KnowU - Policy Driven Access Control in Campus Networks

Gurupragaash Annasamy Mani
University Of Utah
guru@cs.utah.edu

August 28, 2017

1 Overview

University campus infrastructures count among the most complex and sophisticated information technology (IT) deployments; often combining a mix of enterprise, academic, research, and healthcare environments. Different segments of the campus have very different policies and regulations that govern its treatment of sensitive data (e.g., private student/employee information, health care data, financial transactions etc.). Dealing with the security of this complex and highly dynamic environment is extremely challenging. The unique requirements of data-intensive scientific research traffic often require exceptions to conventional IT policies, which typically result in ad-hoc solutions that bypass standard operational methods and procedures, thus leaving both the scientific workflow and the campus as a whole vulnerable to attack. In short, state-of-the-art campus security operations still heavily rely on human domain experts to interpret high level policy documents, attempt to implement those policies through low level mechanisms, manually implement exceptions to these policies to accommodate scientific workflow requirements, interpret reports and alerts from a variety of security point solutions, and be able to react to security events in near real time on a 24-by-7 basis. This state of affairs is clearly untenable.

In order to make the current systems to work, we have to let go of the dynamicity. So a researcher from healthcare cannot access his healthcare resources unless he is connected to the healthcare network by being present in the healthcare domain physically. And this applies for other groups too. In this project we aim to solve this problem by automating the process of policing user groups and their traffic dynamically. Our policy engine is fed information about user group, policies that these user groups should follow and then whenever the user connects to the network, the policy engine kicks in and installs flows based on the policy configured for that group

2 Background

Any campus network can be broken down into three main pieces. 1) Users, who are part of the campus network. These users are generally identified using some authentication mechanism. Most of the network use RADIUS for authentication. 2) Policies, which should be applied over these user/user groups. 3) Hardware(switches/routers) over which these policies are implemented.

In KnowU, we will also be using RADIUS for authenticating/managing the users. Section 1 describes the basic radius interactions when a user connects to the network. Sections 2.2 and 2.3 deals about how the users and policies are represented in KnowU. KnowU implements these access control policies in networks using Openflow rules and hence the network should be openflow capable. But most of the current campus networks don't have openflow capable switches and to replace the entire network with openflow capable switches is not possible. So we have designed KnowU such that, it can implement access control in a campus network provided the edge switches(user facing swithces) are openflow capable.

2.1 RADIUS

The radius interactions are shown in Figure(1). Here the phone/laptop runs a 802.1x supplicant. The Talac switch or any edge switch where the user connects to is called the Network Access Server(NAS). The radius server which we are using is freeradius 3.0. The NAS is configured as valid in the radius server configuration and their shared secret is configured in both the NAS and radius server. Each NAS has its own shared secret with the Radius server. Now when the User connects to the AP(NAS), they use some secure protocol to communicate. It can be EAP, CHAP, MSCHAP, PPP and so on. We are using EAP here. So when the user initiates a EAP Start, the NAS sends EAP request identity message. Now when the user responds with EAP response, the NAS converts this into a RADIUS ACCESS-REQUEST packet and sends it to the radius server after encrypting with the shared key. The radius server decrypts the messages, and gives a challenge to the USER via RADIUS ACCESS-CHALLENGE packet. The NAS converts this ACCESS-CHALLENGE packet to EAP Request Authentication packet and once the user sends a response for this, the NAS converts this to RADIUS ACCESS-CHALLENGE and also add the state variable to let the RADIUS know it is not a new session. The radius then process the response and if its valid will respond with RADIUS ACCEPT-ACCEPT or send a RADIUS ACCEPT-REJECT packet which the NAS will translate accordingly. The radius can also use additional attributes to instruct the NAS, but in our case we dont use any additional attributes.

2.2 What is security zone ?

Security Zone is a collection of users who can communicate with each other and have common policies that they have to adhere when communicating with the other zones. Each zone has a id(number) associated with it and a set of policies to determine the communication. Eg: Flux users can be put in a zone, INSCC users can be put in a zone, Internet is also a zone(but a special one since the set of users cannot be quantified). Now if the Flux users want to communiate with the Internet, they should have an appropriate policy which will allow them to do so.

