# Implementing Scalable Security for Devices Without 802.1x Support

Author: Umer Khan, umer.khan@mail.com
Advisor: Lenny Zeltser

Accepted: November 30, 2022

Abstract

Enterprises often implement 802.1x to control access to wired and wireless networks by authenticating computers using username/password or a digital certificate, but MAC Authentication Bypass (MAB) is used for devices that do not support enterprise 802.1x capability such as printers, industrial control systems, and operating technology machines. MAB is difficult to maintain and devices that use it are often permitted unrestricted network access without other security mitigations. Since MAC addresses are easy to impersonate, this authentication mechanism by itself is ineffective and easy to bypass, thereby granting attackers easy access to enterprise networks. This research analyzes how to implement MAB in a more scalable fashion and how to enhance it with security mitigations such as device fingerprinting for device validation and automated access control lists to restrict network resources a MAB-authenticated device can access.

# 1. Introduction

"Inventory and Control of Enterprise Assets" is the first of the CIS Critical Security Controls, highlighting how foundational it is. This is because it is impossible to secure a network if one does not know and control which systems connect to it. Network and security architects often turn to IEEE 802.1x to control which wired and wireless devices are allowed entry to the network. Most enterprises prefer to allow only approved and "managed" devices onto the network, thereby keeping unknown, unmanaged, and rogue devices off the network. 802.1x generally utilizes user and/or machine credentials (such as username/password, multi-factor authentication, user/machine certificates, or a combination of these) before authorizing a system to gain network access.

A major challenge with 802.1x is that it requires client-side support and, therefore, generally only works on enterprise operating systems such as Windows, Linux, macOS, iOS and Android. 802.1x support is commonly not found on printers, industrial control systems (ICS), and operational technology (OT) devices, as these hosts often run embedded and proprietary operating systems. As a result, while 802.1x works well to serve the needs of software companies, law firms, and other office-type environments where the number of such devices is low and exceptions are rare, it falls apart in operational environments that are industrial in nature, such as factory floors and research laboratories.

The most common method of allowing OT devices on the network is Media Access Control (MAC) Authentication Bypass, called MAB for short. Ethernet and Wi-Fi MAC addresses are unique identifiers burned into physical network interfaces (whether integrated into the motherboard or an add-on card) and are 48-bit numbers generally written as 12 hexadecimal characters for easy readability. MAB requires an organization to maintain a database of MAC addresses of all devices that do not support 802.1x. These devices are then authorized to connect to the network simply by having their MAC addresses in an allowlist. This creates logistical challenges for IT and security teams to constantly maintain a MAB database when devices are added and removed from the environment, but more importantly, it introduces major security issues. Firstly, although MAC addresses are designed to be globally unique, they are easy to spoof, making the

control easy to bypass. While network traffic restrictions could serve as a control to mitigate the security risk of such a weak form of authentication, most organizations typically grant unrestricted network access to these devices because such controls require significant manual effort and further maintenance overhead. This means that simple MAC address impersonation can be used to gain full access to the entire network easily.

While all devices need to perform certain core network functions (such as the ability to request and receive IP addresses from the DHCP server or resolve hostnames to IP addresses using the DNS server), many devices need relatively limited network services beyond that. For example, a printer only needs to be able to communicate to print servers and a video camera only needs to be able to talk to video control and recording servers. If network access control lists could easily be applied to MAB-authenticated devices, they could be locked down to only communicate with what they need ("principle of least privilege"). Thus, even if a malicious actor were to bypass the weak authentication by impersonating a MAC address and gain entry to the network, they would obtain extremely limited access to network resources, and their ability to laterally move through the environment and compromise other enterprise assets would be severely restricted.

## 2. Objectives

While some device manufacturers implement 802.1x, many do not. The objective of this research is to find easy and scalable mechanisms to secure devices that do not support 802.1x authentication. This includes:

- Utilizing an authentication method that is supported by these devices.
- If the authentication method is weak, augmenting it with profiling and identification techniques to validate the device in order to catch any spoofing attempts.
- Classifying such devices into types (manually or automatically) and using these device types to dynamically implement appropriate network access restrictions. Network access does not have to be "all or nothing", and it can be controlled granularly.

The goal of this research is to explore how to achieve the above in a practical and hands-on manner that is tested properly in a lab environment and not merely have a theoretical discussion that may later result in caveats when implemented in the real world. A pragmatic approach should also minimize ongoing IT and security operational involvement by relying upon automation where possible. The solution should be easy to manage, and not overly burdensome for IT and security teams.

# 3. Lab Setup and Initial Configuration

First, a lab environment was built to validate the baseline scenario of using MAB authentication without additional controls. What follows is a description of the hardware and software utilized, the topology, the initial configuration, and initial validation.

## 3.1. Lab Hardware and Software

The specific tools utilized in the lab environment are not so important as the goal is to use standards-based technology. The purpose of this research is to evaluate and demonstrate concepts that extend across enterprise-grade open source or commercial software tools from any vendor. The off-the-shelf tools selected below were easily available for research and were chosen to illustrate and test concepts:

- **An authentication server:** PacketFence, an open-source and full-featured network access control software, installed on a virtual machine. PacketFence runs on Linux and is free, making it easy for anyone to duplicate the work in this research. Commercial solutions at large enterprises can cost millions of dollars. The lab setup used VMware Fusion 12.2.4 running on macOS Big Sur (11.6) on a MacBook Pro (16-inch, 2019), but PacketFence can run on any physical or virtual machine with 4 CPU cores, 16 GB of RAM, and 200 GB of disk space.
- **An authenticator / network switch:** A Cisco Catalyst 3560 switch (WS-C3560CG-8PC-S) running Cisco IOS version 15.2(2)E10. This is a small, quiet, 10-port desktop switch with full enterprise features and Power over Ethernet (POE) support. The features utilized in this lab are generally

available on any enterprise-class end-user switch with no specific dependency on this switch vendor or model.

- **A client with 802.1x support:** An Apple MacBook Pro (16-inch, 2021) based on the Apple M1 Pro chip, running macOS Monterey (version 12.3).
- **Device 1 without 802.1x support:** A Brother HL-L2380DW printer/scanner.
- **Device 2 without 802.1x support:** A Cisco IP Phone 7945 (CP-7945G).
- **Device 3 without 802.1x support:** An Avigilon enterprise security camera.
- **A wireless access point:** A Ubiquiti UniFi AP AC Pro 802.11ac wireless access point.
- **Device 4 to test fingerprinting accuracy:** A Nest T3007ES 3.4 Learning Thermostat 3rd generation Silver.
- **Device 5 to test fingerprinting accuracy:** A 2018 Tesla Model 3 car.
- **Device 6 to test fingerprinting accuracy:** An LG OLED65C7P TV.

