Administration Guide

for PacketFence version 5.3.1

Administration Guide

by Inverse Inc.

Version 5.3.1 - July 2015

Table of Contents

About this Guide

This guide will walk you through the installation and the day to day administration of the PacketFence solution.

The latest version of this guide is available at http://www.packetfence.org/documentation/

Other sources of information

The following documents are included in the package and release tarballs.

Network Devices Configuration Guide (pdf)	Covers switch, controllers and access points configuration.
Developer's Guide (pdf)	Covers captive portal customization, VLAN management customization and instructions for supporting new hardware.
`CREDITS`	This is, at least, a partial file of PacketFence contributors.
`NEWS.asciidoc`	Covers noteworthy features, improvements and bugfixes by release.
`UPGRADE.asciidoc`	Covers compatibility related changes, manual instructions and general notes about upgrading.
`ChangeLog`	Covers all changes to the source code.

Introduction

PacketFence is a fully supported, trusted, Free and Open Source network access control (NAC) system. Boosting an impressive feature set including a captive portal for registration and remediation, centralized wired and wireless management, 802.1X support, layer-2 isolation of problematic devices, integration with IDS, vulnerability scanners and firewalls; PacketFence can be used to effectively secure networks - from small to very large heterogeneous networks.

Features

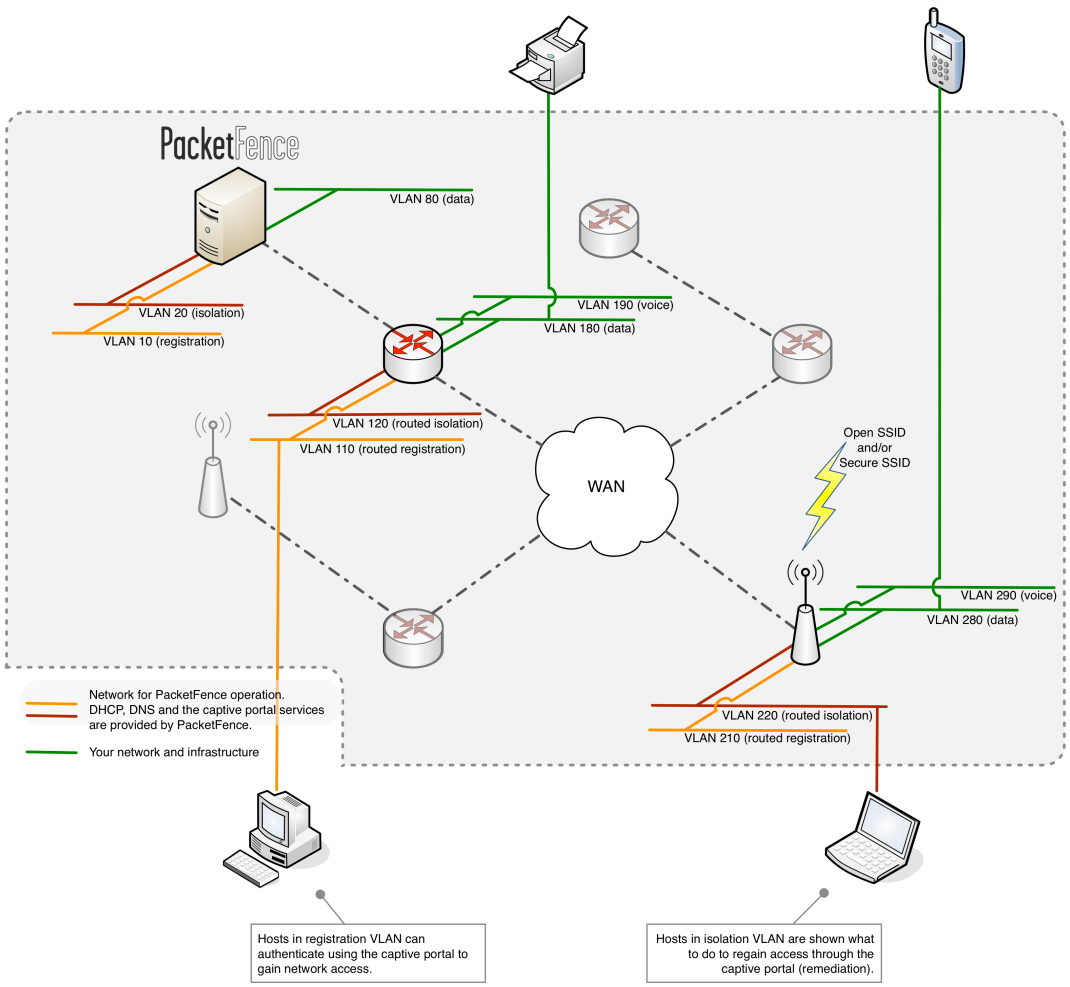
Out of band (VLAN Enforcement)	PacketFence's operation is completely out of band when using VLAN enforcement which allows the solution to scale geographically and to be more resilient to failures.
In Band (Inline Enforcement)	PacketFence can also be configured to be in-band, especially when you have non-manageable network switches or access points. PacketFence can also work with both VLAN and Inline enforcement activated for maximum scalability and security while allowing older hardware to still be secured using inline enforcement. Both layer-2 and layer-3 are supported for inline enforcement.
Hybrid support (Inline Enforcement with RADIUS support)	PacketFence can also be configured as hybrid, if you have a manageable device that supports 802.1X and/or MAC-authentication. This feature can be enabled using a RADIUS attribute (MAC address, SSID, port) or using full inline mode on the equipment.
Hotspot support (Web Auth Enforcement)	PacketFence can also be configured as hotspot, if you have a manageable device that supports an external captive portal (like Cisco WLC or Aruba IAP).
Voice over IP (VoIP) support	Also called IP Telephony (IPT), VoIP is fully supported (even in heterogeneous

	environments) for multiple switch vendors (Cisco, Avaya, HP and many more).
802.1X	802.1X wireless and wired is supported through our FreeRADIUS module.
Wireless integration	PacketFence integrates perfectly with wireless networks through our FreeRADIUS module. This allows you to secure your wired and wireless networks the same way using the same user database and using the same captive portal, providing a consistent user experience. Mixing Access Points (AP) vendors and Wireless Controllers is supported.
Registration	PacketFence supports an optional registration mechanism similar to "captive portal" solutions. Contrary to most captive portal solutions, PacketFence remembers users who previously registered and will automatically give them access without another authentication. Of course, this is configurable. An Acceptable Use Policy can be specified such that users cannot enable network access without first accepting it.
Detection of abnormal network activities	Abnormal network activities (computer virus, worms, spyware, traffic denied by establishment policy, etc.) can be detected using local and remote Snort or Suricata sensors. Beyond simple detection, PacketFence layers its own alerting and suppression mechanism on each alert type. A set of configurable actions for each violation is available to administrators.
Proactive vulnerability scans	Either Nessus , OpenVAS or WMI vulnerability scans can be performed upon registration, scheduled or on an ad-hoc basis. PacketFence correlates the scan engine vulnerability ID's of each scan to the violation configuration, returning content specific web pages about which vulnerability the host may have.
Isolation of problematic devices	PacketFence supports several isolation techniques, including VLAN isolation with VoIP support (even in heterogeneous environments) for multiple switch vendors.
Remediation through a captive portal	Once trapped, all network traffic is terminated by the PacketFence system.

	Based on the node's current status (unregistered, open violation, etc), the user is redirected to the appropriate URL. In the case of a violation, the user will be presented with instructions for the particular situation he/she is in reducing costly help desk intervention.
Firewall integration	PacketFence provides Single-Sign On features with many firewalls. Upon connection on the wired or wireless network, PacketFence can dynamically update the IP/user association on firewalls for them to apply, if required, per-user or per-group filtering policies.
Command-line and Web-based management	Web-based and command-line interfaces for all management tasks.
Guest Access	PacketFence supports a special guest VLAN out of the box. You configure your network so that the guest VLAN only goes out to the Internet and the registration VLAN and the captive portal are the components used to explain to the guest how to register for access and how his access works. This is usually branded by the organization offering the access. Several means of registering guests are possible. PacketFence does also support guest access bulk creations and imports.
Devices registration	A registered user can access a special Web page to register a device of his own. This registration process will require login from the user and then will register devices with pre-approved MAC OUI into a configurable category.

PacketFence is developed by a community of developers located mainly in North America. More information can be found at http://www.packetfence.org.

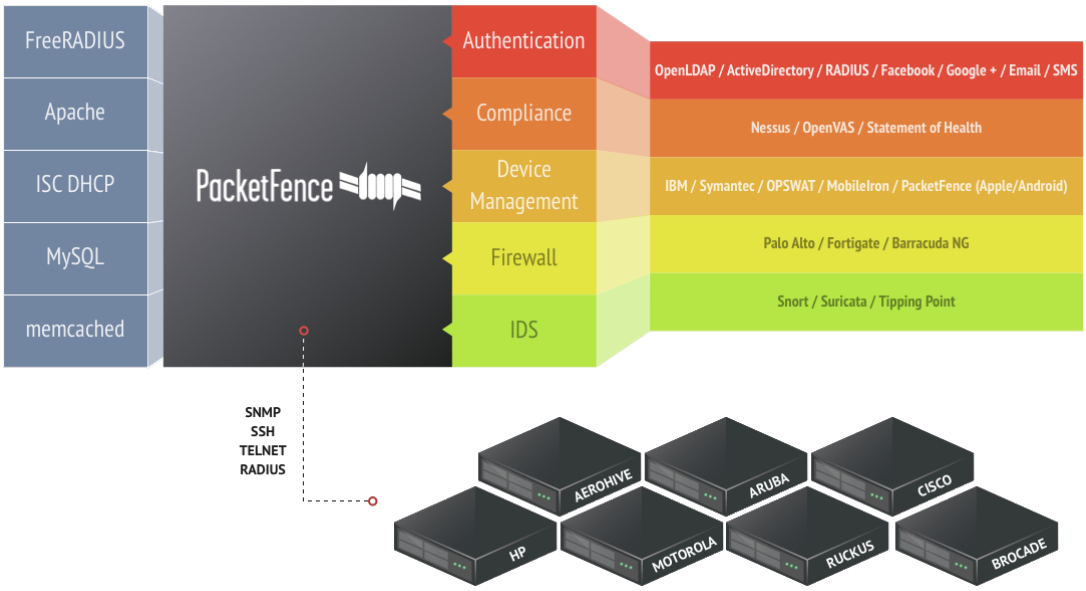
Network Integration



VLAN enforcement is pictured in the above diagram. Inline enforcement should be seen as a simple flat network where PacketFence acts as a firewall / gateway.

Components

PacketFence requires various components to work such as a Web server, a database server, and a RADIUS server. It interacts with external tools to extend its functionalities.


FreeRADIUS
Apache
ISC DHCP
MySQL
memcached
PacketFence
Authentication
Compliance
Device Management
Firewall
IDS
OpenLDAP / ActiveDirectory / RADIUS / Facebook / Google + / Email / SMS
Nessus / OpenVAS / Statement of Health
IBM / Symantec / OPSWAT / MobileIron / PacketFence (Apple/Android)
Palo Alto / Fortigate / Barracuda NG
Snort / Suricata / Tipping Point
SNMP
SSH
TELNET
RADIUS
AEROHIVE
ARUBA
CISCO
HP
MOTOROLA
RUCKUS
BROCADE

System Requirements

Assumptions

PacketFence reuses many components in an infrastructure. Thus, it requires the following ones:

- Database server (MySQL or MariaDB)
- Web server (Apache)
- DHCP server (ISC DHCP)
- RADIUS server (FreeRADIUS)

Depending on your setup you may have to install additional components like:

- NIDS (Snort/Suricata)

In this guide, we assume that all those components are running on the same server (i.e., "localhost" or "127.0.0.1") that PacketFence will be installed on.

Good understanding of those underlying component and GNU/Linux is required to install PacketFence. If you miss some of those required components, please refer to the appropriate documentation and proceed with the installation of these requirements before continuing with this guide.

The following table provides recommendations for the required components, together with version numbers :

MySQL server	MySQL 5.1
Web server	Apache 2.2
DHCP server	DHCP 4.1
RADIUS server	FreeRADIUS 2.2.x
Snort	Snort 2.9.1
Suricata	Suricata 1.4.1

More recent versions of the software mentioned above can also be used.

Minimum Hardware Requirements

The following provides a list of the minimum server hardware recommendations:

- Intel or AMD CPU 3 GHz
- 8 GB of RAM
- 100 GB of disk space (RAID-1 recommended)
- 1 Network card (2 recommended)

Operating System Requirements

PacketFence supports the following operating systems on the x86_64 architectures:

- Red Hat Enterprise Linux 6.x Server
- Community ENTerprise Operating System (CentOS) 6.x
- Debian 7.0 (Wheezy)
- Ubuntu 12.04 LTS (Precise Pangolin)

Make sure that you can install additional packages from your standard distribution. For example, if you are using Red Hat Enterprise Linux, you have to be subscribed to the Red Hat Network before continuing with the PacketFence software installation.

Other distributions such as Fedora and Gentoo are known to work but this document doesn't cover them.

Services start-up

PacketFence takes care of handling the operation of the following services:

- Web server (httpd)
- DHCP server (dhcpd)
- FreeRADIUS server (radiusd)
- Snort/Suricata Network IDS (snort/suricata)
- Firewall (iptables)

Make sure that all the other services are automatically started by your operating system!

Installation

This section will guide you through the installation of PacketFence together with its dependencies.

OS Installation

Install your distribution with minimal installation and no additional packages. Then:

- Disable Firewall
- Disable SELinux
- Disable AppArmor
- Disable resolvconf

Make sure your system is up to date and your yum or apt-get database is updated. On a RHEL-based system, do:

```
yum update
```

On a Debian or Ubuntu system, do:

```
apt-get update
apt-get upgrade
```

Regarding SELinux or AppArmor, even if these features may be wanted by some organizations, PacketFence will not run properly if SELinux or AppArmor are enabled. You will need to explicitly disable SELinux in the `/etc/selinux/config` file and AppArmor with **update-rc.d -f apparmor stop**, **update-rc.d -f apparmor teardown** and **update-rc.d -f apparmor remove**. Regarding resolvconf, you can remove the symlink to that file and simply create the `/etc/resolv.conf` file with the content you want.

RedHat-based systems

Note

Applies to CentOS and Scientific Linux but only the x86_64 architecture is supported.

RHEL 6.x

Note

These are extra steps are required for RHEL 6 systems only, excluding derivatives such as CentOS or Scientific Linux.

RedHat Enterprise Linux users need to take an additional setup step. If you are not using the RHN Subscription Management from RedHat you need to enable the optional channel by running the following as root:

```
rhn-channel --add --channel=rhel-`uname -m`-server-optional-6
```

Debian and Ubuntu

All the PacketFence dependencies are available through the official repositories.

Software Download

PacketFence provides a RPM repository for RHEL / CentOS instead of a single RPM file.

For Debian and Ubuntu, PacketFence also provides package repositories.

These repositories contain all required dependencies to install PacketFence. This provides numerous advantages:

- easy installation
- everything is packaged as RPM/deb (no more CPAN hassle)
- easy upgrade

Software Installation

RHEL / CentOS

In order to use the PacketFence repository :

```
# rpm -Uvh http://packetfence.org/downloads/PacketFence/RHEL6/`uname -i`/RPMS/
packetfence-release-1-2.centos6.noarch.rpm
```

Once the repository is defined, you can install PacketFence with all its dependencies, and the required external services (Database server, DHCP server, RADIUS server) using:

```
yum install --enablerepo=packetfence packetfence
```

Once installed, the Web-based configuration interface will automatically be started. You can access it from https://@ip_of_packetfence:1443/configurator

Debian

You must enable non-free repository:

For non-free, edit the file `/etc/apt/source.list` and add non-free like that:

```
deb http://debian.mirror.iweb.ca/debian/ wheezy main non-free
```

In order to use the repository, create a file named `/etc/apt/sources.list.d/packetfence.list`:

```
echo 'deb http://inverse.ca/downloads/PacketFence/debian wheezy wheezy' > /etc/
apt/sources.list.d/packetfence.list
```

Once the repository is defined, you can install PacketFence with all its dependencies, and the required external services (Database server, DHCP server, RADIUS server) using:

```
sudo apt-key adv --keyserver keys.gnupg.net --recv-key 0x810273C4
sudo apt-get update
sudo apt-get install packetfence
```

Ubuntu

In order to use the repository, create a file named `/etc/apt/sources.list.d/packetfence.list`:

```
echo 'deb http://inverse.ca/downloads/PacketFence/ubuntu precise precise' > /etc/
apt/sources.list.d/packetfence.list
```

Once the repository is defined, you can install PacketFence with all its dependencies, and the required external services (Database server, DHCP server, RADIUS server) using:

```
sudo apt-key adv --keyserver keys.gnupg.net --recv-key 0x810273C4
sudo apt-get update
sudo apt-get install packetfence
```

Once installed, the Web-based configuration interface will automatically be started. You can access it from https://@ip_of_packetfence:1443/configurator

Get off on the right foot

Prior configuring PacketFence, you must chose an appropriate enforcement mode to be used by PacketFence with your networking equipment. The enforcement mode is the technique used to enforce registration and any subsequent access of devices on your network. PacketFence supports the following enforcement modes:

- Inline
- Out-of-band
- Hybrid

It is also possible to combine enforcement modes. For example, you could use the out-of-band mode on your wired switches, while using the inline mode on your old WiFi access points.

The following sections will explain these enforcement modes. If you decide to use the inline mode, please refer to the PacketFence Inline Deployment Quick Guide using ZEN for a complete configuration example. If you device to use the out-of-band mode, please refer to the PacketFence Out-of-Band Deployment Quick Guide using ZEN

Technical introduction to Inline enforcement

Introduction

Before the version 3.0 of PacketFence, it was not possible to support unmanageable devices such as entry-level consumer switches or access-points. Now, with the new inline mode, PacketFence can be use in-band for those devices. So in other words, PacketFence would become the gateway of that inline network, and NAT or route the traffic using IPTables/IPSet to the Internet (or to another section of the network). Let see how it works.

Device configuration

No special configuration is needed on the unmanageable device. That's the beauty of it. You only need to ensure that the device is "talking" on the inline VLAN. At this point, all the traffic will be passing through PacketFence since it is the gateway for this VLAN.

Access control

The access control relies entirely on IPTables/IPSet. When a user is not registered, and connects in the inline VLAN, PacketFence will give him an IP address. At this point, the user will be marked as unregistered in the ipset session, and all the Web traffic will be redirected to the captive portal and other traffic blocked. The user will have to register through the captive portal as in VLAN enforcement. When he registers, PacketFence changes the device´s ipset session to allow the user's mac address to go through it.

Limitations

Inline enforcement because of it's nature has several limitations that one must be aware of.

- Everyone behind an inline interface is on the same Layer 2 LAN
- Every packet of authorized users goes through the PacketFence server increasing the servers' load considerably: Plan ahead for capacity
- Every packet of authorized users goes through the PacketFence server: it is a single point of failure for Internet access
- Ipset can store up to 65536 entries, so it is not possible to have a inline network class upper than B

This is why it is considered a poor man's way of doing access control. We have avoided it for a long time because of the above mentioned limitations. That said, being able to perform both inline and VLAN enforcement on the same server at the same time is a real advantage: it allows users to maintain maximum security while they deploy new and more capable network hardware providing a clean migration path to VLAN enforcement.

Technical introduction to Out-of-band enforcement

Introduction

VLAN assignment is currently performed using several different techniques. These techniques are compatible one to another but not on the same switch port. This means that you can use the more secure and modern techniques for your latest switches and another technique on the old switches that doesn't support latest techniques. As it's name implies, VLAN assignment means that PacketFence is the server that assigns the VLAN to a device. This VLAN can be one of your VLANs or it can be a special VLAN where PacketFence presents the captive portal for authentication or remediation.