2.3 Policies

A Policy is associated with a zone and specifies how this zone's users can communicate with another zone. Apart from specifying the how the zones can communicate, the policies can also have fine grained exceptions like, one particular user of the zone can access another zone; one particular resource is more secure and only a smaller subset of the zone can access and so on. Lets say the flux zone has id 100 and inscc has id 200. And let the policy be flux users can communicate with inscc users and inscc users can communicate with flux users. But the flux printer(printer1) cannot talk with inscc and vice versa, but one inscc user(inscc1) can talk with flux printer. This can be represented using the policy

```
{
    "zoneId": "100",
    "defaultAction": "false",
    "zones": [
        {
            "dstZoneId": "200",
            "isAllowed": "true",
            "exceptions": [
                {
                    "isAllowed": "false",
                    "srcIp": "printer1"
                },
                {
                    "isAllowed": "true",
                    "srcIp": "inscc1",
```

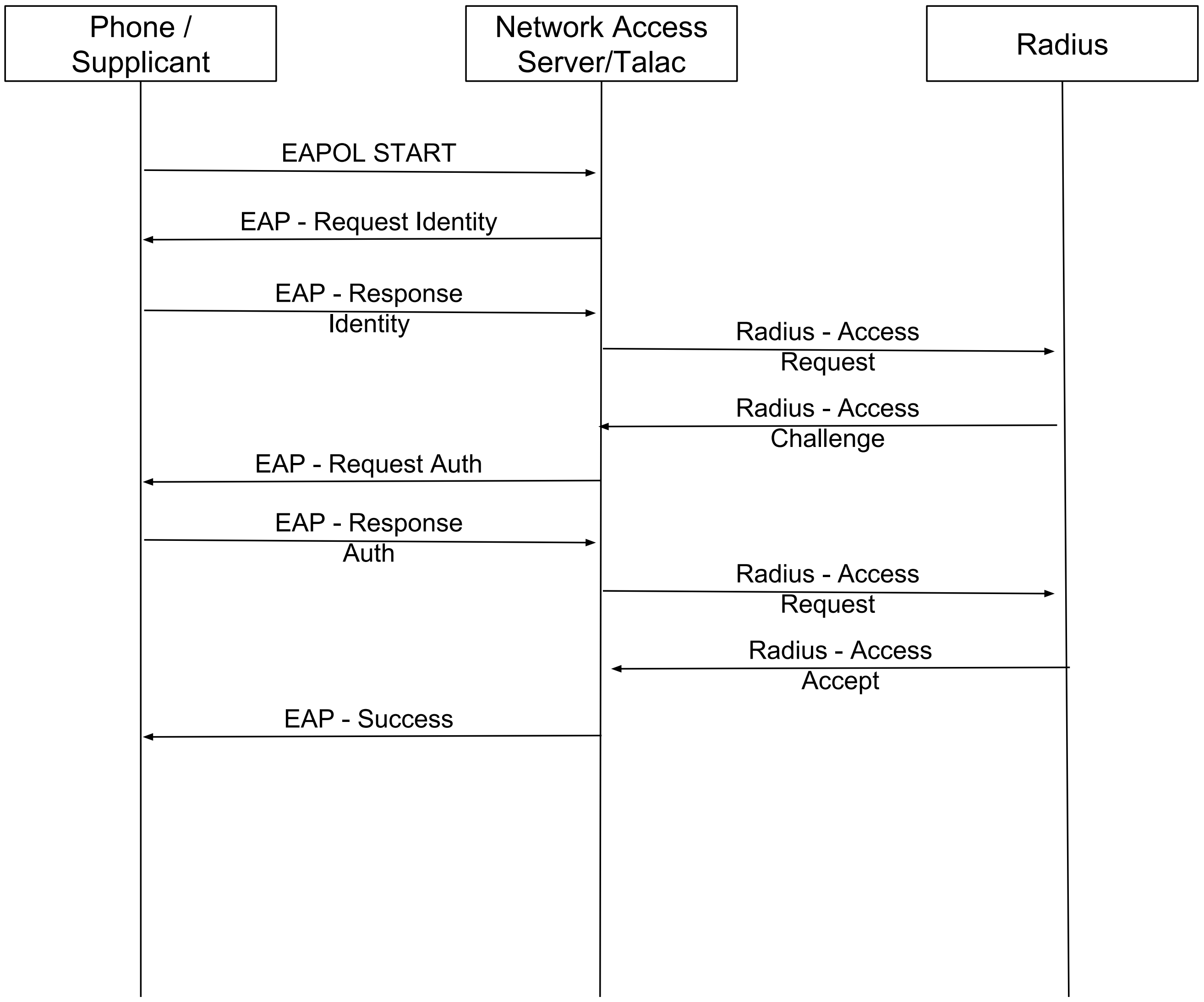

Figure 1: Radius Interaction

```
                    "dstIp": "printer1"
                }
            ]
        },
}
```

