A Model for the Administration of Access Control in Software Defined Networking using Custom Permissions

Abdullah Al-Alaj[1], Ravi Sandhu[1] and Ram Krishnan[2]

[1]Dept. of Computer Science
[2]Dept. of Electrical and Computer Engineering
[1,2]Institute for Cyber Security
[1,2]Center for Security and Privacy Enhanced Cloud Computing (C-SPECC)
University of Texas at San Antonio, TX 78249

The Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA 2020)
December 2, 2020

Agenda

- Motivation
- Access Control Administration in SDN
- SDN-RBACa Administrative Model
- Custom and Proxy Operations
- Custom Permissions
- Task and Role Engineering Custom Permissions
- Use-Case and Administrative Actions
- Evaluation and Comparison
- Conclusion and Future Work

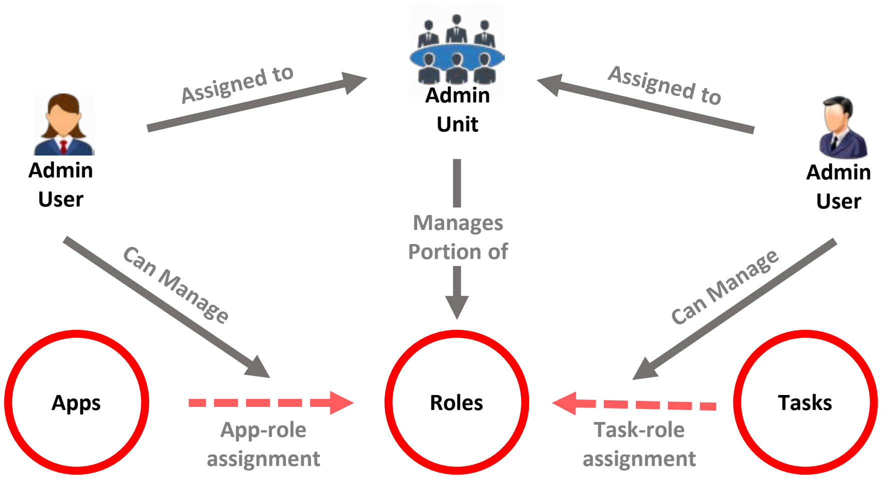
Motivation

- RBAC has been applied in SDN.
- RBAC simplifies the administration of authorizations.
- Currently, role-based approaches for SDN are lacking such administrative model.

- Operations provided by SDN services are coarse grained.
- Extend the capabilities of SDN services and provide fine grained custom permissions.

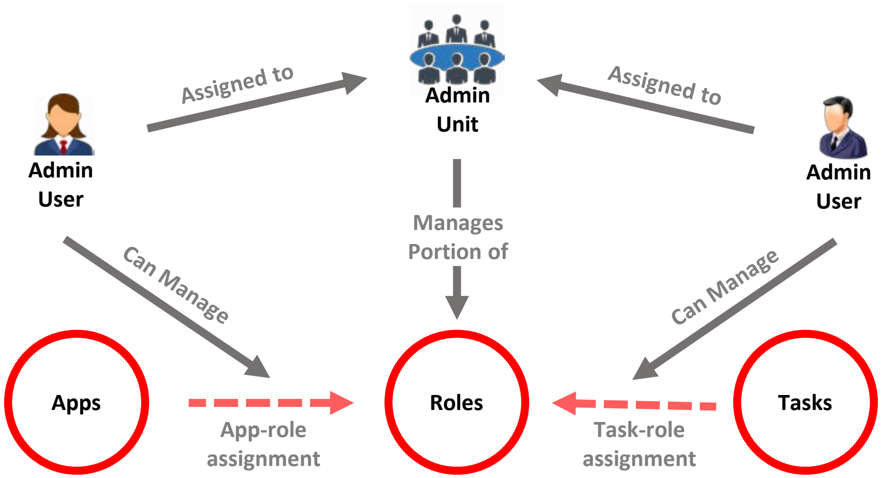
Administrative Units in SDN

- Small SDN networks could be managed by a single administrator or a single admin unit.
- Larger SDNs become more complex to centrally manage all access control associations by a single administrative authority.
- Administration has to be decentralized into multiple AUs.



Access Control Administration in SDN

- App-role and permission-role relations need management.
- In SDN-RBACa administrative model.
  - 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.