VLAN assignment effectively isolate your hosts at the OSI Layer2 meaning that it is the trickiest method to bypass and is the one which adapts best to your environment since it glues into your current VLAN assignment methodology.

VLAN assignment techniques

Wired: 802.1X + MAC Authentication

802.1X provides port-based authentication, which involves communications between a supplicant, authenticator (known as NAS), and authentication server (known as AAA). The supplicant is often software on a client device, such as a laptop, the authenticator is a wired Ethernet switch or wireless access point, and the authentication server is generally a RADIUS server.

The supplicant (i.e., client device) is not allowed access through the authenticator to the network until the supplicant's identity is authorized. With 802.1X port-based authentication, the supplicant provides credentials, such as user name / password or digital certificate, to the authenticator, and the authenticator forwards the credentials to the authentication server for verification. If the credentials are valid (in the authentication server database), the supplicant (client device) is allowed to access the network. The protocol for authentication is called Extensible Authentication Protocol (EAP) which have many variants. Both supplicant and authentication servers need to speak the same EAP protocol. Most popular EAP variant is PEAP-MsCHAPv2 (supported by Windows / Mac OSX / Linux for authentication against AD).

In this context, PacketFence runs the authentication server (a FreeRADIUS instance) and will return the appropriate VLAN to the switch. A module that integrates in FreeRADIUS does a remote call to the PacketFence server to obtain that information. More and more devices have 802.1X supplicant which makes this approach more and more popular.

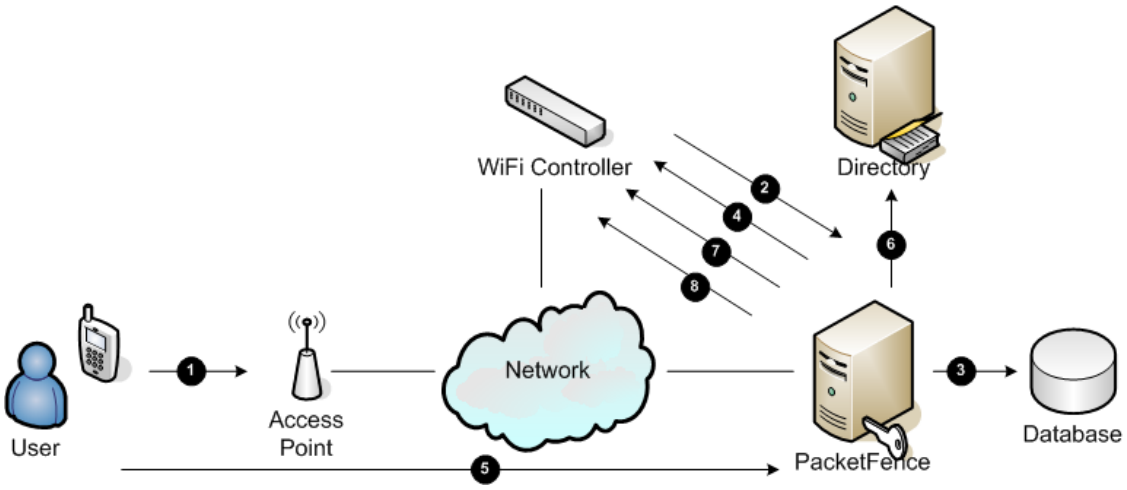
MAC Authentication is a new mechanism introduced by some switch vendor to handle the cases where a 802.1X supplicant does not exist. Different vendors have different names for it. Cisco calls it MAC Authentication Bypass (MAB), Juniper calls it MAC RADIUS, Extreme Networks calls it Netlogin, etc. After a timeout period, the switch will stop trying to perform 802.1X and will fallback to MAC Authentication. It has the advantage of using the same approach as 802.1X except that the MAC address is sent instead of the user name and there is no end-to-end EAP conversation (no strong authentication). Using MAC Authentication, devices like network printer or non-802.1X capable IP Phones can still gain access to the network and the right VLAN.

Wireless: 802.1X + MAC authentication

Wireless 802.1X works like wired 802.1X and MAC authentication is the same as wired MAC Authentication. Where things change is that the 802.1X is used to setup the security keys for encrypted communication (WPA2-Enterprise) while MAC authentication is only used to authorize (allow or disallow) a MAC on the wireless network.

On wireless networks, the usual PacketFence setup dictate that you configure two SSIDs: an open one and a secure one. The open one is used to help users configure the secure one properly and requires authentication over the captive portal (which runs in HTTPS).

The following diagram demonstrates the flow between a mobile enpoint, a WiFi access point, a WiFi controller and PacketFence:



1. User initiates association to WLAN AP and transmits MAC address. If user accesses network via a registered device in PacketFence go to 8

2. The WLAN controller transmits MAC address via RADIUS to the PacketFence server to authenticate/authorize that MAC address on the AP

3. PacketFence server conducts address audit in its database. If it does not recognize the MAC address go to 4. If it does go to 8.

4. PacketFence server directs WLAN controller via RADIUS (RFC2868 attributes) to put the device in an "unauthenticated role“ (set of ACLs that would limit/redirect the user to the PacketFence

captive portal for registration, or we can also use a registration VLAN in which PacketFence does DNS blackholing and is the DHCP server)

5. The user's device issues a DHCP/DNS request to PacketFence (which is a DHCP/DNS server on this VLAN or for this role) which sends the IP and DNS information. At this point, ACLs are limiting/redirecting the user to the PacketFence's captive portal for authentication. PacketFence fingerprints the device (user-agent attributes, DHCP information & MAC address patterns) to which it can take various actions including: keep device on registration portal, direct to alternate captive portal, auto-register the device, auto-block the device, etc. If the device remains on the registration portal the user registers by providing the information (username/password, cell phone number, etc.). At this time PacketFence could also require the device to go through a posture assessment (using Nessus, OpenVAS, etc.)

6. If authentication is required (username/password) through a login form, those credentials are validated via the Directory server (or any other authentication sources - like LDAP, SQL, RADIUS, SMS, Facebook, Google+, etc.) which provides user attributes to PacketFence which creates user +device policy profile in its database.

7. PacketFence performs a Change of Authorization (RFC3576) on the controller and the user must be re-authenticated/reauthorized, so we go back to 1

8. PacketFence server directs WLAN controller via RADIUS to put the device in an "authenticated role", or in the "normal" VLAN

Web Auth mode

Web authentication is a method on the switch that forwards http traffic of the device to the captive portal. With this mode, your device will never change of VLAN ID but only the ACL associated to your device will change. Refer to the Network Devices Configuration Guide to see a sample web auth configuration on a Cisco WLC.

Port-security and SNMP

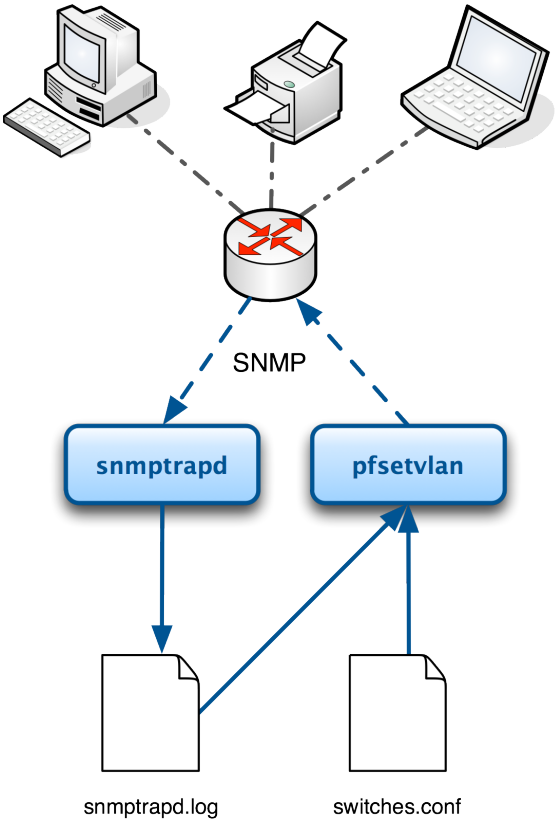
Relies on the port-security SNMP Traps. A fake static MAC address is assigned to all the ports this way any MAC address will generate a security violation and a trap will be sent to PacketFence. The system will authorize the MAC and set the port in the right VLAN. VoIP support is possible but tricky. It varies a lot depending on the switch vendor. Cisco is well supported but isolation of a PC behind an IP Phone leads to an interesting dilemma: either you shut the port (and the phone at the same time) or you change the data VLAN but the PC doesn't do DHCP (didn't detect link was down) so it cannot reach the captive portal.

Aside from the VoIP isolation dilemma, it is the technique that has proven to be reliable and that has the most switch vendor support.

More on SNMP traps VLAN isolation

When the VLAN isolation is working through SNMP traps all switch ports (on which VLAN isolation should be done) must be configured to send SNMP traps to the PacketFence host. On PacketFence,

we use snmptrapd as the SNMP trap receiver. As it receives traps, it reformats and writes them into a flat file: `/usr/local/pf/logs/snmptrapd.log`. The multithreaded `pfsetvlan` daemon reads these traps from the flat file and responds to them by setting the switch port to the correct VLAN. Currently, we support switches from Cisco, Edge-core, HP, Intel, Linksys and Nortel (adding support for switches from another vendor implies extending the `pf::Switch` class). Depending on your switches capabilities, `pfsetvlan` will act on different types of SNMP traps.



You need to create a registration VLAN (with a DHCP server, but no routing to other VLANs) in which PacketFence will put unregistered devices. If you want to isolate computers which have open violations in a separate VLAN, an isolation VLAN needs also to be created.

linkUp/linkDown traps (deprecated)

This is the most basic setup and it needs a third VLAN: the MAC detection VLAN. There should be nothing in this VLAN (no DHCP server) and it should not be routed anywhere; it is just an void VLAN.

When a host connects to a switch port, the switch sends a linkUp trap to PacketFence. Since it takes some time before the switch learns the MAC address of the newly connected device, PacketFence immediately puts the port in the MAC detection VLAN in which the device will send DHCP requests (with no answer) in order for the switch to learn its MAC address. Then pfsetvlan will send periodical SNMP queries to the switch until the switch learns the MAC of the device. When the MAC address is known, pfsetvlan checks its status (existing ? registered ? any violations ?) in the database and puts the port in the appropriate VLAN. When a device is unplugged, the switch sends a *linkDown* trap to PacketFence which puts the port into the MAC detection VLAN.

When a computer boots, the initialization of the NIC generates several link status changes. And every time the switch sends a linkUp and a linkDown trap to PacketFence. Since PacketFence has

to act on each of these traps, this generates unfortunately some unnecessary load on pfsetvlan. In order to optimize the trap treatment, PacketFence stops every thread for a *linkUp trap* when it receives a *linkDown* trap on the same port. But using only linkUp/linkDown traps is not the most scalable option. For example in case of power failure, if hundreds of computers boot at the same time, PacketFence would receive a lot of traps almost instantly and this could result in network connection latency.

MAC notification traps

If your switches support MAC notification traps (MAC learnt, MAC removed), we suggest that you activate them in addition to the linkUp/linkDown traps. This way, pfsetvlan does not need, after a linkUp trap, to query the switch continuously until the MAC has finally been learned. When it receives a linkUp trap for a port on which MAC notification traps are also enabled, it only needs to put the port in the MAC detection VLAN and can then free the thread. When the switch learns the MAC address of the device it sends a MAC learnt trap (containing the MAC address) to PacketFence.

Port Security traps

In its most basic form, the Port Security feature remembers the MAC address connected to the switch port and allows only that MAC address to communicate on that port. If any other MAC address tries to communicate through the port, port security will not allow it and send a port-security trap.

If your switches support this feature, **we strongly recommend to use it rather than linkUp/linkDown and/or MAC notifications**. Why? Because as long as a MAC address is authorized on a port and is the only one connected, the switch will send no trap whether the device reboots, plugs in or unplugs. This drastically reduces the SNMP interactions between the switches and PacketFence.

When you enable port security traps you should not enable linkUp/linkDown nor MAC notification traps.

Technical introduction to Hybrid enforcement

Introduction

In previous versions of PacketFence, it was not possible to have RADIUS enabled for inline enforcement mode. Now with the new hybrid mode, all the devices that supports 802.1X or MAC-authentication can work with this mode. Let's see how it works.

Device configuration

You need to configure inline enforcement mode in PacketFence and configure your switch(es) / access point(s) to use the VLAN assignement techniques (802.1X or MAC-authentication). You also need to take care of a specific parameter in the switch configuration window, "Trigger to enable inline mode". This parameter is working like a trigger and you have the possibility to define different sort of triggers:

ALWAYS , PORT , MAC , SSID
: where ALWAYS means that the device is always in inline mode, PORT specify the ifIndex of the port which will use inline enforcement, MAC a mac address that will be put in inline enforcement technique rather than VLAN enforcement and SSID an ssid name. An example:

```
SSID::GuestAccess,MAC::00:11:22:33:44:55
```

This will trigger all the nodes that connects to the *GuestAccess* SSID to use inline enforcement mode (PacketFence will return a void VLAN or the `inlineVlan` if defined in switch configuration) and the MAC address `00:11:22:33:44:55` client if it connects on another SSID.

Configuration

At this point in the documentation, PacketFence should be installed. You would also have chosen the right enforcement method for you and completed the initial configuration of PacketFence. The following section presents key concepts and features in PacketFence.

PacketFence provides a web-based administration interface for easy configuration and operational management. If you went through PacketFence's web-based configuration tool, you should have set the password for the `admin` user.

Once PacketFence is started, the administration interface is available at: https://@ip_of_packetfence:1443/

The next key steps are important to understand how PacketFence works. In order to get the solution working, you must first understand and configure the following aspects of the solution in this specific order:

1. **roles** - a role in PacketFence will be eventually be mapped to a VLAN, an ACL or an external role. You must define the roles to use in your organization for network access
2. **authentication** - once roles are defined, you must create an appropraite authentication source in PacketFence. That will allow PacketFence to compute the right role to be used for an endpoint, or the user using it
3. **network devices** - once your roles and authentication sources are defined, you must add switches, WiFi controllers or APs to be mananaged by PacketFence. When doing so, you will configure how roles are being mapped to VLAN, ACLs or external roles
4. **portal profiles** - at this point, you are almost ready to test. You will need to set which authentication sources are to be used on the default captive portal, or create an other one to suit your needs
5. test!

Note

If you plan to use 802.1X - please see the *FreeRADIUS Configuration* section below.

Roles Management

Roles in PacketFence can be created from PacketFence administrative GUI - from the **Configuration → Users → Roles** section. From this interface, you can also limit the number of devices users belonging to certain roles can register.

Roles are dynamically computed by PacketFence, based on the rules (ie., a set of conditions and actions) from authentication sources, using a first-match wins algorithm. Roles are then matched to VLAN or internal roles or ACL on equipment from the **Configuration → Network → Switches** module.

Authentication

PacketFence can authenticate users that register devices via the captive portal using various methods. Among the supported methods, there are:

- Active Directory
- Apache htpasswd file
- Email
- Facebook (OAuth 2)
- Github (OAuth 2)
- Google (OAuth 2)
- Kerberos
- LDAP
- LinkedIn (OAuth 2)
- Null
- RADIUS
- SMS
- Sponsored Email
- Twitter (OAuth 2)
- Windows Live (OAuth 2)

Moreover, PacketFence can also authenticate users defined in its own internal SQL database. Authentication sources can be created from PacketFence administrative GUI - from the **Configuration → Users → Sources** section. Alternatively (but not recommended), authentication sources, rules, conditions and actions can be configured from `conf/authentication.conf`.

Each authentication sources you define will have a set of rules, conditions and actions.

Multiple authentication sources can be defined, and will be tested in the order specified (note that they can be reordered from the GUI by dragging it around). Each source can have multiple rules, which will also be tested in the order specified. Rules can also be reordered, just like sources. Finally, conditions can be defined for a rule to match certain criterias. If the criterias match (one

or more), action are then applied and rules testing stop, across all sources as this is a "first match wins" operation.

When no condition is defined, the rule will be considered as a fallback. When a fallback is defined, all actions will be applied for any users that match in the authentication source.

Once a source is defined, it can be used from **Configuration → Portal Profiles**. Each portal profile has a list of authentication sources to use.

Example

Let's say we have two roles: guest and employee. First, we define them **Configuration → Users → Roles**.

Now, we want to authenticate employees using Active Directory (over LDAP), and guests using PacketFence's internal database - both using PacketFence's captive portal. From the **Configuration → Users → Sources**, we select **Add source → AD**. We provide the following information:

- **Name:** ad1
- **Description:** Active Directory for Employees
- **Host:** 192.168.1.2:389 without SSL/TLS
- **Base DN:** CN=Users,DC=acme,DC=local
- **Scope:** One-level
- **Username Attribute:** sAMAccountName
- **Bind DN:** CN=Administrator,CN=Users,DC=acme,DC=local
- **Password:** acme123

Then, we add a rule by clicking on the **Add rule** button and provide the following information:

- **Name:** employees
- **Description:** Rule for all employees
- Don't set any condition (as it's a catch-all rule)
- Set the following **actions:**
  - Set role employee
  - Set unregistration date January 1st, 2020

Test the connection and save everything. Using the newly defined source, any username that actually matches in the source (using the sAMAccountName) will have the employee role and an unregistration date set to January 1st, 2020.

Now, since we want to authenticate guests from PacketFence's internal SQL database, accounts must be provisionned manually. You can do so from the **Users → Create** section. When creating guests, specify "guest" for the **Set role** action, and set an access duration for 1 day.

If you would like to differentiate user authentication and machine authentication using Active Directory, one way to do it is by creating a second authentication sources, for machines:

- **Name:** ad1
- **Description:** Active Directory for Machines
- **Host:** 192.168.1.2:389 without SSL/TLS
- **Base DN:** CN=Computers,DC=acme,DC=local

- **Scope:** One-level
- **Username Attribute:** servicePrincipalName
- **Bind DN:** CN=Administrator,CN=Users,DC=acme,DC=local
- **Password:** acme123

Then, we add a rule:

- Name:* machines
- **Description:** Rule for all machines
- Don't set any condition (as it's a catch-all rule)
- Set the following **actions:**
  - Set role machineauth
  - Set unregistration date January 1st, 2020

Note

When a rule is defined as a catch-all, it will always match if the username attribute matches the queried one. This applies for Active Directory, LDAP and Apache htpasswd file sources. Kerberos and RADIUS will act as true catch-all, and accept everything.

Network Devices Definition (switches.conf)

This section applies only for VLAN enforcement. Users planning to do inline enforcement only can skip this section.

PacketFence needs to know which switches, access points or controllers it manages, their type and configuration. All this information is stored in `/usr/local/pf/conf/switches.conf`. You can modify the configuration directly in the `switches.conf` file or you can do it from the Web Administration panel under **Configuration → Network → Switches** - which is now the preferred way.

The `/usr/local/pf/conf/switches.conf` configuration file contains a default section including:

- Default SNMP read/write communities for the switches
- Default working mode (see the note below about possible working modes)

and a switch section for each switch (managed by PacketFence) including:

- Switch IP/Mac/Range
- Switch vendor/type
- Switch uplink ports (trunks and non-managed IfIndex)
- per-switch re-definition of the VLANs (if required)

Note

`switches.conf` is loaded at startup. A reload is required when changes are manually made to this file `/usr/local/pf/bin/pfcmd configreload`.

Working modes

There are three different working modes for a switch in PacketFence:

Testing	pfsetvlan writes in the log files what it would normally do, but it doesn't do anything.
Registration	pfsetvlan automatically-register all MAC addresses seen on the switch ports. As in testing mode, no VLAN changes are done.
Production	pfsetvlan sends the SNMP writes to change the VLAN on the switch ports.

