ParaSDN: An Access Control Model for SDN Applications based on Parameterized Roles and Permissions

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

[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 6th IEEE International Conference on Collaboration and Internet Computing (CIC 2020)
December 1 - 3, 2020

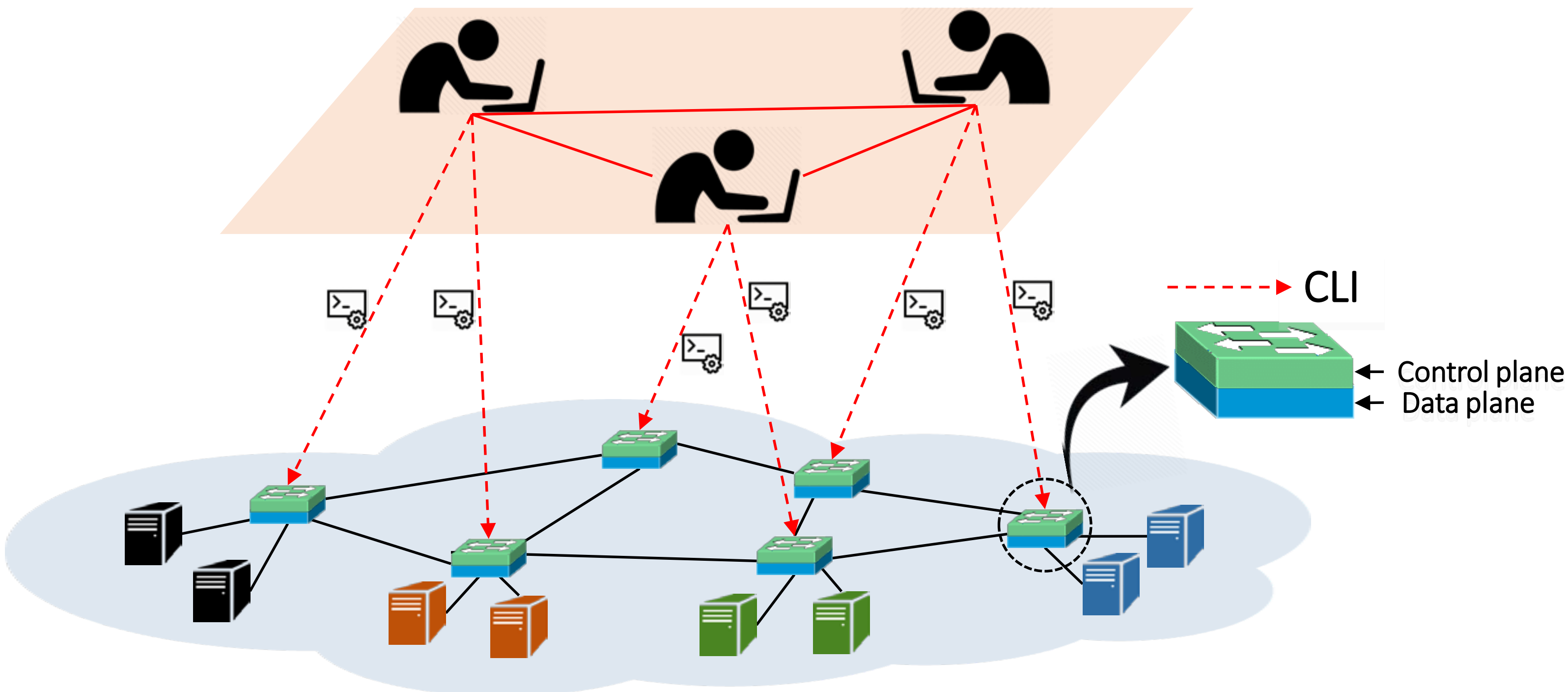
Agenda

- Introduction to Software Defined Networking (SDN)
- Parameterized Roles and Permissions in SDN
- ParaSDN main components
- ParaSDN Parameter Engine
- ParaSDN Conceptual Model and Definitions
- Use-Case Security Configuration in SDN-RBAC
- ParaSDN Implementation & Evaluation
- Conclusion and Future Work