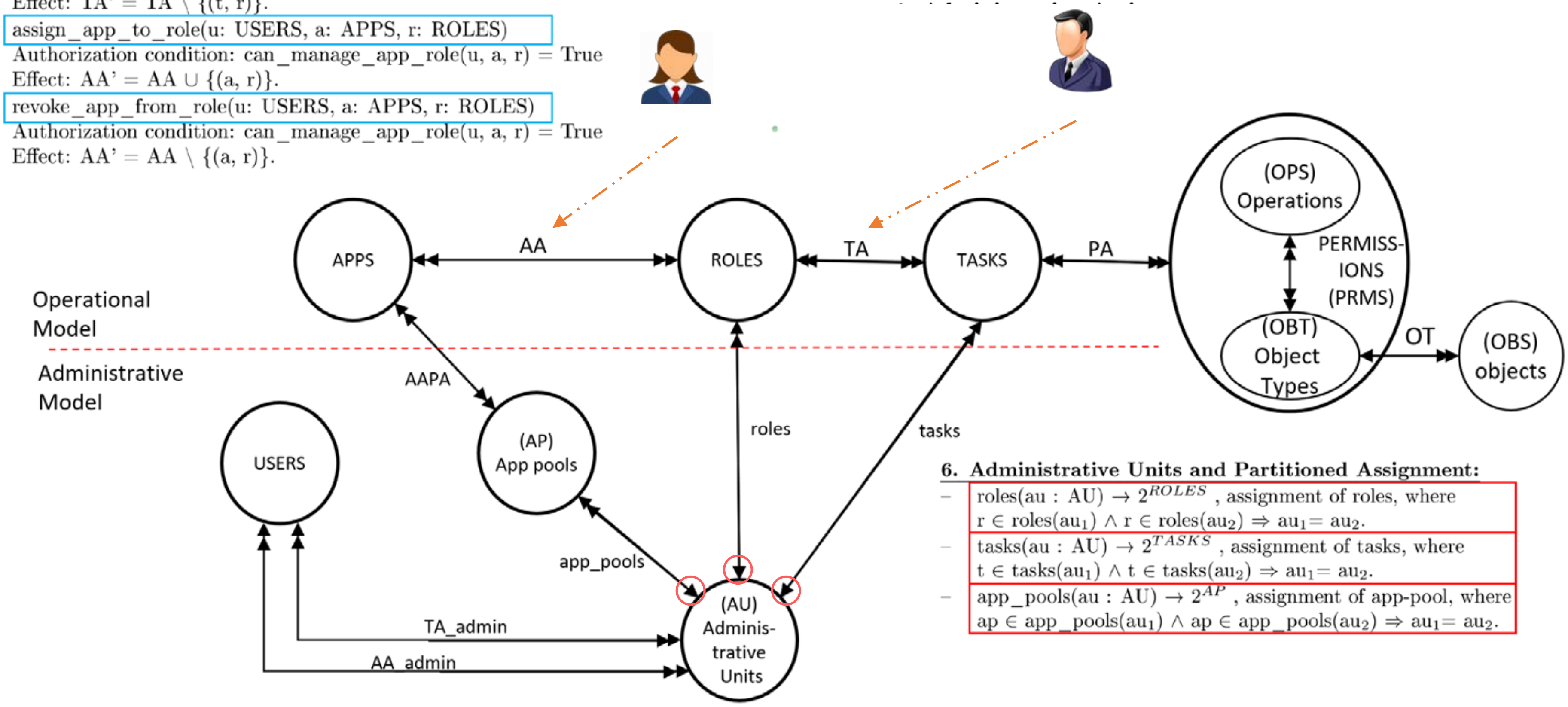
SDN-RBACa Administrative Model (Conceptual)

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: $AA' = AA \setminus \{(a, r)\}$.

8. **Administrative User Authorization Functions:**

- can_manage_task_role(u : USERS, t : TASKS, r : ROLES) = $\exists au \in AU : (u, au) \in TA\_admin \wedge r \in roles(au) \wedge t \in tasks(au)$.
- can_manage_app_role(u : USERS, a : APPS, r : ROLES) = $\exists au \in AU : ((u, au) \in AA\_admin \wedge r \in roles(au)) \wedge$ $\exists ap \in AP : ((a, ap) \in AAPA \wedge ap \in app\_pools(au))$.



6. **Administrative Units and Partitioned Assignment:**

- $roles(au : AU) \rightarrow 2^{ROLES}$, assignment of roles, where $r \in roles(au_1) \wedge r \in roles(au_2) \Rightarrow au_1 = au_2$.
- $tasks(au : AU) \rightarrow 2^{TASKS}$, assignment of tasks, where $t \in tasks(au_1) \wedge t \in tasks(au_2) \Rightarrow au_1 = au_2$.
- $app\_pools(au : AU) \rightarrow 2^{AP}$, assignment of app-pool, where $ap \in app\_pools(au_1) \wedge ap \in app\_pools(au_2) \Rightarrow au_1 = au_2$.

7. **Administrative User Assignment:**

- $TA\_admin \subseteq USERS \times AU$.
- $AA\_admin \subseteq USERS \times AU$.

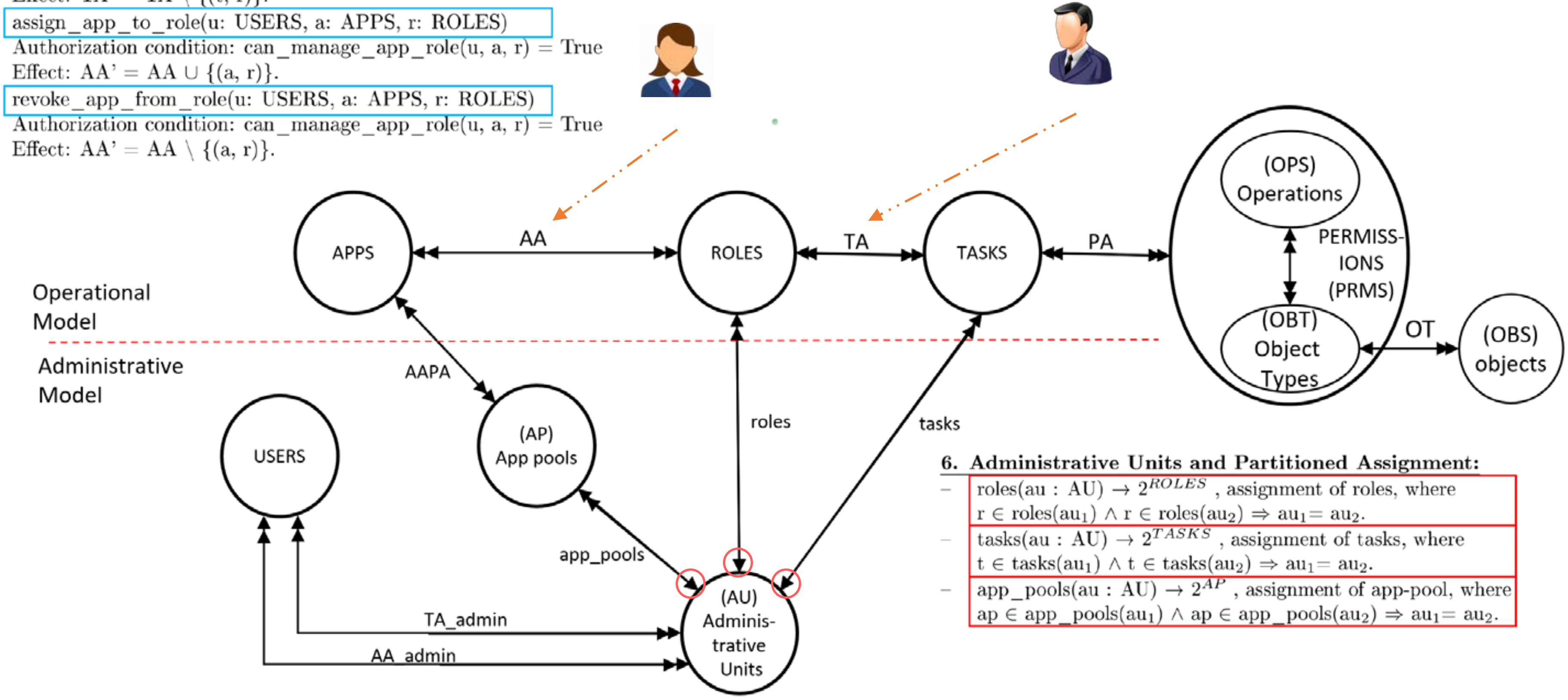
SDN-RBACa Administrative Model (Conceptual)

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)}.