Now lets look at generic policy structure.

```
    list policy {
        leaf zoneId {
            type uint32;
            description "Id used to represent the zone";
            default '0';
        }
        leaf defaultAction {
            type boolean;
            description "This is set to true if connections from other zones can be allowed";
            default false;
        }
        list zones {
            leaf dstZoneId {
                type uint32;
                description "Zone id of the destination";
                default ’0’;
            }
            leaf isAllowed {
                type boolean;
                description "True if connection is allowed to this destination";
                default false;
            }
            list exceptions {
                description "This is used to specify the exceptions with respect to this zone";
                leaf srcUser {
                    type string;
                    description "UID of the user";
                }
                leaf dstUser {
                    type string;
                    description "UID of the user";
                }
                leaf isAllowed {
                    type boolean;
                    description "True if connection is possible";
                    default false;
                }
            }
        }
    }
```

Here the ”zoneId” represents the id of the zone. This id value is unique. If the zone allows communication from all users irrespective of their zone, then defaultAction can be set to true. The ”list zone” is used to specify if the communication is possible between the specified zones or not. If we need more fine grained control with respect to the communication between the zones, then exceptions can be used. For a communication to happen between two zones A and B, there should be two policies. A policy of Zone A which allows its users to talk to Zone B and a policy in Zone B which allows its users to talk to Zone A. If a policy in a zone doesnt have a complimenting policy in its peer zone, then the communication is not possible. This will lead to lots of policies even if a zone doesn’t want any restrictions. This is where the defaultAction will be helpful. Lets say there are 10 zones in the system (100, 200, ... 1000). Now each zone(Eg 100) will have policy structure with zoneId set to 100 and a list destination zones and the restrictions associated with it. But for a zone which allows all communication it can just have the defaultAction set to ”true” and the other values can be empty. And later if the adminstrator decides to prevent the zone from communication to one specific zone, then that zone details can be added to the zone list with the isAllowed of that particular zone set to ”false”.