Introduction
Traditional Networks

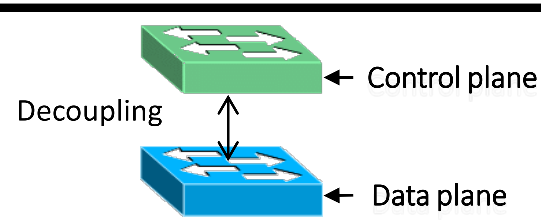
Management Layer

Infrastructure Layer

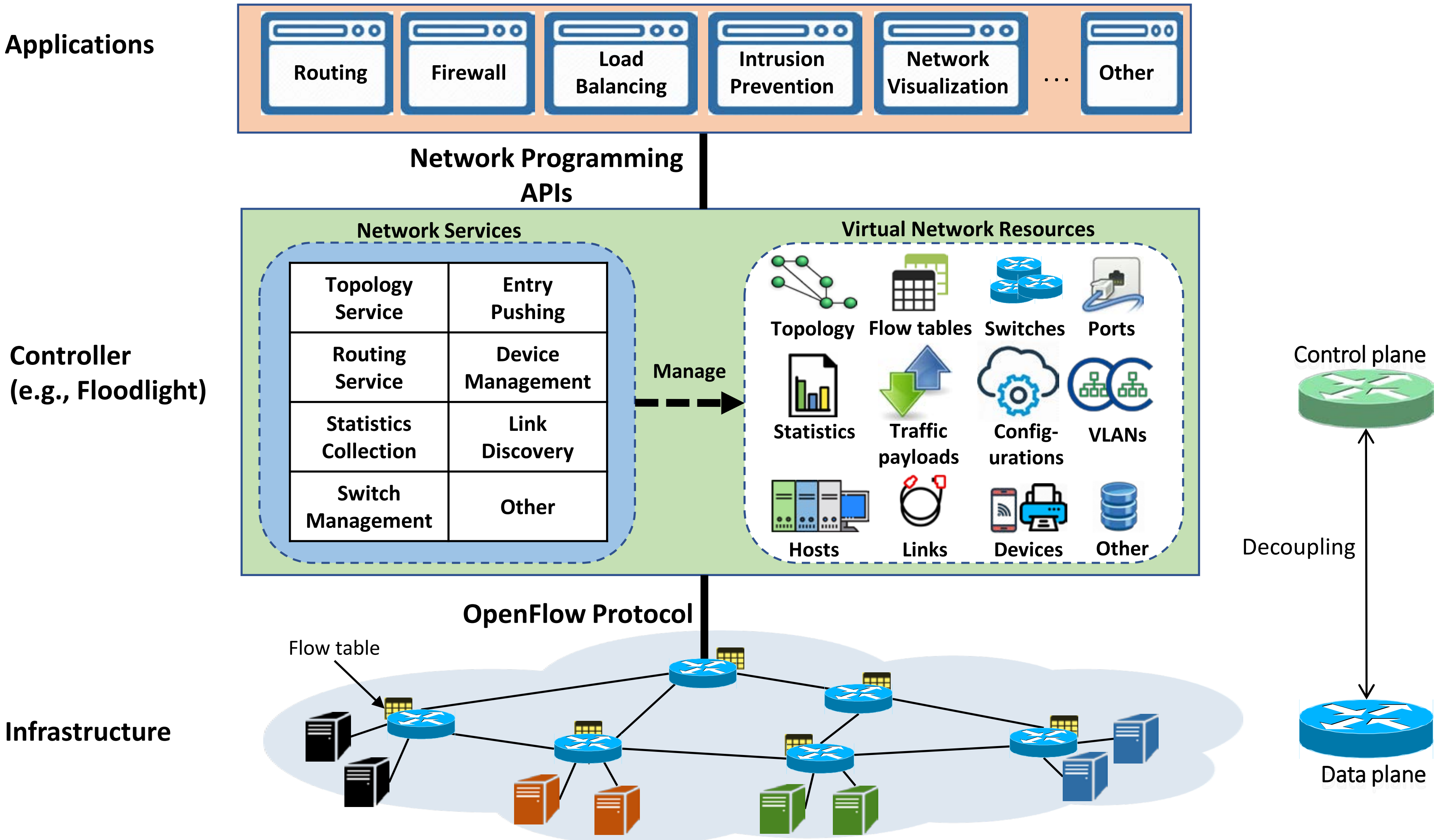


Traditional Network

SDN Idea

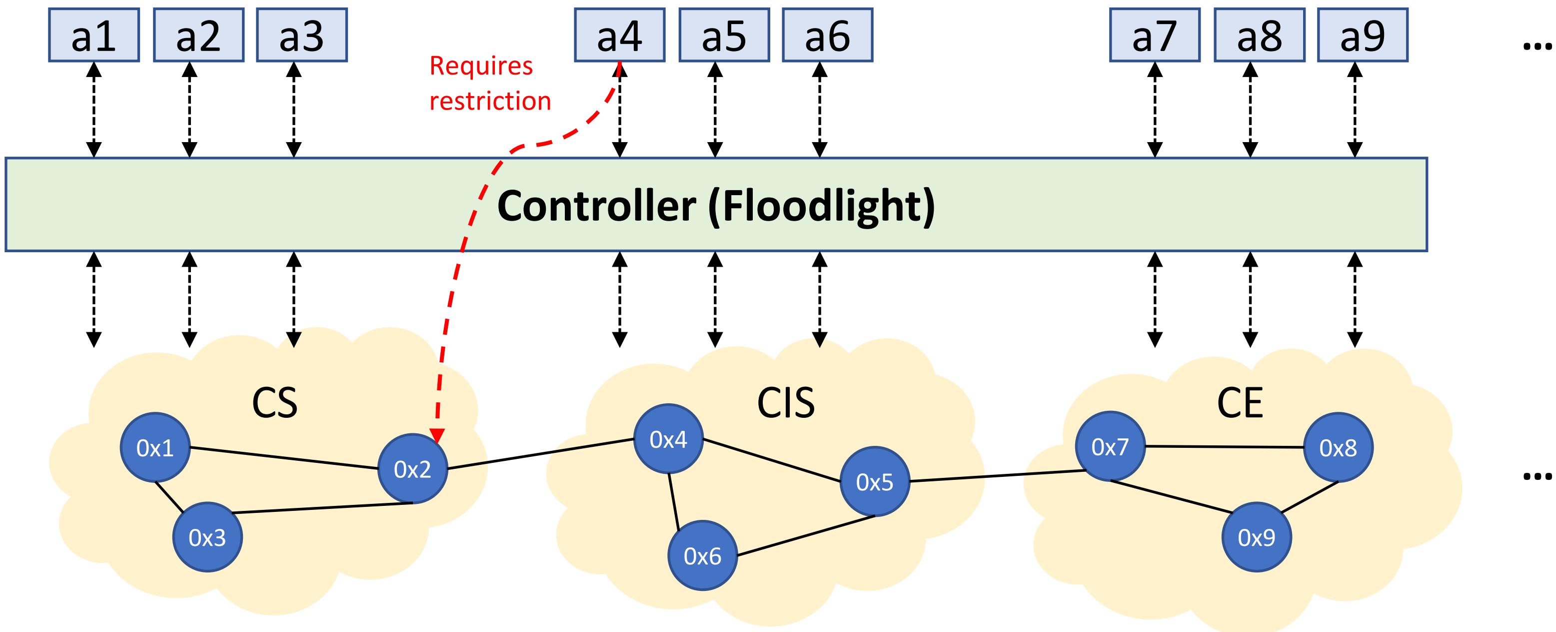


Introduction
Software Defined Networks (SDN)