8. Administrative User Authorization Functions:

- can_manage_task_role(u : USERS, t : TASKS, r : ROLES) = $\exists$au$\in$AU : (u , au) $\in$ TA_admin $\bigwedge$ r $\in$ roles(au) $\bigwedge$ t $\in$ tasks(au).
- can_manage_app_role(u : USERS, a : APPS, r : ROLES) = $\exists$au$\in$AU : ((u, au) $\in$ AA_admin $\bigwedge$ r $\in$ roles(au)) $\bigwedge$ $\exists$ap$\in$AP : ((a , ap) $\in$ AAPA $\bigwedge$ ap $\in$ app_pools(au)).



6. Administrative Units and Partitioned Assignment:

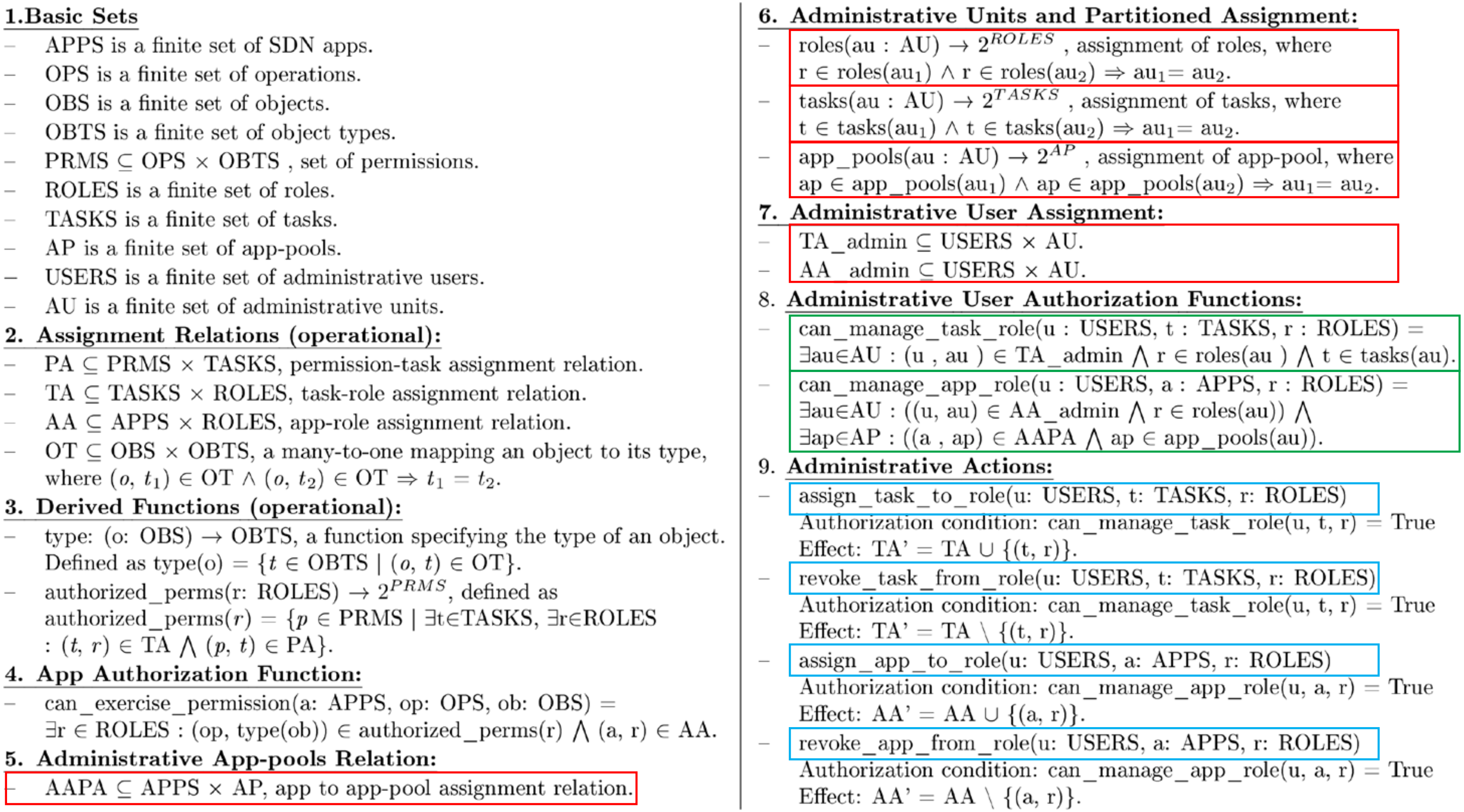
- roles(au : AU) $\rightarrow 2^{ROLES}$, assignment of roles, where r $\in$ roles(au$_1$) $\wedge$ r $\in$ roles(au$_2$) $\Rightarrow$ au$_1$= au$_2$.
- tasks(au : AU) $\rightarrow 2^{TASKS}$, assignment of tasks, where t $\in$ tasks(au$_1$) $\wedge$ t $\in$ tasks(au$_2$) $\Rightarrow$ au$_1$= au$_2$.
- app_pools(au : AU) $\rightarrow 2^{AP}$, assignment of app-pool, where ap $\in$ app_pools(au$_1$) $\wedge$ ap $\in$ 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 Defenition



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 $\subseteq$ OPS $\times$ 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 $\subseteq$ PRMS $\times$ TASKS, permission-task assignment relation.
- TA $\subseteq$ TASKS $\times$ ROLES, task-role assignment relation.
- AA $\subseteq$ APPS $\times$ ROLES, app-role assignment relation.
- OT $\subseteq$ OBS $\times$ OBTS, a many-to-one mapping an object to its type, where $(o, t_1) \in \mathrm{OT} \wedge (o, t_2) \in \mathrm{OT} \Rightarrow t_1 = t_2$.

