where \((o, t_1) \in OT \land (o, t_2) \in OT \Rightarrow t_1 = t_2\).

3. Derived Functions (operational):
   - type: (o: OBS) → OBTS, a function specifying the type of an object.
   - Defined as type(o) = \{t ∈ OBTS | (o, t) ∈ OT\}.
   - authorized_perms(r: ROLES) → 2^{PRMS}, defined as authorized_perms(r) = \{p ∈ PRMS | \exists t ∈ TASKS, \exists r ∈ ROLES : (t, r) ∈ TA \land (p, t) ∈ PA\}.

4. App Authorization Function:
   - can_exercise_permission(a: APPS, op: OPS, ob: OBS) = \exists r ∈ ROLES : (op, type(ob)) ∈ authorized_perms(r) \land (a, r) ∈ AA.

5. Administrative App-pools Relation:
   - AAAP ⊆ APPS × AP, app to app-pool assignment relation.

6. Administrative Units and Partitioned Assignment:
   - roles(au : AU) → 2^{ROLES}, assignment of roles, where \(r ∈ roles(au_1) \land r ∈ roles(au_2) \Rightarrow au_1 = au_2\).
   - tasks(au : AU) → 2^{TASKS}, assignment of tasks, where \(t ∈ tasks(au_1) \land t ∈ tasks(au_2) \Rightarrow au_1 = au_2\).
   - app_pools(au : AU) → 2^{AP}, assignment of app-pool, where \(ap ∈ app_pools(au_1) \land ap ∈ app_pools(au_2) \Rightarrow au_1 = au_2\).

7. Administrative User Assignment:
   - TA_admin ⊆ USERS × AU.
   - AA_admin ⊆ USERS × AU.

8. Administrative User Authorization Functions:
   - can_manage_task_role(u : USERS, t : TASKS, r : ROLES) = \exists au ∈ AU : (u, au) ∈ TA_admin \land r ∈ roles(au) \land t ∈ tasks(au).
   - can_manage_app_role(u : USERS, a : APPS, r : ROLES) = \exists au ∈ AU : ((u, au) ∈ AA_admin \land r ∈ roles(au)) \land \exists ap ∈ AAAP : ((a, ap) ∈ AAAP \land ap ∈ app_pools(au)).

9. Administrative Actions:
   - assign_task_to_role(u : USERS, t : TASKS, r : ROLES)
     Authorization condition: can_manage_task_role(u, t, r) = True
     Effect: TA’ = TA \cup \{(t, r)\}.
   - revoke_task_from_role(u : USERS, t : TASKS, r : ROLES)
     Authorization condition: can_manage_task_role(u, t, r) = True
     Effect: TA’ = TA \setminus \{(t, r)\}.
   - assign_app_to_role(u : USERS, a : APPS, r : ROLES)
     Authorization condition: can_manage_app_role(u, a, r) = True
     Effect: AA’ = AA \cup \{(a, r)\}.
   - revoke_app_from_role(u : USERS, a : APPS, r : ROLES)
     Authorization condition: can_manage_app_role(u, a, r) = True
     Effect: AA’ = AA \setminus \{(a, r)\}.
In large SDNs, specialized apps control/analyze and monitor/inspect specific network traffic type.

These apps should be authorized to access only traffic type they handle and not other type (via roles).

**Use Case using SDN-RBACca - Introduction**

### Apps

- **Web-specific apps:**
  - Web Load Balancers
  - Web Firewalls
  - etc.

- **VoIP-specific apps:**
  - VoIP Load Balancers
  - VoIP Firewalls
  - etc.

- **FTP-specific apps:**
  - FTP Load Balancers
  - FTP Firewalls
  - etc.

- **Email-specific apps:**
  - Email Load Balancers
  - Email Firewalls
  - etc.

### Roles

- **Web-specific roles:**
  - Web Flow Mod
  - Web Load Balancing
  - etc.

- **VoIP-specific roles:**
  - VoIP Flow Mod
  - VoIP Load Balancing
  - etc.

- **FTP-specific roles:**
  - Ftp Flow Mod
  - Ftp Load Balancing
  - etc.

- **Email-specific roles:**
  - Email Flow Mod
  - Email Load Balancing
  - etc.
In large SDNs, specialized apps control/analyze and monitor/inspect specific network traffic type.

These apps should be authorized to access only traffic type they handle and not other type (via roles).