Motivation

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



Role: Flow Mod$_1$
Assigned Perms:
(addFlow, flow_rule^{sw0x1})
(addFlow, flow_rule^{sw0x2})
(addFlow, flow_rule^{sw0x3})

Role: Flow Mod$_2$
Assigned Perms:
(addFlow, flow_rule^{sw0x4})
(addFlow, flow_rule^{sw0x5})
(addFlow, flow_rule^{sw0x6})

Role: Flow Mod$_3$
Assigned Perms:
(addFlow, flow_rule^{sw0x7})
(addFlow, flow_rule^{sw0x8})
(addFlow, flow_rule^{sw0x9})

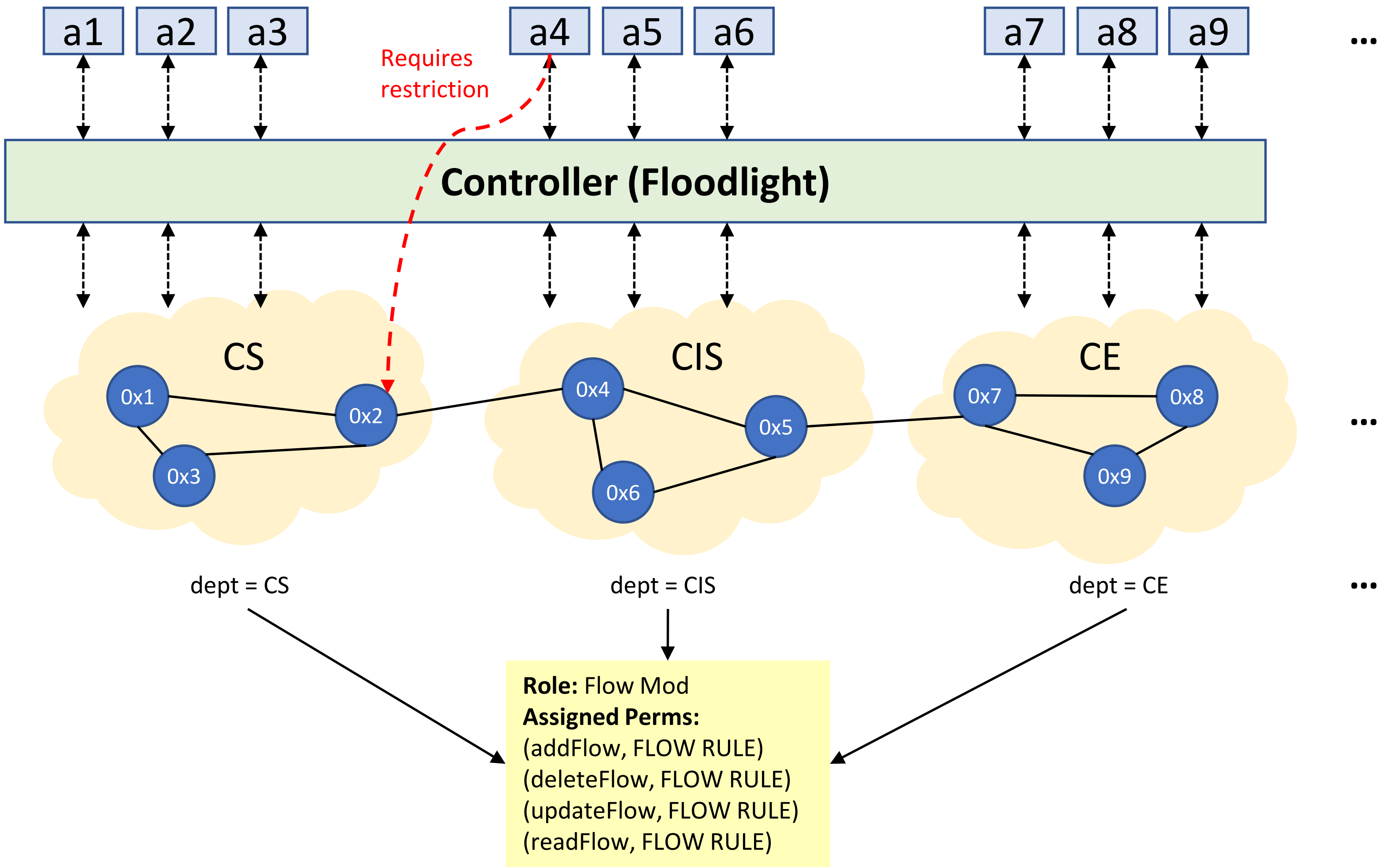
...

- 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



Proposed work

- Formal Access control model for SDN enhanced with role and permission parameters
- Authorization Framework extended with parameter engine and enforcement in SDN controller.