Devices 1, 2, and 3 were selected as a representative subset of type of devices in an organization without 802.1x support. In addition to these, devices 4, 5, and 6 were chosen to test fingerprinting capability, i.e., the ability to analyze network traffic to identify the type of device.

## 3.2. Lab Topology

As described in Figure 1, the lab equipment was connected on a flat network. This enabled testing of traffic between hosts on a single subnet, which is important because MAB-connected devices often sit on the same network as each other as well as 802.1x-connected devices. The Wi-Fi devices were out of the scope of testing MAB authentication but were used to test fingerprinting capabilities.

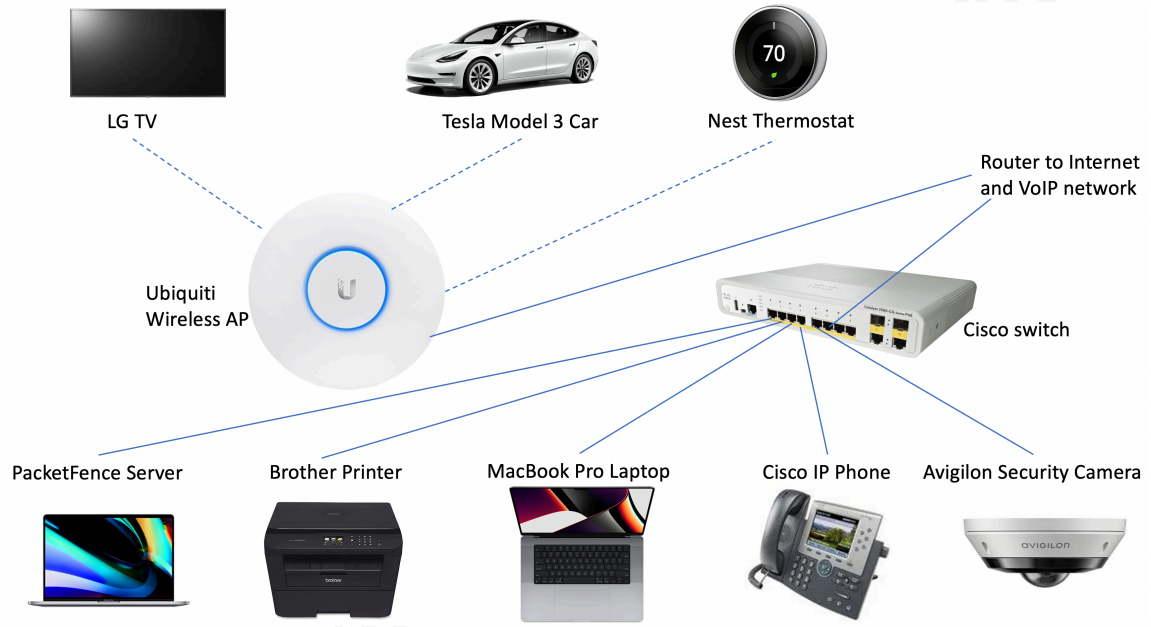



FIGURE 1: THE LAB TOPOLOGY CONNECTS WIRELESS AND WIRED DEVICES ON A FLAT SUBNET.

## 3.3. Basic Lab Configuration

It is assumed that configuring an authentication server (such as PacketFence) and network switches and clients (supplicants) for basic 802.1x authentication are simple tasks and common functions that many enterprises have already performed, so these configurations are outside the primary scope of this research paper. Instructions for setting up these core components of the lab can be found in Appendix I, II, and III for easy duplication of the lab environment.

The testing began by configuring each Ethernet port to support both 802.1x authentication and MAB. The idea here is that every Ethernet port on each end-user-facing switch in the enterprise would be configured this way, making interface configurations easy by not having to worry about which switch ports may have 802.1x-supported devices connected to them, and which ones do not. This allows for devices to move around in a company without the need to make any configuration changes on switch ports. The following configuration was applied to all switch ports:

```
interface GigabitEthernet0/8
 switchport mode access
 authentication host-mode single-host
 authentication order dot1x mab
```

Umer Khan, umer.khan@mail.com

```
authentication priority dot1x mab
authentication port-control auto
authentication periodic
authentication timer restart 10800
authentication timer reauthenticate 10800
mab
no snmp trap link-status
dot1x pae authenticator
dot1x timeout quiet-period 2
dot1x timeout tx-period 3
```

At this point, when any device was connected to a switch port, the switch would first attempt authentication using 802.1x. If that failed (for example, due to lack of support), the device would appear as an unregistered "node" in PacketFence. This configuration was tested by connecting the MacBook Pro to the switch with a successful 802.1x authentication then using the same device but bypassing the 802.1x login screen (by clicking "Cancel") to create a failed 802.1x authentication to test MAB. The MacBook Pro initially also failed MAB since it was not "registered" in PacketFence. Its "Status" was then changed (under "Nodes") to "Registered" in PacketFence which caused MAB to succeed, thereby granting it full access to the network.

With basic 802.1x and MAB functionality validated, the three different wired device types that don't support 802.1x (the printer, the phone, and the security camera) were plugged in. Each device was marked as "registered" in PacketFence to allow MAB to function. Network connectivity was tested for each device by sending a print job to the printer, making a voice call on the phone, and pulling up live video from the security camera on the video server.

## 3.4. Demonstrating MAC Spoofing as a Baseline

To demonstrate the ease of spoofing a MAC address to gain full access to the network in this baseline scenario, the printer was removed from the network and the MacBook Pro was instead configured to spoof the printer's MAC address:

```
admin@mbp ~ % ifconfig en10 |grep ether
      ether 00:e0:4c:02:ad:fd
admin@mbp ~ % sudo ifconfig en10 ether 10:08:b1:9c:b6:f0
```

```
admin@mbp ~ % ifconfig en10 |grep ether
      ether 10:08:b1:9c:b6:f0
```

As expected, when the MacBook Pro was plugged back into the network, it gained immediate and full access to the network by simply impersonating the printer's MAC address. Unrestricted network access was validated by pinging the print server, the IP phone server, and the video recording server:

```
umer@umer-mbp ~ % ping -c 1 -t 1 192.168.1.200 | grep icmp
64 bytes from 192.168.1.200: icmp_seq=0 ttl=62 time=0.162 ms
umer@umer-mbp ~ % ping -c 1 -t 1 192.168.1.201 | grep icmp
64 bytes from 192.168.1.200: icmp_seq=0 ttl=62 time=0.159 ms
umer@umer-mbp ~ % ping -c 1 -t 1 192.168.1.202 | grep icmp
64 bytes from 192.168.1.200: icmp_seq=0 ttl=62 time=0.161 ms
```

## 4. Network Segmentation Options

Given that MAB is a weak type of authentication that is easily spoofed, additional controls are needed to improve security. One of these controls is to limit the attack surface by minimizing what a MAB-authenticated device has access to on the network using access control lists. The goal is to classify MAB-authenticated devices into different types and then allow only the network traffic necessary for that type of device to function. There are several ways to implement such network restrictions, including:

- **Host firewalls**, which are a type of network access control capability built into the host operating system.
- **Network firewalls**, whether physical or virtual appliances. These may be from commercial providers such as Palo Alto Networks, Juniper, or Cisco. Alternatively, they can be "build-it-yourself" appliances using open-source technology such as pfSense, firewalld, or iptables.
- **Access Control Lists (ACLs)** applied at Switched Virtual Interfaces (SVIs), which are the layer-3 interfaces that exist between IP networks / VLANs on routers/switches. These are often called "router ACLs."