RADIUS

To set the RADIUS secret, set it from the Web administrative interface when adding a switch. Alternatively, edit the switch config file (`/usr/local/pf/conf/switches.conf`) and set the following parameters:

```
radiusSecret = secretPassPhrase
```

Moreover, the RADIUS secret is required to support the RADIUS Dynamic Authentication (Change of authorization or Disconnect) as defined in RFC3576.

SNMP v1, v2c and v3

PacketFence uses SNMP to communicate with most switches. PacketFence also supports SNMP v3. You can use SNMP v3 for communication in both directions: from the switch to PacketFence and from PacketFence to the switch. SNMP usage is discouraged, you should now use RADIUS. However, even if RADIUS is being used, some switches might also require SNMP to be configured to work properly with PacketFence.

From PacketFence to a switch

Edit the switch config file (`/usr/local/pf/conf/switches.conf`) and set the following parameters:

```
SNMPVersion = 3
SNMPUserNameRead = readUser
SNMPAuthProtocolRead = MD5
SNMPAuthPasswordRead = authpwdread
SNMPPrivProtocolRead = AES
SNMPPrivPasswordRead = privpwdread
SNMPUserNameWrite = writeUser
SNMPAuthProtocolWrite = MD5
SNMPAuthPasswordWrite = authpwdwrite
SNMPPrivProtocolWrite = AES
SNMPPrivPasswordWrite = privpwdwrite
```

From a switch to PacketFence

Edit the switch config file (`/usr/local/pf/conf/switches.conf`) and set the following parameters:

```
SNMPVersionTrap = 3
SNMPUserNameTrap = readUser
SNMPAuthProtocolTrap = MD5
SNMPAuthPasswordTrap = authpwdread
SNMPPrivProtocolTrap = AES
SNMPPrivPasswordTrap = privpwdread
```

Switch Configuration

Here is a switch configuration example in order to enable SNMP v3 in both directions on a Cisco Switch.

```
snmp-server engineID local AA5ED139B81D4A328D18ACD1
snmp-server group readGroup v3 priv
snmp-server group writeGroup v3 priv read v1default write v1default
snmp-server user readUser readGroup v3 auth md5 authpwdread priv aes 128
 privpwdread
snmp-server user writeUser writeGroup v3 auth md5 authpwdwrite priv aes 128
 privpwdwrite
snmp-server enable traps port-security
snmp-server enable traps port-security trap-rate 1
snmp-server host 192.168.0.50 version 3 priv readUser  port-security
```

Command-Line Interface: Telnet and SSH

Warning

Privilege detection is disabled in the current PacketFence version due to some issues (see #1370). So make sure that the `cliUser` and `cliPwd` you provide always get you into a privileged mode (except for Trapeze hardware).

PackeFence needs sometimes to establish an interactive command-line session with a switch. This can be done using Telnet. You can also use SSH. In order to do so, edit the switch configuration file (`/usr/local/pf/conf/switches.conf`) and set the following parameters:

```
cliTransport = SSH (or Telnet)
cliUser = admin
cliPwd = admin_pwd
cliEnablePwd =
```

It can also be done through the Web Administration Interface under **Configuration → Switches**.

Web Services Interface

PackeFence sometimes needs to establish a dialog with the Web Services capabilities of a switch. In order to do so, edit the switch config file (`/usr/local/pf/conf/switches.conf`) and set the following parameters:

```
wsTransport = http (or https)
wsUser = admin
wsPwd = admin_pwd
```

It can also be done through the Web Administration Interface under **Configuration → Switches**.

Role-based enforcement support

Some network devices support the assignment of a specific set of rules (firewall or ACLs) to a user. The idea is that these rules can be a lot more accurate to control what a user can or cannot do compared to VLAN which have a larger network management overhead.

PacketFence supports assigning roles on devices for switches and WiFi controllers that support it. The current role assignment strategy is to assign it along with the VLAN (that may change in the future). A special internal role to external role assignment must be configured in the switch configuration file (`/usr/local/pf/conf/switches.conf`).

The current format is the following:

```
Format: <rolename>Role=<controller_role>
```

And you assign it to the global `roles` parameter or the per-switch one. For example:

```
adminRole=full-access
engineeringRole=full-access
salesRole=little-access
```

would return the `full-access` role to the nodes categorized as admin or engineering and the role `little-access` to nodes categorized as sales. It can also be done through the Web Administration Interface under **Configuration → Switches**.

Caution

Make sure that the roles are properly defined on the network devices prior to assigning roles!

Portal Profiles

PacketFence comes with a default portal profile. The follow parameters are important to configure no matter if you use the default portal profile or create a new one:

- Redirect URL under **Configuration → Portal Profile → Portal Name**

For some browsers, it is preferable to redirect the user to a specific URL instead of the URL the user originally intended to visit. For these browsers, the URL defined in `redirecturl` will be the one where the user will be redirected. Affected browsers are Firefox 3 and later.

- IP under **Configuration → Captive portal**

This IP is used as the web server who hosts the `common/network-access-detection.gif` which is used to detect if network access was enabled. It cannot be a domain name since it is used in registration or quarantine where DNS is black-holed. It is recommended that you allow your users to reach your PacketFence server and put your LAN's PacketFence IP. By default we will make this reach PacketFence's website as an easier and more accessible solution.

In some cases, you may want to present a different captive portal (see below for the available customizations) according to the SSID, the VLAN, the switch IP/MAC or the URI the client connects to. To do so, PacketFence has the concept of portal profiles which gives you this possibility.

When configured, portal profiles will override default values for which it is configured. When no values are configured in the profile, PacketFence will take its default ones (according to the "default" portal profile).

Here are the different configuration parameters that can be set for each portal profiles. The only mandatory parameter is "filter", otherwise, PacketFence won't be able to correctly apply the portal profile. The parameters must be set in conf/profiles.conf:

```
[profilename1]
description = the description of your portal profile
filter = the name of the SSID for which you'd like to apply the profile, or the
 VLAN number
billing_engine = either enabled or disabled
sources = comma-separated list of authentications sources (IDs) to use
```

Portal profiles should be managed from PacketFence's Web administrative GUI - from the **Configuration → Portal Profiles** section. Adding a portal profile from that interface will correctly copy templates over - which can then be modified as you wish.

- Filters under **Configuration → Portal Profile → Portal Name → Fitlers**

PacketFence offers the following filters: Connection Type, Network, Node Role, Port, realm, SSID, Switch, Switch Port, URI and VLAN.

Example with the most common ones:

- **SSID:** Guest-SSID
- **VLAN:** 100

Caution

Node role will take effect only with a 802.1x connection or if you use VLAN filters.

PacketFence relies extensively on Apache for its captive portal, administrative interface and Web services. The PacketFence´s Apache configuration are located in `/usr/local/pf/conf/httpd.conf.d/`.

In this directory you have three important files: `httpd.admin`, `httpd.portal`, `httpd.webservices`, `httpd.aaa`.

- `httpd.admin` is used to manage PacketFence admin interface

- `httpd.portal` is used to manage PacketFence captive portal interface
- `httpd.webservices` is used to manage PacketFence webservices interface
- `httpd.aaa` is use to manage incoming RADIUS request

These files have been written using the Perl language and are completely dynamic - so they activate services only on the network interfaces provided for this purpose.

The other files in this directory are managed by PacketFence using templates, so it is easy to modify these files based on your configuration. SSL is enabled by default to secure access.

Upon PacketFence installation, self-signed certificates will be created in `/usr/local/pf/conf/ssl` (`server.key` and `server.crt`). Those certificates can be replaced anytime by your 3rd-party or existing wildcard certificate without problems. Please note that the CN (Common Name) needs to be the same as the one defined in the PacketFence configuration file (`pf.conf`).

FreeRADIUS Configuration

This section presents the FreeRADIUS configuration steps. In some occasions, a RADIUS server is mandatory in order to give access to the network. For example, the usage of WPA2-Enterprise (Wireless 802.1X), MAC authentication and Wired 802.1X all require a RADIUS server to authenticate the users and the devices, and then to push the proper roles or VLAN attributes to the network equipment.

Option 1: Authentication against Active Directory (AD)

Caution

If you are using an Active/Active or Active/Passive cluster, please follow the instructions under *Option 1b* since the instructions below do not currently work in a cluster.

In order to have domain authentication working properly, you need to enable IP forwarding on your server. To do it permanently, look in the `/etc/sysctl.conf`, and set the following line:

```
# Controls IP packet forwarding
net.ipv4.ip_forward = 1
```

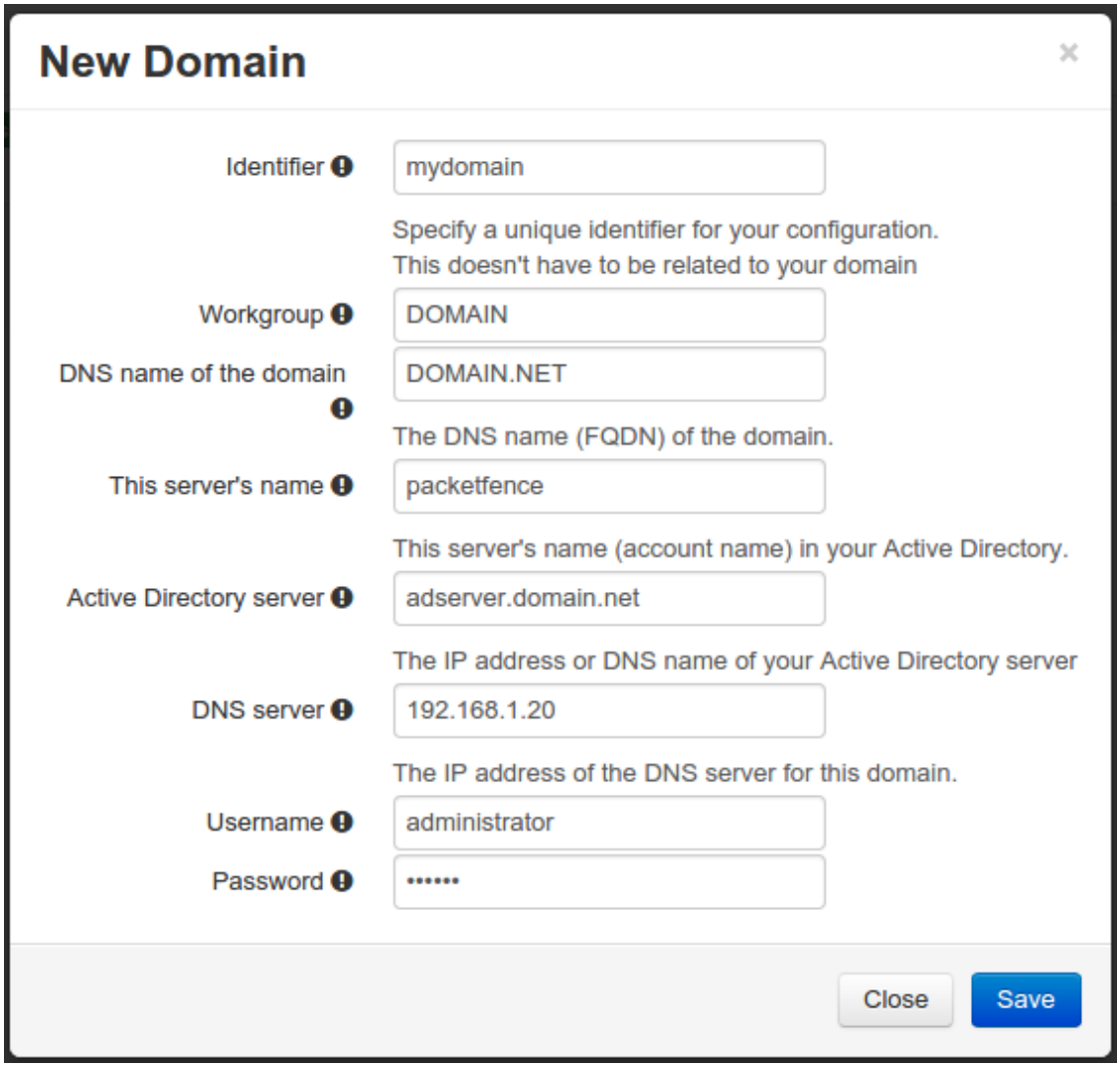
Now execute `sysctl -p` to apply the configuration

Next, go in the Administration interface under **Configuration → Domains**.

Note

If you can't access this section and you have previously configured your server to bind to a domain externally to PacketFence, make sure you run `/usr/local/pf/addons/AD/migrate.pl`

Click **Add Domain** and fill in the informations about your domain.



Where :

- **Identifier** is a unique identifier for your domain. It's purpose is only visual.
- **Workgroup** is the workgroup of your domain in the old syntax (like NT4).
- **DNS name of the domain** is the FQDN of your domain. The one that suffixes your account names.
- **This server's name** is the name that the server's account will have in your Active Directory.
- **DNS server** is the IP address of the DNS server of this domain. Make sure that the server you put there has the proper DNS entries for this domain.
- **Username** is the username that will be used for binding to the server. This account must be a domain administrator.
- **Password** is the password for the username defined above.

Troubleshooting

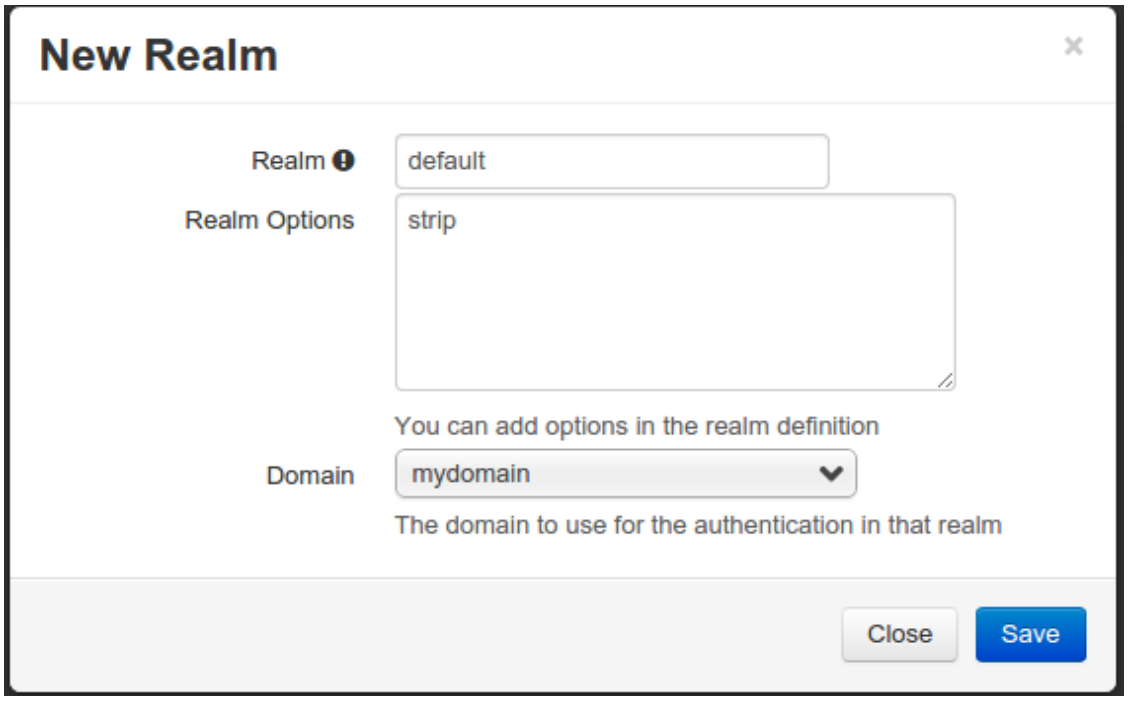
- In order to troubleshoot unsuccessful binds, please refer to the following file : `/chroots/<mydomain>/var/log/samba<mydomain>/log.winbindd`. Replace `<mydomain>` with the identifier you set in the domain configuration.
- You can validate the domain bind using the following command : `chroot /chroots/<mydomain> wbinfo -u`
- You can test the authentication process using the following command `chroot /chroots/<mydomain> ntlm_auth --username=administrator`

Note

Under certain conditions, the test join may show as unsuccessful in the Administration interface but the authentication process will still work properly. Try the test above before doing any additionnal troubleshooting

Default domain configuration

You should now define the domain you want to use as the default one by creating the following realm in **Configuration → Realms**



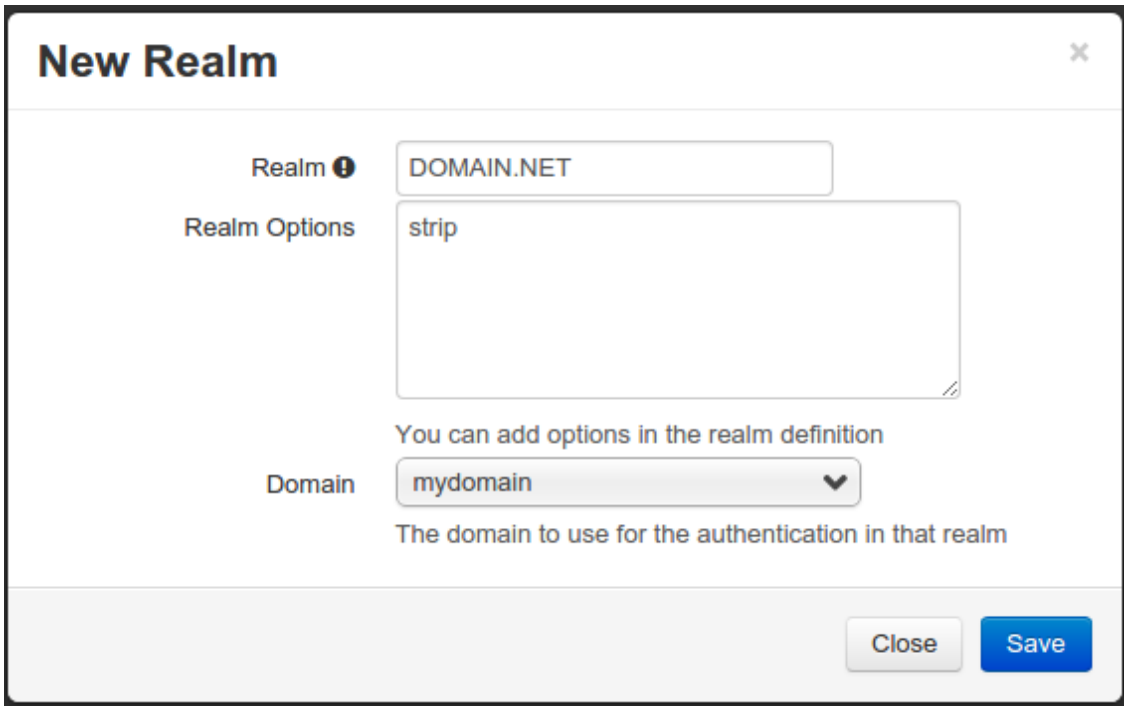
Next, restart Packetfence in **Status → Services**

Multiple domains authentication

First configure your domains in **Configuration → Domains**.

Once they are configured, go in **Configuration → Realms**.

Create a new realm that matches the DNS name of your domain **AND** one that matches your workgroup. In the case of this example, it will be DOMAIN.NET and DOMAIN.