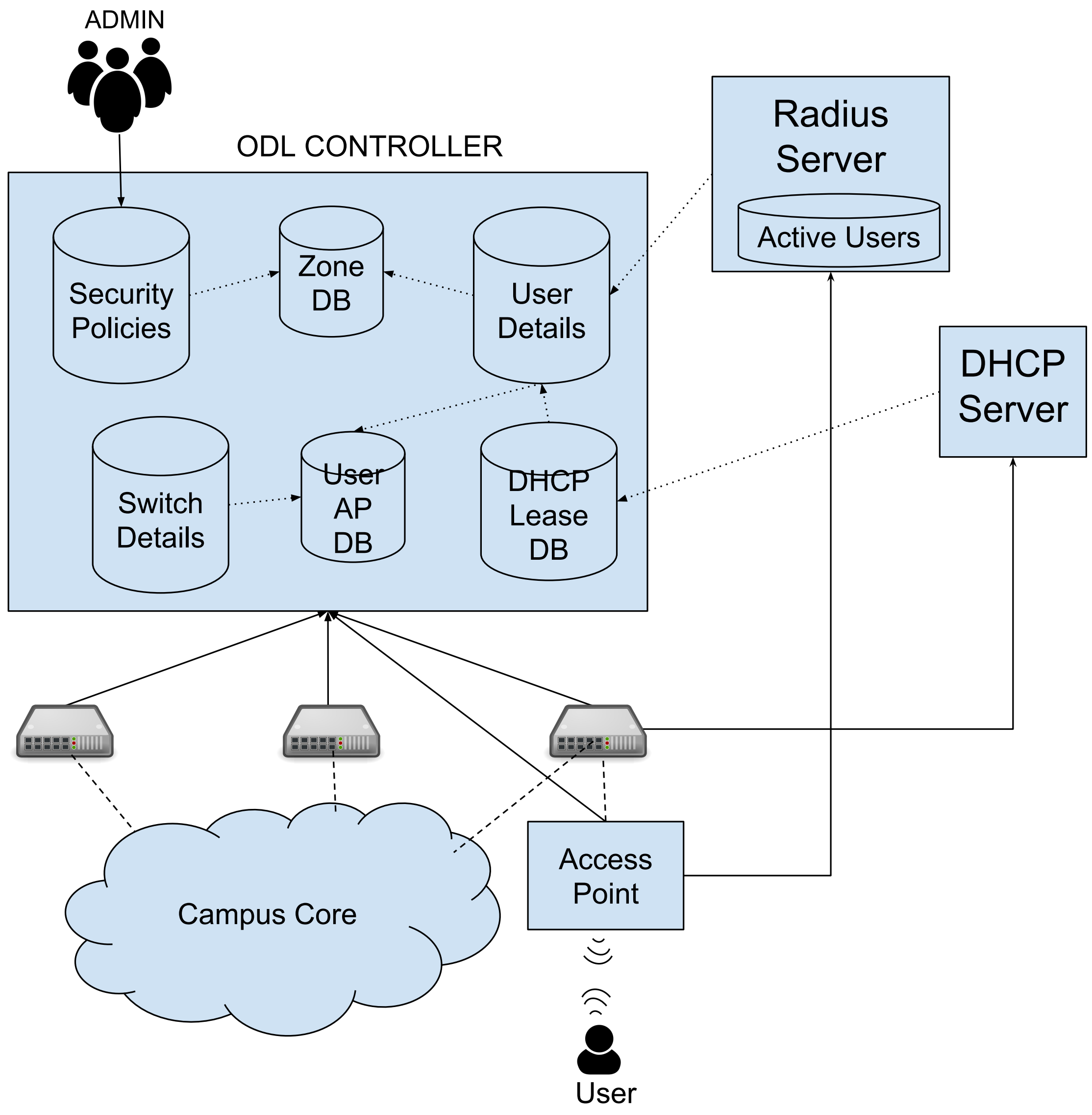

Figure 2: KnowU System Overview

Figure 3 shows how the system architecture maps to the KnowNet Architecture

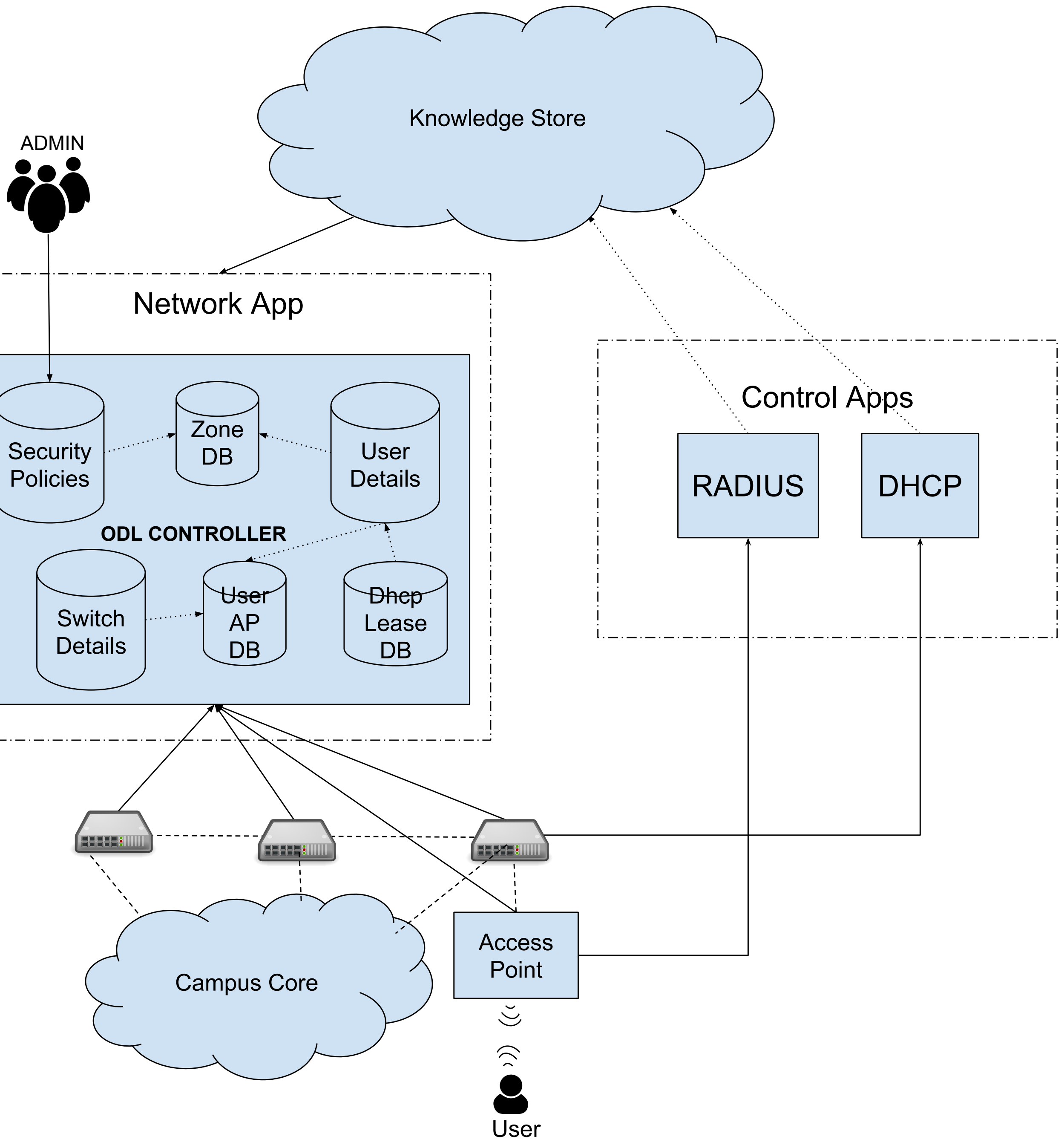

Figure 3: System Architecture Mapped to KnowNet Architecture

2.4 Databases

Below is the description of the databases used in the system. How these database are used is explained in the architecture section.