### Apps
- **Web-specific apps:**
  - Web Load Balancers
  - Web Firewalls
  - etc.
- **VoIP-specific apps:**
  - VoIP Load Balancers
  - VoIP Firewalls
  - etc.
- **FTP-specific apps:**
  - FTP Load Balancers
  - FTP Firewalls
  - etc.
- **Email-specific apps:**
  - Email Load Balancers
  - Email Firewalls
  - etc.

### Roles
- **Flow Mod**
- **Load Balancing**
- etc.

@ Abdullah Al-Alaj
Relations between apps and roles should be managed by different administrative units.

**Administrative Units**

- **Web Admin Unit**
  - **Roles**: {Web Flow Mod, Web Load Balancing, etc.}
  - **App-Pools**: {Web Security, Web Load Balance, etc.}

- **Email Admin Unit**
  - **Roles**: {Email Flow Mod, Email Load Balancing, etc.}
  - **App-Pools**: {Email Security, Email Load Balance}

- **VoIP Admin Unit**
  - **Roles**: {Email Flow Mod, VoIP Load Balancing, etc.}
  - **App-Pools**: {VoIP Security, VoIP Load Balance}

- **FTP Admin Unit**
  - **Roles**: {FTP Mod Email, FTP Load Balancing, etc.}
  - **App-Pools**: {FTP Security, FTP Load Balance}
Custom and Proxy Operations

3. Create proxy operations

\[ \text{OP}_{\text{Proxy1}} \]
\[ \text{OP}_{\text{Proxy2}} \]
\[ \text{OP}_{\text{Proxy3}} \]

Proxy group

1. Clone operation

2. Refine (verify access to appropriate content)

3. Create proxy operations and call custom operation by passing actual value.

```
addWebFlow(...)
  call addFlow(..., web)

addVoIPFlow(...)
  call addFlow(..., voip)

addFtpFlow(...)
  call addFlow(..., ftp)
```

AddFlow

provides restrictive access to specific traffic type.

(ensure that flow rule handles correct traffic type)
Custom Permissions

• Custom permissions are those permissions that are created using proxy operations.

\[(\text{OP}_{\text{Proxy}_1}, \text{ot})\]
\[(\text{OP}_{\text{Proxy}_2}, \text{ot})\]
\[(\text{OP}_{\text{Proxy}_3}, \text{ot})\]
...

Examples:

\[(\text{addWebFlow}, \text{FLOW-RULE})\]
\[(\text{addVoIPFlow}, \text{FLOW-RULE})\]
\[(\text{addFtpFlow}, \text{FLOW-RULE})\]
\[(\text{createWebMember}, \text{LB-POOL-MEMBER})\]
\[(\text{createVoIPMember}, \text{LB-POOL-MEMBER})\]
\[(\text{createFtpMember}, \text{LB-POOL-MEMBER})\]
\[(\text{readWebPacketInPayload}, \text{PI-PAYLOAD})\]
\[(\text{readVoIPPacketInPayload}, \text{PI-PAYLOAD})\]
...

@ Abdullah Al-Alaj
Task and Role Engineering Custom Permissions

SDN Apps | Roles | Tasks  | Custom Permissions | OP\text{\small{Proxy}} | OP\text{\small{Custom}} | OP\text{\small{Target}}
---|---|---|---|---|---|---
App1 | r1 | t1 | p1 | x11 | val1 | clone | op1
   |   |   | p2 | x12 | val2 |   |   
   |   |   | p3 | x13 | val3 |   |   
App2 | r2 | t2 | p4 | x21 | val1 | clone | op2
   |   |   | p5 | x22 | val2 |   |   
   |   |   | p6 | x23 | val3 |   |   
App3 | r3 | t3 | p7 | x31 | val1 | clone | op3
   |   |   | p8 | x32 | val2 |   |   
   |   |   | p9 | x33 | val3 |   |   

Actual value passed to custom operation

@ Abdullah Al-Alaj
Task and Role Engineering using Custom Permissions - Example

SDN Apps
- Web Intrusion Prevention
- VoIP Load Balancer
- FTP Application Firewall

Roles
- Web Flow Mod
- VoIP Flow Mod
- FTP Flow Mod

Tasks
- Web Traffic Forwarding Task
- VoIP Traffic Forwarding Task
- FTP Traffic Forwarding Task
- Web Flow Viewing Task
- VoIP Flow Viewing
- FTP Flow Viewing Task