Fine-Grained and Scalable Access Control for SDN

Parameterized Permissions and Roles

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

- **Parameterized Roles:**

  $(r_i , \{(par_1 , val_1), (par_2 , val_2), ...\})$

Example:

(Flow Mod, {(dept, ⊥), (traffic, ⊥)})

- **Parameterized Permissions:**

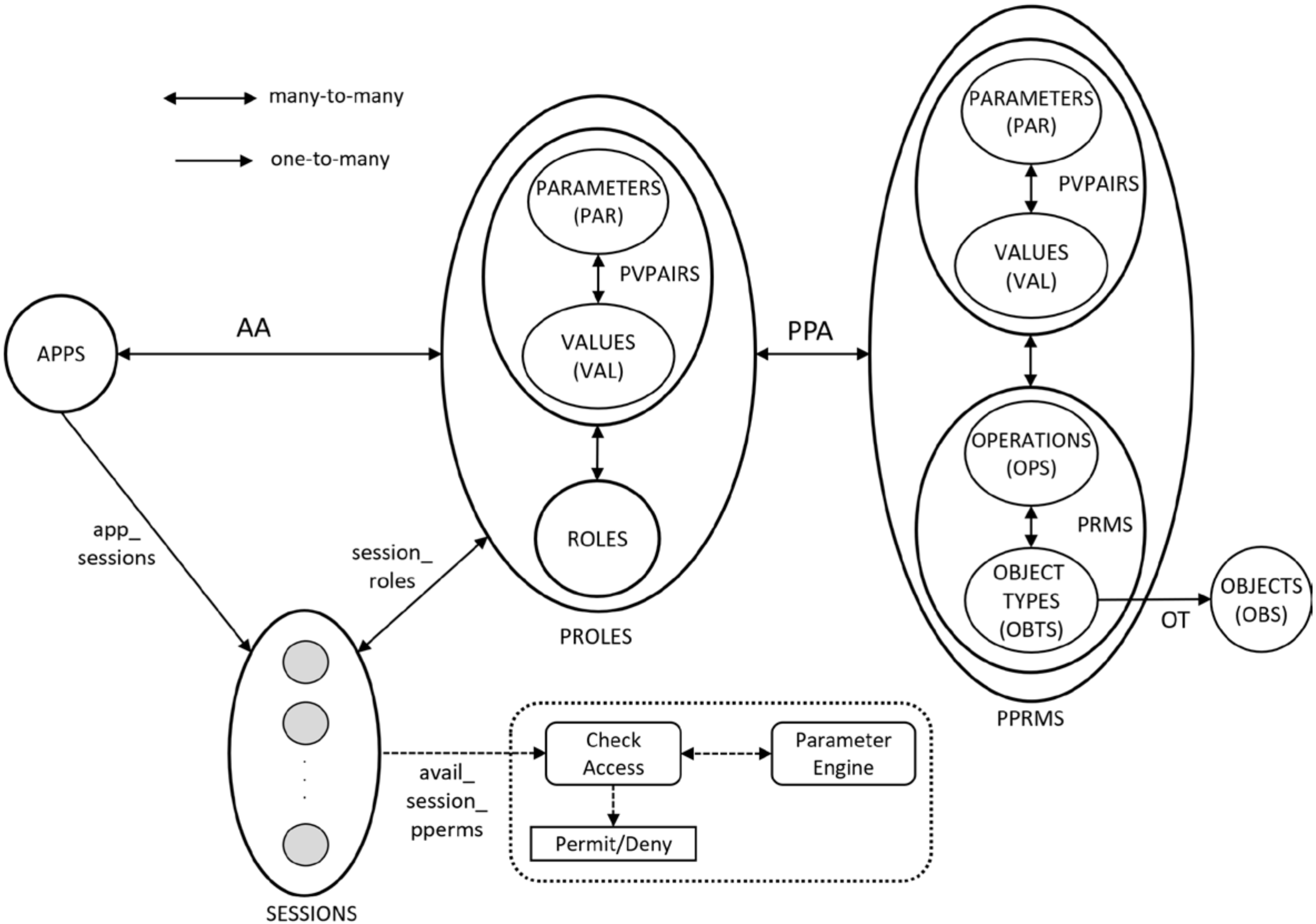
  $((op_i, ot_i), \{(par_1 , val_1), (par_2 , val_2), ...\})$

Example:

((addFlow, FLOW-RULE), {(dept, ⊥), (traffic, ⊥)})

⊥ = *Unknown.*

ParaSDN Conceptual Model



I·C·S The Institute for Cyber Security

ParaSDN Formal Model Definition

C·SPECC Center for Security and Privacy Enhanced Cloud Computing

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 "$\perp$" to indicate that the value of a parameter is unknown.
- parType: PAR $\rightarrow$ {set, atomic} specifies parameter type as set of atomic valued.
- PRMS $\subseteq$ OPS $\times$ OBTS, set of ordinary permissions.
- SESSIONS, set of sessions.

2.Assignment Relations:

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

3.Derived Functions:

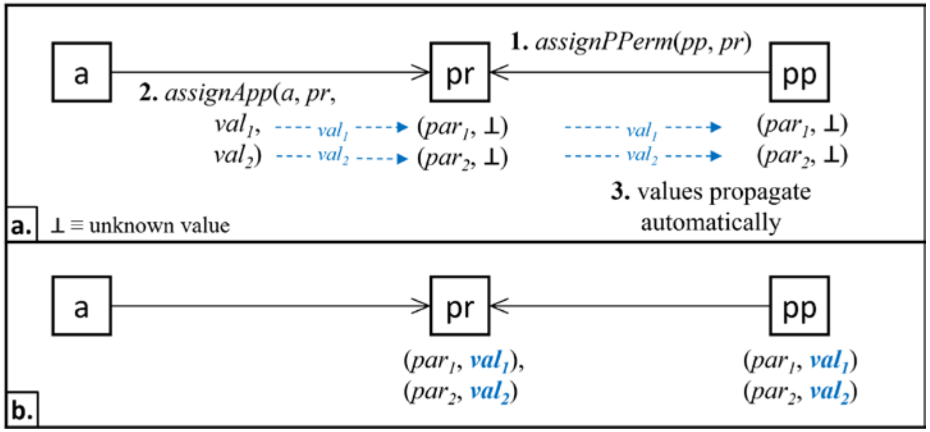
- assigned_pperms: PROLES $\rightarrow$ 2^{PPRMS}, the mapping of parameterized role into a set of parameterized permissions. Formally, assigned_pperms(pr) = {pp $\in$ PPRMS — (pp, pr) $\in$ PPA}.
- app_sessions: APPS $\rightarrow$ $2^{SESSIONS}$, the mapping of an app into a set of sessions.
- session_app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