- User DB - A database which maintains the users connected in the system and their attributes.

1. key - user ip - Ip associated with the user
2. Data
    (a) zone id - The zone user belongs to
    (b) attributes - Till now the implemented doesnt use this field. Additional user details can be stored here. Could be used to add exceptions
    (c) ap-ip - The switch/ap the user is connected to
    (d) name - UID associated with the user
    (e) mac - Mac address associated with this ip/user

- Switch Config DB - A database which maintains the list of switches present in the system. The administrator knows the topology of the network and knows which ports of the switch is network facing and this information is added to the ODL system using this database. When the Switches connect to the ODL controller, this database is referred to find which are the uplink and downlink ports and then this info is saved in the Switch DB

    1. ip - Ip address of the switch
    2. Uplink Ports - The interface which connects the AP to the core network.

- Switch DB - A database which maintains the list of switches connected with the controller

    1. ip - Ip address of the switch
    2. Node-ref - A pointer maintained by opendaylight to access the switch
    3. name - The details of the manufacturer
    4. Downlink Ports - The interface which connect the AP and the hosts.
    5. Uplink Ports - The interface which connects the AP to the core network.

- Dhcp-Lease DB - A database which is populated based on dhcp alloc events. Whenever Dhcp server allocates an IP for a mac, an entry is added in this DB. This information is crucial for the system and uses this to bind the user with the IP at runtime.

    1. mac - Mac address to which IP was allocated
    2. ip - IP address allocated to the above MAC address

- User-AP DB - A database which maintains the association between the user and the ap the user is connected

    1. ip - user ip
    2. Node-ref - an opendaylight reference to the AP associated with user

- Zone DB- A database which maintains security zone details using

    1. key - ZoneId - Security Zone id
    2. Data
        (a) vlanId - Vlan Associated with the security zone
        (b) userList - List of users associated with this security zone
        (c) flowList - Opendaylight flow data structures used to install flows associated with this security zone
            i. Flow - ODL datastructure which represent a flow
            ii. FlowId - ODL datastructure to represent the id of the flow
            iii. nodeId - ODL reference to the switch where the flow was installed

3 Architecture

Our system can work only with the openflow enabled switches. As soon as the switch connects to the controller, the details of the switch (the interfaces present in the switch, it capabilities) are fetched by the ODL's inventory module. Once this fetch is done, our module gets the notification and we walk through the inventory modules database, fetch the information required by us and save them in the switch details database. When the user connects to wifi access point/physical port and authenticates himself, the details of the user is got from DHCP and RADIUS. This new user detail is updated in both the user details database and the user ap database. Now whenever we need to know the zone corresponding to the user, we can look up the user details database and when we need to install flows with respect to the user, we find the switch associated to the user using the user ap database. Apart from these, there is a security policy database which holds all security policy associated with zones which was described above. As the admin updates the security policies, the policy database is populated. In the system, we install flows reactively for intra zone communication and proactively for inter zone communication.

4 Workflow

- The openflow switches are configured to talk to the controller.
- The Access Points(APs) are configured with the Radius/DHCP Relay details.
- If the AP doesnt support DHCP relay, it will be configured in the first hop switch of the AP.
- When the OF switches establish a connection with the controller, the details of the switch is updated in the switch db.
- For all the OF switches connected to the controller, a low priority rule which matches any IP packet and sends the packet to the controller and a low priority L2 rule which does a MAC LEARING/FLOOD are installed in the switch.
- The user connects to the AP and and uses his credentials to authenticate.
- AP(Radius Authenticator) passes the users details to the RADIUS server which verifies the users and adds the user details in its DB.
- Upon authentication by radius, the user sends a dhcp request, which are collected by the dhcp relay agent. The packets are sent to dhcp server and the dhcp reply is propagated to the user.
- Once the dhcp server assigns the ip, this information is updated in the DhcpLeases table in the ODL. This triggers the ODL to query the Radius DB to find the entry associated with the MAC Addess. Once the user entry from the radius is fetched, the user information from radius is binded with the ip from the dhcp lease and this is updated in the UserDB.
- Now the ODL has the information that a new user with this ID belonging to this zone from the radius update and also IP associate with the user from the Dhcp update. Now we bind these two information and update the user's detail in the UsersDB. This late binding of userid with the ip is how the system manages to install the policy rules which are represented in the abstract domain(User id's and zone ids) into IP flow rules at runtime. This property makes the system more generic and now it can police the user based on his ID and not based on his ip.
- The user's security zone is identified and if this is the first user in that security zone, no flow rules are added. If there are already users in that security zone, then flows are installed in the following ways(proactive flow programming) - Lets say the current user has ip 'A' and the associated AP is 'AP1", and the existing users of the same zone are B, C, D. The flow with "srcip = A and dstip = B—C—D" and "srcip = B—C—D and dstip = A" are created in AP1 and similar flows associated with A is installed in AP's associated with B.