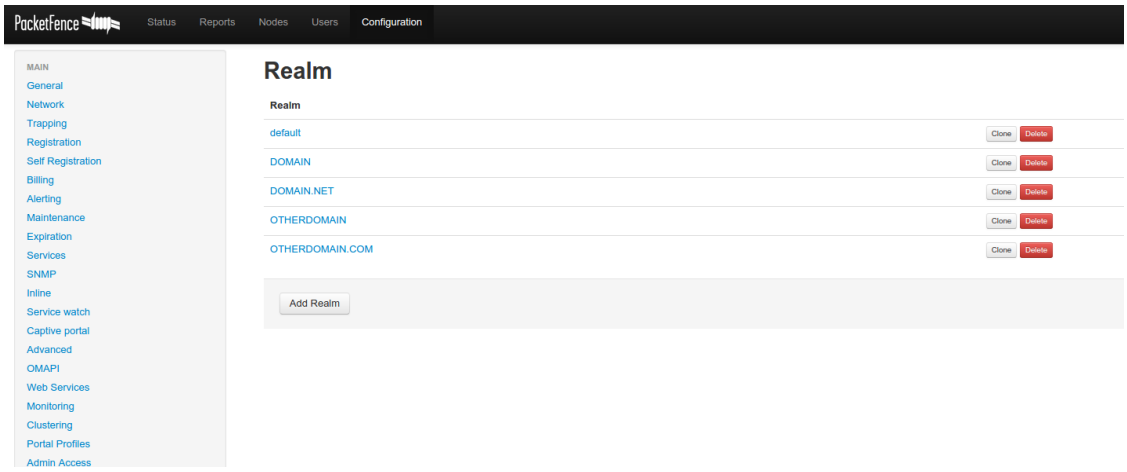


Where :

- **Realm** is either the DNS name (FQDN) of your domain or the workgroup
- **Realm options** are any realm options that you want to add to the FreeRADIUS configuration
- **Domain** is the domain which is associated to this realm

Now create the two other realms associated to your other domains.

You should now have the following realm configuration



Option 1b: Authentication against Active Directory (AD) in a cluster

Samba / Kerberos / Winbind

Install Samba. You can either use the sources or use the package for your OS. For RHEL/CentOS, do:

```
yum install samba krb5-workstation
```

For Debian and Ubuntu, do:

```
apt-get install samba winbind krb5-user
```

Note

If you have Windows 7 PCs in your network, you need to use Samba version 3.5.0 (or greater).

When done with the Samba install, modify your `/etc/hosts` in order to add the FQDN of your Active Directory servers. Then, you need to modify `/etc/krb5.conf`. Here is an example for the `DOMAIN.NET` domain for Centos/RHEL:

```
[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 default_realm = DOMAIN.NET
 dns_lookup_realm = false
 dns_lookup_kdc = false
 ticket_lifetime = 24h
 forwardable = yes

[realms]
 DOMAIN.NET = {
  kdc = adserver.domain.net:88
  admin_server = adserver.domain.net:749
  default_domain = domain.net
 }
[domain_realm]
 .domain.net = DOMAIN.NET
 domain.net = DOMAIN.NET

[appdefaults]
 pam = {
   debug = false
   ticket_lifetime = 36000
   renew_lifetime = 36000
   forwardable = true
   krb4_convert = false
 }
```

For Debian and Ubuntu:

```
  [logging]
  default = FILE:/var/log/krb5libs.log
  kdc = FILE:/var/log/krb5kdc.log
  admin_server = FILE:/var/log/kadmind.log
  [libdefaults]
  default_realm = DOMAIN.NET
  ticket_lifetime = 24h
  forwardable = yes
  [appdefaults]
  pam = {
  debug = false
  ticket_lifetime = 36000
  renew_lifetime = 36000
  forwardable = true
  krb4_convert = false
  }
```

Next, edit `/etc/samba/smb.conf`. Again, here is an example for our `DOMAIN.NET` for Centos/RHEL:

```
 [global]
 workgroup = DOMAIN
 server string = %h
 security = ads
 passdb backend = tdbsam
 realm = DOMAIN.NET
 encrypt passwords = yes
 winbind use default domain = yes
 client NTLMv2 auth = yes
 preferred master = no
 domain master = no
 local master = no
 load printers = no
 log level = 1 winbind:5 auth:3
 winbind max clients = 750
 winbind max domain connections = 15
```

For Debian and Ubuntu:

```
[global]
 workgroup = DOMAIN
 server string = Samba Server Version %v
 security = ads
 realm = DOMAIN.NET
 password server = 192.168.1.1
 domain master = no
 local master = no
 preferred master = no
 winbind separator = +
 winbind enum users = yes
 winbind enum groups = yes
 winbind use default domain = yes
 winbind nested groups = yes
 winbind refresh tickets = yes
 template homedir = /home/%D/%U
 template shell = /bin/bash
 client use spnego = yes
 client ntlmv2 auth = yes
 encrypt passwords = yes
 restrict anonymous = 2
 log file = /var/log/samba/log.%m
 max log size = 50
```

Issue a `kinit` and `klist` in order to get and verify the Kerberos token:

```
# kinit administrator
# klist
```

After that, you need to start samba, and join the machine to the domain:

```
# service smb start
# chkconfig --level 345 smb on
# net ads join -U administrator
```

Note that for Debian and Ubuntu you will probably have this error:

```
# kinit succeeded but ads_sasl_spnego_krb5_bind failed: Invalid credentials
# Join to domain is not valid: Invalid credentials
```

For Centos/RHEL:

```
# usermod -a -G wbpriv pf
```

Finally, start `winbind`, and test the setup using `ntlm_auth` and `radtest`:

```
# service winbind start
# chkconfig --level 345 winbind on
```

For Debian and Ubuntu:

```
# usermod -a -G winbindd_priv pf
# ntlm_auth --username myDomainUser
# radtest -t mschap -x myDomainUser myDomainPassword localhost:18120 12
 testing123
  Sending Access-Request of id 108 to 127.0.0.1 port 18120
      User-Name = "myDomainUser"
      NAS-IP-Address = 10.0.0.1
      NAS-Port = 12
      Message-Authenticator = 0x00000000000000000000000000000000
      MS-CHAP-Challenge = 0x79d62c9da4e55104
      MS-CHAP-Response =
 0x000100000000000000000000000000000000000000000000000091c843b420f0dec4228ed2f26bff07d5e49ad9a297
  rad_recv: Access-Accept packet from host 127.0.0.1 port 18120, id=108,
 length=20
```

Option 2: Local Authentication

Add your user's entries at the end of the `/usr/local/pf/raddb/users` file with the following format:

```
username Cleartext-Password := "password"
```

Option 3: EAP authentication against OpenLDAP

To authenticate 802.1x connection against OpenLDAP you need to define the ldap connection in `/usr/local/pf/raddb/modules/ldap` and be sure that the userpassword is define as a NTHASH or as cleartext.

```
        ldap openldap {
                server = "ldap.acme.com"
                identity = "uid=admin,dc=acme,dc=com"
                password = "password"
                basedn = "dc=district,dc=acme,dc=com"
                filter = "(uid=%{mschap:User-Name})"
                ldap_connections_number = 5
                timeout = 4
                timelimit = 3
                net_timeout = 1
                tls {
                }
                dictionary_mapping = ${confdir}/ldap.attrmap
                edir_account_policy_check = no

                keepalive {
                        # LDAP_OPT_X_KEEPALIVE_IDLE
                        idle = 60

                        # LDAP_OPT_X_KEEPALIVE_PROBES
                        probes = 3

                        # LDAP_OPT_X_KEEPALIVE_INTERVAL
                        interval = 3
                }
        }
```

Next in `/usr/local/pf/raddb/sites-available/packetfence-tunnel` add in the authorize section:

```
authorize {
        suffix
        ntdomain
        eap {
                ok = return
        }
        files
        openldap
    }
```

Option 4: EAP Guest Authentication on email, sponsor and SMS registration

The goal here is to be able to use the credential PacketFence created on guest access and use this one on a secure connection. First create a guest SSID with the guest access you want to use (Email, Sponsor or SMS) and check Add user on email registration and/or Add user on sponsor registration in Configuration → Self Registration section. At the end of the guest registration, PacketFence will send an email with the credentials for Email and Sponsor. For SMS use your phone number and the PIN code.

Note that this option doesn't currently work with the *Reuse dot1x credentials* option of the captive portal.

In `/usr/local/pf/raddb/sites-available/packetfence-tunnel` there is an example on how to configure RADIUS to enable this feature (uncomment to make it work).

In this example we activate this feature on a specific SSID name (Secure-Wireless), disabled by default NTLM Auth, test email credential (pfguest), test sponsor (pfsponsor) and test sms (pfsms). If all failled then we reactivate NTLM Auth.

```
authorize {
        suffix
        ntdomain
        eap {
                ok = return
        }
        files
####Activate local user eap authentication based on a specific SSID ####
## Set Called-Station-SSID with the current SSID
#        set.called_station_ssid
#        if (Called-Station-SSID == 'Secure-Wireless') {
## Disable ntlm_auth
#            update control {
#                MS-CHAP-Use-NTLM-Auth := No
#            }
## Check password table with email and password for a sponsor registration
#            pfguest
#            if (fail || notfound) {
## Check password table with email and password for a guest registration
#                pfsponsor
#                if (fail || notfound) {
## Check activation table with phone number and PIN code
#                    pfsms
#                    if (fail || notfound) {
#                        update control {
#                           MS-CHAP-Use-NTLM-Auth := Yes
#                        }
#                    }
#                }
#            }
#        }
```

Note

For this feature to work, the users' passwords must be stored in cleartext in the database.

Option 5: EAP Local user Authentication

The goal here is to use the local user you created in the admin GUI for EAP authentication. The logic is exactly the same than in option 4, the difference is that we use another SSID and we only use local accounts.

Edit `/usr/local/pf/raddb/sites-available/packetfence-tunnel`

In this example we activate this feature on a specific SSID name (Secure-local-Wireless), disabled by default NTLM Auth and test local account. If it failled then we reactivate NTLM Auth.

```
####Activate local user eap authentication based on a specific SSID ####
## Set Called-Station-SSID with the current SSID
#        set.called_station_ssid
#        if (Called-Station-SSID == 'Secure-local-Wireless') {
## Disable ntlm_auth
#            update control {
#                MS-CHAP-Use-NTLM-Auth := No
#            }
## Check password table for local user
#            pflocal
#            if (fail || notfound) {
#                update control {
#                    MS-CHAP-Use-NTLM-Auth := Yes
#                }
#            }
#        }
```

Caution

You will need to deasactivate password hashing in the database for local authentication to work. In the administration interface, go in *Configuration* → *Advanced* and set *Database passwords hashing method* to `plaintext`

Tests

Test your setup with `radtest` using the following command and make sure you get an `Access-Accept` answer:

```
# radtest dd9999 Abcd1234 localhost:18120 12 testing123
Sending Access-Request of id 74 to 127.0.0.1 port 18120
  User-Name = "dd9999"
  User-Password = "Abcd1234"
  NAS-IP-Address = 255.255.255.255
  NAS-Port = 12
rad_recv: Access-Accept packet from host 127.0.0.1:18120, id=74, length=20
```

Debugging

Log files

Here are the most important PacketFence log files:

- `/usr/local/pf/logs/packetfence.log` — PacketFence Core Log
- `/usr/local/pf/logs/httpd.portal.access` — Apache – Captive Portal Access Log
- `/usr/local/pf/logs/httpd.portal.error` — Apache – Captive Portal Error Log
- `/usr/local/pf/logs/httpd.admin.access` — Apache – Web Admin/Services Access Log
- `/usr/local/pf/logs/httpd.admin.error` — Apache – Web Admin/Services Error Log
- `/usr/local/pf/logs/httpd.webservices.access` — Apache – Webservices Access Log
- `/usr/local/pf/logs/httpd.webservices.error` — Apache – Webservices Error Log
- `/usr/local/pf/logs/httpd.aaa.access` — Apache – AAA Access Log
- `/usr/local/pf/logs/httpd.aaa.error` — Apache – AAA Error Log

There are other log files in `/usr/local/pf/logs/` that could be relevant depending on what issue you are experiencing. Make sure you take a look at them.

The main logging configuration file is `/usr/local/pf/conf/log.conf`. It contains the configuration for the `packetfence.log` file (`Log::Log4Perl`) and you normally don't need to modify it. The logging configuration files for every service are located under `/usr/local/pf/conf/log.conf.d/`.

RADIUS Debugging

First, check the FreeRADIUS logs. The file is located at `/usr/local/pf/logs/radius.log`.

If this didn't help, run FreeRADIUS in debug mode. To do so, start it using the following command:

```
# radiusd -X -d /usr/local/pf/raddb
```

Additionally there is a `raddebug` tool that can extract debug logs from a running FreeRADIUS daemon. PacketFence's FreeRADIUS is preconfigured with such support.

In order to have an output from `raddebug`, you need to either:

a. Make sure user `pf` has a shell in `/etc/passwd`, add `/usr/sbin` to PATH (`export PATH=/usr/sbin:$PATH`) and execute `raddebug` as `pf`

b. Run `raddebug` as root (less secure!)

Now you can run `raddebug` easily:

```
raddebug -t 300 -d /usr/local/pf/raddb
```

The above will output FreeRADIUS' debug logs for 5 minutes. See `man raddebug` for all the options.

More on VoIP Integration

VoIP has been growing in popularity on enterprise networks. At first sight, the IT administrators think that deploying VoIP with a NAC poses a huge complicated challenge to resolve. In fact, depending of the hardware you have, not really. In this section, we will see why.

CDP and LLDP are your friend

For those of you who are unaware of the existence of CDP or LLDP (or LLDP-MED), I suggest you start reading on this topic. Cisco Discovery Protocol (CDP) is device-discovery protocol that runs on all Cisco-manufactured equipment including routers, access servers, bridges, and switches. Using CDP, a device can advertise its existence to other devices and receive information about other devices on the same LAN or on the remote side of a WAN. In the world of VoIP, CDP is able to determine if the connecting device is an IP Phone or not, and tell the IP Phone to tag its ethernet frame using the configured voice VLAN on the switchport.

On many other vendors, you are likely to find LLDP or LLDP-MED support. Link Layer Discovery Protocol (LLDP) is a vendor-neutral Link Layer protocol in the Internet Protocol Suite used by network devices for advertising their identity, capabilities, and neighbors. Same as CDP, LLDP can tell an IP Phone which VLAN id is the voice VLAN.

VoIP and VLAN assignment techniques

As you already know, PacketFence supports many VLAN assignment techniques such as port-security, mac authentication or 802.1X. Let's see how VoIP is doing with each of those.

Port-security

Using port-security, the VoIP device rely on CDP/LLDP to tag its ethernet frame using the configured voice VLAN on the switch port. After that, we ensure that a security trap is sent from the voice VLAN so that PacketFence can authorize the mac address on the port. When the PC connects, another security trap will be sent, but from the data VLAN. That way, we will have 1 mac address authorized on the voice VLAN, and 1 on the access VLAN.

Note

Not all vendors support VoIP on port-security, please refer to the Network Configuration Guide.

Mac Authentication and 802.1X

Cisco hardware

On Cisco switches, we are looking at the multi-domain configuration. The multi-domain means that we can have one device on the VOICE domain, and one device on the DATA domain. The domain assignment is done using a Cisco VSA. When the phone connects to the switchport, PacketFence will respond with the proper VSA only, no RADIUS tunneled attributes. CDP then tells the phone to tag its ethernet frames using the configured voice VLAN on the port. When a PC connects, the RADIUS server will return tunneled attributes, and the switch will place the port in the provided access VLAN.

Non-Cisco hardware

On other vendor hardware, it is possible to make VoIP work using RADIUS VSAs. When a phone connects to a switchport, PacketFence needs to return the proper VSA to tell the switch to allow tagged frames from this device. When the PC will connect, we will be able to return standard RADIUS tunnel attributes to the switch, that will be the untagged VLAN.

Note

Again, refer to the Network Configuration Guide to see if VoIP is supported on your switch hardware.

What if CDP/LLDP feature is missing

It is possible that your phone doesn't support CDP or LLDP. If it's the case, you are probably looking at the "DHCP way" of provisionning your phone with a voice VLAN. Some models will ask for a specific DHCP option so that the DHCP server can give the phone a voice VLAN id. The phone will then reboot, and tag its ethernet frame using the provided VLAN tag.

In order to make this scenario work with PacketFence, you need to ensure that you tweak the registration and your production DHCP server to provide the DHCP option. You also need to make sure there is a voice VLAN properly configured on the port, and that you auto-register your IP Phones (On the first connect, the phone will be assigned on the registration VLAN).

Advanced topics

This section covers advanced topics in PacketFence. Note that it is also possible to configure PacketFence manually using its configuration files instead of its Web administrative interface. It is still recommended to use the Web interface.

In any case, the `/usr/local/pf/conf/pf.conf` file contains the PacketFence general configuration. For example, this is the place where we inform PacketFence it will work in VLAN isolation mode.

All the default parameters and their descriptions are stored in `/usr/local/pf/conf/pf.conf.defaults`.

In order to override a default parameter, define it and set it in `pf.conf`.

`/usr/local/pf/conf/documentation.conf` holds the complete list of all available parameters.

All these parameters are also accessible through the web-based administration interface under the Configuration tab. It is highly recommended that you use the web-based administration interface of PacketFence for any configuration changes.

Apple and Android Wireless Provisioning

Apple devices such as iPhones, iPads, iPods and Mac OS X (10.7+) support wireless profile importation using a special XML file format (mobileconfig). Android is also able to support this feature by importing the wireless profile with the Android PacketFence Agent. In fact, installing such file on your Apple device will automatically configure the wireless settings for a given SSID. This feature is often used when the SSID is hidden, and you want to ease the configuration steps on the mobile device (because it is often painful to configure manually). In PacketFence, we are going further, we generate the profile according to the administrator's preference and we pre-populate the file with the user's credentials (without the password). The user simply needs to install its generated file and he will be able to use the new SSID.

Configure the feature

First of all, you need to configure the SSID that your devices will use after they go though the authentication process.

In order to do that, in the administration interface, go in *Configuration/Provisioners*. Then select the *android* provisioner. Enter the SSID and save.

Now do the same thing for the iOS provisioner.

After, you simply need to add the *Android* and *iOS* provisioners to your *Portal Profile* configuration.

For Android, you must allow passthroughs in your `pf.conf` configuration file:

```
[trapping]
passthrough=enabled
passthroughs=*.ggpht.com,*.googleusercontent.com,android.clients.google.com,*.googleapis.com,*.and
```

Profile generation

Upon registration, instead of showing the default release page, the user will be showing another version of the page saying that the wireless profile has been generated with a clickable link on it. To install the profile, Apple user owner simply need to click on that link, and follow the instructions on their device. Android user owner simply click to the link and will be forwarded to Google Play to install PacketFence agent. Simply launch the application and click to configure will create the secure SSID profile. It is that simple.

Billing Engine

PacketFence integrates the ability to use a payment gateway to bill users to gain access to the network. When configured, the user who wants to access the network / Internet is prompted by a page asking for it's personnal information as well as it's credit card information.

PacketFence currently supports two payment gateways: Authorize.net and Mirapay.

The configuration to use the feature is fairly simple. The general configuration to enable / disable the billing engine can be done through the Web administration GUI (**Configuration → Portal Profiles and Pages**) or from the `conf/profiles.conf` file:

```
[default]
billing_engine = enabled
...
```

Billing engine parameters are specified in `conf/pf.conf` or from **Configuration → Billing**:

```
[billing]
gateway = authorize_net
authorizenet_posturl = The payment gateway processing URL
authorizenet_login = The merchant's unique API Login ID
authorizenet_trankey = The merchant's unique Transaction Key
```

It is also possible to configure multiple network access with different prices. For example, you may want to provide basic Internet access with a decent speed at a specific price and another package with high speed connection at another price.

Caution

The use of different billing tiers requires different roles in PacketFence. Make sure to create these roles first otherwise you will run into problems.

To do so, some customizations is needed to the billing module. You'll need to redefined the `getAvailableTiers` method in the `lib/pf/billing/custom.pm` file. An example is already in place in the file.