- **Port Access Control Lists (PACLs)**, which are applied to individual Ethernet ports on a network switch and can even control traffic between hosts on the same subnet.

Each of these network traffic restriction mechanisms is discussed in greater detail below.

Rather than implement segmentation on network devices, traffic restrictions can also be applied by using host firewalls on enterprise operating systems and managing them centrally through configuration management tools (such as Microsoft Endpoint Manager for Windows, Jamf for macOS, or Puppet for Linux). However, hosts that do not support 802.1x also tend not to support host firewalling as they do not run enterprise operating systems.

Network firewalls have their many applications for a variety of use cases, but they don't scale well in performance or cost for MAB-authenticated devices which tend to be spread across the enterprise network edge. Enterprises run high-performance layer-2 and layer-3 network switches with large backplanes and switching capacities (both in terms of packets per second throughput and the amount of bandwidth they can handle through them) and dedicated network firewalls do not scale well to such high speeds. Additionally, network firewalls create chokepoints in the network as all traffic must be sent through them. This adds latency, increases jitter, and decreases the total amount of available bandwidth between nodes on a network. If an enterprise has many sites or buildings, that either further reduces performance by bringing all the traffic back to a more central chokepoint or it increases cost with the need to purchase many more firewalls. In addition to the direct upfront cost of the appliance, there are indirect costs for rack space, power, cooling, annual vendor hardware maintenance and software support, and labor for ongoing administration and management.

Another disadvantage of network firewalls is that they only filter traffic between network segments or Virtual Local Area Networks (VLANs). They do not provide a mechanism to block traffic between hosts *within* a particular network segment or VLAN. This also creates the issue of having to cleanly organize hosts into VLANs, which can be

difficult for enterprises that have large already-built networks where VLANs may not have been designed and implemented with security in mind.

A final problem with firewalls is the complexity of layer-3 network configuration. The firewall either directly must serve as the default gateway of a network segment (which is not desired as firewalls often do not provide the same level of "networking" features as routers/switches do) or they require complex Virtual Router and Forwarding (VRF) configuration for the router/switch to continue serving as the default gateway while enabling the firewall to be inserted into the traffic flow.

Router ACLs work like network firewalls with some differences. Like network firewalls, they filter traffic at the network segment / VLAN level. Therefore, they, too, require hosts to be organized cleanly into VLANs. They are also not able to block traffic between hosts on the same network segment. However, because they are built into routers and switches, the biggest advantage they provide is no degradation in performance (modern routers and switches can handle access lists at layer-4 with no impact on performance) and no additional cost. Their disadvantages over firewalls are that they do not support stateful inspection of traffic, nor do they support the superb inspection capabilities of the modern Next Generation Firewalls (NGFWs) that operate at the application layer. They also do not support the robust logging capabilities of network firewalls.

PACLs have been around for about two decades and provide incredibly powerful capabilities. They are applied on a network switch at a layer-2 Ethernet port, but they can operate and filter traffic at layer-3 and layer-4. Because they are applied right at the edge of the network, they can filter traffic immediately as it enters the switch, thereby providing the ability to protect hosts on the same network segment / VLAN from each other. PACLs are handled entirely in the switch ASIC hardware and have no impact on performance (neither to the maximum throughput of traffic nor to the number of packets per second that can be switched). Additionally, since the traffic does not need to be brought back to a central router interface or firewall, there is no increase in network latency, no impact to jitter, and no creation of uplink bandwidth bottlenecks.

There are two additional reasons why PACLs are desirable for restricting traffic from MAB-authenticated devices that don't support 802.1x:

- PACLs can be implemented in an existing enterprise without any need to create new VLANs or move devices between VLANs. Moving a device generally necessitates changing its IP address, which can become a major one-time operational burden as it requires identifying the owner of the device, coordinating an outage, moving the device to the new VLAN, re-IP addressing the device itself (which is often a manual process that involves physically touching the device), and updating all systems that may reference the device by IP address.
- PACLs also remove a requirement that exists with router ACLs or host firewalls: after a host comes online and is identified and authenticated using MAB, its switch port must be shut down and then brought back online for the device to go through a VLAN change. This is because router ACLs or host firewalls (which only filter traffic crossing VLANs) require the device to be placed in a specific VLAN. When a host first connects to a switch port, it is placed in the default VLAN by the switch. The switch then communicates with the RADIUS server and determines the specific destination VLAN in which to place the device (based on its type/role). Because embedded devices do not handle such VLAN transitions well (where the port first comes up in one VLAN and then after RADIUS communication is moved to another VLAN), the switch port must generally be "shut" and then quickly "unshut" for the device to request and receive an IP address in the new VLAN. This process increases the total amount of time it takes for such a device to finally come online on a network. With PACLs, a device can remain in the VLAN it initially connected to and the PACL handles traffic filtering without needing any VLAN changes or port shut / unshut.

The primary disadvantages of PACLs compared to network firewalls are that they do not offer stateful inspection, application layer inspection, or logging, but these are not necessary capabilities for this use case. Stateful inspection allows a firewall to track

source and destination IP addresses, source and destination ports, and other connection information. The firewall can intelligently drop certain types of traffic which may not be recognized as “bad” by non-stateful firewalls by recognizing if particular network traffic is initiating a conversation or in the middle of a conversation. Application layer inspection allows a firewall to validate that the network traffic truly demonstrates the expected features and characteristics of the application that is supposedly running on a particular TCP or UDP port. While both are useful advanced firewalling features, they are not as vital for MAB-authenticated devices on the internal network, and some compromise in these abilities is acceptable.