3. Derived Functions (operational):

- type: (o: OBS) $\rightarrow$ OBTS, a function specifying the type of an object. Defined as type(o) $= \{t \in \mathrm{OBTS} \mid (o, t) \in \mathrm{OT}\}$.
- authorized_perms(r: ROLES) $\rightarrow 2^{PRMS}$, defined as authorized_perms$(r) = \{p \in \mathrm{PRMS} \mid \exists$t$\in$TASKS, $\exists$r$\in$ROLES $: (t, r) \in \mathrm{TA} \bigwedge (p, t) \in \mathrm{PA}\}$.

4. App Authorization Function:

- can_exercise_permission(a: APPS, op: OPS, ob: OBS) = $\exists$r $\in$ ROLES : (op, type(ob)) $\in$ authorized_perms(r) $\bigwedge$ (a, r) $\in$ AA.

5. Administrative App-pools Relation:

- AAPA $\subseteq$ APPS $\times$ AP, app to app-pool assignment relation.

6. Administrative Units and Partitioned Assignment:

- roles(au : AU) $\rightarrow 2^{ROLES}$, assignment of roles, where r $\in$ roles(au$_1$) $\wedge$ r $\in$ roles(au$_2$) $\Rightarrow$ au$_1$= au$_2$.
- tasks(au : AU) $\rightarrow 2^{TASKS}$, assignment of tasks, where t $\in$ tasks(au$_1$) $\wedge$ t $\in$ tasks(au$_2$) $\Rightarrow$ au$_1$= au$_2$.
- app_pools(au : AU) $\rightarrow 2^{AP}$, assignment of app-pool, where ap $\in$ app_pools(au$_1$) $\wedge$ ap $\in$ 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.

8. Administrative User Authorization Functions:

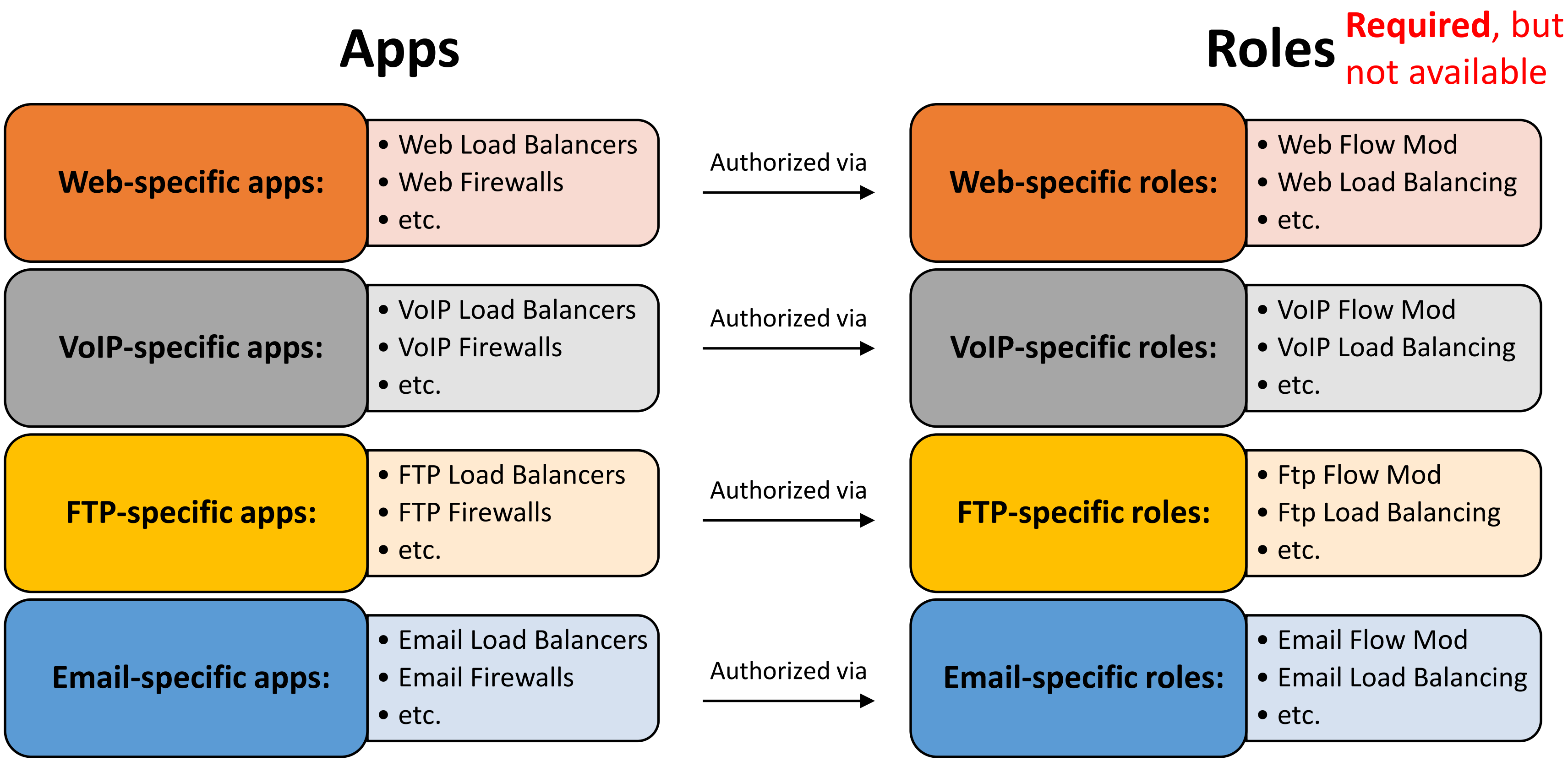
- can_manage_task_role(u : USERS, t : TASKS, r : ROLES) = $\exists$au$\in$AU : (u , au) $\in$ TA_admin $\bigwedge$ r $\in$ roles(au) $\bigwedge$ t $\in$ tasks(au).
- can_manage_app_role(u : USERS, a : APPS, r : ROLES) = $\exists$au$\in$AU : ((u, au) $\in$ AA_admin $\bigwedge$ r $\in$ roles(au)) $\bigwedge$ $\exists$ap$\in$AP : ((a , ap) $\in$ AAPA $\bigwedge$ ap $\in$ 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)}.