To assign a role by tiers (example: slow, medium and fast), edit the file `lib/pf/billing/custom.pm`

```
my %tiers = (
        tier1 => {
            id => "tier1",
            name => "Tier 1",
            price => "1.00",
            timeout => "7D",
            usage_duration => '1D',
            category => '',
            description => "Tier 1 Internet Access", destination_url => "http://
www.packetfence.org"
        },
);
```

id is used as the item value of the billing table.

name is the name of the tier used on billing.html.

price is amount charged on the credit card.

timeout is used to compute the unregistration date of the node.

usage_duration is the amount of non-contignuous access time for the node, set as the time_balance value of the node table.

category is the role in which to put the node.

description will appear on the billing.html.

destination_url is the url that the device will be redirected after a successful authentication.

Devices Registration

Users have the possibility to register their devices (Microsoft XBOX/XBOX360, Nintendo DS/Wii, Sony PlayStation and so on) right from a special portal page. When accessing this page, users will be prompted to login as if they were registering themselves. Once logged in, the portal will ask them to enter the device MAC address that will then be matched against a predefined list of authorized MAC OUI. The device will be registered with the user's id and can be assigned into a specific category for easier management.

Here's how to configure the whole thing. The portal page can be accessed by the following URL: https://YOUR_PORTAL_HOSTNAME/device-registration This URL is accessible from within the network, in any VLAN that can reach the PacketFence server.

The following can be configured by editing the pf.conf file:

```
[registration]
device_registration = enabled
device_registration_role = gaming
```

Make sure the role exists in PacketFence otherwise you will encounter registration errors. Moreover, make sure the role mapping for your particular equipment is done.

These parameters can also be configured from the **Configuration → Registration** section.

Note

A *portal* interface type is required to use this feature. A *portal* interface type can be added to any network interface using the web admin GUI.

Eduroam

> eduroam (education roaming) is the secure, world-wide roaming access service developed for the international research and education community.
>
> eduroam allows students, researchers and staff from participating institutions to obtain Internet connectivity across campus and when visiting other participating institutions by simply opening their laptop.
>
> — eduroam *https://www.eduroam.org/*

PacketFence supports integration with eduroam and allows participating institutions to authenticate both locally visiting users from other institutions as well as allowing other institutions to authenticate local users.

In order for PacketFence to allow eduroam authentication, the FreeRADIUS configuration of PacketFence must be modified to allow the eduroam servers to connect to it as clients as well as to proxy RADIUS authentication requests for users from outside institutions.

First, modify the /usr/local/pf/raddb/clients.conf file to allow the eduroam servers to connect to your PacketFence server. Add the eduroam servers as clients and make sure to add the proper RADIUS secret. Set a shortname to refer to these clients as you will later need it to exclude them from some parts of the PacketFence configuration.

clients.conf example:

```
client tlrs1.eduroam.us {
        secret = useStrongerSecret
        shortname = tlrs1
}

client tlrs2.eduroam.us {
        secret = useStrongerSecret
        shortname = tlrs2
}
```

Secondly, modify the list of domains and proxy servers in /usr/local/pf/raddb/proxy.conf. You will need to define each of your domains as well as the DEFAULT domain. The DEFAULT realm will apply to any client that attempts to authenticate with a realm that is not otherwise defined in proxy.conf and will be proxied to the eduroam servers.

Define one or more home servers (servers to which eduroam requests should be proxied).

proxy.conf example:

```
home_server tlrs1.eduroam.us {
        type = auth
        ipaddr = 257.128.1.1
        port = 1812
        secret = useStrongerSecret
        require_message_authenticator = yes
}
```

Define a pool of servers to group your eduroam home servers together.

proxy.conf example:

```
home_server_pool eduroam {
        type = fail-over
        home_server = tlrs1.eduroam.us
        home_server = tlrs2.eduroam.us
}
```

Define realms to select which requests should be proxied to the eduroam server pool. There should be one realm for each of your domains, and possibly one more per domain if you intend to allow usernames of the DOMAIN\user form.

The REALM is set based on the domain found by the suffix or ntdomain modules (see raddb/modules/realm). The suffix or ntdomain modules try to find a domain either with an @domain or suffix\username.

- If none is found, the REALM is NULL.
- If a domain is found, FreeRADIUS tries to match one of the REALMS defined in this file.
- If the domain is either example.edu or EXAMPLE FreeRADIUS sets the corresponding REALM, i.e. example.edu or EXAMPLE.
- If the REALM does not match either (and it isn't NULL), that means there was a domain other than EXAMPLE or example.edu and we assume it is meant to be proxied to eduroam. FreeRADIUS sets the DEFAULT realm (which is proxied to the eduroam authentication pool).

The REALM determines where the request is sent to. If the REALM authenticates locally the requests are processed entirely by FreeRADIUS. If the REALM sets a different home server pool, the requests are proxied to the servers defined within that pool.

proxy.conf example:

```
# This realm is for requests which don't have an explicit realm
# prefix or suffix.  User names like "bob" will match this one.
# No authentication server is defined, thus the authentication is
# done locally.
realm NULL {
}

# This realm is for ntdomain users who might use the domain like
# this "EXAMPLE\username".
# No authentication server is defined, thus the authentication is
# done locally.
realm EXAMPLE {
}

# This realm is for suffix users who use the domain like this:
# "username@example.edu".
# No authentication server is defined, thus the authentication is
# done locally.
realm example.edu {
}

# This realm is for ALL OTHER requests. Meaning in this context,
# eduroam. The auth_pool is set to the eduroam pool and so the
# requests will be proxied.
realm DEFAULT {
    auth_pool  = eduroam
    nostrip
}
```

Thirdly, you must configure the packetfence FreeRADIUS virtual servers to treat the requests properly.

In /usr/local/pf/raddb/sites-enabled/packetfence, modify the authorize section like this:

raddb/sites-enabled/packetfence example:

```
authorize {
        # pay attention to the order of the modules. It matters.
        ntdomain
        suffix
        preprocess

        # uncomment this section if you want to block eduroam users from
        # you other SSIDs. The attribute name ( Called-Station-Id ) may
        # differ based on your controller
        #if ( Called-Station-Id !~ /eduroam$/i) {
        #    update control {
        #        Proxy-To-Realm := local
        #    }
        #}

        eap {
                ok = return
        }


        files
        expiration
        logintime
        packetfence
}
```

In /usr/local/pf/raddb/sites-enabled/packetfence-tunnel, modify the post-auth section like this. If you omit this change the request will be sent to PacketFence where it will be failed since the eduroam servers are not part of your configured switches.

raddb/sites-enabled/packetfence-tunnel example:

```
post-auth {
        exec

        # we skip packetfence when the request is coming from the eduroam servers
        if ( "%{client:shortname}" != "tlrs1" && \
             "%{client:shortname}" != "tlrs2"    ) {
                packetfence
        }

        Post-Auth-Type REJECT {
                attr_filter.access_reject
        }
}
```

Finally, make sure that the realms module is configured this way (see /usr/local/pf/raddb/modules/realm):

raddb/modules/realm example:

```
# 'username@realm'
realm suffix {
        format = suffix
        delimiter = "@"
}

#  'domain\user'
realm ntdomain {
        format = prefix
        delimiter = "\\"
        ignore_null = yes
}
```

Fingerbank integration

Fingerbank, a great device profiling tool developed alongside of PacketFence, now integrates with it to power-up the feature set allowing a PacketFence administrator to easily trigger violations based on different device types, device parents, DHCP fingerprints, DHCP vendor IDs, MAC vendors and browser user agents.

The core of that integration resides in the ability for a PacketFence system, to interact with the Fingerbank upstream project, which then allow a daily basis fingerprints database update, sharing unknown data so that more complex algorithms can process that new data to integrate it in the global database, querying the global upstream database in the case of an unknown match and much more.

Since the Fingerbank integration is now the "de facto" device profiling tool of PacketFence, it was a requirement to make it as simple as possible to configure and to use. From the moment a working PacketFence system is in place, Fingerbank is also ready to be used, but only in a "local" mode, which means, no interaction with the upstream Fingerbank project.

Onboarding

To benefit from all the advantages of the Fingerbank project, the onboarding step is required to create an API key that will then allow interaction with the upstream project. That can easily be done only by going in the "Settings" menu item under the "Fingerbank" section of the PacketFence "Configuration" tab. From there, an easy process to create and save an user/organization specific API key can be followed. Once completed, the full feature set of Fingerbank can be used.

Update Fingerbank database

Updating the Fingerbank data can't be easier. The only requirement is the onboarding process which allows you to interact with upstream project. Once done, an option to "Update Fingerbank DB" can be found on top of every menu item sections under "Fingerbank". Process may take a minute or two, depending on the size of the database and the internet connectivity, after which a success or error message will be show accordingly. "Local" records are NOT being modified during this process.

Submit unknown data

Saying that we don't know everything is not false modesty. In that sense, the "Submit Unknown/ Unmatched Fingerprints" option is made available (after onboarding) so that unknown fingerprinting data going in and out on your network can easily be submitted to the upstream Fingerbank project for further analysis and integration the in the global database.

Upstream interogation

By default, PacketFence is configured to interogate the upstream Fingerbank project (if onboarding has been completed) to fullfill a query with unmatched local results. Unmatched local results can result of an older version of the Fingerbank database or a requirement for a more complex algorithm due to the data set. That behavior is completely transparent and can be modified using the "Settings" menu item under the "Fingerbank"section of the PacketFence "Configuration" tab.

Local entries

It is possible for an administrator who wants to customize an existing record (or create a new one) to do so using the "Local" entries. An upstream record (DHCP Fingerprint, DHCP Vendor, MAC Vendor, User Agent, Device type, even a Combination) can be cloned and then modified on a local basis if needed. Local records are always matched first since their purpose is to *override* an existing one. A local combination can be created to match either "Local" or "Upstream" or both entries to allow identification of a device.

Settings

Fingerbank settings can easily be modified from the "Settings" menu item under the "Fingerbank" section of the PacketFence "Configuration" tab. There's documentation for each an every parameter that allow easier understanding.

Floating Network Devices

Starting with version 1.9, PacketFence now supports floating network devices. A Floating network device is a device for which PacketFence has a different behaviour compared to a regular device. This functionality was originally added to support mobile Access Points.

Caution

Right now PacketFence only supports floating network devices on Cisco and Nortel switches configured with port-security.

For a regular device, PacketFence put it in the VLAN corresponding to its status (Registration, Quarantine or Regular Vlan) and authorizes it on the port (port-security).

A floating network device is a device that PacketFence does not manage as a regular device.

When a floating network device is plugged, PacketFence will let/allow all the MAC addresses that will be connected to this device (or appear on the port) and if necessary, configure the port as multi-vlan (trunk) and set PVID and tagged VLANs on the port.

When an floating network device is unplugged, PacketFence will reconfigure the port like before it was plugged.

How it works

Configuration:

- floating network devices have to be identified using their MAC address.
- linkup/linkdown traps are not enabled on the switches, only port-security traps are.

When PacketFence receives a port-security trap for a floating network device, it changes the port configuration so that:

- it disables port-security
- it sets the PVID
- it eventually sets the port as multi-vlan (trunk) and sets the tagged Vlans
- it enables linkdown traps

When PF receives a linkdown trap on a port in which a floating network device was plugged, it changes the port configuration so that:

- it enables port-security
- it disables linkdown traps

Identification

As we mentioned earlier, each floating network device has to be identified. There are two ways to do it:

- by editing `conf/floating_network_device.conf`
- through the Web GUI, in **Configuration → Network → Floating devices**

Here are the settings that are available:

MAC Address	MAC address of the floating device
IP Address	IP address of the floating device (not required, for information only)
trunkPort	Yes/no. Should the port be configured as a muti-vlan port?
pvid	VLAN in which PacketFence should put the port
taggedVlan	Comma separated list of VLANs. If the port is a multi-vlan, these are the Vlans that have to be tagged on the port.

OAuth2 Authentication

The captive portal of PacketFence allows a guest/user to register using his Google, Facebook, LinkedIn, Windows Live, Twitter or Github account.

For each providers, we maintain an allowed domain list to punch holes into the firewall so the user can hit the provider login page. This list is available in each OAuth2 authentication source.

In order to have oauth2 working properly, you need to enable IP forwarding on your servers. To do it permanently, look in the `/etc/sysctl.conf`, and set the following line:

```
# Controls IP packet forwarding
net.ipv4.ip_forward = 1
```

Save the file, and issue a `sysctl -p` to update the OS config.

You must also enable the passthrough option in your PacketFence configuration (trapping.passthrough in pf.conf).

Google

In order to use Google as a OAuth2 provider, you need to get an API key to access their services. Sign up here : http://code.google.com/apis/console. Make sure you use this URI for the "Redirect URI" field : https://YOUR_PORTAL_HOSTNAME/oauth2/google. Of course, replace the hostname with the values from `general.hostname` and `general.domain`.

You can keep the default configuration, modify the App ID & App Secret (Given by Google on the developper plateform) and Portal URL (https://YOUR_PORTAL_HOSTNAME/oauth2/google).

Also, add the following Authorized domains : *.google.com, *.google.ca, *.google.fr, *.gstatic.com,googleapis.com,accounts.youtube.com (Make sure that you have the google domain from your country like Canada *.google.ca, France *.google.fr, etc...)

Once you have your client id, and API key, you need to configure the OAuth2 provider. This can be done by adding a Google OAuth2 authentication source from **Configuration → Sources**.

Moreover, don't forget to add Google as a registration mode from your portal profile definition, available from **Configuration → Portal Profiles and Pages**.

Facebook

To use Facebook, you also need an API code and a secret key. To get one, go here: https://developers.facebook.com/apps. When you create your App, make sure you specify the following as the Website URL: https://YOUR_PORTAL_HOSTNAME/oauth2/facebook

Of course, replace the hostname with the values from `general.hostname` and `general.domain`.

You can keep the default configuration, modify the App ID & App Secret (Given by FaceBook on the developper plateform) and Portal URL (https://YOUR_PORTAL_HOSTNAME/oauth2/facebook).

Also, add the following Authorized domains : *.facebook.com, *.fbcdn.net, *.akamaihd.net (May change)

Once you have your information, you need to configure the OAuth2 provider. This can be done by adding a Facebook OAuth2 authentication source from **Configuration → Sources**.

Moreover, don't forget to add Facebook as a registration mode from your portal profile definition, available from **Configuration → Portal Profiles and Pages**.

Caution

By allowing OAuth through Facebook, you will give Facebook access to the users while they are sitting in the registration VLAN.

GitHub

To use GitHub, you also need an API code and a secret key. To get one, you need to create an App here: https://github.com/settings/applications. When you create your App, make sure you specify the following as the Callback URL https://YOUR_PORTAL_HOSTNAME/oauth2/github

Of course, replace the hostname with the values from `general.hostname` and `general.domain`.

Once you have your information, you need to configure the OAuth2 provider. This can be done by adding a GitHub OAuth2 authentication source from **Configuration → Sources**.

Moreover, don't forget to add GitHub as a registration mode from your portal profile definition, available from **Configuration → Portal Profiles and Pages**.

LinkedIn

To use LinkedIn, you also need an API code and a secret key. To get one, you need to create an App here: https://developer.linkedin.com/. When you create your App, make sure you specify the following as the Callback URL https://YOUR_PORTAL_HOSTNAME/oauth2/linkedin

Of course, replace the hostname with the values from `general.hostname` and `general.domain`.

Once you have your information, you need to configure the OAuth2 provider. This can be done by adding a LinkedIn OAuth2 authentication source from **Configuration → Sources**.

Moreover, don't forget to add LinkedIn as a registration mode from your portal profile definition, available from **Configuration → Portal Profiles and Pages**.

Also, LinkedIn requires a *state* parameter for the authorization URL. If you modify it, make sure to add it at the end of your URL.

Twitter

To use Twitter, you also need an API code and a secret key which Twitter calls *consumer key* and *consumer secret*. Obtain this information by creating an new application from your Twitter Apps Management page. When you create your App, make sure you specify the following as the *Callback URL* https://YOUR_PORTAL_HOSTNAME/oauth2/twitter

Of course, replace the hostname with the values from `general.hostname` and `general.domain`.

Once you have your information, you need to configure the OAuth2 provider. This can be done by adding a Twitter OAuth2 authentication source from **Configuration → Sources**.

Moreover, don't forget to add Twitter as a registration mode from your portal profile definition, available from **Configuration → Portal Profiles and Pages**.

Windows Live

To use Windows live, you also need an API code and a secret key. To get one, you need to create an App here: https://account.live.com/developers/applications. When you create your App, make sure you specify the following as the Callback URL https://YOUR_PORTAL_HOSTNAME/oauth2/windowslive

Of course, replace the hostname with the values from `general.hostname` and `general.domain`.

Once you have your information, you need to configure the OAuth2 provider. This can be done by adding a WindowsLive OAuth2 authentication source from **Configuration → Sources**.

Moreover, don't forget to add WindowsLive as a registration mode from your portal profile definition, available from **Configuration → Portal Profiles and Pages**.

Passthrough

In order to use the passthrough feature in PacketFence, you need to enable it from the GUI in **Configuration → Trapping** and check **Passthrough**.

There are two solutions for passthroughs - one using DNS resolution and iptables and the other one using Apache's mod_proxy module. When enabled, PacketFence will use `pfdns` if you defined **Passthroughs**, or Apache mod-proxy if you defined **Proxy Passthroughs** to allow trapped devices to reach external websites.

- **DNS passthrough:** Add a new FQDN (should be a wildcard domain like *.google.com) in the Passthroughs section. When PacketFence receives a DNS request for this domain, it will answer the real IP address and punch a hole in the firewall (using iptables) to allow access. With this method, PacketFence must be the default gateway of your device.
- **mod_proxy passthrough:** Add a new FQDN (should be a wildcard domain like *.google.com) in the Proxy Passthroughs section. For this FQDN, PacketFence will answer the IP address of the captive portal and when a device hits the captive portal, PacketFence will detect that this FQDN has a passthrough configuration and will forward the traffic to mod_proxy.

These two methods can be used together but DNS-based passthroughs have higher priority.

Production DHCP access

In order to perform all of its access control duties, PacketFence needs to be able to map MAC addresses into IP addresses.

For all the networks/VLANs where you want PacketFence to have the ability to isolate a node or to have IP information about nodes, you will need to perform **one** of the techniques below.

Also note that this doesn't need to be done for the registration, isolation VLANs and inline interfaces since PacketFence acts as the DHCP server in these networks.

IP Helpers (recommended)

If you are already using IP Helpers for your production DHCP in your production VLANs this approach is the simplest one and the one that works the best.

Add PacketFence's management IP address as the last `ip helper-address` statement in your network equipment. At this point PacketFence will receive a copy of all DHCP requests for that VLAN and will record what IP were distributed to what node using a `pfdhcplistener` daemon.

By default no DHCP Server should be running on that interface where you are sending the requests. This is by design otherwise PacketFence would reply to the DHCP requests which would be a bad thing.

Obtain a copy of the DHCP traffic

Get a copy of all the DHCP Traffic to a dedicated physical interface in the PacketFence server and run `pfdhcplistener` on that interface. It will involve configuring your switch properly to perform port mirroring (aka network span) and adding in PacketFence the proper interface statement at the operating system level and in `pf.conf`.

`/etc/sysconfig/network-scripts/ifcfg-eth2`:

```
DEVICE=eth2
ONBOOT=yes
BOOTPROTO=none
```

Add to `pf.conf`: (IPs are not important they are there only so that PacketFence will start)

```
[interface eth2]
mask=255.255.255.0
type=dhcp-listener
gateway=192.168.1.5
ip=192.168.1.1
```

Restart PacketFence and you should be good to go.