Custom Permissions
- (addWebFlow, FLOW-RULE)
- (addVoipFlow, FLOW-RULE)
- (addFtpFlow, FLOW-RULE)
- (deleteWebFlow, FLOW-RULE)
- (deleteVoipFlow, FLOW-RULE)
- (deleteFtpFlow, FLOW-RULE)
- (readWebFlow, FLOW-RULE)
- (readVoipFlow, FLOW-RULE)
- (readFtpFlow, FLOW-RULE)

OPProxy
- addWebFlow
- addVoipFlow
- addFtpFlow

OPCustom
- deleteWebFlow
- deleteVoipFlow
- deleteFtpFlow

OPTarget
- readWebFlow
- readVoipFlow
- readFtpFlow

OPProx
- addFlow
- passed to custom operation

OPCustom
- deleteFlow

OPTarget
- readFlow
Use-Case and Administrative Actions

Tasks, roles, and app-pools in white are exclusively managed by: **Web Admin Unit**
Tasks, roles, and app-pools in gray are exclusively managed by: **VoIP Admin Unit**

Administrative User Assignment:
\[ \text{TA\_admin} = \{ \text{(web\_functions\_admin\_user, Web Admin Unit), (voip\_functions\_admin\_user, VoIP Admin Unit)} \} \].

Example:

1. Administrative Action to assign task to a role:
   \[ \text{assign\_task\_to\_role(web\_functions\_admin\_user, Web Traffic Forwarding Task, Web Flow Mod)} \] is allowed.

   → Authorization Function:
   \[ \text{can\_manage\_task\_role(web\_functions\_admin\_user, Web Traffic Forwarding Task, Web Flow Mod)} = \text{True}. \]

   Reason:
   \[ \exists \text{Web Admin Unit} \in \text{AU} : ((\text{web\_functions\_admin\_user, Web Admin Unit}) \in \text{TA\_admin}) \land \text{Web Flow Mod} \in \text{roles(Web Admin Unit)} \land \text{Web Traffic Forwarding Task} \in \text{tasks(Web Admin Unit)}. \]
Evaluation and Comparison

- Evaluation of SDN-RBACa operational model with tasks and proxy permissions.
- Test app with 50 proxy operations ops covered by 10 different roles.
- Report authorization time for all 50 requests.
- Different security policies.
- Test repeated 100 times for each security policy.
- Average authorization time is calculated.

- Operational model of SDN-RBACa adds an average of 0.0252 ms overhead on the floodlight controller while SDN-RBAC adds 0.0245 ms on average.
- Using tasks in SDN-RBACa operational model introduces additional variance in the authorization check time.
- The operational model of SDN-RBACa introduces acceptable overhead to the controller for the sake of access control administration.
Conclusion and Future Work

• We presented SDN-RBAC, a model for enabling role based authorization for SDN. SDN-RBAC is implemented and enforced in Floodlight controller.

• We presented ParaSDN, a fine-Grained and Scalable Access Control for SDN Enhanced with Role and Permission Parameters. The Authorization Framework includes Parameter Engine and Enforcement in SDN Controller.

• We presented SDN-RBACa, an administrative model for SND enhanced with Proxy Operations and Custom Permissions.

Future Work:

• Access Control for SDN-Enabled technologies.
• Risk-Aware Access Control for SDN Apps.
Dissertation Publications

Published:


Submitted for review:


Thank you!

Questions?
Backup Slides
<table>
<thead>
<tr>
<th>Role</th>
<th>General Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Handler</td>
<td>permissions for querying the controller about devices</td>
</tr>
<tr>
<td>Bandwidth Monitoring</td>
<td>permissions to read the bandwidth consumption for switch ports.</td>
</tr>
<tr>
<td>Flow Mod</td>
<td>permissions to insert/update/delete flow rules into a switch’s flow tables.</td>
</tr>
<tr>
<td>Link Handler</td>
<td>permissions to get information about network links</td>
</tr>
<tr>
<td>Device Tracking</td>
<td>permissions to get notifications about changes on network devices (added, removed, Moved, Address Changed, etc.)</td>
</tr>
<tr>
<td>Port Handler</td>
<td>permissions to read information about ports and their status</td>
</tr>
<tr>
<td>Routing</td>
<td>permissions to get and compute routes between various source and destination nodes</td>
</tr>
</tbody>
</table>
SDN-RBAC - Check Access

request from session \( (se) \)