4.Parameter Verification Functions:

- VERIFIERS = {V$_1$, V$_2$, ..., V$_n$} a finite set of Boolean functions. For each V$_i$ $\in$ VERIFIERS.V$_i$: SESSIONS $\times$ OPS $\times$OBS $\times$ PVPAIRS $\rightarrow$ {True, False}.
- param_verifier: OBTS $\times$ PAR $\rightarrow$ VERIFIERS, a function that maps a combination of object type and parameter to the corresponding verification function needs to be evaluated.

Parameter Value Assignment

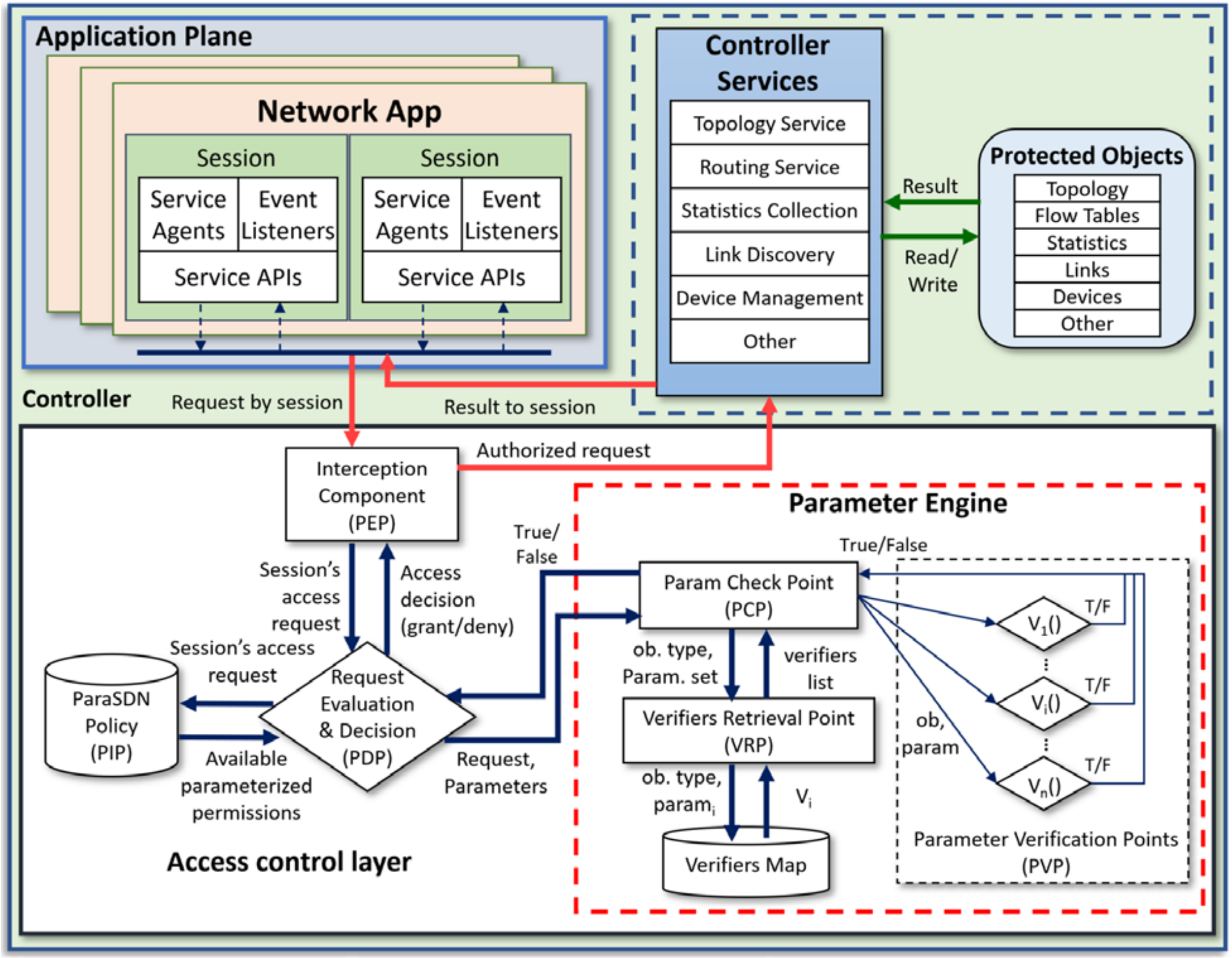
- Parameter values assigned via assignApp administrative action propagate automatically from role parameters to permission parameters



Function	Authorization Condition	Update
assignPPerm(pp, pr)	pp $\in$ PPRMS $\bigwedge$ pr $\in$ PROLES $\bigwedge$ (pp, pr) $\notin$ PPA	PPA' = PPA $\bigcup$ {(pp, pr)}

