NCSU Internet2 SDN NAC Application

This document describes an application developed by Communication Technologies within the Office of Information Technology at North Carolina State University and has been selected by Internet2 as an example of an Innovative SDN Application.

0. Background

With the “Bring Your Own Device” (BYOD) phenomenon there is an increased need to permit customer/employee/student owned devices onto enterprise networks. In many cases a lightweight authentication mechanism would suffice to permit these devices to obtain access to the network. While industry standard solutions such as 802.1X exist to handle these scenarios they are often difficult to implement because they require supplicant configuration on the client endpoint.

Other Network Admission Control (NAC) solutions work by sending messages to campus switches in order to manipulate traffic flows or to force user authentication. The issue with these systems is that they often require changes (sometimes significant) to be made to each switch on campus where a user attempting to connect a BYOD device might be located. Some NAC solutions overcome this by implementing central services to manipulate traffic flows for known and unknown users but this is done in software on traditional servers and often results in throughput limitations ultimately impacting scalability. We present a different mechanism in the remainder of this document.

1. SDN Application

One of the primary benefits of SDN capable switching equipment is that the control plane can be decoupled from the equipment while the high performance hardware based forwarding plane remains. This inflection point in networking technology ushers in a completely new set of open source applications that were previously not viable due to the need for line rate forwarding (where software based solutions implemented with NICs on general purpose computing systems are insufficient).

We propose utilizing OpenFlow capable switches to implement a NAC solution capable of permitting users to log into the network in order to implement BYOD while still being able to know who and what is on the network.

2. Architecture

There are multiple architectural options for this project - many are still being considered but at least one alternative will be described in subsections below. Please note that multiple options could be implemented over time to permit a highly flexible open source application that could be deployed in multiple environments with different requirements.

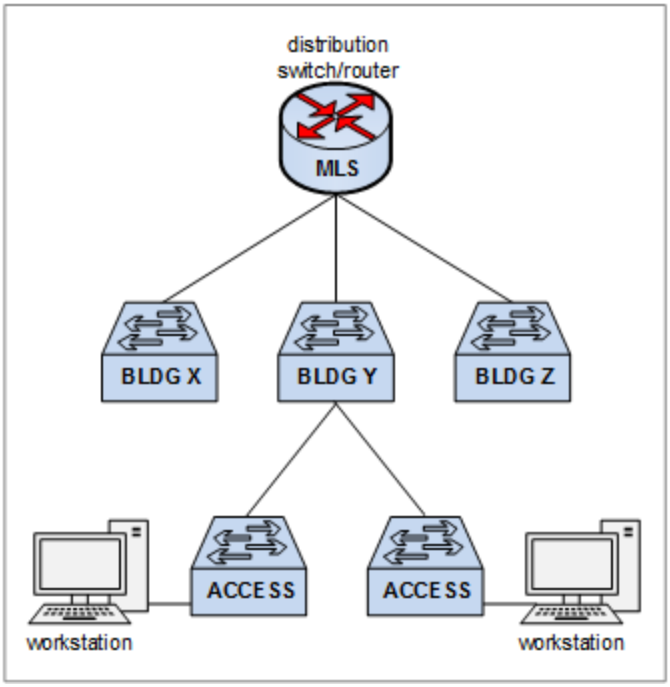
Assume we are interested in tracking the tuple consisting of (username, MAC, IP) as a session.

If the MAC or IP is unknown or changes, we assume the traffic is not part of a valid or known authenticated session and force authentication before permitting subsequent frames to be transmitted.

2.1 OpenFlow Switch as a VLAN Swapping Mechanism

Many campus networks are designed utilizing a classic textbook “Core, Distribution, Access” architecture to permit scalability and to minimize the impact of well-known problems. Within these architectures there is often a tradeoff as to where the distribution router is located. In some cases, each building contains a distribution router. In other cases, the distribution router is located in geographic regions of campus and the buildings within that region connect to the regional distribution router through a building aggregation switch. Through discussion with colleagues we have found the latter option to be the one typically implemented. The costs are lower in that a routing license (and associated hardware) does not need to be purchased for each building. An example of this architecture is shown in Figure 2.1 - Typical Campus Architecture.