SDN-RBAC checkAccess

avail_session_perms

session_roles

assigned_perms

\[ \text{session_roles}(se) \]

\[ r_i \in \text{session_roles}(se) \]

\[ \text{assigned_perms}(r_i) \]

\[ \text{per p_i matches request?} \]

true/false

\[ \text{true/false} \]

\[ \text{true/false} \]
- \textit{dces} = DataCapEnforcingSession

SDN-RBAC App Authorization - Example

\begin{itemize}
  \item SDN-RBAC checkAccess
  \item avail\_session\_perms
  \item session\_roles
  \item assigned\_perms
\end{itemize}

request from session \((dces)\)

addFlow, \(fr_{i}\)

\(dces\)

\(dces\)

\{Flow Mod\}

\{(addFlow, FLOW-RULE), \ldots\}

\{(addFlow, FLOW-RULE), \ldots\}

matches request?

true

true

\(dces\)

\{(addFlow, FLOW-RULE), \ldots\}

\{(addFlow, FLOW-RULE), \ldots\}

true
## SDN-RBAC: Specifications of System Functions

### Session Creation/Deletion
- **Function**: `createSession(a : APPS, s : SESSIONS, ars : 2ROLES)`
- **Authorization Condition**: \( a r s \subseteq \{ r \in ROLES | (a, r) \in AR \} \land s \notin SESSIONS \)
- **Update**:  \( \forall s \in \text{app}\_sessions(a), \text{SESSIONS}' = \text{SESSIONS} \cup \{s\}, \text{app}\_sessions'(a) = \text{app}\_sessions(a) \cup \{s\}, \text{session}\_roles'(s) = ars \)

### Adding/Dropping Active Role
- **Function**: `deleteSession(a : APPS, s : SESSIONS)`
- **Authorization Condition**: \( s \in \text{app}\_sessions(a) \)
- **Update**:  \( \forall s \in \text{app}\_sessions(a), \text{SESSIONS}' = \text{SESSIONS} \setminus \{s\}, \text{app}\_sessions'(a) = \text{app}\_sessions(a) \setminus \{s\}, \text{SESSIONS}' = \text{SESSIONS} \setminus \{s\} \)

### Access Check
- **Function**: `addActiveRole(a : APPS, s : SESSIONS, r : ROLES)`
- **Authorization Condition**: \( s \in \text{app}\_sessions(a) \land (a, r) \in AR \land r \notin \text{session}\_roles(s) \)
- **Update**:  \( \forall s \in \text{app}\_sessions(a), \text{SESSIONS}' = \text{SESSIONS} \setminus \{s\}, \text{SESSIONS}' = \text{SESSIONS} \setminus \{s\} \land r \}

### Drop Active Role
- **Function**: `dropActiveRole(a : APPS, s : SESSIONS, r : ROLES)`
- **Authorization Condition**: \( s \in \text{app}\_sessions(a) \land r \in \text{session}\_roles(s) \)
- **Update**:  \( \forall s \in \text{app}\_sessions(a), \text{SESSIONS}' = \text{SESSIONS} \setminus \{s\}, \text{SESSIONS}' = \text{SESSIONS} \setminus \{s\} \land r \}

### Check Access
- **Function**: `checkAccess(s : SESSIONS, op : OPS, ob : OBS)`
- **Authorization Condition**: \( \exists r \in ROLES : r \in \text{session}\_roles(s) \land ((op, type(ob)), r) \in PR \)

Apps are authorized based on object type.
Methods for Inter-session Interaction for SDN-RBAC

Atomic sessions  Two sessions access shared data  Conditional session creation  Interaction via inter-session interaction APIs  Active role set sent from master to slave sessions

(a) App
   Session1

(b) App
   Session1  Session2
            r/w  r/w
 Internal shared data

(c) App
   Session1
   Session2
   c

(d) App
   Session1
   Session2
   l
   Session3

(e) App
   Session1
   Session2
   a
   a
   a

: creates a session (From the creator to the created session).
 accessibility: access shared data.
 : session interaction via session interaction API.
 w/r: read/write operation.
 c: condition that triggers session creation.
 l: session interaction API (managed by the system).
 a: active role set sent along with session creation request.
Session Handling Approaches

• Who is responsible of specifying:
  - (T) the **tasks** and corresponding sessions.
  - (C) the **condition** for session creation/deletion.
  - (A) the **active** role set.
  - (R) role to be added/dropped during execution.