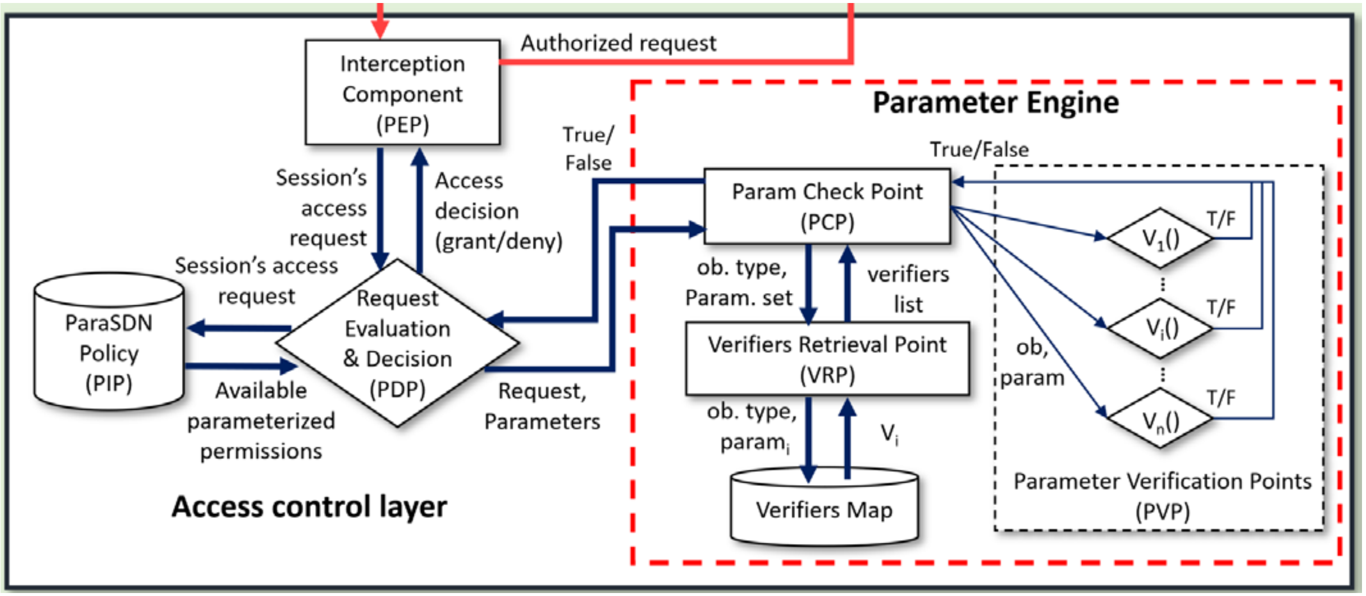
assignApp(a, pr, valset)	a $\in$ APPS $\bigwedge$ pr $\in$ PROLES $\bigwedge$ valset $\in$ VAL $\bigwedge$(a, pr) $\notin$ AA	//Assign values to role parameters.
		For each pr_pvpair$_i$ $\in$ pr.PVPAIRS, v$_i$ $\in$ valset, $1 \leq i \leq$ |pr.PVPAIRS| **do**
		pr_pvpair$_i$.val = v$_i$
		//Pass parameter values from pr to its member parameterized permissions.
		For each pp $\in$ PPRMS : (pp, pr) $\in$ PPA **do**
		For each pr_pvpair$_i$ $\in$ pr.PVPAIRS, pp_pvpair$_i$ $\in$ pp.PVPAIRS, $1 \leq i \leq$ |pr.PVPAIRS| **do**
		pp_pvpair$_i$.val = pr_pvpair$_i$.val
		AA' = AA $\bigcup$ {(a, pr)}

ParaSDN Framework Architecture



ParaSDN Parameter Engine

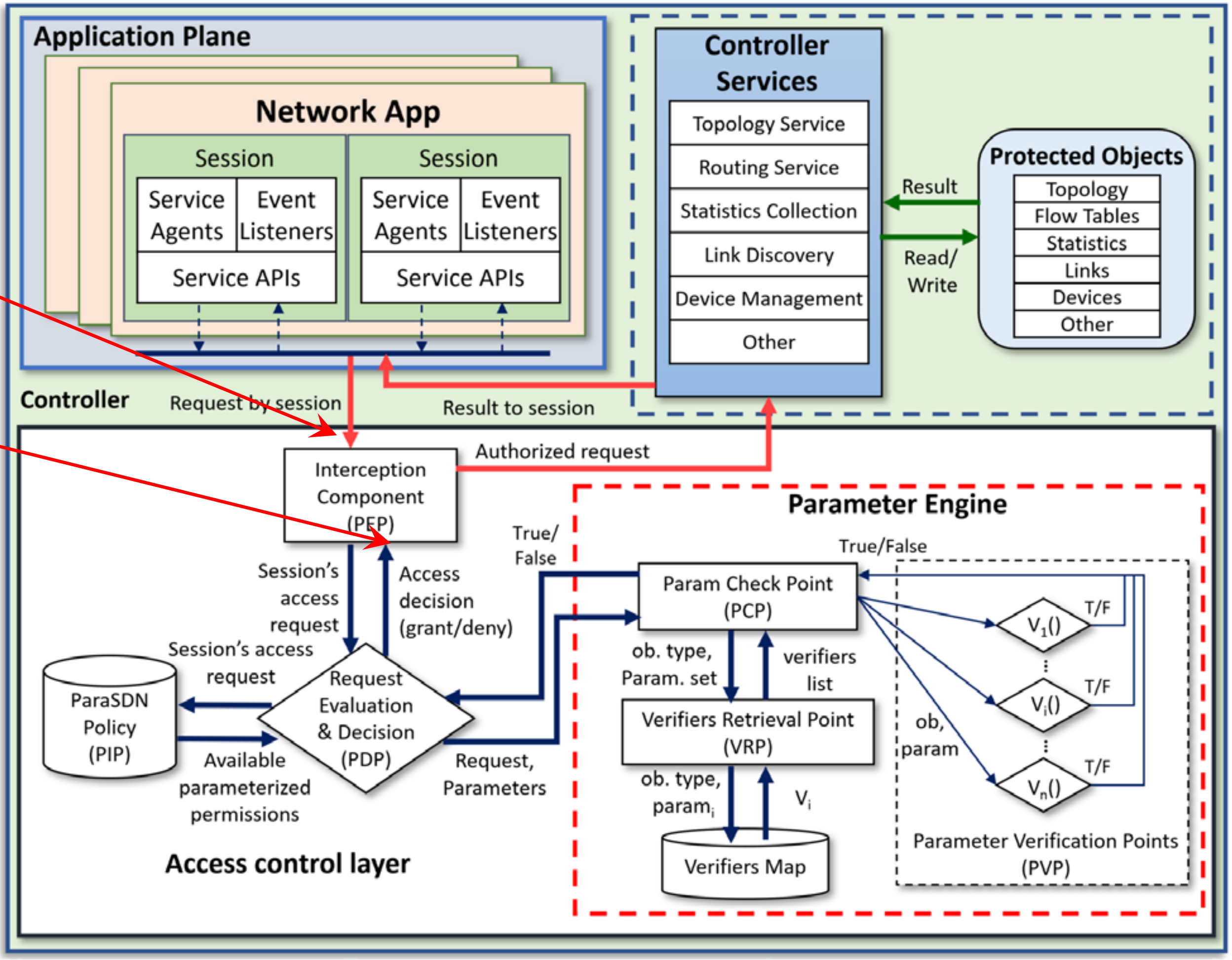
- The General functionality of the Parameter Engine is distributed among multiple components:
  - Parameter Check Point (PCP),
  - Verifiers Retrieval Point (VRP), and
  - multiple Parameter Verification Points (PVPs).