Use Case using SDN-RBACa -
Motivation

- 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

Roles **Required**, but not available

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

Authorized via

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

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

Authorized via

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

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

Authorized via

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

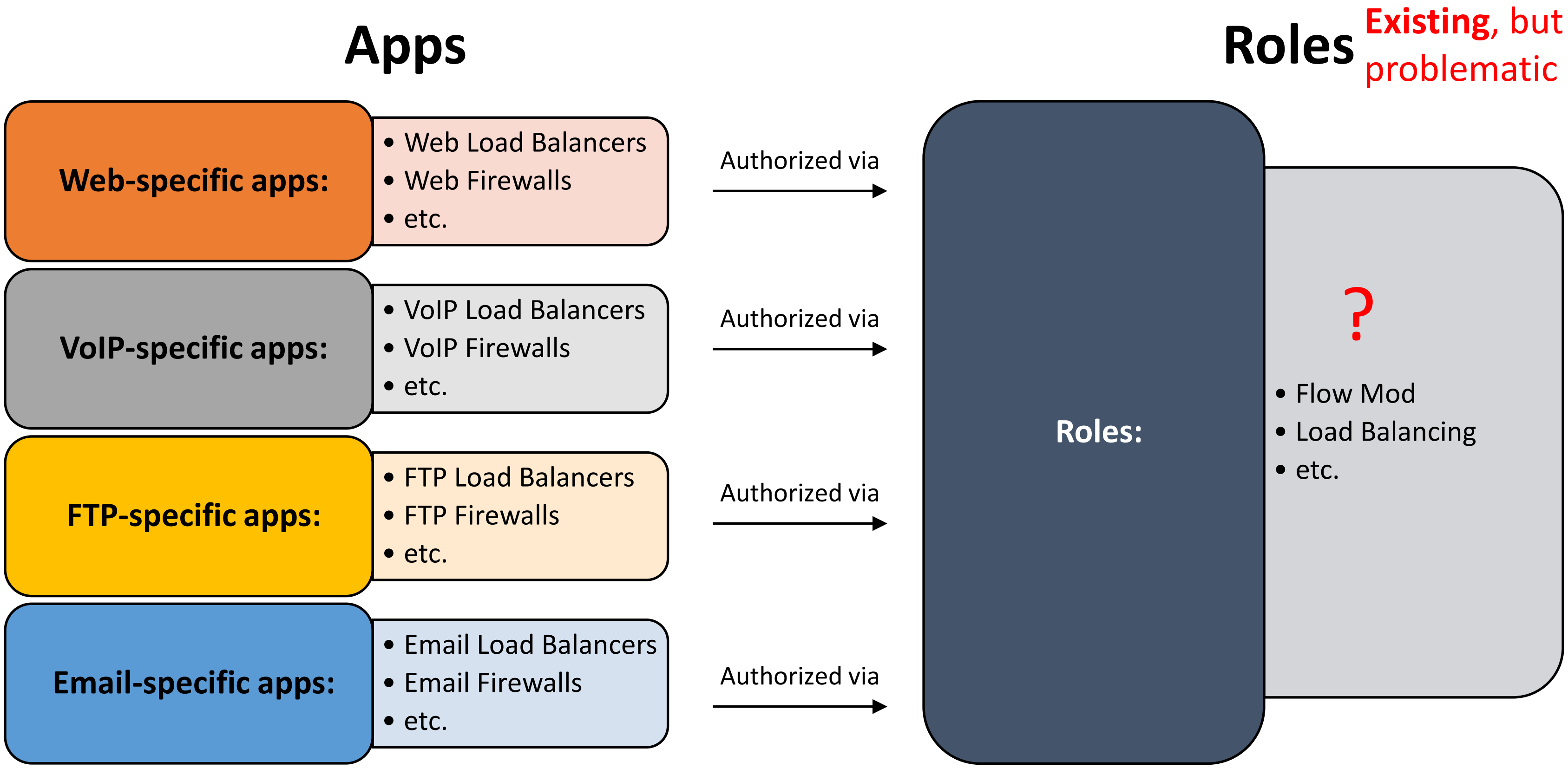
Email-specific apps:
- Email Load Balancers
- Email Firewalls
- etc.

Authorized via

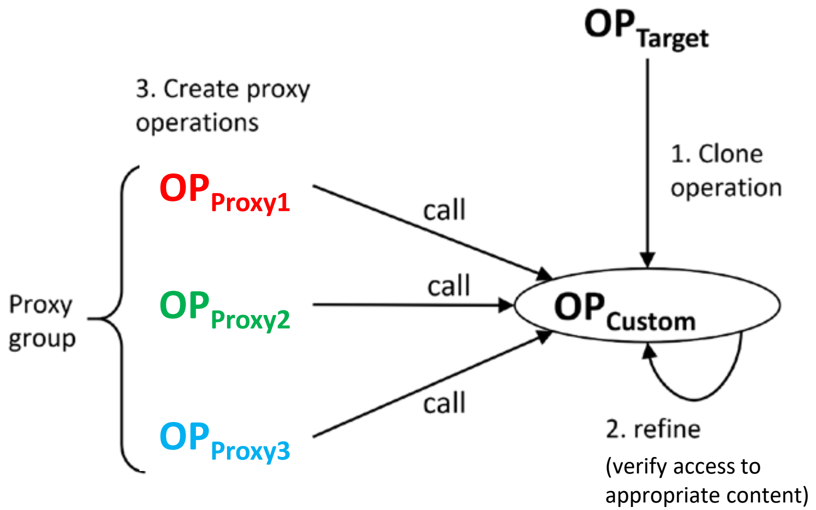
Email-specific roles:
- Email Flow Mod
- Email Load Balancing
- etc.