The table below summarizes the advantages and disadvantages of using these three segmentation methods for MAB devices on an 802.1x network.

| **Physical or virtual network firewalls** | **“Router ACLs” on SVIs** | **Port Access Control Lists (PACLs)** |
|---|---|---|
| Work at VLAN / layer-3 boundaries only (inter-VLAN traffic) | Work at VLAN / layer-3 boundaries only (inter-VLAN traffic) | Works at every switch port (all traffic goes through them) |
| Require possible move of devices to "appropriate" VLAN, re-IP-addressing | Require possible move of devices to "appropriate" VLAN, re-IP-addressing | Can be applied without having to "move" or "re-IP" a device |
| Require "VLAN change" and shut/no-shut on the port | Require "VLAN change" and shut/no-shut on the port | Can be applied without having to "change VLAN" and shut/no-shut port |
| Provide best inspection and logging abilities | Inspection / logging nowhere near as good as a real firewall | Inspection / logging nowhere near as good as a real firewall |

| | | |
|---|---|---|
| (stateful; based on deep packet inspection, etc.) | (mostly layer 3 and layer 4) | (mostly layer 3 and layer 4) |
| Require adding a “hop” / bottleneck to the network. Scale challenges. | Scale without impact in performance | Scale without impact in performance |
| VLANs have scaling limitations (too many spanning tree instances, etc.) | VLANs have scaling limitations (too many spanning tree instances, etc.) | Work fine with a small number of VLANs |
| Do not depend on PACL support in switches | Do not depend on PACL support in switches | Switches must support PACLs and have large enough TCAMs |

TABLE 1: COMPARISON OF NETWORK SEGMENTATION OPTIONS FOR MAB-AUTHENTICATED DEVICES.

Enterprises can select the best-fit network segmentation method for their business based on Table 1. With a desire to maintain a high-performance network and minimize upfront work by not having to “re-VLAN” devices, this research proceeds with port access control lists.

## 5. Applying Port Access Control Lists

Once it was determined that PACLs will be used for this research, the next step was to figure out how to apply them to switch ports “dynamically.” After all, most networks do not maintain a “static” configuration, where a particular device is always connected to a specific switch and port. Because hosts move around, mapping ACLs statically to switch ports would result in the wrong ACLs being applied to the wrong hosts, or worse, no ACLs applied to certain hosts at all.

## 5.1. Options for Applying ACLs

Researched revealed three broad mechanisms for dynamically applying a PACL to the switch port of a MAB-authenticated device based on device type:

1. Create ACLs for each device type on every network switch as a part of the global switch configuration, and dynamically apply these ACLs to individual switch ports (when a device connects to them and performs MAB-based authentication) using RADIUS attributes.
2. Create ACLs for each device type centrally on the authentication server, and have the switch dynamically download these ACLs and apply them to individual switch ports using RADIUS attributes.
3. Create ACLs for each device type centrally on the authentication server or another system and dynamically create them on the switch and apply them to individual switch ports through an out-of-band mechanism such as SSH or SNMP.

The first two mechanisms were available in the lab environment as they are supported by the PacketFence authentication server, so they are discussed in detail in this research paper. The third mechanism is not supported by PacketFence so instead it was tested on a commercial Forescout appliance (outside the lab environment) and the discoveries are summarized for educational purposes as follows:

- When the first device of a kind is connected to a network switch, the authentication server initiates an SSH connection to the switch and creates the ACL on the switch line-by-line. It then applies that ACL to the switch port where the device is connected.
- When another device of the same kind is connected to the same network switch, the authentication server initiates an SSH connection to the switch. It simply applies the pre-existing ACL to the switch port where the new device is connected.
- When a device is disconnected, the authentication server initiates an SSH connection to the switch and un-applies the ACL from that switch port.

- When the last device of a kind on a switch is disconnected, the authentication server initiates an SSH connection to the switch, removes the application of the ACL from the switch port, and completely deletes the ACL from that switch.
- If any changes are made to the ACL, the authentication server initiates an SSH connection to the switch, removes the application of the ACL from all relevant switch ports, completely deletes the existing ACL, re-enters the new ACL, and re-applies the ACL to all relevant ports.

Since the above changes are all performed *after* a host comes online or offline and made over SSH, as if a user was entering keystrokes on the network switch through an interactive terminal session, there is some delay (of a few seconds or minutes) for these changes to take place. Forescout does not support creating, applying, and removing ACLs using SNMP, but SNMP MIBs exist for ACL management and could serve as a viable mechanism. Overall, the delays associated with out-of-band ACL management (through SSH, SNMP, or any other such mechanism) are not ideal as they leave a switch port with unrestricted network traffic for too long, whereas RADIUS installs an access control list before any traffic is allowed through the switch port.

In this research, both RADIUS mechanisms were tested in the lab environment: the switch fetching the ACL name from the authentication server to apply a pre-existing ACL to the port, and the switch dynamically downloading the entire ACL from the authentication server and then applying it to the port. To determine the ACL, packet traces were captured and analyzed, and the following IP protocols and ports were identified as necessary for the three types of MAB-authenticated devices to function:

| **Device type** | **Source protocol** | **Destination IP address** | **Destination protocol** |
|---|---|---|---|
| Printer, security camera, IP phone | BOOTPC (UDP) | Any | BOOTPS (UDP) |
| Printer, security camera, IP phone | DNS (UDP, TCP) | Any | Any (TCP, UDP) |

| | | | |
|---|---|---|---|
| Printer | Any (TCP) | Print server (192.168.1.200) | Any (TCP) |
| Security camera | Any (UDP) | Recording server (192.68.1.201) | Any (UDP) |
| IP Phone | Any (TCP, UDP) | Communication Manager server (192.168.1.202) | Any (TCP, UDP) |
| IP Phone | Any (UDP) | VLAN with other phones and VoIP devices | Any (UDP) |

TABLE 2: PROTOCOLS/PORTS IDENTIFIED AS NEEDED FOR MAB-AUTHENTICATED DEVICES

Note: In the last four rows of the table, it is possible to restrict the traffic flows further by specifying the exact TCP and UDP port numbers, but it was deemed sufficient to restrict traffic to certain destination IP addresses. Each of these use cases was found to use a range of port numbers, so a balance was sought between restricting fully to the minimum TCP/UDP ports required and simplifying the ACL configurations for ease of administration.

As discussed earlier, VLANs with router ACLs are not granular enough, still allowing communication between hosts on a given network segment. Out of the box, PacketFence supports VLANs-based controls. To use PACLs instead, “roles” (i.e., types of devices/users) were defined in PacketFence. This was done under Configuration > Policies and Access Control > Network Devices > Switches, by clicking on the switch IP address to edit it, and then clicking the "Roles" tab, disabling "Role by VLAN ID", and enabling "Role by Access List", and then clicking “Save”.

## 5.2. Applying Pre-Existing ACLs on the Switch Port

As a first configuration scenario, access control lists were created on the switch and “roles” were used on the authentication server. This means that when a device of a particular type (“role”) is connected to the switch, the authentication server (through RADIUS) specifies a pre-existing access list on the switch by name to be applied to the Ethernet port for that device.