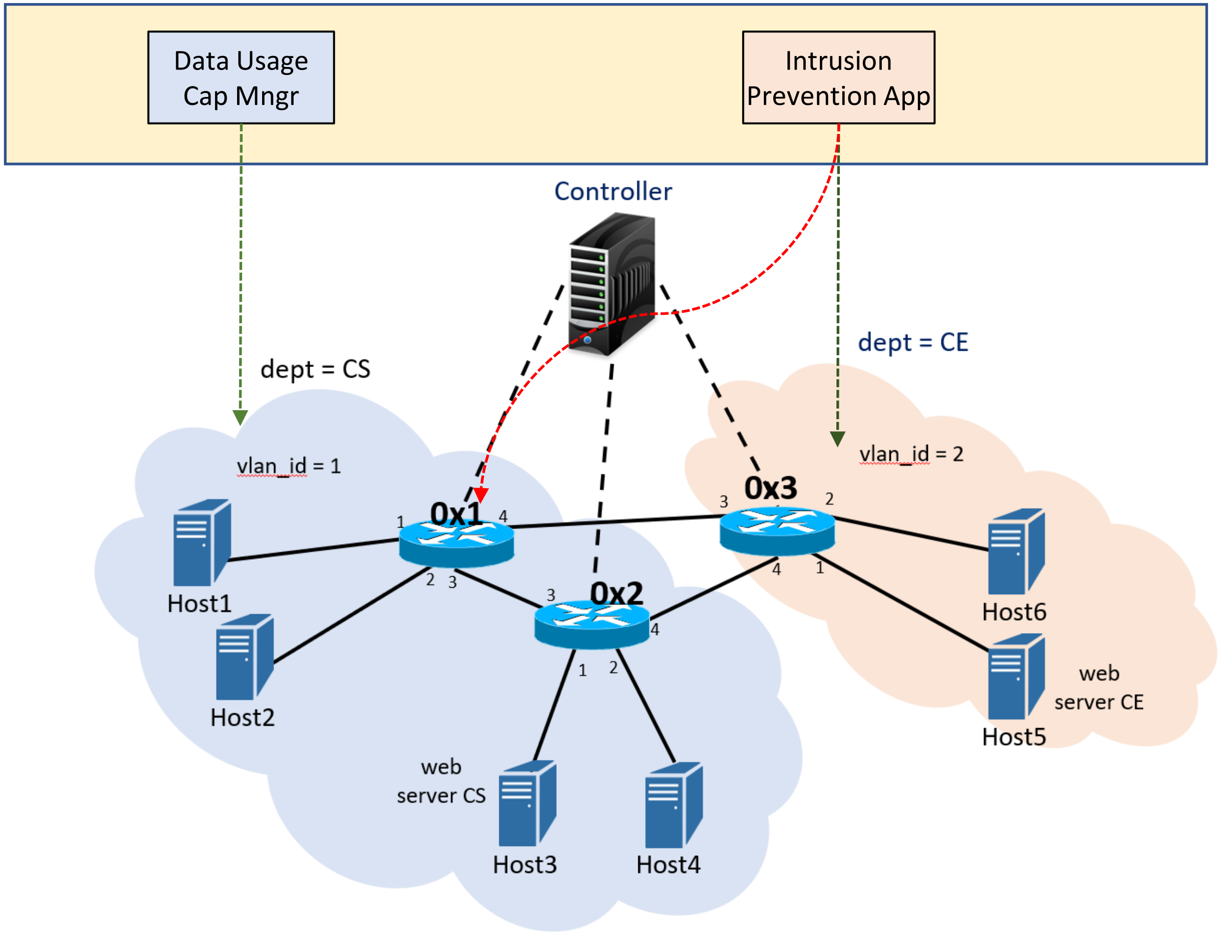
ParaSDN Implementation

Timer Started

Timer Ended



Use-Case & Security Configuration in ParaSDN Model



Use-Case Security Configuration in ParaSDN Model

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, 0x2:1, 0x2:2, 0x3:1}. Range(dept) = {CS, CE}. Range(traffic) = {web}.
- parType(vlan_id) = atomic. parType(attachment_point) = set. parType(dept) = set. parType(traffic) = 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} ⋃ {(ps, PORT-STATS) : ps ∈ PS} ⋃ {(fr, FLOW-RULE) : fr ∈ FR} ⋃ {(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, ⊥)})}
- PPA = {(((queryDevice, DEVICE), {(vlan_id, ⊥)}), (Device Handler, {(vlan_id, ⊥)})), (((getBandwidthConsumption, PORT-STATS), {(attachment_point, ⊥)}), (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, (Device Handler, {(vlan_id, 1)})), (Data Usage Cap Mngr, (Bandwidth Monitoring, {(attachment_point, {0x1:1, 0x1:2, 0x2:1, 0x2:2})})), (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)}))}.

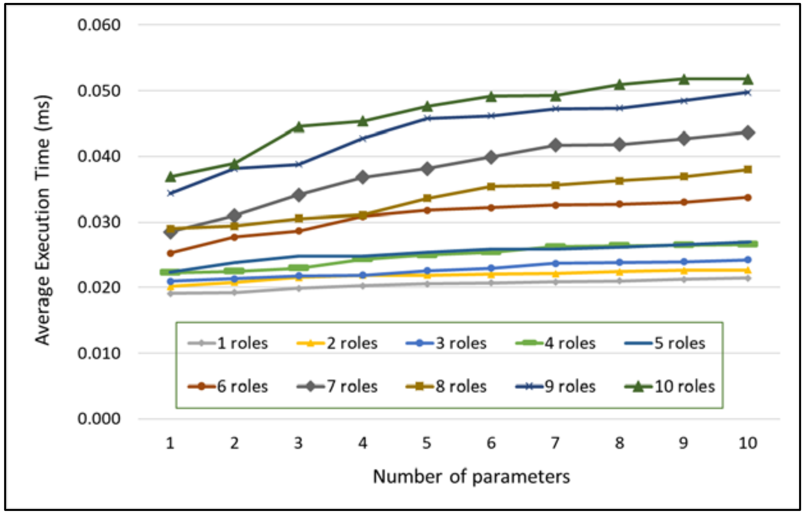
3. Derived Functions:

- 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}.
  app_sessions(Intrusion Prevention App) = {IntrusionPreventionSession}.
- session_roles(DataUsageAnalysisSession) = {(Device Handler, {(vlan_id, 1)}), (Bandwidth Monitoring, {(attachment_point, {0x1:1, 0x1:2, 0x2:1})})}.
  session_roles(DataCapEnforcingSession) = {(Flow Mod, {(dept, {CS}), (traffic, web)})}.