Interface in every VLAN

Because DHCP traffic is broadcast traffic, an alternative for small networks with few local VLANs is to put a VLAN interface for every VLAN on the PacketFence server and have a `pfdhcplistener` listen on that VLAN interface.

On the network side you need to make sure that the VLAN truly reaches all the way from your client to your DHCP infrastructure up to the PacketFence server.

On the PacketFence side, first you need an operating system VLAN interface like the one below. Stored in `/etc/sysconfig/network-scripts/ifcfg-eth0.1010`:

```
# Engineering VLAN
DEVICE=eth0.1010
ONBOOT=yes
BOOTPROTO=static
IPADDR=10.0.101.4
NETMASK=255.255.255.0
VLAN=yes
```

Then you need to specify in `pf.conf` that you are interested in that VLAN's DHCP by setting type to `dhcp-listener`.

```
[interface eth0.1010]
mask=255.255.255.0
type=dhcp-listener
gateway=10.0.101.1
ip=10.0.101.4
```

Repeat the above for all your production VLANs then restart PacketFence.

Host production DHCP on PacketFence

It's an option. Just modify `conf/dhcpd.conf` so that it will host your production DHCP properly and make sure that a `pfdhcplistener` runs on the same interface where production DHCP runs. However, please note that this is **NOT** recommended. See this ticket to see why.

Proxy Interception

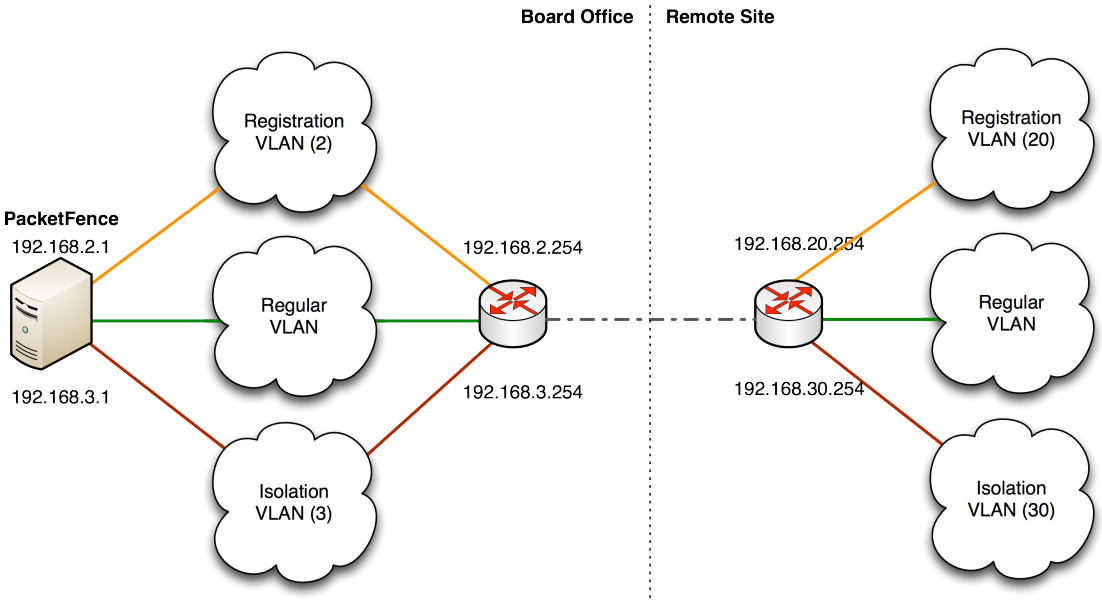
PacketFence enables you to intercept proxy requests and forward them to the captive portal. It only works one layer-2 networks because PacketFence must be the default gateway. In order to use the Proxy Interception feature, you need to enable it from the GUI in **Configuration → Trapping** and check **Proxy Interception**.

Add the port you want to intercept (like 8080 or 3128) and add a new entry in the `/etc/hosts` file to resolve the fully qualified domain name (fqdn) of the captive portal to the IP address of the registration interface. This modification is mandatory in order for Apache to receives the proxy requests.

Routed Networks

If your isolation and registration networks are not locally-reachable (at layer 2) on the network, but routed to the PacketFence server, you'll have to let the PacketFence server know this.

PacketFence can even provide DHCP and DNS in these routed networks and provides an easy to use configuration interface.



	Registration VLAN	Isolation VLAN	Isolation VLAN	Registration VLAN
VLAN ID	2	3	30	20
Network	192.168.2.0/24	192.168.3.0/24	192.168.30.0/24	192.168.20.0/24
Gateway	192.168.2.254	192.168.3.254	192.168.30.254	192.168.20.254
DHCP	192.168.2.1	192.168.3.1	192.168.3.1	192.168.2.1
DNS	192.168.2.1	192.168.3.1	192.168.3.1	192.168.2.1
PacketFence	192.168.2.1	192.168.3.1		

For dhcpd, make sure that the clients DHCP requests are correctly forwarded (IP Helpers in the remote routers) to the PacketFence server. Then make sure you followed the instructions in the DHCP and DNS Server Configuration (networks.conf) for your locally accessible network.

If we consider the network architecture illustrated in the above schema, `conf/pf.conf` will include the local registration and isolation interfaces only.

```
[interface eth0.2]
enforcement=vlan
ip=192.168.2.1
type=internal
mask=255.255.255.0
```

```
[interface eth0.3]
enforcement=vlan
ip=192.168.3.1
type=internal
mask=255.255.255.0
```

Note

PacketFence will not start unless you have at least one *internal* interface, so you need to create local registration and isolation VLANs even if you don't intend to use them.

Also, the *internal* interfaces are the only ones on which dhcpd listens, so the remote registration and isolation subnets need to point their DHCP helper-address to those particular IPs.

Then you need to provide the routed networks information to PacketFence. You can do it through the GUI in **Administration → Networks** (or in `conf/networks.conf`).

`conf/networks.conf` will look like this:

```
[192.168.2.0]
netmask=255.255.255.0
gateway=192.168.2.1
next_hop=
domain-name=registration.example.com
dns=192.168.2.1
dhcp_start=192.168.2.10
dhcp_end=192.168.2.200
dhcp_default_lease_time=300
dhcp_max_lease_time=600
type=vlan-registration
named=enabled
dhcpd=enabled
```

```
[192.168.3.0]
netmask=255.255.255.0
gateway=192.168.3.1
next_hop=
domain-name=isolation.example.com
dns=192.168.3.1
dhcp_start=192.168.3.10
dhcp_end=192.168.3.200
dhcp_default_lease_time=300
dhcp_max_lease_time=600
type=vlan-isolation
named=enabled
dhcpd=enabled
```

```
[192.168.20.0]
netmask=255.255.255.0
gateway=192.168.20.254
next_hop=192.168.2.254
domain-name=registration.example.com
dns=192.168.2.1
dhcp_start=192.168.20.10
dhcp_end=192.168.20.200
dhcp_default_lease_time=300
dhcp_max_lease_time=600
type=vlan-registration
named=enabled
dhcpd=enabled
```

```
[192.168.30.0]
netmask=255.255.255.0
gateway=192.168.30.254
next_hop=192.168.3.254
domain-name=isolation.example.com
dns=192.168.3.1
dhcp_start=192.168.30.10
dhcp_end=192.168.30.200
dhcp_default_lease_time=300
dhcp_max_lease_time=600
type=vlan-isolation
named=enabled
dhcpd=enabled
```

DHCP clients on the registration and isolation networks receive the PF server IP as their DNS server (dns=x.x.x.x), and PF spoofs DNS responses to force clients via the portal. However, clients could manually configure their DNS settings to escape the portal. To prevent this you will need to apply an ACL on the access router nearest the clients, permitting access only to the PF server and local DHCP broadcast traffic.

For example, for the VLAN 20 remote registration network:

```
ip access-list extended PF_REGISTRATION
 permit ip any host 192.168.2.1
 permit udp any any eq 67
 deny ip any any log
interface vlan 20
 ip address 192.168.20.254 255.255.255.0
 ip helper-address 192.168.2.1
 ip access-group PF_REGISTRATION in
```

If your edge switches support *vlan-isolation* you can also apply the ACL there. This has the advantage of preventing machines in isolation from attempting to attack each other.

Statement of Health (SoH)

The Statement of Health (SoH) is product that has been developed by Microsoft. In the Microsoft world, this is named Network Access Protection or NAP. On Windows versions from XP SP2 to Windows 7, there is a NAP service installed that can relay health information (Anti-Virus update status, Windows Update status, etc) to a RADIUS Server or a DHCP server. The section below explains you how to do SoH policies with PacketFence.

Installation

By default, we turn SoH off. To enable its support, simply uncomment the following lines in `/usr/local/pf/conf/radiusd/eap.conf`.

```
soh=yes
soh-virtual-server = "soh-server"
```

Restart the RADIUS service afterward.

On the client side, to enable SoH for EAP, do the following (Windows 7 example):

```
sc config napagent start=auto
sc start napagent

:: Wired 802.1X
sc config dot3svc start=auto depend=napagent
sc start dot3svc

netsh nap client show config

:: get the "ID" value for the "EAP Quarantine Enforcement Client"
netsh nap client set enforce id=$ID admin=enable
```

The last step is to select the "Enforce Network Access Protection" checkbox under the EAP profile settings. Those steps can be easily configured using GPOs.

Configuration of SoH policy

In order to enforce a SoH policy, we need to create it first. This is done using the **Configuration → Compliance → Statement of Health** module.

Policy example

Let's walk through an example situation. Suppose you want to display a remediation page to clients that do not have an anti-virus enabled.

The three broad steps are: create a violation class for the condition, then create an SoH filter to trigger the violation when "anti-virus is disabled", and finally, reload the violations.

First, create the proper violation either via the Admin UI, or by editing the `conf/violations.conf` files:

```
[4000001]
desc=No anti-virus enabled
url=/remediation.php?template=noantivirus
actions=trap,email,log
enabled=Y
```

Note

You may also want to set other attributes such as auto_enable, grace, etc.

When done with the violation, visit the Web Administration under **Configuration → Compliance → Statement of Health** and (edit the filter named **Default**, or) use the **Add a filter** button to create

a filter named **antivirus**. Click on **antivirus** in the filter list, and select **Trigger violation** in the action drop-down. Enter the vid of the violation you created above in the input box that appears.

Next, click on **Add a condition**, and select **Anti-virus**, **is**, and **disabled** in the drop-down boxes that appear. Click on the **Save filters** button. Finally, reload the violations either by restarting PacketFence or using the `pfcmd reload violations` command.

The last step is to create a new remediation template called `noantivirus.php` on the filesystem in the `html/captive-portal/violations` folder. Edit it to include the text you want to display to the users.

VLAN Filter Definition

We added the ability to specify filters directly in the portion of code that re-evaluates the VLAN or do a call to the API.

These rules are available in different scopes:

```
ViolationVlan
RegistrationVlan
NormalVlan
InlineVlan
AutoRegister
NodeInfoForAutoReg
```

And can be defined using different criteria like:

```
node_info
switch
ifIndex
mac
connection_type
username
ssid
time
owner
radius_request
```

For example, lets define a rule that prevents a device from connecting when its category is the "default", when the SSID is "SECURE" and when the current time is between 11am and 2pm: from Monday to Friday when it try to connect as a registered device :

```
[category]
filter = node_info
attribute = category
operator = is
value = default
```

```
[ssid]
filter = ssid
operator = is
value = SECURE
```

```
[time]
filter = time
operator = is
value = wd {Mon Tue Wed Thu Fri} hr {11am-2pm}
```

```
[1:category&ssid&time]
scope = NormalVlan
role = nointernet
```

The second example will create a violation if the SSID is OPEN and the owner is igmout

```
[igmout]
filter = owner
operator = is
value = igmout
```

```
[open]
filter = ssid
operator = is
value = OPEN
```

```
[2:igmout&ssid]
scope = NormalVlan
action = trigger_violation
action_param = mac = $mac, tid = 1100012, type = INTERNAL
```

The third example will autoregister the device and assign the role staff to each device where the username is igmout.

```
[igmout]
filter = username
operator = is
value = igmout
```

```
[secure]
filter = ssid
operator = is
value = SECURE
```

```
[3:igmout&secure]
scope = AutoRegister
role = staff
```

```
[4:igmout&secure]
scope = NodeInfoForAutoReg
role = staff
```

You can have a look in the file vlan_filters.conf, there are some examples on how to use and define filters.

Optional components

Blocking malicious activities with violations

Policy violations allow you to restrict client system access based on violations of certain policies. For example, if you do not allow P2P type traffic on your network, and you are running the appropriate software to detect it and trigger a violation for a given client, PacketFence will give that client a "blocked" page which can be customized to your wishes.

In order to be able to block malicious activities, you need to install and configure the SNORT or Suricata IDS to talk with PacketFence.

Snort

Installation

The installation procedure is quite simple for SNORT. We maintain a working version on the PacketFence repository. To install it, simply run the following command:

```
yum install snort
```

Configuration

PacketFence provides a basic `snort.conf` template that you may need to edit depending of the Snort version. The file is located in `/usr/local/pf/conf`. It is rarely necessary to change anything in that file to make Snort work and trap alerts. DO NOT edit the `snort.conf` located in `/usr/local/pf/var/conf`, all the modification will be destroyed on each PacketFence restart.

Suricata

Installation

Since the suricata IDS is not packaged with the distros (except maybe Fedora, which we do not officially support), you need to build it the "old" way.

The OISF provides a really well written how-to for that. It's available here: https://redmine.openinfosecfoundation.org/projects/suricata/wiki/CentOS5

Configuration

PacketFence will provide you with a basic `suricata.yaml` that you can modify to suit you own needs. The file is located in `/usr/local/pf/conf`.

Violations

In order to make PacketFence react to the Snort alerts, you need to explicitly tell the software to do so. Otherwise, the alerts will be discarded. This is quite simple to accomplish. In fact, you need to create a violation and add the Snort alert SID in the trigger section of a Violation.

PacketFence policy violations are controlled using the `/usr/local/pf/conf/violations.conf` configuration file. The violation format is as follows:

```
[1234]
desc=Your Violation Description
priority=8
template=<template>
enable=Y
trigger=Detect::2200032,Nessus::11808
actions=email,log,trap
vlan=isolationVlan
whitelisted_categories=
```

`[1234]` The violation ID. Any integer except 1200000-120099 which is reserved for required administration violations.

`desc` single line description of violation

`priority` Range 1-10, with 1 the higest priority and 10 the lowest. Higher priority violations will be addressed first if a host has more than one.

`template` Template name to use while in violation. It must match a HTML file name (without the extension) of the `violations` templates directory.

`enable` If enable is set to *N*, this violation is disabled and no additional violations of this type will be added.

`trigger` Method to reference external detection methods. Trigger is formatted as follows `type::ID`. The type can be `Detect` (Snort), `Nessus`, `OpenVAS`, `OS` (DHCP Fingerprint Detection), `UserAgent` (Browser signature), `VendorMAC` (MAC address class), `SoH` (Statement of Health filter), `Accounting`, etc. In the above example, 2000032 is the Snort ID and 11808 is the Nessus plugin number. The Snort ID does NOT have to match the violation ID.

`actions` This is the list of actions that will be executed on a violation addition. The actions can be:

- `log` Log a message to the file specified in `[alerting].log`
- `email` Email the address specified in `[alerting].emailaddr`, using `[alerting].smtpserver`. Multiple `emailaddr` can be sperated by comma.
- `trap` Isolate the host and place them in violation. It opens a violation and leaves it open. If trap is not there, a violation is opened and then automatically closed.

	`winpopup`	send a windows popup message. You need to configure `[alerting].winserver`, `[alerting].netbiosname` in `pf.conf` when using this option.
	`external`	execute an external command, specified in `[paths].externalapi`.
	`close`	close the violation ID specified in the vclose field.
	`role`	change the node's role to the one specified in the target_category field.
	`autoreg`	register the node.
	`unreg`	deregister the node.
`vlan`	Destination VLAN where PacketFence should put the client when a violation of this type is open. The VLAN value can be:	
	`isolationVlan`	Isolation VLAN as specified in `switches.conf`. This is the recommended value for most violation types.
	`registrationVlan`	Registration VLAN as specified in `switches.conf`.
	`normalVlan`	Normal VLAN as specified in `switches.conf`. Note: It is preferable not to trap than to trap and put in normal VLAN. Make sure you understand what you are doing.
	`whitelisted_categories`	Nodes in a category listed in `whitelisted_categories` won't be affected by a violation of this type. Format is a comma separated list of category names.

Also included in `violations.conf` is the defaults section. The defaults section will set a default value for every violation in the configuration. If a configuration value is not specified in the specific ID, the default will be used:

```
[defaults]
priority=4
max_enable=3
actions=email,log
auto_enable=Y
enable=N
grace=120m
delay_by=0
window=0
vclose=
target_category=
button_text=Enable Network
snort_rules=local.rules,bleeding-attack_response.rules,bleeding-
exploit.rules,bleeding-p2p.rules,bleeding-scan.rules,bleeding-virus.rules
vlan=isolationVlan
whitelisted_categories=
```

`max_enable` — Number of times a host will be able to try and self remediate before they are locked out and have to call the help desk. This is useful for users who just *click through* violation pages.

`auto_enable` — Specifies if a host can self remediate the violation (enable network button) or if they can not and must call the help desk.

`grace` — Amount of time before the violation can reoccur. This is useful to allow hosts time (in the example 2 minutes) to download tools to fix their issue, or shutoff their peer-to-peer application.

`delay_by` — Amount of time before the violation action will run.

`window` — Amount of time before a violation will be closed automatically. Instead of allowing people to reactivate the network, you may want to open a violation for a defined amount of time instead. You can use the allowed time modifiers or the dynamic keyword. Note that the dynamic keyword only works for accounting violations. Dynamic will open the violation according to the time you set in the accounting violation (ie. You have an accounting violation for 10GB/month. If you bust the bandwidth after 3 days, the violation will open and the release date will be set for the last day of the current month.)

`vclose` — When selecting the "close" action, triggering the violation will close the one you select in the vclose field. This is an experimentalworkflow for Mobile Device Management (MDM).

`target_category` — When selecting the "role" action, triggering the violation will change the node's role to the one you select in the target_category field.

`button_text` — Text displayed on the violation form to hosts.

`snort_rules`	The Snort rules file is the administrators responsibility. Please change this to point to your violation rules file(s). If you do not specify a full path, the default is `/usr/local/pf/conf/snort`. If you need to include more than one file, just separate each filename with a comma.

Note

`violations.conf` is loaded at startup. A restart is required when changes are made to this file.

Example violation

In our example we want to isolate people using Limewire. Here we assume Snort is installed and configured to send alerts to PacketFence. Now we need to configure PacketFence isolation.

Enable Limewire violation in `/usr/local/pf/conf/violations.conf` and configure it to trap.

```
[2001808]
desc=P2P (Limewire)
priority=8
template=p2p
actions=log,trap
enable=Y
max_enable=1
trigger=Detect::2001808
```

Compliance Checks

PacketFence supports either Nessus, OpenVAS and WMI as a scanning engine for compliance checks. Since PacketFence v5.1 you are now able to create multiples scan engines configuration and assign them on specific captive portals. It mean per example that you are now able to active a scan for specific Operating System only on a specific SSID.

Installation

Nessus

Please visit http://www.nessus.org/download/ to download Nessus v5 and install the Nessus package for your operating system. You will also need to register for the HomeFeed (or the ProfessionalFeed) in order to get the plugins.

After you installed Nessus, follow the Nessus documentation for the configuration of the Nessus Server, and to create a user for PacketFence.

Note

You may run into some issue while using Nessus with the Net::Nessus::XMLRPC module (which is the default behavior in PacketFence). Please refer to the bug tracking system for more information.