- Due to the above flows, the communication is possible only between nodes belonging to the same security zone.
- Whenever a user(U1) from security zone A tries to contact user (U2) from security zone B, there is no flow to forward the packet and hence this will be caught by the low priority catch all IP packet flow installed in step 3 and will reach the controller(reactive flow programming).
- The controller then gets src ip, dst ip from packet and determines users and the zones associated with them. With the zone information it queries the policy db to check if a connection between the entities is possible. If so creates appropriate flows and a reference to these flows is maintained in the flowList of the security zone db.
- Now whenever a policy associated with the security zone is updated, the flows associated with the zones are identified and the flows are removed. Since our system works in the reactive mode, the flows will be installed again whenever ever there is packets received. But then the new flows will follow the updated policy.
- As we already mentioned, Internet is also a Zone, but a special one which doesnt have quantified number of users. So if we create a flow for each communication to internet, the flow table in the switch is going to explode immediately. So we treat Internet Zone in a special way. Lets say the ip space controlled by our system is 100.x.x.x, then we will install rule which matches this prefix in src/dst and its action is set to forward to controller. Lets call this rule R1. Now we install another rule which will match this userip, dstip is wildcarded and forward the packet to the core. This rule's priority is set lower than R1's priority. We also create the complimentary rule, src=* dstip = userip which forwards packets to the user. Let these rules be R2. Now if the packet from internet arrives this will be matched by the R2 and will be forwarded, but if the user try to initiate communicate with another security zone, it will match R1 and will be lifted to controller which will make sure if the communication is possible or not
- If a user disconnects, the change is propagated to ODL using a radius trigger. So the user is removed from the security zone list and the flows associated with the user is removed. And here the user mean one particular instance of the (user,ip) combination. If the user had multiple devices connected to the network, the disconnect will only remove the flow of that (user, ip) pair.

5 Deployment

Figure 4 shows how the deployment will be. The lower layer comprises of open flow based talac switches, which are controlled by our controller. The next level also comprises openflow enabled switches which are connected to a separate controller which is configured to do basic l2 learning/switching. These switches connect the talac switches to the core network. The policy restrictions are enforced by installing openflow rules on the talac switches. Thus we enforce the policy restrictions at the source itself when the source is a part of our network. If the source is internet or is connect to network via a switch which is not controlled by us, then we cannot enforce source restriction and hence this will be handled at the destination which will be a talac switch controlled by us.

Figure 5 shows the actual deployment at Flux. Here Talac and DELL form the edge AP. There is no second level OF switch. The APs directly connect to the switch/router which connects the FLUX network to campus core.

6 Deployment Issues

- The talac AP takes time to bring its wireless interfaces. So if the controller is running and the Talac is turned on, it connects to the controller even before the wireless lan is up and hence the it doesnt inform