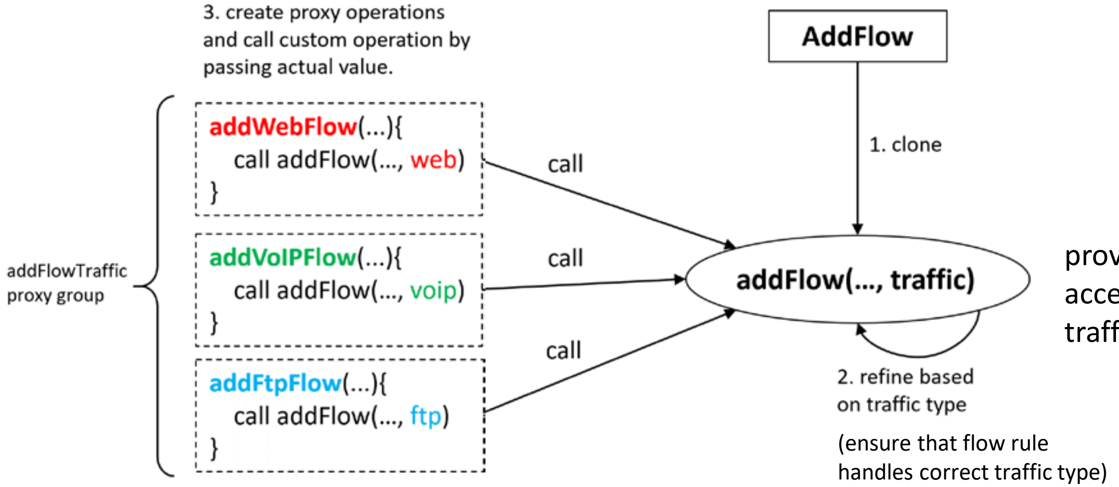
Use Case using SDN-RBACa -
Motivation

- 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).



Custom and Proxy Operations




3. create proxy operations and call custom operation by passing actual value.
AddFlow
1. clone
addFlowTraffic proxy group
call
call
call
addFlow(..., traffic)
2. refine based on traffic type
(ensure that flow rule handles correct traffic type)

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

```
addVoIPFlow(...){
   call addFlow(..., voip)
}
```

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

provides restrictive access to specific traffic type.

Custom Permissions

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

(OP_{Proxy_1}, ot)
(OP_{Proxy_2}, ot)
(OP_{Proxy_3}, ot)
…

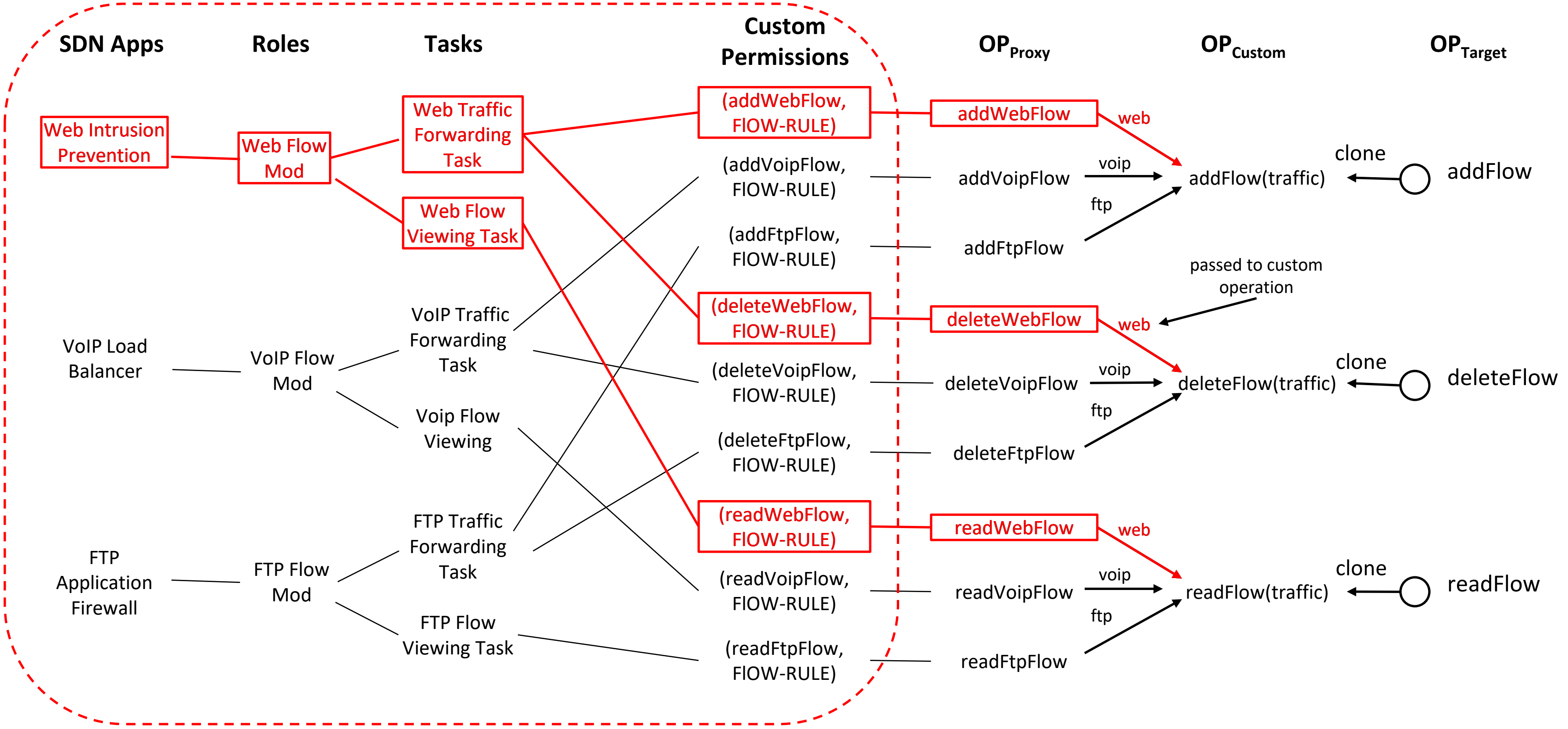
A permission
created using target operation.