ACLs were created on the switch as follows:

```
ip access-list extended PRINTER
 permit udp any eq bootpc any eq bootps
 permit udp any any eq domain
 permit tcp any any eq domain
 permit tcp any host 192.168.1.200
 deny ip any any
ip access-list extended CAMERA
 permit udp any eq bootpc any eq bootps
 permit udp any any eq domain
 permit tcp any any eq domain
 permit udp any host 192.168.1.201
 deny ip any any
ip access-list extended PHONE
 permit udp any eq bootpc any eq bootps
 permit udp any any eq domain
 permit tcp any any eq domain
 permit tcp any host 192.168.1.202
 permit udp any host 192.168.1.202
 permit udp any 192.168.2.0 0.0.0.255
 deny ip any any
```

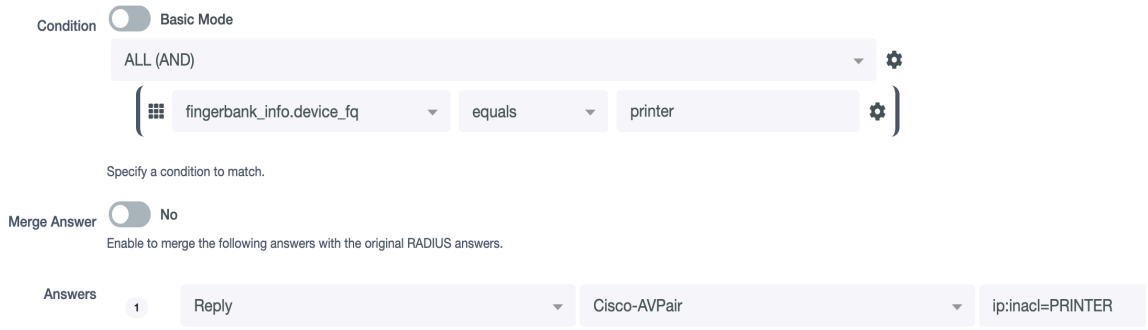
These access lists map to the network traffic observed from these devices (listed in Table 2), and are relatively simple to understand, administer, and manage. By not overcomplicating them, significant security value can be gained while not introducing much of a management burden.

A role was then created for each device type ("printer", "camera", "phone") in PacketFence under Configuration > Policies and Access Control > Roles, and the configuration for each of the three MAB-authenticated devices was modified to use its respective role under "Nodes."

Finally, RADIUS filters were set up in PacketFence to map each device type ("role") to its respective ACL name on the switch:

1. Log in to PacketFence at https://ip_address:1443 using a web browser. Use username admin.

2. Click on the "Configuration" menu.
3. Expand "Advanced Access Configuration" and click "Filter Engines".
4. Select "RADIUS Filters."
5. Click "New Filter".
6. For the "Name," enter "Printers-apply" and description, "Apply pre-existing ACL to printers"
7. Click on the settings icon and click "Add Value".
8. Create a condition where the "user_role" for the field "equals" for the operator and "printer" for the value.
9. Click "Add Answer"
10. Create an answer where the first type is "Reply", the second type is "Cisco-AVPair" and the value is "ip:inacl=PRINTER":




Similar mappings were created for cameras and phones.

RADIUS debugging was enabled on the network switch (in order to observe the ACL name being fetched by the switch from the authentication server) and the three Ethernet ports were shut down and brought back online. As expected, an appropriate ACL was automatically applied to the switch port for each device. As an example, for the printer:

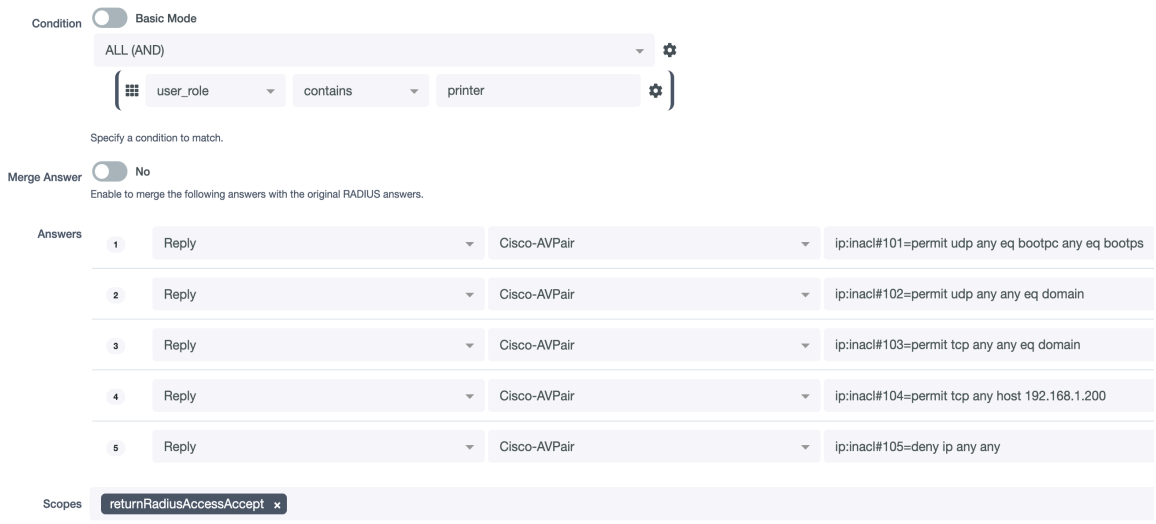
```
*Jan  2 06:39:59.592: RADIUS(00000000): Sending a IPv4 Radius
Packet
*Jan  2 06:39:59.592: RADIUS(00000000): Started 2 sec timeout
```

```
*Jan  2 06:39:59.671: RADIUS: Received from id 1645/12
192.168.1.4:1812, Access-Accept, len 78
*Jan  2 06:39:59.671: RADIUS:  authenticator 26 F2 64 F9 06 14 EE
2F - 7A 70 84 BF A7 8C 54 63
*Jan  2 06:39:59.676: RADIUS:  Vendor, Cisco      [26]  24
*Jan  2 06:39:59.676: RADIUS:   Cisco AVpair      [1]   18
"ip:inacl=PRINTER"
*Jan  2 06:39:59.676: RADIUS:  Reply-Message      [18]  34
*Jan  2 06:39:59.676: RADIUS:   52 65 71 75 65 73 74 20 70 72 6F
63 65 73 73 65  [Request processe]
*Jan  2 06:39:59.676: RADIUS:   64 20 62 79 20 50 61 63 6B 65 74
46 65 6E 63 65  [ d by PacketFence]
*Jan  2 06:39:59.676: RADIUS(00000000): Received from id 1645/12
*Jan  2 06:39:59.676: RADIUS/DECODE: Reply-Message fragments, 32,
total 32 bytes
```

Tests were performed to validate that each device only had the minimum amount of network access needed for that type of device and that it could not pass any other traffic through the switch port, even if the destination was within the same VLAN. Allowed traffic was tested through a print job, a voice call, and a live video feed. Denied traffic was tested from the 802.1x-connected MacBook Pro. As expected, the laptop was unable to ping the printer, phone, and security camera due to the ACLs applied for these MAB-connected devices.