Figure 2.1 - Typical Campus Architecture



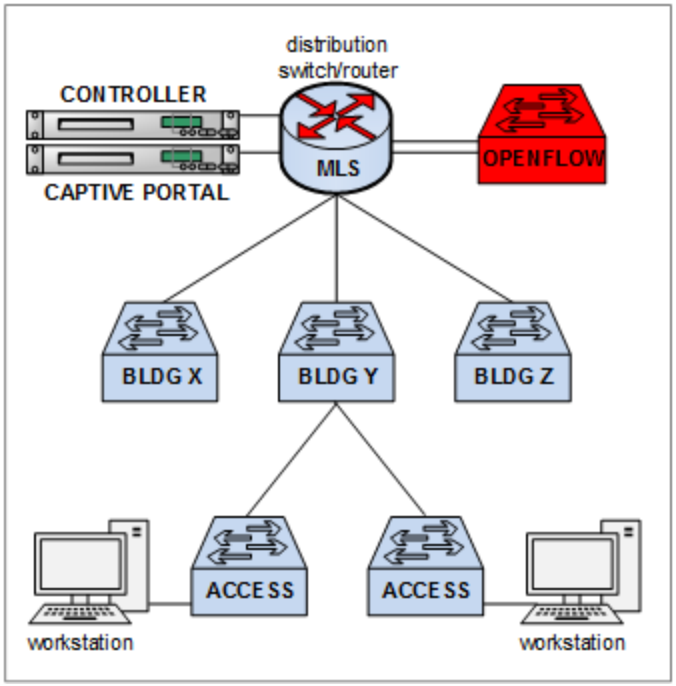
In the example in Figure 2.1, a given building may contain a number of VLANs. The router interface providing default gateway services for hosts within these VLANs lives within the distribution router at the top of the diagram.

To implement a hardware based NAC solution within this environment multiple options are available - but the one discussed here is the option that results in minimal cost and minimal operational changes to the network. The idea involves removing the routed interface for the VLANs presented to the building thus removing the ability for the distribution router to natively provide default gateway services to the hosts within the building. These VLANs containing hosts within the building without a routed interface will be referred to as untrusted VLANs going forward.

A new set of router interfaces will be created on the distribution router that are associated with new VLANs that are different than those used within the building. The same subnets that were previously associated with the building VLANs are then associated with these new interfaces. We refer to these as trusted VLANs going forward.

An OpenFlow capable switch is attached to the distribution switch/router and both the untrusted and trusted VLANs are trunked to a port on this switch (as well as captive portal VLANs). In addition, an OpenFlow controller and web based captive portal are available to the switch. This is illustrated in Figure 2.1.1 below:

Figure 2.1.1 - VLAN Swapping NAC Architecture



The OpenFlow switch is then configured by the OpenFlow controller to send traffic not matching any known flows to the captive portal by mapping traffic from the trusted VLAN to a captive portal VLAN instead of the trusted VLAN. The captive portal presents a web interface with instructions and a username/password dialog box. When a user supplies valid credentials, the controller

installs flows onto the OpenFlow switch such that traffic is mapped from the untrused VLAN to the trusted VLAN for the user in question.

3. Implementation

As mentioned previously, multiple valid architectures are available to implement the NAC solution utilizing SDN. For each architecture there are multiple valid implementation options. This section presents an implementation option for each architecture described previously.

3.1 OpenFlow Switch as a VLAN Swapping Mechanism

The goal of this implementation is to minimize the number of changes to the existing physical network. Consider an organization with the following networks associated with a distribution router in a region of campus:

Figure 3.1.1 - Example Network Topology

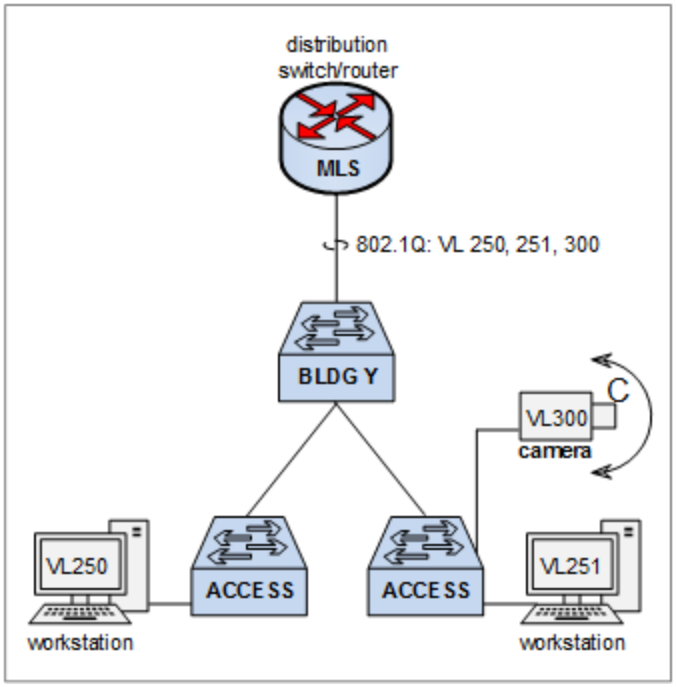


Table 3.1.1 - Example Network Allocation

VLAN	Subnet	Description
250	10.0.1.0/24	Engineering
251	10.0.2.0/24	Sales
300	10.1.0.0/24	Infrastructure