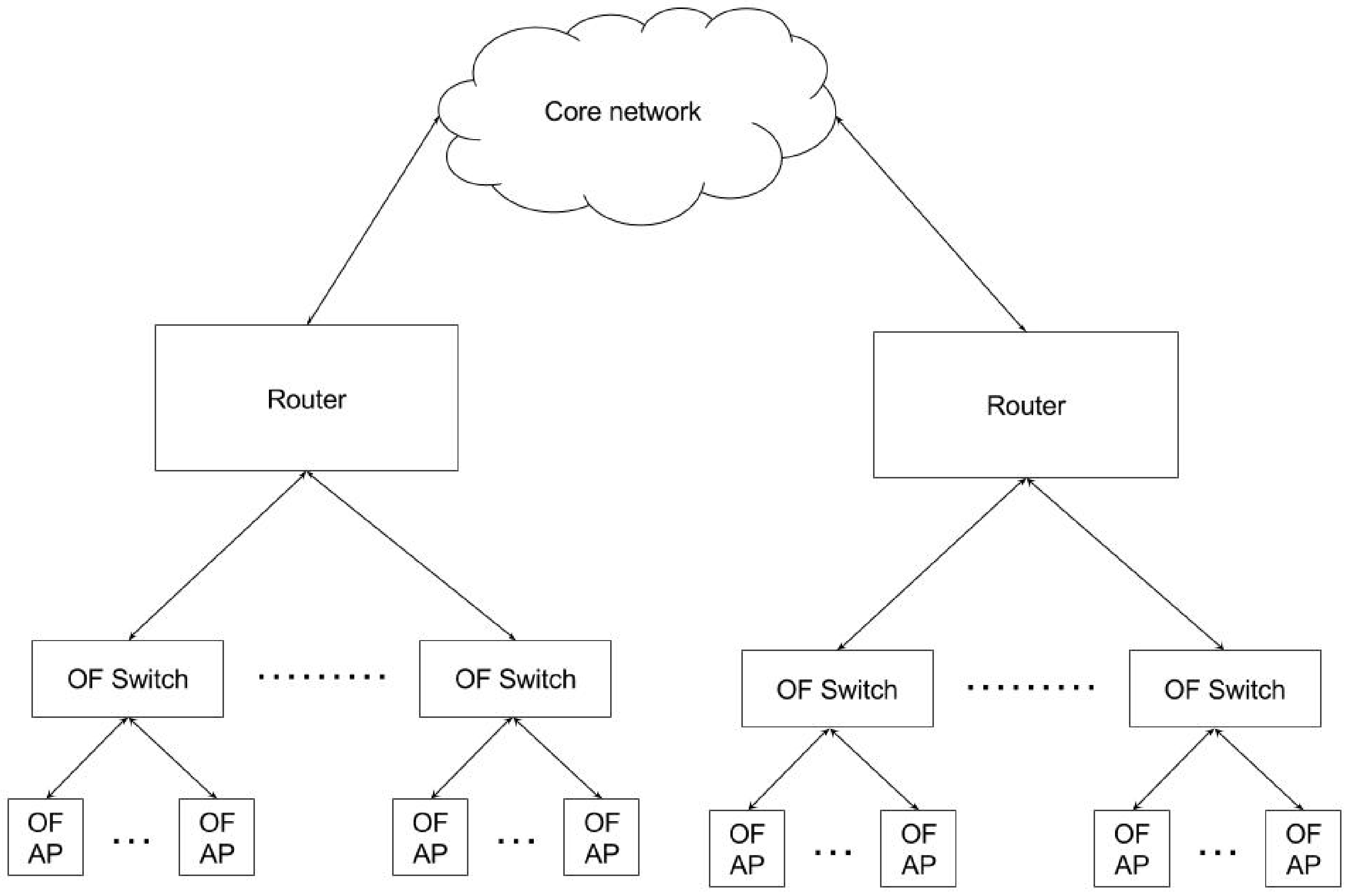

Figure 4: Deployment setup

about its wlan interface during the connection time but is later updated. But the controller expects the wlan interface to up in the beginning itself. So either we should have the talac switches connected first and then turn on the controller(not a good solution) or listen to the interface up notification from talac via openflow and then update the switch db and then trigger the communication. And also if the wireless lan is turned off for some reason, the controller doesnt handle the situation. The notification should be processed and the switch's wlan should not be used to send packets till the wlan is up. In short, the INTERFACE NOTIFICATIONS should be handled

- Unable to restrict flows between hosts connected to the same ap
- In the wired host scenario, its better to find which port the host is connected to and then send packets to that interface instead of sending it to all the interfaces facing the hosts(current scenario)
- Some times, radius gives a user disconnected message even though user is still connected. As of now since the main set of users will be wired machines in the lab, I assume this wont happen with wired clients. So handle this issue later ?