## 5.3. Downloading Dynamic ACLs to the Switch Port

The three RADIUS filters configured in Section 5.2 were designed to test the first configuration scenario: selecting a pre-existing ACL on the switch by name. They were now disabled in PacketFence to test the second configuration: having the switch download the entire ACL from the authentication server. Three new RADIUS filters were created to test ACL downloads by the switch from the authentication server. For example, the printer filter had the following condition and answers:




The three switch ports were disabled and re-enabled to force a new MAB / RADIUS authentication. As expected, the switch downloaded the appropriate ACL from the authentication server during the RADIUS negotiation and applied it to each switch port. As an example, for the printer:

```
*Jan  2 07:09:24.366: RADIUS: Received from id 1645/15
192.168.1.4:1812, Access-Accept, len 301
*Jan  2 07:09:24.366: RADIUS:  authenticator 9B 00 7E 4B 8E 83 9A
D5 - 59 FD 3E 40 E5 38 90 CF
*Jan  2 07:09:24.366: RADIUS:  Reply-Message       [18]  34
*Jan  2 07:09:24.366: RADIUS:   52 65 71 75 65 73 74 20 70 72 6F
63 65 73 73 65  [Request processe]
*Jan  2 07:09:24.366: RADIUS:   64 20 62 79 20 50 61 63 6B 65 74
46 65 6E 63 65  [ d by PacketFence]
*Jan  2 07:09:24.366: RADIUS:  Vendor, Cisco       [26]  59
*Jan  2 07:09:24.366: RADIUS:   Cisco AVpair       [1]   53
"ip:inacl#101=permit udp any eq bootpc any eq bootps"
*Jan  2 07:09:24.366: RADIUS:  Vendor, Cisco       [26]  49
*Jan  2 07:09:24.366: RADIUS:   Cisco AVpair       [1]   43
"ip:inacl#102=permit udp any any eq domain"
*Jan  2 07:09:24.366: RADIUS:  Vendor, Cisco       [26]  49
```

```
*Jan  2 07:09:24.366: RADIUS:    Cisco AVpair       [1]   43
"ip:inacl#103=permit tcp any any eq domain"
*Jan  2 07:09:24.366: RADIUS:  Vendor, Cisco       [26]  54
*Jan  2 07:09:24.366: RADIUS:    Cisco AVpair       [1]   48
"ip:inacl#104=permit tcp any host 192.168.1.200"
*Jan  2 07:09:24.366: RADIUS:  Vendor, Cisco       [26]  36
*Jan  2 07:09:24.366: RADIUS:    Cisco AVpair       [1]   30
"ip:inacl#105=deny ip any any"
*Jan  2 07:09:24.371: RADIUS(00000000): Received from id 1645/15
*Jan  2 07:09:24.371: RADIUS/DECODE: Reply-Message fragments, 32,
total 32 bytes
```

Again, tests were performed to validate that the ACLs worked as expected. Additionally, to validate improved security, the MacBook Pro was once again configured to spoof the printer. This time, while it gained network access and was able to ping the printer server, it was not able to ping any other device on the local network or the Internet.

```
umer@umer-mbp ~ % ping -c 1 -t 1 192.168.1.200 | grep icmp
64 bytes from 192.168.1.200: icmp_seq=0 ttl=62 time=0.165 ms
umer@umer-mbp ~ % ping -c 1 -t 1 192.168.1.201 | grep icmp
umer@umer-mbp ~ % ping -c 1 -t 1 192.168.1.202 | grep icmp
umer@umer-mbp ~ % ping -c 1 -t 1 www.google.com | grep icmp
```

This was a significant improvement. If a MAB-authenticated device was now spoofed, network access was only allowed to the minimum necessary for the spoofed device.

## 6. Device Profiling as an Added Control

802.1x server software (such as PacketFence or other commercial solutions) often includes a “device type detection” feature to automatically identify and classify devices as they join the network. When a device comes online on the network, its network traffic can be analyzed and profiled to validate that it behaves like devices of its type are expected to. This feature may look at a combination of the MAC address, DHCP request data, HTTP User-Agent request header, and TCP and UDP sockets to classify devices. However, this same feature can also be used to provide a layer of protection against a

malicious actor spoofing the MAC address of a permitted device on the network. For example, a particular MAC address populated by an IT person in an enterprise's MAB database is registered as a conference room iPad that only needs access to the scheduling server, but it is detected by the authentication server to be sending web requests with the user agent of a Linux machine. This would point to a mismatch between the device type in the database and the actual host connected to the network and therefore indicate potential spoofing.

The first half (24 bits) of an Ethernet MAC address is known as the Organizationally Unique Identifier (OUI), and it identifies a unique vendor or manufacturer. To help identify and classify a device, this part of the MAC address can be combined with information gathered from the "DHCP request" that the device sends to look for a DHCP server and request an IP address. When a device comes online on the network, if it is not configured with a static IP address, it generally uses Dynamic Host Configuration Protocol (DHCP) to dynamically request an IP address from a DHCP server. As a part of this DHCP request, the client may ask for certain DHCP "options" (such as the DNS server, the default gateway, etc.). It turns out that the specific options that are requested and the order in which they are requested are relatively unique (particularly when combined with the MAC address) and can help identify the operating system and/or type of device (Fingerbank, 2022). DHCP for IPv6 (DHCPv6) also includes the options in a specific order and further adds an enterprise identifier which further helps with identification and classification (Fingerbank, 2022).

While not necessary, capturing the HTTP User-Agent of the device from the network traffic can enhance and improve the accuracy of device identification (Fingerbank, 2022). The User-Agent is a string that a Web browser (or any sort of embedded HTTP agent on a host) sends to web servers and it often contains details about the operating system and web browser version. Certain patterns found in TCP and UDP sockets can also help improve device identification accuracy (Fingerbank, 2022).

PacketFence supports integration with Fingerbank, a cloud-based fingerprinting database / API offered as Software as a Service (SaaS). A free Fingerbank account was created for this research, an API key was received, and it was configured in PacketFence

(see Appendix IV for detailed configuration steps). This allowed PacketFence to start polling the Fingerbank cloud API and automatically identify all connected "nodes", i.e., the devices it discovered on the network.