(addFlow, FLOW-RULE)

Custom permissions
created using proxy operations:

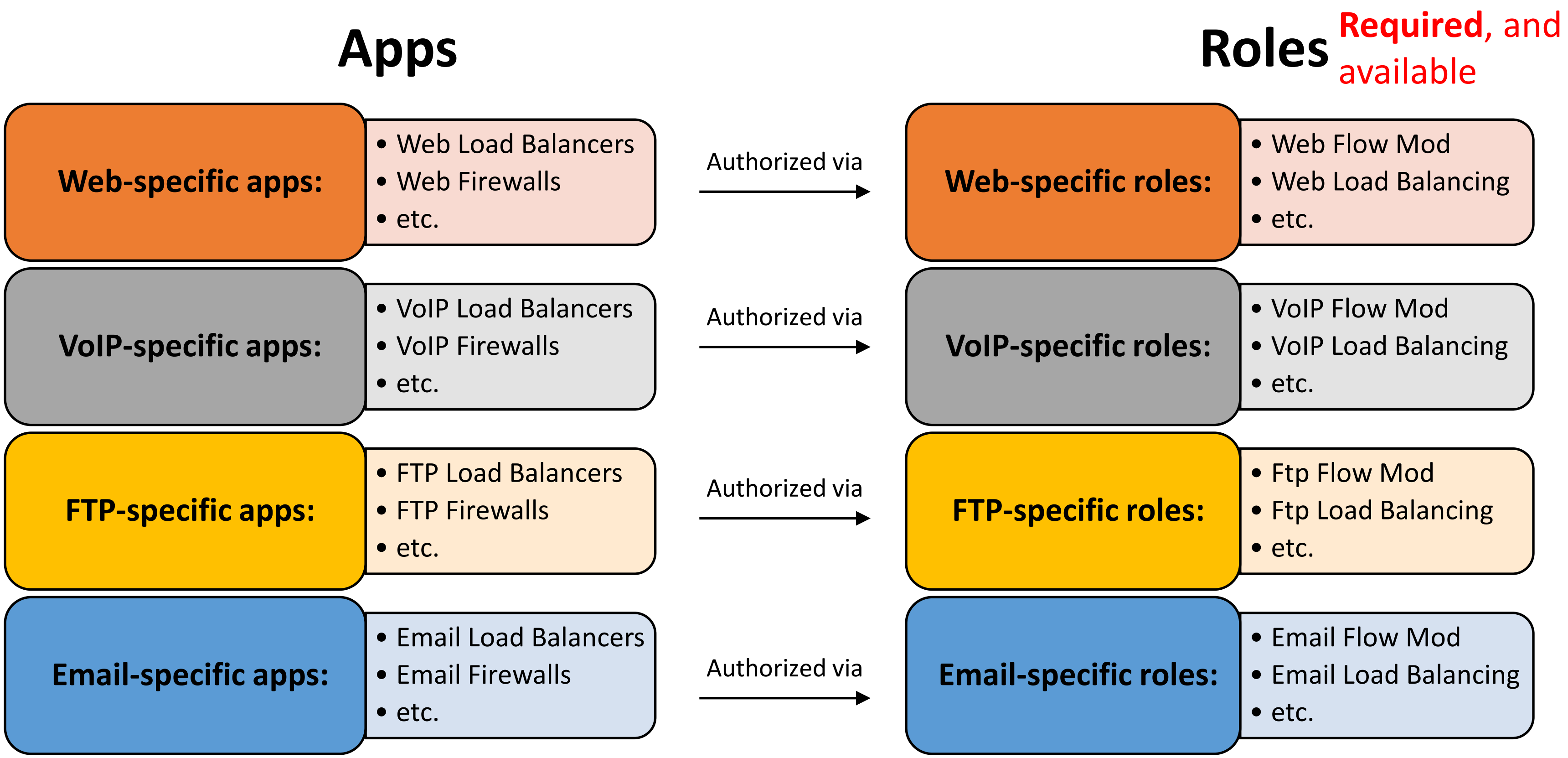
(addWebFlow, FLOW-RULE)
(addVoIPFlow, FLOW-RULE)
(addFtpFlow, FLOW-RULE)
(createWebMember, LB-POOL-MEMBER)
(createVoIPMember, LB-POOL-MEMBER)
(createFtpMember, LB-POOL-MEMBER)
(readWebPacketInPayload, PI-PAYLOAD)
(readVoIPPacketInPayload, PI-PAYLOAD)
…

Task and Role Engineering using Custom Permissions - Example



Roles Engineered after using Custom Permissions

- 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

Roles **Required**, and available

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

Authorized via →

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

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

Authorized via →

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

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

Authorized via →

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

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

Authorized via →

Email-specific roles:
- Email Flow Mod
- Email Load Balancing
- etc.