**Session Handling Approaches**

- **Developer-driven Approach**
  - (T) DD
  - (C) DD
  - (A) DD
  - (R) DD

- **System-driven Approach**
  - (T) CR
  - (C) CR
  - (A) CR
  - (R) CR

- **Session-driven Approach**
  - (T) DD
  - (C) SR
  - (A) SR
  - (R) SR

DD = determined by Developer at Design-time.
CR = determined by Controller at Run-time.
SR = determined by Session at Run-time.

© Abdullah Al-Alaj
Developer-driven Session Handling (1)

• Developer has full and prior knowledge of
  • all possible sessions
  • active role set required for each session to achieve its task.
• This information is provided to the controller before app execution.
• The controller knows in advance:
  • what session instances will be created.
  • the tasks that will execute in each session.
  • active role set required for each session.
To show that SDN-RBAC authorization system can identify and reject any unauthorized operations:

- We forced `DataUsageAnalysisSession` to read link information via operation `getAllLinks`.
- The permission `(getAllLinks, LINK)` is assigned to the role `LinkHandler`.
- Role `LinkHandler` is not a member of the active role set of `DataUsageAnalysisSession`.
- A snapshot of the execution result is shown below.

```
The method net.floodlightcontroller.topology.ITopologyService.getAllLinks
is called by session net.floodlightcontroller.datausagemngr.DataUsageAnalysisSession
Unauthorized access requested by session (DataUsageAnalysisSession)
Reason: None of session active roles contains a corresponding permission
Active roles set for this session: [Device Handler, Bandwidth Monitoring]
```

Snapshot of authorization check result for `getAllLinks()` operation requested by `DataUsageAnalysisSession` - Access Denied.
We forced DataUsageAnalysisSession to read device statistics via operation getBandwidthConsumption.

The permission (getBandwidthConsumption, PORT-STATS) is assigned to the role BandwidthMonitoring.

Role BandwidthMonitoring is a member of the active role set of DataUsageAnalysisSession.

A snapshot of the execution result is shown below.

The snapshot below shows how DataUsageAnalysisSession was able to pass the authorization.

```
The method net.floodlightcontroller.statistics.IStatisticsService.getBandwidthConsumption
is called by session net.floodlightcontroller.datausage.DataUsageAnalysisSession
```

Snapshot of authorization check result for getBandwidthConsumption() operation requested by DataUsageAnalysisSession - Access Granted.
Parameter Checking Functions

A. Verifiers:
Language LVerify is used to define each verifier $V_i(s$: SESSIONS, $op$: OPS, $ob$: OBS, pvpair : PVPairs) in VERIFIERS.

B. CandidateVerifiers: a function that maps each object type to its applicable set of verifiers.
CandidateVerifiers($ot$: OBTS, pvpairs : $2^{PVPairs}$)

\[
\begin{align*}
\text{verifiers} &= \{\}\; ; \\
\text{For each } pvpair_i \in pvpairs & \text{ do} \\
& \quad V_i = \text{param\_verifier}(ot, pvpair_i, \text{par}) \\
& \quad \text{verifiers} := \text{verifiers} \cup \{(V_i \times pvpair_i)\}; \\
& \quad \text{return } \text{verifiers}; 
\end{align*}
\]

C. ParamCheck: a function that checks an object against all candidate verifiers until the first failure is discovered or a true is returned as the final outcome.
ParamCheck($s$: SESSIONS, $op$: OPS, $ob$: OBS, pvpairs : $2^{PVPairs}$)

\[
\begin{align*}
\text{For each } (V_i \times pvpair_i) \in \text{CandidateVerifiers}(type(ob), pvpairs) & \text{ do} \\
& \quad \text{if } \neg V_i(s, op, ob, pvpair_i) \\
& \quad \quad \text{return } \text{false}; \\
& \quad \text{return } \text{true}; 
\end{align*}
\]
## App Authorization Function

<table>
<thead>
<tr>
<th>Function</th>
<th>Authorization Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkAccess(s: SESSIONS, op: OPS, ob: OBS)</td>
<td>$\exists pr \in PROLES : pr \in session_roles(s), \exists pp \in PPRMS : (pp, pr) \in PPA \land (op, type(ob)) = (pp.op, pp.ot) \land ParamCheck(s, op, ob, pp.PVPAIRS) = True.$</td>
</tr>
</tbody>
</table>
A.3. VRULESWITCH(s: SESSIONS, op: OPS, ob: OBS, pvpair : PVPAIRS) {
    // assume a request from app Data Usage Cap Mngr via DataCapEnforcingSession with the following:
    // ob = flow_rule[switch_id=0x2,tcp_dst=80,...]
    // pvpair = (dept, {CS})
    // switches(CS) = {0x1, 0x2}
    (\exists d \in pvpair.val : ob.switch_id \in switches(d)); // will return true
}
Parameter Value Assignment

- Parameter values, assigned via assignApp administrative action, propagate automatically from role parameters to permission parameters.