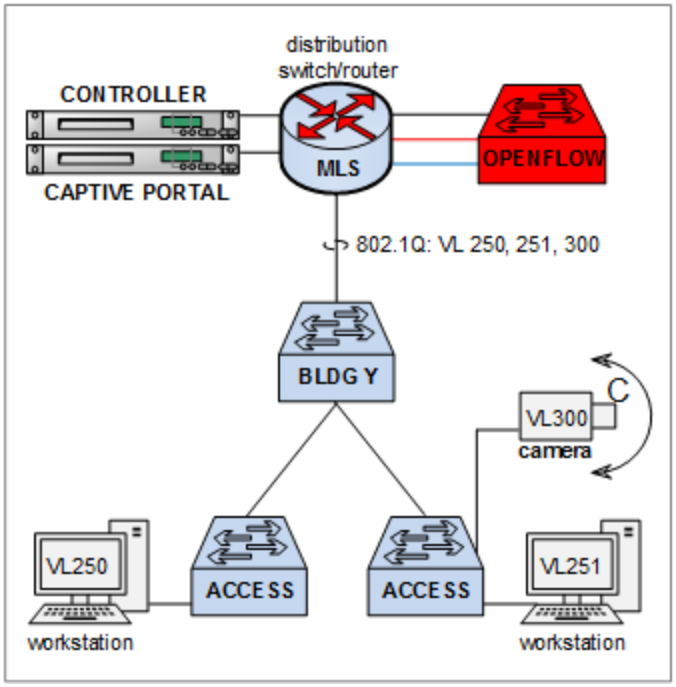
As shown, all VLANs are currently routed via a multi-layer switch (MLS) serving as a regional distribution router. The goal for this example is to deploy NAC functionality on VLANs 250 and 251 and to leave VLAN 300 as-is since it is for hardwired infrastructure within secure locations and is not available for commodity computing use.

As a result, we need to allocate some additional VLANs so that we can remove the router interfaces from VLANs 250 and 251.

Table 3.1.2 - Network Allocation Modification

VLAN	Subnet	Description
250	10.0.0.0/24	Engineering
1250	10.0.0.0/24	Engineering Routed
2250	N/A	Engineering Captive Portal
251	10.0.1.0/24	Sales
1251	10.0.1.0/24	Sales Routed
2251	N/A	Sales Captive Portal
300	10.1.0.0/24	Infrastructure

Figure 3.1.2 - Network Topology Modification



As shown in Figure 3.1.2, all of the changes are to the regional distribution router - not to the rest of the infrastructure within the building. To proceed we attach an OpenFlow capable switch to the MLS. One link (black in Figure 3.1.2) is for management purposes only. The management link permits the OpenFlow switch to be monitored by network operations staff and more importantly permits the OpenFlow switch to establish a connection to an OpenFlow controller.

If the OpenFlow switch that is selected permits traffic to be hairpinned in and out a single port (via OFPP_IN_PORT), only a single data link is required between the MLS and the OpenFlow switch. If the OpenFlow switch does not permit traffic to be hairpinned in and out a single port then two links are required. In Figure 3.1.2, the red link indicates the original untrusted traffic from the building that is to be subject to NAC and therefore carries VLANs 250 and 251 while the blue link indicates VLAN swapped traffic that can flow towards the new VLANs 1250 and 1251 which contain the original router interfaces or towards VLANs 2250 and 2251 which contain the captive portal. Note that VLAN 300 which is not subject to NAC is not modified in any way and continues to have a router interface on the MLS.

In addition to the OpenFlow switch, a pair of servers are required. Although Figure 3.1.2 shows physical servers a single host running multiple VMs is also sufficient. One server runs the POX OpenFlow controller and must have L3 connectivity to the management interface on the OpenFlow switch. The other server provides captive portal services to unauthenticated users and must be L2 adjacent to the MLS (or with slight modification it could have an interface plugged directly into the OpenFlow switch). The captive portal server must have L3 connectivity to the server running the OpenFlow controller so that authentication notifications can be passed

to the controller.