Functional Administrative Units for SDN

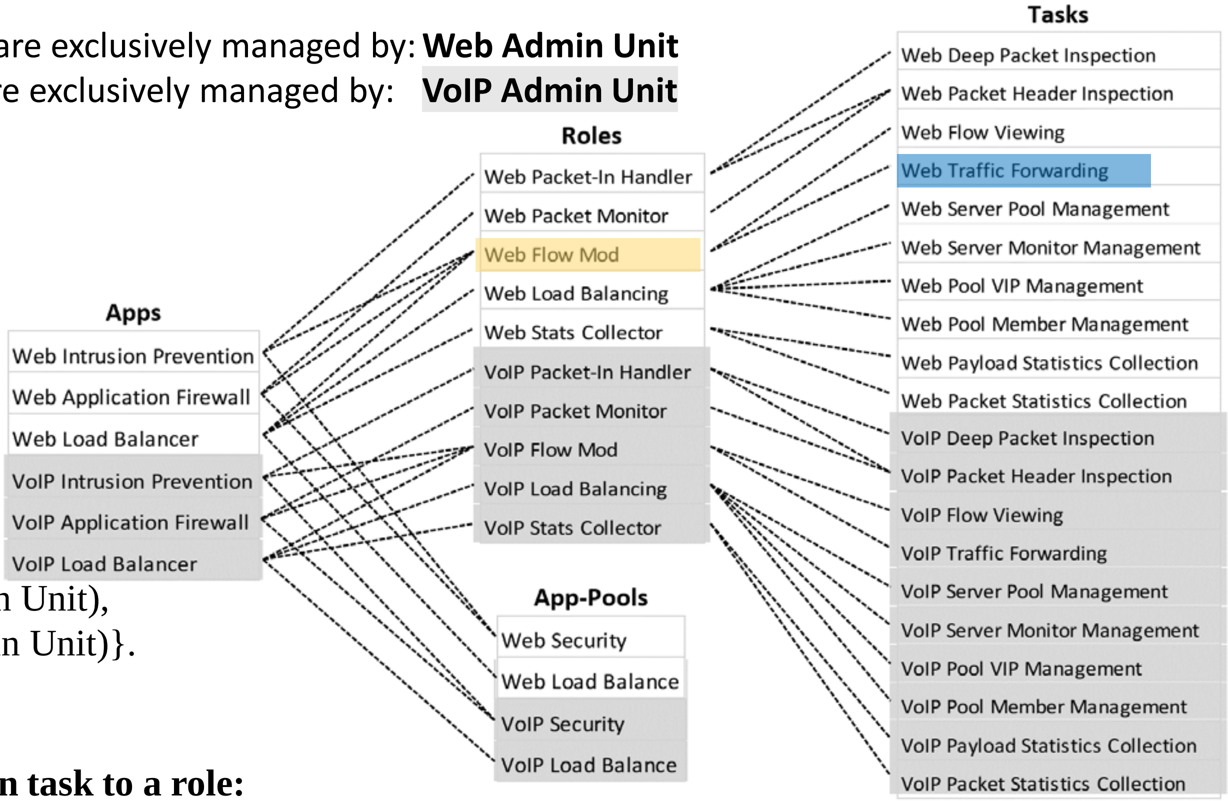
- Relations between apps and roles should be managed by different administrative units.

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}

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**



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

Roles
- Web Packet-In Handler
- Web Packet Monitor
- Web Flow Mod
- Web Load Balancing
- Web Stats Collector
- VoIP Packet-In Handler
- VoIP Packet Monitor
- VoIP Flow Mod
- VoIP Load Balancing
- VoIP Stats Collector

App-Pools
- Web Security
- Web Load Balance
- VoIP Security
- VoIP Load Balance

Tasks
- Web Deep Packet Inspection
- Web Packet Header Inspection
- Web Flow Viewing
- Web Traffic Forwarding
- Web Server Pool Management
- Web Server Monitor Management
- Web Pool VIP Management
- Web Pool Member Management
- Web Payload Statistics Collection
- Web Packet Statistics Collection
- VoIP Deep Packet Inspection
- VoIP Packet Header Inspection
- VoIP Flow Viewing
- VoIP Traffic Forwarding
- VoIP Server Pool Management
- VoIP Server Monitor Management
- VoIP Pool VIP Management
- VoIP Pool Member Management
- VoIP Payload Statistics Collection
- VoIP Packet Statistics Collection

Administrative User Assignment:
TA_admin = {
(web_functions_admin_user, Web Admin Unit),
(voip_functions_admin_user, VoIP Admin Unit)}.

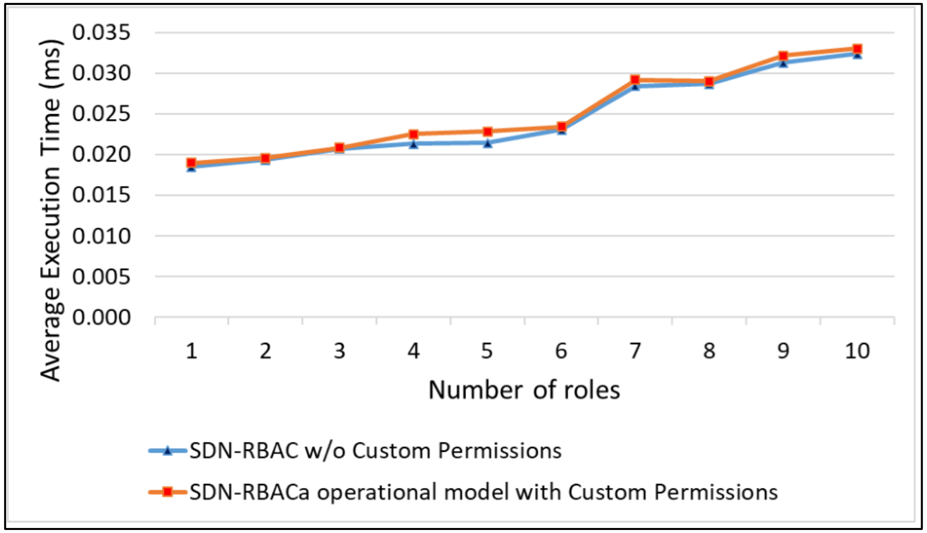
Example:

1. **Administrative Action to assign task to a role:**
assign_task_to_role(web_functions_admin_user, Web Traffic Forwarding Task, Web Flow Mod) **is allowed**.

→**Authorization Function:**
can_manage_task_role(web_functions_admin_user, Web Traffic Forwarding Task, Web Flow Mod) = True.

Reason:
∃Web Admin Unit ∈ AU : ((web_functions_admin_user, Web Admin Unit) ∈ TA_admin) ∧
Web Flow Mod ∈ roles(Web Admin Unit) ∧
Web Traffic Forwarding Task ∈ 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.
- Overhead of around 2.9% on average to the authorization framework.

Conclusion and Future Work

- This work presented SDN-RBACa, an administrative model for role based access control in SDN.
- An approach to extend the capabilities of SDN services for creating custom SDN permissions specialized for the administration of access control in SDN.
- Through proof of concept prototype implementation and use cases, we demonstrated the usability of custom permissions.

- In future work, custom permissions can be further refined and demonstrated in more use cases and implementations of the administrative model.

Thank you