Example:

\[ \text{assignPPerm}(pp, pr) \]

\[ \text{assignApp(a = Data Usage Cap Mngr, pr, } \{ \text{CS}, \text{ web} \}) \]

\[ \text{assignApp(a = Data Usage Cap Mngr, pr, } \{ \text{CS}, \text{ web} \}) \]

\[ \frac{pr}{ (\text{Flow Mod, } \{(\text{dept}, \bot), (\text{traffic}, \bot)\}) } \]

\[ \frac{pp}{ (\text{Flow Mod, } \{(\text{dept}, \bot), (\text{traffic}, \bot)\}) } \]

1. Parameterized permission assigned to parameterized role. Parameter values are unknown.

2. Parameter value assigned to parameterized role via assignApp administrative action.

3. Parameter values propagate from parameterized role to parameterized permission.
ParaSDN: App Authorization Example

- `checkAccess(DataCapEnforcingSession, addFlow, flow_rule[switch_id=0x2, tcp_dst=80, ...])`\
  \[\equiv\]\
  `\exists (Flow Mod, ((dept, {CS}), (traffic, web))) \in PROLES :`\
  `\exists ((addFlow, FLOW-RULE), ((dept, {CS}), (traffic, web))) \in PPRMS :`\
  `\land`\
  `ParamCheck(DataCapEnforcingSession, addFlow, flow_rule[switch_id=0x2, tcp_dst=80, ...], ((dept, {CS}), (traffic, web))) = True.`

\[\text{PPA} = \{\ldots, ((addFlow, FLOW-RULE), ((dept, {CS}), (traffic, web))), (Flow Mod, ((dept, {CS}), (traffic, web))), \ldots\} \]
Parameter Engine - Use Case Example

DataCapEnforcingSession,
addFlow,
flow_rule\{switch\_id=0x2, tcp\_dst=80,...\},
{(dept, \{CS\}), (traffic, web)}

\text{type(addFlow, flow\_rule\{switch\_id=0x2, tcp\_dst=80,...\}),}
{(dept, \{CS\}), (traffic, web)}

\{VRuleSwitch, VRuleTraffic\}

\text{flow\_rule\{switch\_id=0x2, tcp\_dst=80,...\},}
(dept, \{CS\})

\text{true/false}

\text{VRuleSwitch}

\text{\{Id \in pvpair.val : flow\_rule.switch\_id \in switches(d))\};}

\text{true/false}

\text{flow\_rule\{switch\_id=0x2, tcp\_dst=80,...\},}
(traffic, web)

\text{VRuleTraffic}

\text{\{flow\_rule.tcp\_dst \in protocol\_ports(pvpair.val)\}}

\text{true/false}

\text{true}

\text{true/false}

\text{true/false}

\text{true/false}

\text{\text{verifier's result = false}}

\text{\text{true}}
Administrative Actions - Examples

1. Examples of Authorization Functions:
   - can_manage_task_role(web_functions_admin_user, Web Traffic Forwarding Task, Web Flow Mod) = True
     Reason:
     $\exists$ Web Admin Unit $\in$ AU : $\{(web\_functions\_admin\_user, Web\ Admin\ Unit) \in TA\_admin\} \land$
     Web Flow Mod $\in$ roles(Web Admin Unit) $\land$
     Web Traffic Forwarding Task $\in$ tasks(Web Admin Unit).
   - can_manage_task_role(voip_functions_admin_user, Web Server Pool Management Task, Web Load Balancing) = False
     Reason:
     Web Load Balancing $\in$ roles(Web Admin Unit) $\land$
     Web Server Pool Management Task $\in$ tasks(Web Admin Unit),
     however, (voip_functions_admin_user, Web Admin Unit) $\notin$ TA_admin.
   - can_manage_app_role(web_apps_admin_user, Web Intrusion Prevention App, Web Flow Mod) = True
     Reason:
     $\exists$ Web Admin Unit $\in$ AU : $\{(web\_apps\_admin\_user, Web\ Admin\ Unit) \in AA\_admin\} \land$
     Web Flow Mod $\in$ roles(Web Admin Unit) $\land$
     $\exists$ Web Security Pool $\in$ AP: (Web Intrusion Prevention App, Web Security Pool) $\in$ AAPA $\land$
     Web Security Pool $\in$ app_pools(Web Admin Unit).
   - can_manage_app_role(web_apps_admin_user, VoIP Application Firewall App, VoIP Flow Mod) = False
     Reason:
     VoIP Flow Mod $\in$ roles(VoIP Admin Unit) $\land$
     (VoIP Application Firewall App, VoIP Security) $\in$ AAPA $\land$ VoIP Security $\in$ app_pools(VoIP Admin Unit),
     however, (web_apps_admin_user, VoIP Admin Unit) $\notin$ AA_admin.