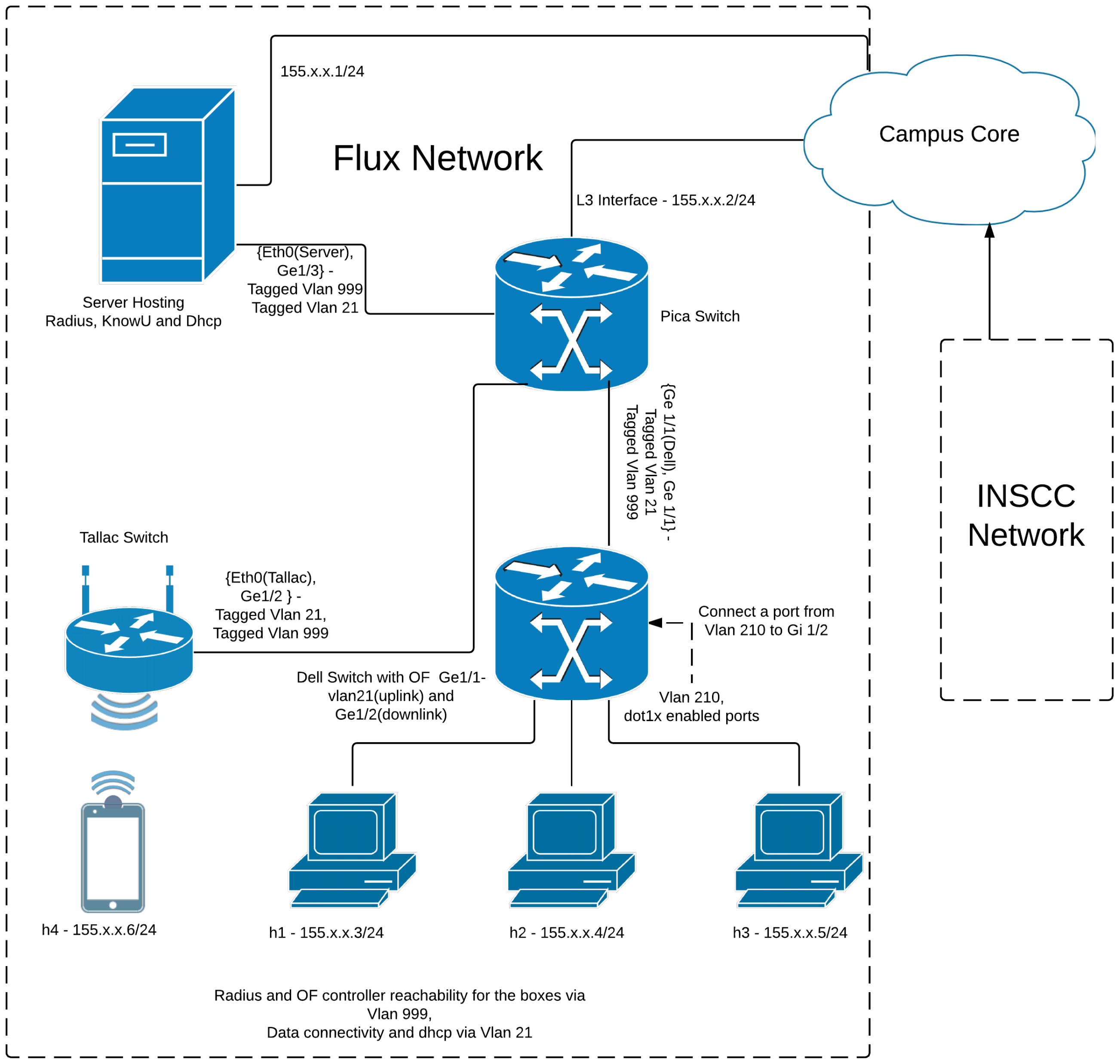
155.x.x.1/24
Flux Network
Campus Core
L3 Interface - 155.x.x.2/24
{Eth0(Server), Ge1/3} - Tagged Vlan 999 Tagged Vlan 21
Server Hosting Radius, KnowU and Dhcp
Pica Switch
{Ge 1/1(Dell), Ge 1/1} - Tagged Vlan 21 Tagged Vlan 999
INSCC Network
Tallac Switch
{Eth0(Tallac), Ge1/2 } - Tagged Vlan 21, Tagged Vlan 999
Connect a port from Vlan 210 to Gi 1/2
Dell Switch with OF Ge1/1-vlan21(uplink) and Ge1/2(downlink)
Vlan 210, dot1x enabled ports
h4 - 155.x.x.6/24
h1 - 155.x.x.3/24
h2 - 155.x.x.4/24
h3 - 155.x.x.5/24
Radius and OF controller reachability for the boxes via Vlan 999,
Data connectivity and dhcp via Vlan 21

Figure 5: Flux Deployment

7 Future work

- The current policies are just in the ip level. It can be made much granular. For example instead of saying one user can access the server in another zone, we can limit further by saying this user can only access the server via this port number.
- Assigning Ip's in a structured way so that, rules can be combined in the edge switches.(If there are n users which fall into 2 zones, give them ips in a way such that each zone gets a continuous ip set, so

that common flows can be combined

- As of now, the user can only be a part of one security group. What if he wants to be a part of multiple security group. ?
- Extend the policy engine to specify the network path that this communication can/should use instead of just saying allow communication
- Allow the policy engine to contain abstract user set instead of a concrete one. Eg: Zone 100 = Allow users who have finished security tests. The users will get added/moved to the zone upon an trigger from the security application. This will bring in more dynamicity to the system
- Currently the policies to a zone can only take another zone as input. Add support to add users also.
- Allow the policy engine to work with more than just IPs. A way to support data based policies. A way to implement something like a taint system developed by Kirk

8 Acknowledgement

I thank my advisor Jacobus Van Der Merwe for guiding me throughout the Master's program, without whom this project would not have been possible. I also thank Joseph R Breen III for his suggestions and feedback which helped me a lot towards this project.

This project is based upon work supported by the National Science Foundation under awards #1302688 and #1642158.