After waiting a few minutes, as could be seen under "Nodes", clicking a device, and then clicking the "Fingerbank" tab, PacketFence had already accurately classified the various devices on the network:

| **MAC Address** | **DHCP Fingerprint (i.e. DHCP options and their sequence in the DHCP request)** | **Device Class** | **Device Manufacturer** | **Device Type** | **Fully Qualified Device Name** |
|---|---|---|---|---|---|
| 00:e0:4c:02:ad:fd | 1,121,3,6,15,108,114,119,252,95,44,46 | Mac OS X or macOS | REALTEK SEMICONDUCTOR CORP. | Apple MacBook Pro | Operating System/Apple OS/Mac OS X or macOS/Mac OS X/Apple MacBook Pro |
| cc:88:26:54:44:49 | 1,3,26,252,42,15,6,12 | Automotive, Energy and Tools | LG Innotek | Tesla Model 3 | Automotive, Energy and Tools/Automotive/Tesla/Tesla Model 3 |
| 10:08:b1:9c:b6:f0 | 6,3,1,15,66,67,13,44,2,42,12 | Printer or Scanner | Hon Hai Precision Ind. Co.,Ltd. | Brother Printer | Printer or Scanner/Brother Printer |
| 34:db:fd:cd:5d:eb | 1,66,6,3,15,150,35 | VoIP Device | Cisco Systems, Inc | Cisco IP Phone CP- | VoIP Device/Cisco VoIP/Cisco IP |

| | | | | | |
|---|---|---|---|---|---|
| | (DHCP vendor: Cisco Systems, Inc. IP Phone CP-7945G) | | | 7945 G | Phone/Cisco IP Phone CP-7945G |
| 00:1f:92: 01:04:3e | 1,3,6,12,15,28,42 (DHCP vendor: udhcp 1.22.1) | Audio, Imagi ng or Video Equip ment | Motorola | Avigil on H4F Dome Came ra | Audio, Imaging or Video Equipment/Camer a/Surveillance Camera/Avigilon/ Avigilon H4F Dome Camera |
| 18:b4:30: 6d:53:16 | 3,1,252,42,15,6,12 | Intern et of Things (IoT) | Nest Labs Inc. | Nest Ther mosta t | Internet of Things (IoT)/Thermostat/ Nest Thermostat |
| b4:e6:2a: 40:eb:e8 | 252,3,42,15,6,1,12 | Audio, Imagi ng or Video Equip ment | LG Innotek | LG TV - Web OS | Audio, Imaging or Video Equipment/Televi sion/LG TV/LG TV - WebOS |

TABLE 3: DEVICE FINGERPRINTING RESULTS

The next step was to automatically map devices to device types. Each device (originally set as “printer”, “camera”, “phone”) in PacketFence was changed to the “default” role under “Nodes”, basically removing the statically-configured mappings between devices and device types. The RADIUS filters in PacketFence were then adjusted to use Fingerbank fingerprint matching instead of static roles:

```
node_info.device_type contains Printer
node_info.device_type contains IP Phone
node_info.device_type contains Camera
```

After these changes, the Ethernet ports were disabled and re-enabled, as expected, ACLs were downloaded and applied by the switch to each port as expected.

The MacBook Pro was again configured to spoof the MAC address of the printer. After connecting to the switch and bypassing the 802.1x login screen, it succeeded MAC address bypass. However, PacketFence detected it as a "Mac OS X or macOS" device within seconds and the switch denied it any access to the network.

# 7. Summary of Findings and Recommendations

Through this research in the lab environment, the security of MAB-authenticated devices was significantly enhanced by applying a set of scalable and easy-to-manage controls.

## 7.1. Findings

The baseline scenario was set up to allow MAB for devices such as printers, security cameras, and desktop phones. These devices were easily impersonated by spoofing a MAC address and unrestricted access to the network was gained.

Many types of network access control were evaluated, including host firewalls, router ACLs, network firewalls, and port access control lists. It was determined that port access control lists were the best fit for MAB-authenticated devices as they provide the ability to restrict traffic within network segments without any compromise on network performance.

Two types of port access control lists were tested – those that are pre-defined on the network switch and selected by the switch by ACL name based on RADIUS communication with the authentication server, and those that exist on the authentication server and are downloaded by the switch when needed. Devices were classified into different "roles" (device types) manually, and the appropriate access list was applied to the switch port for each device type. With access control lists, the MAB-authenticated devices were impersonated by spoofing a MAC address, but the network access gained was extremely restricted, making lateral movement much more difficult, while alerting defenders and giving them the opportunity to detect and react to such an incident.

As the final control and as a step towards more automation, device profiling and fingerprinting was implemented in order to avoid having to classify devices into roles manually. PacketFence, through the help of the Fingerbank database, was accurately able to profile and identify all device types and assign ACLs dynamically and then apply them to the switch port. Once again, spoofing a MAB-authenticated device was attempted, but this time no network access was gained.

## 7.2. Recommendations and Topics for Further Research

While the tests performed in this research were real and hands-on, further work should be performed before scaling such a solution to a large enterprise. These questions can be future topics of research:

- The final lab set up was able to block all network traffic in the case of simple MAC address spoofing. What would happen if a sophisticated actor crafted DHCP packets to emulate the device type of an authorized MAB-authenticated device? How hard would it be to bypass these controls?
- What security operations alerts can be created to help detect MAC address spoofing and other types of attacks on 802.1x or MAB authentication?
- How much can the industry rely upon fingerprinting from a service such as Fingerbank? Results in our lab were very accurate with a small sample size. How accurate are the results when applied across tens of thousands of devices of hundreds of types?
- Can tools such as nmap or Nessus be used for more advanced fingerprinting, for example by identifying the TCP/UDP ports, studying responses to traffic against those ports, and looking into the banners received, etc.?
- What are the best mechanisms to determine the protocols and ports needed in the network access control lists? Is it NetFlow? Should packet captures be used? Is there some other better mechanism?
- Port access control lists make use of Ternary Content Addressable Memory (TCAM) in a network switch. Is the amount of TCAM in common network

switches large enough to accommodate the access control lists required for a typical number of MAB-authenticated devices in an enterprise?

## 8. Conclusion

Organizations that have large numbers of devices lacking 802.1x support find it burdensome to manage network authentication and authorization. This research has shown that MAC Authentication Bypass (MAB), device fingerprinting, classification, and automated access control lists can reasonably secure these devices and significantly mitigate risk. Device identification through profiling and fingerprinting proved to be accurate and can aid in validating devices and detecting spoofing. Network access control lists of different types were evaluated for feasibility and scale. Port Access Control Lists (PACLs) were perhaps the most scalable and easy mechanism while being the most secure as they apply to all traffic in and out of switch ports and not just network traffic that traverses VLAN boundaries. PACLs were tested using multiple mechanisms and applied dynamically to different device types. It is highly recommended that enterprises apply all these additional mitigations on their network to strengthen their implementation of CIS Critical Security Control 1, and in particular, sub-controls 1.6 (to address unauthorized assets) and 1.7 (to deploy port-level access control).