OpenVAS

Please visit http://www.openvas.org/install-packages.html#openvas4_centos_atomic to configure the correct repository to be able to install the latest OpenVAS scanning engine.

Once installed, please make sure to follow the instructions to correctly configure the scanning engine and create a scan configuration that will fit your needs. You'll also need to create a user for PacketFence to be able to communicate with the server.

It is important to get the correct scan config ID and NBE report format ID to populate the parameters in the PacketFence configuration file. The easiest way to get these IDs is by downloading both of the scan configuration and report format from the OpenVAS web gui and retrieve the IDs in the filenames.

For example `report-format-f5c2a364-47d2-4700-b21d-0a7693daddab.xml` gives report format ID `f5c2a364-47d2-4700-b21d-0a7693daddab`.

WMI

You just have to enable wmi on each windows devices with a GPO from Active Directory.

Configuration

In order for the compliance checks to correctly work with PacketFence (communication and generate violations inside PacketFence), you need to configure these sections:

Scanner Definition

First go in Configuration and Scanner Definition:

Then add a scan:

There are common parameters for each scan engines:

```
Name: the name of your scan engine
Roles: Only devices with these role(s) will be affected (Optional)
OS: Only devices with this Operating System will be affected (Optional)
Duration: Approximate duration of scan (Progress bar on the captive portal)
Scan before registration: Trigger the scan when the device appear on the
 registration vlan
Scan after registration: Trigger the scan just after registration on the captive
 portal
Scan after registration: Trigger the scan on the production network
 (pfdhcplistener must receive production dhcp traffic)
802.1x: Even if the auto-registration has been enabled, the scan will be trigger
 on a EAP connection
802.1x types: comma delimited EAP type that will trigger the scan if 802.1x above
 has been enabled
```

Specific to Nessus:

```
Hostname or IP Address: Hostname or IP Address where Nessus is running
Username: Username to connect to Nessus scan
Password: Password to connect to Nessus scan
Port of the service: port to connect (default 8834)
Nessus client policy: the name of the policy to use for the scan (Must be define
 on the Nessus server)
```

Specific to OpenVAS:

```
Hostname or IP Address: Hostname or IP Address where OpenVAS is running
Username: Username to connect to OpenVAS scan
Password: Password to connect to OpenVAS scan
Port of the service: port to connect (default 9390)
OpenVAS config ID: the ID of scanning configuration on the OpenVAS server
```

Specific to WMI:

```
Username: A username from Active Directory that is allowed to connect to wmi
Domain: Domain of the Active Directory
Password: Password of the account
WMI Rules: Ordered list of WMI rules you defined in Configuration -> WMI Rules
 Definition
```

WMI Rules Definition

If you have configured a WMI scan engine then you need to define WMI Rules. WMI is a sort of database on each windows devices, to retreive information on the device you need to know the sql request. In order to help you to find and make a rule you can use a third party tool like WMI Explorer.

Go in configuration → WMI Rules Definition:

There are already 3 rules defined:

```
Software_Installed
logged_user
Process_Running
```

Let's take the Software_Installed rule:

```
request: select * from Win32_Product
```

```
Rules Actions:
```

```
[Google]
attribute = Caption
operator = match
value =Google
```

```
[1:Google]
action=trigger_violation
action_param = mac = $mac, tid = 888888, type = INTERNAL
```

This rule will do the following:

```
retreive all the installed software on the device and test if the attribute
 Caption contain Google.
if it matched then we will trigger a violation (with the trigger
 internal::888888) for the mac address of the device.
```

The second one, logged_user:

```
request: select UserName from Win32_ComputerSystem
```

```
Rules Actions:
```

```
[UserName]
attribute = UserName
operator = match
value = (.*)
```

```
[1:UserName]
action = dynamic_register_node
action_param = mac = $mac, username = $result->{'UserName'}
```

This rule will do the following:

```
retreive the current logged user on the device and register the device based on
 the user account.
```

The last one, Process_Running:

```
request: select Name from Win32_Process
```

```
Rules Actions:
```

```
[explorer]
attribute = Name
operator = match
value = explorer.exe
```

```
[1:explorer]
action = allow
```

This rule will do the following:

```
retreive all the running process on the device and if one match explorer.exe then
 we bypass the scan.
```

- Rules syntax

  ```
  the syntax of the rules are simple to understand:
  ```

  ```
  the request is the sql request you will launch on the remote device, you must
   know what the request will return
  to write the test.
  ```

  ```
  Inside the Rules Actions we define 2 sorts of blocs:
  The test bloc (ie [explorer]) and the action bloc (ie [1:explorer])
  ```

  ```
  The test bloc is a simple test based on the result of the request:
  - attribute is the attribute you want to test
  - operator can be:
    is
    is_not
    match
    match_not
  - value is the value you want to compare
  ```

  ```
  Feel free to define multiples test blocs
  ```

  ```
  The action bloc is where you will define your logic, per example let's take
   this one [1:google&explorer], this mean that if the google test is
  true and explorer is true then we execute the action.
  The logic can be more complex and can be something like that [1:!google|
  (explorer&memory)] that mean if not google or (explorer and memory)
  ```

Violations definition

You need to create a new violation section and have to specify:

Using Nessus:

```
trigger=Nessus::<violationId>
```

Using OpenVAS:

```
trigger=OpenVAS::<violationId>
```

Where `violationId` is either the ID of the Nessus plugin or the OID of the OpenVAS plugin to check for. Once you have finished the configuration, you need to reload the violation related database contents using:

```
$ pfcmd reload violations
```

Note

Violations will trigger if the plugin is higher than a low severity vulnerability.

Assign Scan definition to portal profiles

The last step is to assign one or more scanner you configured to one or more portal profiles. Go in Configuration → Portal Profiles → Edit a Portal → Add Scan

Hosting Nessus / OpenVAS remotely

Because of the CPU intensive nature of an automated vulnerability assessment, we recommend that it is hosted on a separate server for large environments. To do so, a couple of things are required:

- PacketFence needs to be able to communicate to the server on the port specified by the vulnerability engine used
- The scanning server need to be able to access the targets. In other words, registration VLAN access is required if scan on registration is enabled.

If you are using the OpenVAS scanning engine:

- The scanning server need to be able to reach PacketFence's Admin interface (on port 1443 by default) by its DNS entry. Otherwise PacketFence won't be notified of completed scans.
- You must have a valid SSL certificate on your PacketFence server

If you are using the Nessus scanning engine:

- You just have to change the host value by the Nessus server IP.

RADIUS Accounting

RADIUS Accounting is usually used by ISPs to bill clients. In PacketFence, we are able to use this information to determine if the node is still connected, how much time it has been connected, and how much bandwitdh the user consumed.

Violations

Using PacketFence, it is possible to add violations to limit bandwidth abuse. The format of the trigger is very simple:

```
Accounting::[DIRECTION][LIMIT][INTERVAL(optional)]
```

Let's explain each chunk properly:

- `DIRECTION`: You can either set a limit to inbound(IN), outbound(OUT), or total(TOT) bandwidth
- `LIMIT`: You can set a number of bytes(B), kilobytes(KB), megabytes(MB), gigabytes(GB), or petabytes(PB)
- `INTERVAL`: This is actually the time window we will look for potential abuse. You can set a number of days(D), weeks(W), months(M), or years(Y).

Example triggers

- Look for Incoming (Download) traffic with a 50GB/month

```
Accounting::IN50GB1M
```

- Look for Outgoing (Upload) traffic with a 500MB/day

```
Accounting::OUT500MB1D
```

- Look for Total (Download + Upload) traffic with a 200GB limit in the last week

```
Accounting::TOT200GB1W
```

Grace period

When using such violation feature, setting the grace period is really important. You don't want to put it too low (ie. A user re-enable his network, and get caught after 1 bytes is tranmitted!) or too high. We recommend that you set the grace period to one interval window.

Oinkmaster

Oinkmaster is a perl script that enables the possibility to update the different snort rules very easily. It is simple to use, and install. This section will show you how to implement Oinkmaster to work with PacketFence and Snort.

Please visit http://oinkmaster.sourceforge.net/download.shtml to download oinkmaster. A sample oinkmaster configuration file is provided at `/usr/local/pf/addons/snort/oinkmaster.conf`.

Configuration

Here are the steps to make Oinkmaster work. We will assume that you already downloaded the newest oinkmaster archive:

1. Untar the freshly downloaded Oinkmaster
2. Copy the required perl scripts into `/usr/local/pf/oinkmaster`. You need to copy over `contrib` and `oinkmaster.pl`
3. Copy the `oinkmaster.conf` provided by PacketFence (see the section above) in `/usr/local/pf/conf`
4. Modify the configuration to suit your own needs. Currently, the configuration file is set to fetch the bleeding rules.

Rules update

In order to get periodic updates for PacketFence Snort rules, we simply need to create a `crontab` entry with the right information. The example below shows a `crontab` entry to fetch the updates daily at 23:00 PM:

```
0 23 * * * (cd /usr/local/pf; perl oinkmaster/oinkmaster.pl -C conf/
oinkmaster.conf -o conf/snort/)
```

Guests Management

PacketFence supports the ability to manage guests by establishing expire dates and assign different roles which will permit different accesses to the network resources.

Guests can self-register themselves using an activation code sent to their mobile phone or they can use their email address and receive and activation link to activate their network access.

PacketFence has the option to have guests sponsored their access by local staff. Once a guest requests a sponsored access an email is sent to the sponsor and the sponsor must click on a link and authenticate in order to enable his access.

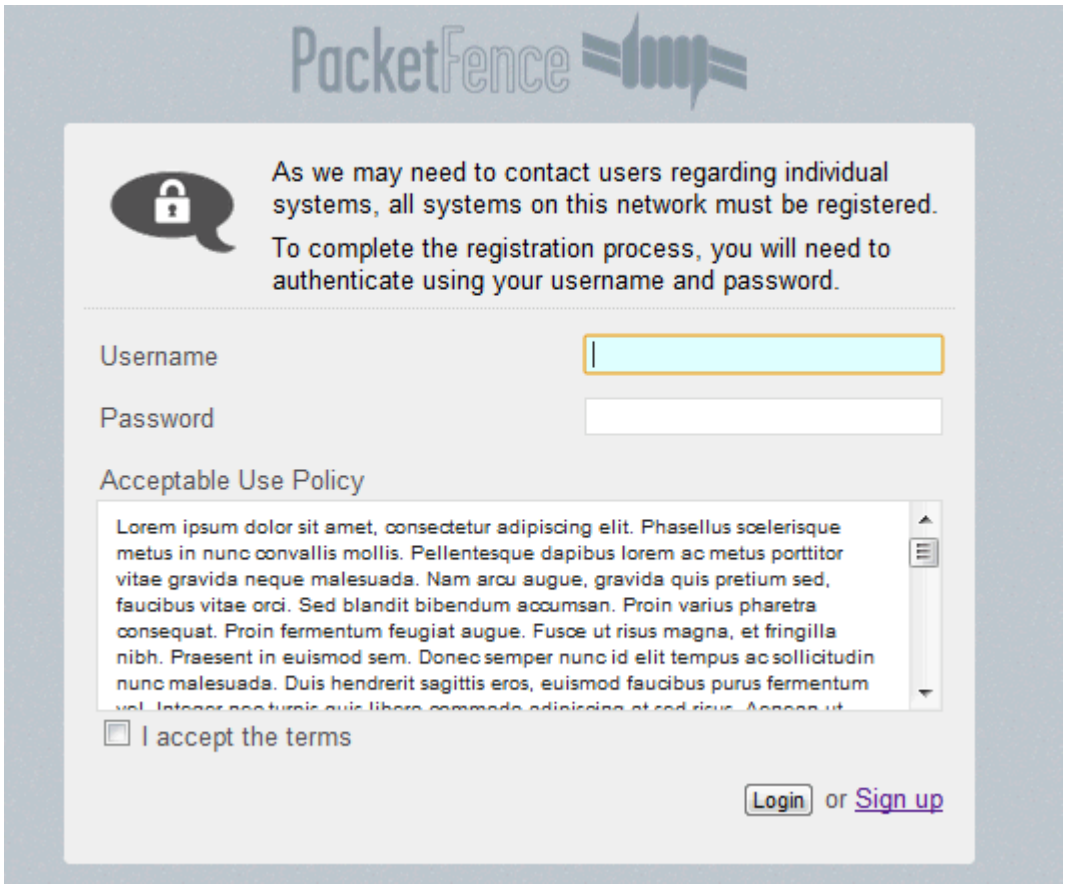
Moreover, PacketFence also has the option for guests to request their access in advance. Confirmation by email and by a sponsor are the two pre-registration techniques supported at this point.

The admin GUI allow PacketFence administrators or guests managers to create single accounts, multiple accounts using a prefix (ie.: guest1, guest2, guest3...) or import data from a CSV to create accounts. Access duration and expected arrival date are also customizable.

Usage

Guest self-registration

Self-registration is enabled by default. It is part of the captive portal profile and can be accessed on the registration page by clicking the **Sign up** link.



Managed guests

Part of the web administration interface, the guests management interface is enabled by default. It is accessible through the **Users → Create** menu.

Guest pre-registration

Pre-registration is disabled by default. Once enabled, PacketFence's firewall and Apache ACLs allow access to the `/signup` page on the portal even from a remote location. All that should be required from the administrators is to open up their perimeter firewall to allow access to PacketFence's management interface IP on port 443 and make sure a domain name to reach said IP is configured (and that the SSL cert matches it). Then you can promote the pre-registration link from your extranet web site: https://<hostname>/signup.

Caution

Pre-registration increases the attack surface of the PacketFence system since a subset of it's functionnality is exposed on the Internet. Make sure you understand the risks, apply the critical operating system updates and apply PacketFence's security fixes.

Note

A *portal* interface type is required to use this feature. A *portal* interface type can be added to any network interface using the web admin GUI.

Configuration

Guest self-registration

It is possible to modify the default values of the guest self-registration feature by editing `/usr/local/pf/conf/pf.conf`.

Default values are located in `/usr/local/pf/conf/pf.conf.defaults` and documentation for every settings is available in `/usr/local/pf/conf/documentation.conf`.

```
[guests_self_registration]
guest_pid=email
preregistration=disabled
sponsorship_cc=
```

These parameters can also be configured from the **Configuration → Self Registration** section of the Web admin interface.

Available registration modes are defined on a per-portal-profile basis. These are configurable from **Configuration → Portal Profiles**. To disable the self-registration feature, simply remove all self-registration sources from the portal profile definition. Notice however that if your default portal profile has no source, it will use all authentication sources.

Caution

A valid MTA configured in PacketFence is needed to correctly relay emails related to the guest module. If *localhost* is used as smtpserver, make sure that a MTA is installed and configured on the server.

Note

A *portal* interface type is required to use this feature. A *portal* interface type can be added to any network interface using the web admin GUI.

Self-registered guests are added under the users tab of the PacketFence Web administration interface.

Managed guests

It is possible to modify the default values of the guests created from the Web admin interface by editing `/usr/local/pf/conf/pf.conf`.

Default values are located in `/usr/local/pf/conf/pf.conf.defaults` and documentation for every settings is available in `/usr/local/pf/conf/documentations.conf`.

```
[guests_admin_registration]
access_duration_choices=1h,3h,12h,1D,2D,3D,5D
default_access_duration=12h
```

The format of the duration is as follow:

```
<DURATION><DATETIME_UNIT>[<PERIOD_BASE><OPERATOR><DURATION><DATE_UNIT>]
```

Let's explain the meaning of each parameter:

- `DURATION`: a number corresponding to the period duration.
- `DATETIME_UNIT`: a character corresponding to the units of the date or time duration; either s (seconds), m (minutes), h (hours), D (days), W (weeks), M (months), or Y (years).
- `PERIOD_BASE`: either F (fixed) or R (relative). A relative period is computed from the beginning of the period unit. Weeks start on Monday.
- `OPERATOR`: either + or -. The duration following the operator is added or subtracted from the base duration.
- `DATE_UNIT`: a character corresponding to the units of the extended duration. Limited to date units (D (days), W (weeks), M (months), or Y (years)).

These parameters can also be configured from the **Configuration → Admin Registration** section of the Web admin interface.

From the Users page of the PacketFence Web admin interface, it is possible to set the access duration of users, change their password and more.

Guest pre-registration

To minimally configure guest pre-registration, you must make sure that the following statement is set under `[guests_self_registration]` in `/usr/local/pf/conf/pf.conf`:

```
[guests_self_registration]
preregistration=enabled
```

This parameter can also be configured from the **Configuration → Self Registration** section.

Finally, it is advised that you read the whole guest self-registration section since pre-registration is simply a twist of the self-registration process.

Caution

A valid MTA configured in PacketFence is needed to correctly relay emails related to the guest module. If *localhost* is used as smtpserver, make sure that a MTA is installed and configured on the server.

ActiveDirectory Integration

Deleted Account

Create the script `unreg_node_deleted_account.ps1` on the Windows Server with the following content. Make sure to change `@IP_PACKETFENCE` to the IP address of your PacketFence server. You'll also need to change the username and password as they must match the credentials defined in the Web admin interface under **Configuration → Web Services**.

```
##############################################################################################
#Powershell script to unregister deleted Active Directory account based on the
 UserName.#
##############################################################################################

Get-EventLog -LogName Security -InstanceId 4726 |
   Select ReplacementStrings,"Account name"|
   % {
    $url = "https://@IP_PACKETFENCE:9090/"
    $username = "admin" # Username for the webservices
    $password = "admin" # Password for the webservices
    [System.Net.ServicePointManager]::ServerCertificateValidationCallback =
 {$true}
    $command = '{"jsonrpc": "2.0", "method": "unreg_node_for_pid", "params":
 [{"pid": "'+$_.ReplacementStrings[0]+'"}]}'

    $bytes = [System.Text.Encoding]::ASCII.GetBytes($command)
    $web = [System.Net.WebRequest]::Create($url)
    $web.Method = "POST"
    $web.ContentLength = $bytes.Length
    $web.ContentType = "application/json-rpc"
    $web.Credentials = new-object System.Net.NetworkCredential($username,
 $password)
    $stream = $web.GetRequestStream()
    $stream.Write($bytes,0,$bytes.Length)
    $stream.close()

    $reader = New-Object System.IO.Streamreader -ArgumentList
 $web.GetResponse().GetResponseStream()
    $reader.ReadToEnd()
    $reader.Close()
   }
```

Create the scheduled task based on an event ID

Start → Run → Taskschd.msc

Task Scheduler → Task Scheduler Library → Event Viewer Task → Create Task

General

```
Name: PacketFence-Unreg_node-for-deleted-account
Check: Run whether user is logged on or not
Check: Run with highest privileges
```

Triggers → New

```
Begin on the task: On an event
Log: Security
Source: Microsoft Windows security auditing.
Event ID: 4726
```

Actions → New

```
Action: Start a program
Program/script: powershell.exe
Add arguments (optional): C:\scripts\unreg_node_deleted_account.ps1
```

Settings:

```
At the bottom, select in the list "Run a new instance in parallel" in order to
 unregister multiple nodes at the same time.
```

Validate with Ok and give the account who will run this task. (Usually *DOMAIN\Administrator*)