2. Examples of Administrative Actions:
   - assign_task_to_role(web_functions_admin_user, Web Traffic Forwarding Task, Web Flow Mod) is allowed
     Reason:
     can_manage_task_role(web_functions_admin_user, Web Traffic Forwarding Task, Web Flow Mod) = True
   - revoke_task_from_role(voip_functions_admin_user, Web Server Pool Management Task, Web Load Balancing) is not allowed
     Reason:
     can_manage_task_role(voip_functions_admin_user, Web Server Pool Management Task, Web Load Balancing) = False
   - assign_app_to_role(web_apps_admin_user, Web Intrusion Prevention App, Web Flow Mod) is allowed
     Reason:
     can_manage_app_role(web_apps_admin_user, Web Intrusion Prevention App, Web Flow Mod) = True
   - revoke_app_from_role(web_apps_admin_user, VoIP Application Firewall App, VoIP Flow Mod) is not allowed
     Reason:
     can_manage_app_role(web_apps_admin_user, VoIP Application Firewall App, VoIP Flow Mod) = False

Administrative User Assignment:
   - TA_admin = \{(web\_functions\_admin\_user, Web\ Admin\ Unit), (voip\_functions\_admin\_user, VoIP\ Admin\ Unit)\}.
   - AA_admin = \{(web\_apps\_admin\_user, Web\ Admin\ Unit), (voip\_apps\_admin\_user, VoIP\ Admin\ Unit)\}.  

62
# Configuration of the SDN-RBACa in a use case (1)

1. Basic Sets

- \( \text{APPS} = \{ \text{Web Intrusion Prevention App, Web Application Firewall App, Web Load Balancer App} \} \)
- \( \text{OBS} = \{ \text{PI-PAYLOAD, PI-HEADER, FLOW-RULE, LB-POOL, LB-MONITOR, LB-VIP, LB-POOL-MEMBER, and FLOW-STATS} \} \)
- \( \text{OBTS} = \{ \text{PI-PAYLOAD, PI-HEADER, FLOW-RULE, LB-POOL, LB-MONITOR, LB-VIP, LB-POOL-MEMBER, FLOW-STATS} \} \)
- \( \text{ROLES} = \{ \text{Web Packet-In Handler, Web Packet Monitor, Web Flow Mod, Web Load Balancing, Web Stats Collector} \} \)
- \( \text{AP} = \{ \text{Web Load Balance Pool, Web Security Pool} \} \)
- \( \text{USERS} = \{ \text{web_functions_admin_user, web_apps_admin_user} \} \)
- \( \text{AU} = \{ \text{Web Admin Unit} \} \)
Configuration of the SDN-RBACa in a use case (2)