Umer Khan, umer.khan@mail.com

# References

Fingerbank, *About Fingerbank.* (2022, 11 15). Retrieved from https://www.fingerbank.org/about/

Cisco, *Understanding 802.1x and NAC: 3 Problems to Avoid*. (2022, 11 08). Retrieved from https://www.fortinet.com/content/dam/fortinet/assets/white-papers/wp-understanding-nac.pdf

Conrad, E., Henderson J., & Valanzuela, I. (2021) Defensible Security Architecture and Engineering. SANS Institute.

*Cisco Wired 802.1x Deployment Guide.* (2022, 11 08). Retrieved from https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Security/TrustSec_1-99/Dot1X_Deployment/Dot1x_Dep_Guide.html

Gronberg, K, & Howard, S. (2022, 11 08) Retrieved from https://www.cisa.gov/uscert/sites/default/files/ICSJWG-Archive/QNL_DEC_20/Forescout_Jacobs_ICSJWG%20Contributed%20Article_FINAL_S508C.pdf

PacketFence, *PacketFence Installation Guide*. (2022, 11 08). Retrieved from https://www.packetfence.org/doc/PacketFence_Installation_Guide.html

PacketFence, *PacketFence Network Devices Configuration Guide.* (2022, 11 08). Retrieved from https://www.packetfence.org/doc/PacketFence_Network_Devices_Configuration_Guide.html

Umer Khan, umer.khan@mail.com

*What Is Micro-Segmentation? (2022, 11 08).* Retrieved from https://www.cisco.com/c/en/us/products/security/what-is-microsegmentation.html#~benefits

# Appendix I
# Installing and Configuring PacketFence

1. Download the PacketFence Debian (production) ISO image from https://www.packetfence.org/download.html.
2. Ensure that you are connected to the local network with DHCP / Internet, and start VMware Fusion.
3. From the "File" menu, select "New".
4. Drag the PacketFence ISO image (e.g., PacketFence-ISO-v12.0.0.iso) from macOS Finder onto the "Install from disc or image" button.
5. Click "Continue".
6. Leave the default boot firmware of "Legacy BIOS" and click "Continue".
7. Click "Customize Settings" and enter the filename of "packetfence" and click "Save".
8. Click "Processors & Memory" and change "1 processor core" to "4 processor cores" and "16384 MB" instead of "2048 MB " memory. Click "Show All" to return to the main Settings panel.
9. Click "Network Adapter" and select the Ethernet adapter under "Bridged Networking" (Do not rely on "AutoDetect" as it may connect to the wireless network instead). Click "Show All" to return to the main Settings panel.
10. Click "Head Disk" and change the disk size from "20.00 GB" to "200.00 GB" and click "Apply". Click "Show All" to return to the main Settings panel.
11. Click "Printer" and uncheck "Share Mac printers with "Linux". Click "Show All" to return to the main Settings panel.
12. Close the "Settings" panel and start the virtual machine by clicking the "Play" button.
13. From the Debian installer menu, select "Install PacketFence"
14. When prompted for the hostname, leave the default of "packetfence" and select "Continue"

15. When prompted for the domain name, type in "localdomain" and select "Continue"
16. When prompted for the root password, choose an appropriate password and select "Continue"
17. Re-enter the password to verify it and select "Continue"
18. It will take some time for PacketFence to install. The system will then reboot and bring you to a login screen. Log in as root. Use the "hostname -I" command to determine the IP address of the machine.
19. Connect to the configuration wizard using a web browser (https://ip_address:1443) to configure PacketFence basics. Accept the self-signed certificate and proceed to the web site.
20. Click on "Detect Management Interface" to make eth0 the management interface and click "Next Step"
21. Select the appropriate "Timezone" and choose an appropriate Administrator password and click "Next Step"
22. It will take a few moments for PacketFence to update its configuration. Click on "Next Step" when the button is available.
23. The database root and user account passwords will be displayed on screen. Copy these off and store these in a safe place. Click on "Start PacketFence".
24. It will take some time for PacketFence to start. When it is finished, you will be presented with a login screen.

# Appendix II
# Adding a Switch to PacketFence

1. Log in to PacketFence at https://ip_address:1443 using a web browser. Use username admin.
2. Click on the "Configuration" menu.
3. Expand "Policies and Access Control" and click "Switches."
4. Click "New Switch" and select "default" as the switch group.
5. Enter in the IP "address" of the switch.
6. For the switch type, select the appropriate model. For the lab, it was "Cisco Catalyst 3560G."
7. Click the "RADIUS" tab and enter in the "Secret Passphrase". For the lab, it was "useStrongerSecret". Also enter "3799" for the "CoA Port".
8. Click the "CLI" tab and select "SSH" as the "Transport" and enter in the appropriate "Username", "Password", and "Enable Password" for the switch. For the lab, all three were "cisco."
9. Click "Create."



Umer Khan, umer.khan@mail.com

# Appendix III
# Configuring a Switch for 802.1x

```
! Configure an IP address, passwords, disable http, enable ssh
int vlan 1
 ip address dhcp
username cisco password cisco
enable secret cisco
no ip http server
ip domain-name localdomain
! Use 1024 bits in the modulus when prompted for the next command
crypto key generate rsa
aaa new-model
line vty 0 4
 transport input ssh
! Configure 802.1x authentication
! The PacketFence server IP address was 192.168.1.4 in the lab
dot1x system-auth-control
aaa new-model
aaa group server radius packetfence
 server 192.168.1.4 auth-port 1812 acct-port 1813
aaa authentication login default local
aaa authentication dot1x default group packetfence
aaa authorization network default group packetfence
radius-server host 192.168.1.4 auth-port 1812 acct-port 1813
timeout 2 key useStrongerSecret
radius-server vsa send authentication
snmp-server community public RO
snmp-server community private RW
```

# Appendix IV
# Configuring Fingerbank in PacketFence

Fingerbank was set up by registering for the service at https://api.fingerbank.org/ and clicking “Login on Signup” on the top-right. After creating an account, an API key was received and configured Fingerbank in PacketFence as follows:

1. Log in to PacketFence at https://ip_address:1443 using a web browser. Use username admin.
2. Click on the "Configuration" menu.
3. Under Compliance > Fingerbank Profiling, click "General Settings."
4. Enter the "API Key" and click "Save."
5. Click on the "Status" menu.
6. Click "Services.”
7. Click "Start" next to the "fingerbank-collector" service.



Umer Khan, umer.khan@mail.com