3.1.1 Controller Server

In the example implementation the controller server runs the following software:

- Centos 6.4
- POX betta release
- Python 2.7.5
- MySQL

The MySQL database is used to keep track of authentication sessions. When a user logs into the network via the captive portal, their username, MAC address and IP address are recorded into an entry within the database along with a timestamp. When their session is expired due to inactivity the controller updates the record with a session end timestamp.

The POX controller needs to process authentication updates from the captive portal server so that end-user traffic can be permitted to flow. The POX JSON interface was attempted initially but there were some issues. The POX JSON API permits the replacement of the entire flow table but not the addition of individual flows. In addition, the POX JSON API currently does not permit the setting of flags when adding a flow so that when flows are expired a call back can be triggered (to update the MySQL database in our use case).

Modifications were originally made to POX to permit the above changes, but the logging of the initial session into MySQL still would need to be completed by the captive portal.

Instead of making all of these modifications, it was decided to roll all of them back and to launch an XMLRPC server from within the NAC module running in POX. This XMLRPC server listens to requests from the captive portal server to permit flows from authenticated users, set flags on those flows for expiration notification, and to permit the controller to insert the initial authentication session record into the MySQL database. The use of the XMLRPC interface removed a significant amount of code and configuration data from the captive portal server resulting in a cleaner design.

The NAC appliance is delivered as a single file that is installed into the pox/ext directory and can be launched as follows:

python2.7 pox/pox.py log.level --DEBUG nac

Please note that additional configuration and deployment details are shown in the file titled “NCSU-Internet2-SDN-NAC-README-v1.2”.

3.1.2 Captive Portal Server

In the example implementation the captive portal server runs the following software:

- Centos 6.4
- HTTPD
- PHP

Note that the captive portal server should have a default interface “eth0” that is used for server administration, L3 connectivity to the controller server, etc. In addition, the captive portal server has the following network requirements:

- IP forwarding must be enabled
    - sysctl -w net.ipv4.ip_forward=1
- An interface must be created for each client network that is to be subject to the captive portal functionality. This interface should have the same MAC address as the router interface associated with the client network. The IP address used on this interface is not important and should be from RFC1918 space.
    - eth1: 192.168.0.10 255.255.255.0 # VLAN 2250
    - eth2: 192.168.1.10 255.255.255.0 # VLAN 2251
- Proxy ARP must be configured on each interface other than eth0
    - echo 1 > /proc/sys/net/ipv4/conf/eth1/proxy_arp
    - echo 1 > /proc/sys/net/ipv4/conf/eth1/proxy_arp_pvlan
    - echo 1 > /proc/sys/net/ipv4/conf/eth2/proxy_arp
    - echo 1 > /proc/sys/net/ipv4/conf/eth2/proxy_arp_pvlan
- Routes must be added for each of the customer facing networks
    - route add -net 10.0.0.0/24 dev eth1
    - route add -net 10.0.1.0/24 dev eth2
- IPTABLES must be configured to perform destination based NAT (DNAT)
    - /sbin/iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j DNAT --to 192.168.0.10:80
    - /sbin/iptables -t nat -A PREROUTING -i eth2 -p tcp --dport 80 -j DNAT --to 192.168.1.10:80

Since users attempting to visit http://www.example.com/abc/def/ghi will be sent to the captive portal page and content must be returned to them, a special .htaccess file must be installed onto the www root directory of the captive portal server:

```
Options +FollowSymLinks -MultiViews
RewriteEngine On
RewriteBase /

RewriteCond %{REQUEST_FILENAME} -f [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule .+ - [L]

RewriteRule ([a-z]+) /index.php [L]
```

The above .htaccess file file server any files or directories that actually exist on the captive portal server but will rewrite all other requests to /index.php.

The index.php file mentioned above displays a greeting to the user indicating that they have reached a captive portal page. A hyperlink is presented to take the user to the /nac/index.html page which will require authentication. Multiple authentication mechanisms exist - since this is a web authentication application Shibboleth can be used. For the sake of the demo environment HTTP basic auth is used:

```
# contents of /nac/.htaccess
AuthUserFile /var/www/html/nac/.htpasswd
AuthName Network-Admission-Control
AuthType Basic
require valid-user


# contents of /nac/.htpasswd  (usenrame: guest, password: welcome)
guest:vEu.mKqbIl9no
```

Please note that additional configuration and deployment details are shown in the file titled “NCSU-Internet2-SDN-NAC-README-v1.2”.