  session_roles(IntrusionPreventionSession) = {(Device Handler, {(vlan_id, 2)}), (Packet-In Handler, {(attachment_point, {0x3:1})}), (Flow Mod, {(dept, {CE}), (traffic, web)})}.
- avail_session_pperms(DataUsageAnalysisSession) = {((queryDevice, DEVICE), {(vlan_id, 1)}), ((getBandwidthConsumption, PORT-STATS), {(attachment_point, {0x1:1, 0x1:2, 0x2:1})})}.
  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})}), ((addFlow, FLOW-RULE), {(dept, {CE}), (traffic, web)})}.

4. Parameter Verification Functions:

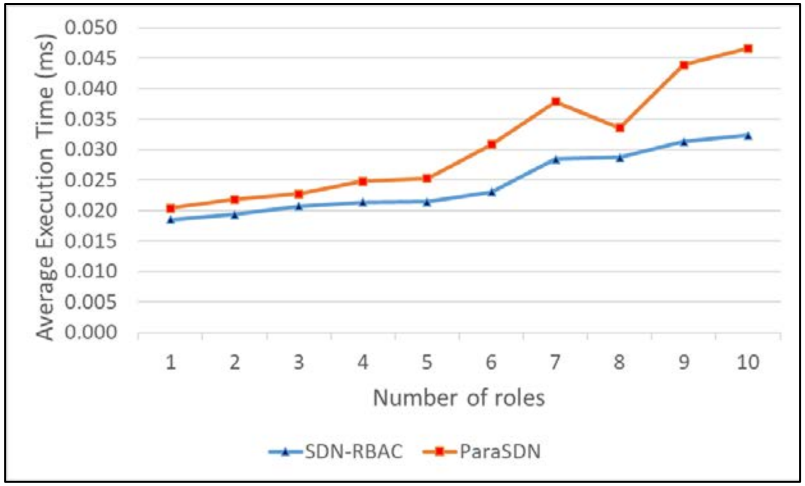
- VERIFIERS = {VDeviceVlan, VStatsAttachpoint, VRuleSwitch, VRuleTraffic, VPInAttchpoint}.
- param_verifier((DEVICE, vlan_id)) = VDeviceVlan.
  param_verifier((PORT-STATS, attachment_point)) = VStatsAttachpoint.
  param_verifier((FLOW-RULE, dept)) = VRuleSwitch.
  param_verifier((FLOW-RULE, traffic)) = VRuleTraffic.
  param_verifier((PI-PAYLOAD, attachment_point)) = VPInAttchpoint.

ParaSDN Evaluation



- 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.

ParaSDN Evaluation



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

Conclusion and Future Work

In this work:

- We proposed ParaSDN, a formal access control model that provides fine grained capabilities for SDN using the concept of parameterized roles and permissions.
- We implemented a proof of concept prototype in an SDN controller.

Future research

- Extend the model to suit the needs for multi-controller environments in SDN-Enabled technologies like IoT and Cloud infrastructures.