2. Assignment Relations (operational):

- PA =
  
  \{ (readWebPacketInPayload, PI-PAYLOAD), (readWebPacketHeader, PI-HEADER),
  (readWebFlow, FLOW-RULE) \times \{ Web Deep Packet Inspection Task \} \cup
  (readWebPacketHeader, PI-HEADER), (readWebFlow, FLOW-RULE) \times
  \{ Web Packet Header Inspection Task \} \cup
  \{ (readWebFlow, FLOW-RULE) \times \{ Web Flow Viewing Task \} \cup
  \{ addWebFlow, FLOW-RULE \}, updateWebFlow, FLOW-RULE),
  deleteWebFlow, FLOW-RULE) \times
  \{ Web Traffic Forwarding Task \} \cup
  \{ createWebPool, LB-POOL\}, \{ listWebPools, LB-POOL\}, \{ removeWebPool, LB-POOL\},
  \{ updateWebPool, LB-POOL\} \times \{ Web Server Pool Management Task \} \cup
  \{ createWebMonitor, LB-MONITOR\}, \{ listWebMonitors, LB-MONITOR\},
  \{ removeWebMonitor, LB-MONITOR\}, \{ updateWebMonitor, LB-MONITOR\} \times
  \{ Web Server Monitor Management Task \} \cup
  \{ createWebVip, LB-VIP\}, \{ listWebVips, LB-VIP\}, \{ removeWebVip, LB-VIP\},
  \{ updateWebVip, LB-VIP\} \times \{ Web Pool VIP Management Task \} \cup
  \{ createWebMember, LB-POOL-MEMBER\}, \{ listWebMembersByPool, LB-POOL-MEMBER\},
  \{ removeWebMember, LB-POOL-MEMBER\}, \{ updateWebMember, LB-POOL-MEMBER\} \times
  \{ Web Pool Member Management Task \} \cup
  \{ createWebFlowByteCount, FLOW-STATS\}, \{ readAggWebFlowByteCount, FLOW-STATS\} \times
  \{ Web Payload Statistics Collection Task \} \cup
  \{ readWebFlowPacketCount, FLOW-STATS\}, \{ readAggWebFlowPacketCount, FLOW-STATS\} \times
  \{ Web Packet Statistics Collection Task \} \}.

- TA = \{ (Web Deep Packet Inspection Task, Web Packet Header Inspection Task) \times \{ Web Packet-In Handler \} \cup
  \{ Web Packet Header Inspection Task \} \times \{ Web Packet Monitor \} \cup
  \{ Web Flow Viewing Task, Web Traffic Forwarding Task \} \times \{ Web Flow Mod \} \cup
  \{ Web Server Pool Management Task, Web Server Monitor Management Task, Web Pool VIP Management Task,
  Web Pool Member Management Task \} \times \{ Web Load Balancing \} \cup
  \{ Web Payload Statistics Collection Task, Web Packet Statistics Collection Task \} \times \{ Web Stats Collector \} \}.

- AA = \{ (Web Intrusion Prevention App) \times \{ Web Packet-In Handler, Web Flow Mod \} \cup
  \{ Web Application Firewall App \} \times \{ Web Packet Monitor, Web Flow Mod \} \cup
  \{ Web Load Balancer App \} \times \{ Web Flow Mod, Web Load Balancing, Web Stats Collector \} \}.

- OT = \{ (all payloads in packet-in message, PI-PAYLOAD), (all packet header objects, PI-HEADER),
  (all flow rules, FLOW-RULE), (all server pools, LB-POOL), (all server monitors, LB-MONITOR),
  (all pool virtual IPs, LB-VIP), (all pool members, LB-POOL-MEMBER),
  (all flow statistics in flow rules, FLOW-STATS) \}.

3. Administrative App-pools Relation:

- AAAA = \{ (Web Intrusion Prevention App, Web Security Pool),
  (Web Application Firewall App, Web Security Pool),
  (Web Load Balancer App, Web Load Balance Pool) \}.

4. Administrative Units and Partitioned Assignment:

- roles(Web Admin Unit) = \{ Web Packet-In Handler, Web Packet Monitor, Web Flow Mod,
  Web Load Balancing, Web Stats Collector \}.

- tasks(Web Admin Unit) = \{ Web Deep Packet Inspection Task, Web Packet Header Inspection Task,
  Web Flow Viewing Task, Web Traffic Forwarding Task, Web Server Pool Management Task,
  Web Server Monitor Management Task, Web Pool VIP Management Task, Web Pool Member Management Task,
  Web Payload Statistics Collection Task, Web Packet Statistics Collection Task \}.

- app_pools(Web Admin Unit) = \{ Web Load Balance Pool, Web Security Pool \}.

5. Administrative User Assignment:

- TA_admin = \{ web_functions_admin_user, Web Admin Unit \}.

- AA_admin = \{ web_apps_admin_user, Web Admin Unit \}.

@ Abdullah Al-Alaj

UTSA

Computer Science
The method net.floodlightcontroller.staticflowentry.IStaticFlowEntryPusherService.addWebFlow is called by session net.floodlightcontroller.webtestapp.WebTestAppSession.
Active roles set for this session: [Web Flow Mod]
01:34:14.824 INFO [n.f.l.i.LinkDiscoveryManager:Scheduled-3] Sending LLDP packets out of all