Disabled Account

Create the script `unreg_node_disabled_account.ps1` on the Windows Server with the following content. Make sure to change `@IP_PACKETFENCE` to the IP address of your PacketFence server. You'll also need to change the username and password as they must match the credentials defined in the Web admin interface under **Configuration → Web Services**.

```
#############################################################################################
#Powershell script to unregister disabled Active Directory account based on the
 UserName.#
#############################################################################################

Get-EventLog -LogName Security -InstanceId 4725 |
   Select ReplacementStrings,"Account name"|
   % {
    $url = "https://@IP_PACKETFENCE:9090/"
    $username = "admin" # Username for the webservices
    $password = "admin" # Password for the webservices
    [System.Net.ServicePointManager]::ServerCertificateValidationCallback =
 {$true}
    $command = '{"jsonrpc": "2.0", "method": "unreg_node_for_pid", "params":
 [{"pid": "'+$_.ReplacementStrings[0]+'"}]}'

    $bytes = [System.Text.Encoding]::ASCII.GetBytes($command)
    $web = [System.Net.WebRequest]::Create($url)
    $web.Method = "POST"
    $web.ContentLength = $bytes.Length
    $web.ContentType = "application/json-rpc"
    $web.Credentials = new-object System.Net.NetworkCredential($username,
 $password)
    $stream = $web.GetRequestStream()
    $stream.Write($bytes,0,$bytes.Length)
    $stream.close()

    $reader = New-Object System.IO.Streamreader -ArgumentList
 $web.GetResponse().GetResponseStream()
    $reader.ReadToEnd()
    $reader.Close()

   }
```

Create the scheduled task based on an event ID

Start → Run → Taskschd.msc

Task Scheduler → Task Scheduler Library → Event Viewer Task → Create Task

General

```
Name: PacketFence-Unreg_node-for-disabled-account
Check: Run whether user is logged on or not
Check: Run with highest privileges
```

Triggers → New

```
Begin on the task: On an event
Log: Security
Source: Microsoft Windows security auditing.
Event ID: 4725
```

Actions → New

```
Action: Start a program
Program/script: powershell.exe
Add arguments (optional): C:\scripts\unreg_node_disabled_account.ps1
```

Settings:

```
At the bottom, select in the list "Run a new instance in parallel"
```

Validate with Ok and give the account who will run this task. (Usually *DOMAIN\Administrator*)

Locked Account

Create the script `unreg_node_locked_account.ps1` on the Windows Server with the following content. Make sure to change `@IP_PACKETFENCE` to the IP address of your PacketFence server. You'll also need to change the username and password as they must match the credentials defined in the Web admin interface under **Configuration → Web Services**.

```
##############################################################################################
#Powershell script to unregister locked Active Directory account based on the
 UserName.#
##############################################################################################

Get-EventLog -LogName Security -InstanceId 4740 |
   Select ReplacementStrings,"Account name"|
   % {
    $url = "https://@IP_PACKETFENCE:9090/"
    $username = "admin" # Username for the webservices
    $password = "admin" # Password for the webservices
    [System.Net.ServicePointManager]::ServerCertificateValidationCallback =
 {$true}
   $command = '{"jsonrpc": "2.0", "method": "unreg_node_for_pid", "params":
 [{"pid": "'+$_.ReplacementStrings[0]+'"}]}'

    $bytes = [System.Text.Encoding]::ASCII.GetBytes($command)
    $web = [System.Net.WebRequest]::Create($url)
    $web.Method = "POST"
    $web.ContentLength = $bytes.Length
    $web.ContentType = "application/json-rpc"
    $web.Credentials = new-object System.Net.NetworkCredential($username,
 $password)
    $stream = $web.GetRequestStream()
    $stream.Write($bytes,0,$bytes.Length)
    $stream.close()

    $reader = New-Object System.IO.Streamreader -ArgumentList
 $web.GetResponse().GetResponseStream()
    $reader.ReadToEnd()
    $reader.Close()

  }
```

Create the scheduled task based on an event ID

Start → Run → Taskschd.msc

Task Scheduler → Task Scheduler Library → Event Viewer Task → Create Task

General

```
Name: PacketFence-Unreg_node-for-locked-account
Check: Run whether user is logged on or not
Check: Run with highest privileges
```

Triggers → New

```
Begin on the task: On an event
Log: Security
Source: Microsoft Windows security auditing.
Event ID: 4740
```

Actions → New

```
Action: Start a program
Program/script: powershell.exe
Add arguments (optional): C:\scripts\unreg_node_locked_account.ps1
```

Settings:

```
At the bottom, select in the list "Run a new instance in parallel"
```

Validate with Ok and give the account who will run this task. (Usually *DOMAIN\Administrator*)

DHCP remote sensor

The DHCP remote sensor consists of a lightweight binary that is installed on your production DHCP server in order to replicate the DHCP traffic 1 to 1 to the PacketFence server. This solution is more reliable than the DHCP relaying since PacketFence receives a copy of all your DHCP traffic and not only the broadcasted DHCP traffic. Supported DHCP servers are Microsoft DHCP server and CentOS 6 and 7.

These sensors work by capturing the packets at the lowest level possible on your DHCP server and forward them to the PacketFence management interface

Microsoft DHCP sensor

You will first need to download and install *WinPcap* available from http://www.winpcap.org/install/

You will also need to download and install *Microsoft Visual C++ 2010 Redistributable* available from http://www.microsoft.com/download/details.aspx?id=5555

Note

You absolutely need to install the 32-bit version of *Microsoft Visual C++ 2010 Redistributable* even if you are using a 64-bit operating system.

Then get the remote sensor from Inverse's download website http://inverse.ca/downloads/PacketFence/udp-reflector/udp_reflector.exe

Create the directory `C:\udp-reflector` and move the downloaded file inside.

Now we will create a task so the reflector starts on boot.

Go in **Startup/Task Scheduler** and create a new Task.

In General:

- Name it **dhcp-reflector**
- Select *Run whether the user is logged on or not*
- Check *Run with highest privileges*

In Triggers:

- Create a new trigger
- In *Begin the task:*, select *At startup*
- Leave the other options as default and save.

In Actions:

- Create a new action
- Set the action to *Start a program*
- In *Program/script* set it to `C:\udp-reflector\udp_reflector.exe`,
- In *Add arguments* put `-s pcap0:67 -d 192.168.1.5:767 -b 25000`
  - where `pcap0` is the interface where your DHCP is listening (use `C:\udp-reflector\udp_reflector.exe -l` to list interfaces)
  - where `192.168.1.5` is the management IP of PacketFence

In Settings:

- Uncheck *Stop the task if it runs longer than:*

Then press *OK* and save the task.

Now start the task manually by right clicking on it and choosing *Run*.

The DHCP traffic should now be reflected on your PacketFence server.

Linux based sensor

First download the RPM on your DHCP server.

CentOS 6 and 7 servers

For CentOS 6:

```
# for x86_64
# wget http://inverse.ca/downloads/PacketFence/CentOS6/extra/x86_64/RPMS/udp-
reflector-1.0-6.1.x86_64.rpm
```

For CentOS 7:

```
# for x86_64
# wget http://inverse.ca/downloads/PacketFence/CentOS7/extra/x86_64/RPMS/udp-
reflector-1.0-6.1.x86_64.rpm
```

Now install the sensor:

```
# rpm -i udp-reflector-*.rpm
```

Compiling the sensor from source on a Linux system

First make sure you have the following packages installed:

- libpcap
- libpcap-devel
- gcc-c++

Get the source code of the sensor:

```
# mkdir -p ~/udp-reflector && cd ~/udp-reflector
# wget http://inverse.ca/downloads/PacketFence/udp-reflector/udp_reflector.cpp
# g++ udp_reflector.cpp -o /usr/local/bin/udp_reflector -lpcap
```

Configuring the sensor

Place the following line in `/etc/rc.local`

- where `pcap0` is the pcap interface where your DHCP server listens on. (List them using `udp_reflector -l`)
- where `192.168.1.5` is the management IP of your PacketFence server

```
/usr/local/bin/udp_reflector -s pcap0:67 -d 192.168.1.5:767 -b 25000 &
```

Start the sensor:

```
# /usr/local/bin/udp_reflector -s pcap0:67 -d 192.168.1.5:767 -b 25000 &
```

The DHCP traffic should now be reflected on your PacketFence server.

Operating System Best Practices

IPTables

IPTables is now entirely managed by PacketFence. However, if you need to perform some custom rules, you can modify `conf/iptables.conf` to your own needs. However, the default template should work for most users.

Log Rotations

PacketFence can generate a lot of log entries in huge production environments. This is why we recommend to use `logrotate` to periodically rotate your logs. A working logrotate script is provided with the PacketFence package. This script is located in `/usr/local/pf/addons`, and it's configured to do a weekly log rotation and keeping old logs with compression. It has been added during PacketFence initial installation.

Performance optimization

SNMP Traps Limit

PacketFence mainly rely on SNMP traps to communicate with equipment. Due to the fact that traps coming in from approved (configured) devices are all processed by the daemon, it is possible for someone who want to generate a certain load on the PacketFence server to force the generation of non-legitimate SNMP traps or a switch can randomly generate a high quantity of traps sent to PacketFence for an unknown reason.

Because of that, it is possible to limit the number of SNMP traps coming in from a single switch port and take action if that limit is reached. For example, if over 100 traps are received by PacketFence from the same switch port in a minute, the switch port will be shut and a notification email will be sent.

Here's the default config for the SNMP traps limit feature. As you can see, by default, PacketFence will log the abnormal activity after 100 traps from the same switch port in a minute. These configurations are in the `conf/pf.conf` file:

```
[vlan]
trap_limit = enabled
trap_limit_threshold = 100
trap_limit_action =
```

Alternatively, you can configure these parameters from the PacketFence Web administrative GUI, in the **Configuration → SNMP** section.

MySQL optimizations

Tuning MySQL

If you're PacketFence system is acting very slow, this could be due to your MySQL configuration. You should do the following to tune performance:

Check the system load

```
# uptime
11:36:37 up 235 days,  1:21,  1 user, load average: 1.25, 1.05, 0.79
```

Check iostat and CPU

```
# iostat 5
avg-cpu:  %user   %nice    %sys %iowait   %idle
           0.60    0.00    3.20   20.20   76.00
Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
cciss/c0d0       32.40         0.00      1560.00          0       7800
avg-cpu:  %user   %nice    %sys %iowait   %idle
           0.60    0.00    2.20    9.20   88.00
Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
cciss/c0d0        7.80         0.00        73.60          0        368
avg-cpu:  %user   %nice    %sys %iowait   %idle
           0.60    0.00    1.80   23.80   73.80
Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
cciss/c0d0       31.40         0.00      1427.20          0       7136
avg-cpu:  %user   %nice    %sys %iowait   %idle
           0.60    0.00    2.40   18.16   78.84
Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
cciss/c0d0       27.94         0.00      1173.65          0       5880
```

As you can see, the load is 1.25 and IOWait is peaking at 20% - this is not good. If your IO wait is low but your MySQL is taking +%50 CPU this is also not good. Check your MySQL install for the following variables:

```
mysql> show variables;
| innodb_additional_mem_pool_size | 1048576      |
| innodb_autoextend_increment     | 8            |
| innodb_buffer_pool_awe_mem_mb   | 0            |
| innodb_buffer_pool_size         | 8388608      |
```

PacketFence relies heavily on InnoDB, so you should increase the `buffer_pool` size from the default values.

Shutdown PacketFence and MySQL

```
# /etc/init.d/packetfence stop
Shutting down PacketFence...
[...]
# /etc/init.d/mysql stop
Stopping MySQL:                                            [  OK  ]
```

Edit `/etc/my.cnf` (or your local `my.cnf`):

```
[mysqld]
# Set buffer pool size to 50-80% of your computer's memory
innodb_buffer_pool_size=800M
innodb_additional_mem_pool_size=20M
innodb_flush_log_at_trx_commit=2
innodb_file_per_table
# allow more connections
max_connections=700
# set cache size
key_buffer_size=900M
table_cache=300
query_cache_size=256M
# enable slow query log
log_slow_queries = ON
```

Start up MySQL and PacketFence

```
# /etc/init.d/mysqld start
Starting MySQL:                                            [  OK  ]
# /etc/init.d/packetfence start
Starting PacketFence...
[...]
```

Wait 10 minutes for PacketFence to initial the network map and re-check iostat and CPU

```
# uptime
12:01:58 up 235 days,  1:46,  1 user, load average: 0.15, 0.39, 0.52
# iostat 5
Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
cciss/c0d0        8.00         0.00        75.20          0        376

avg-cpu:  %user   %nice    %sys %iowait   %idle
           0.60    0.00    2.99   13.37   83.03

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
cciss/c0d0       14.97         0.00       432.73          0       2168
avg-cpu:  %user   %nice    %sys %iowait   %idle
           0.20    0.00    2.60    6.60   90.60

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
cciss/c0d0        4.80         0.00        48.00          0        240
```

MySQL optimization tool

We recommend that you run the MySQL Tuner on your database setup after a couple of weeks to help you identify MySQL configuration improvement. The tool is bundled with PacketFence and can be run from the command-line:

```
# /usr/local/bin/pftest mysql
```

Keeping tables small

Over time, some of the tables will grow large and this will drag down performance (this is especially true on a wireless setup).

One such table is the `locationlog` table. We recommend that closed entries in this table be moved to the archive table `locationlog_archive` after some time. A closed record is one where the `end_time` field is set to a date (strickly speaking it is when `end_time` is not null and not equals to 0).

We provide a script called `database-backup-and-maintenance.sh` located in `addons/` that performs this cleanup in addition to optimize tables on Sunday and daily backups.

Avoid "Too many connections" problems

In a wireless context, there tends to be a lot of connections made to the database by our `freeradius` module. The default MySQL value tend to be low (100) so we encourage you to increase that value to at least 300. See http://dev.mysql.com/doc/refman/5.0/en/too-many-connections.html for details.

Avoid "Host <hostname> is blocked" problems

In a wireless context, there tend to be a lot of connections made to the database by our freeradius module. When the server is loaded, these connection attempts can timeout. If a connection times out during connection, MySQL will consider this a connection error and after 10 of these (by default) he will lock the host out with a:

```
Host 'host_name' is blocked because of many connection errors. Unblock with
 'mysqladmin flush-hosts'
```

This will grind PacketFence to a halt so you want to avoid that at all cost. One way to do so is to increase the number of maximum connections (see above), to periodically flush hosts or to allow more connection errors. See http://dev.mysql.com/doc/refman/5.0/en/blocked-host.html for details.

Captive Portal Optimizations

Avoid captive portal overload due to non-browser HTTP requests

By default we allow every query to be redirected and reach PacketFence for the captive portal operation. In a lot of cases, this means that a lot of non-user initiated queries reach PacketFence and waste its resources for nothing since they are not from browsers. (iTunes, Windows update, MSN Messenger, Google Desktop, ...).

Since version 4.3 of PacketFence, you can define HTTP filters for Apache from the configuration of PacketFence.

Some rules have been enabled by default, like one to reject requests with no defined user agent. All rules, including some examples, are defined in the configuration file apache_filters.conf.

Filters are defined with at least two blocks. First are the tests. For example:

```
[get_ua_is_dalvik]
filter = user_agent
method = GET
operator = match
value = Dalvik
```

```
[get_uri_not_generate204]
filter = uri
method = GET
operator = match_not
value = /generate_204
```

The last block defines the relationship between the tests and the desired action. For example:

```
[block_dalvik:get_ua_is_dalvik&get_uri_not_generate204]
action = 501
redirect_url =
```

This filter will return an error code (501) if the user agent is Dalvik and the URI doesn't contain _/generate_204.

Additional Information

For more information, please consult the mailing archives or post your questions to it. For details, see:

- packetfence-announce@lists.sourceforge.net: Public announcements (new releases, security warnings etc.) regarding PacketFence
- packetfence-devel@lists.sourceforge.net: Discussion of PacketFence development
- packetfence-users@lists.sourceforge.net: User and usage discussions

Commercial Support and Contact Information

For any questions or comments, do not hesitate to contact us by writing an email to: support@inverse.ca.

Inverse (http://inverse.ca) offers professional services around PacketFence to help organizations deploy the solution, customize, migrate versions or from another system, performance tuning or aligning with best practices.

Hourly rates or support packages are offered to best suit your needs.

Please visit http://inverse.ca/ for details.

GNU Free Documentation License

Please refer to http://www.gnu.org/licenses/fdl-1.2.txt for the full license.

Appendix A. Administration Tools

pfcmd

`pfcmd` is the command line interface to most PacketFence functionalities.

When executed without any arguments `pfcmd` returns a basic help message with all main options:

```
Usage:
    pfcmd <command> [options]

     Commands
      cache                        | manage the cache subsystem
      checkup                      | perform a sanity checkup and report any
 problems
      class                        | view violation classes
      configfiles                  | push or pull configfiles into/from database
      configreload                 | reload the configution
      floatingnetworkdeviceconfig | query/modify floating network devices
 configuration parameters
      help                         | show help for pfcmd commands
      ifoctetshistorymac           | accounting history
      ifoctetshistoryswitch        | accounting history
      ifoctetshistoryuser          | accounting history
      import                       | bulk import of information into the database
      ipmachistory                 | IP/MAC history
      locationhistorymac           | Switch/Port history
      locationhistoryswitch        | Switch/Port history
      networkconfig                | query/modify network configuration parameters
      node                         | manipulate node entries
      pfconfig                     | interact with pfconfig
      portalprofileconfig          | query/modify portal profile configuration
 parameters
      reload                       | rebuild fingerprint or violations tables
 without restart
      service                      | start/stop/restart and get PF daemon status
      schedule                     | Nessus scan scheduling
      switchconfig                 | query/modify switches.conf configuration
 parameters
      version                      | output version information
      violationconfig              | query/modify violations.conf configuration
 parameters

    Please view "pfcmd help <command>" for details on each option
```

The node view option shows all information contained in the node database table for a specified MAC address

```
# /usr/local/pf/bin/pfcmd node view 52:54:00:12:35:02
mac|pid|detect_date|regdate|unregdate|lastskip|status|user_agent|computername|
notes|last_arp|last_dhcp|switch|port|vlan|dhcp_fingerprint
52:54:00:12:35:02|1|2008-10-23 17:32:16||||unreg||||2008-10-23 21:12:21|||||
```

pfcmd_vlan

`pfcmd_vlan` is the command line interface to most VLAN isolation related functionality.

Again, when executed without any arguments, a help screen is shown.

```
Usage:
    pfcmd_vlan command [options]

     Command:
       -deauthenticate      de-authenticate a dot11 client
       -deauthenticateDot1x de-authenticate a dot1x client (pass ifIndex for
 wired 802.1x and mac for wireless 802.1x)
       -getAlias            show the description of the specified switch port
       -getAllMACs          show all MACS on all switch ports
       -getHubs             show switch ports with several MACs
       -getIfOperStatus     show the operational status of the specified switch
 port
       -getIfType           show the ifType on the specified switch port
       -getLocation         show at which switch port the MAC is found
       -getSwitchLocation   show SNMP location of specified switch
       -getMAC              show all MACs on the specified switch port
       -getType             show switch type
       -getUpLinks          show the upLinks of the specified switch
       -getVersion          show switch OS version
       -getVlan             show the VLAN on the specified switch port
       -getVlanType         show the VLAN type on the specified port
       -help                brief help message
       -isolate             set the switch port to the isolation VLAN
       -man                 full documentation
       -reAssignVlan        re-assign a switch port VLAN
       -reevaluateAccess    reevaluate the current VLAN or firewall rules of a
 given MAC
       -runSwitchMethod     run a particular method call on a given switch (FOR
 ADVANCED PURPOSES)
       -setAlias            set the description of the specified switch port
       -setDefaultVlan      set the switch port to the default VLAN
       -setIfAdminStatus    set the admin status of the specified switch port
       -setVlan             set VLAN on the specified switch port
       -setVlanAllPort      set VLAN on all non-UpLink ports of the specified
 switch

     Options:
       -alias               switch port description
       -ifAdminStatus       ifAdminStatus
       -ifIndex             switch port ifIndex
       -mac                 MAC address
       -showPF              show additional information available in PF
       -switch              switch description
       -verbose             log verbosity level
                             0 : fatal messages
                             1 : warn messages
                             2 : info messages
                             3 : debug
                             4 : trace
       -vlan                VLAN id
       -vlanName            VLAN name (as in switches.conf)
```