PacketFence – version 1.7.5

Developer's Guide

Version 1.7.5 – December 2008

Contents

Chapter 1

About this Guide

This guide will help you modifying PacketFence to your particular needs. It also contains information on how to add support for new switches.

The instructions are based on version 1.7.5 of PacketFence.

The latest version of this guide is available online at http://inverse.ca/uploads/docs/PacketFence_Developers_Guide.pdf.

Chapter 2

System Requirements

Assumptions

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

- ❑ Database server (MySQL)
- ❑ Web server (Apache)

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

- ❑ DHCP server (ISC DHCP)
- ❑ DNS server (BIND)
- ❑ NIDS (Snort)

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 and configuration of these requirements before continuing with this guide.

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

MySQL server	MySQL 4.1 or 5.1
Web server	Apache 2
ISC DHCP	DHCP 3
ISC BIND	BIND 9
Snort	Snort 2.8

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

Chapter 2

Minimum Hardware Requirements

The following table provides hardware recommendations for the server and desktops :

Server	■ Intel or AMD CPU 3 GHz ■ 2048 MB of RAM ■ 20 GB of disk space (RAID 1) ■ 3 Network cards

Chapter 2

Operating System Requirements

Currently PacketFence 1.7.5 supports the following 32-bit operating systems:

- Red Hat Enterprise Linux 5.x Server
- Community ENTerprise Operating System (CentOS) 5.x

Make sure the required components are started automatically (except Snort that is controlled by PacketFence) at boot time and that they are running before proceeding with the PacketFence configuration. Also make sure that you can install additional packages from your standard distribution. For example, if you are using Red Hat Enterprise Linux 5, you have to be subscribed to the Red Hat Network before continuing with the PacketFence software installation.

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

Chapter 3

Customizing PacketFence

Registration Pages

Translations

The language of the user registration pages is selected through the `general.locale` configuration parameter.

The internationalization process uses gettext. If you are new to gettext, please consult http://www.gnu.org/software/gettext/manual/gettext.html#Overview for a quick introduction.

Currently, PacketFence has language files for Dutch, English, French and Spanish. The PO files are stored in `/usr/local/pf/conf/locale`.

If you want to add support for a new language, please follow these steps:

- create a new language subdirectory in `/usr/local/pf/conf/locale`
- change into your newly created directory
- create a new subdirectory `LC_MESSAGES`
- change into your newly created directory
- copy the file `/usr/local/pf/conf/locale/en/LC_MESSAGES/packetfence.po` into your directory
- translate the message strings in `packetfence.po`
- create the MO file by executing

```
/usr/bin/msgfmt packetfence.po
```

Please consider submitting your new translation to the PacketFence project by contacting us at packetfence-devel@lists.sourceforge.net.

Content

Since version 1.7 of PacketFence, the registration pages use Template Toolkit (http://template-toolkit.org/).

All the template files are located in `/usr/local/pf/html/user/content/templates`. The first template you might want to customize is surely the registration template `/usr/local/pf/html/user/content/templates/register.html`. You can freely edit the

HTML code in this file (and all other temlate files). However, if you want to customize the pages beyond the HTML template (for example by adding new variables to it), you'll need to look into the `/usr/local/pf/lib/pf/web.pm` Perl module. This module contains one function per termplate file. So if you want to modify the registration page, you'll have to modify the `generate_registration_page` function.

Remediation Pages

The remediation page shown to the user during isolation is specified through the `url` parameter of the given violation in `/usr/local/pf/conf/violations.conf`. In its default configuration, PacketFence uses PHP templates located in the directory `/usr/local/pf/html/user/content/violations/`.

Adding custom fields to the database

You can, if needed, add additional fields to the PacketFence database. Keep in mind though that this might lead to more work when you upgrade to the next PacketFence version. Depending on the degree of integration of these fields with PacketFence, you'll have to execute one or more of the following steps

Adding a field to the database only

In this case, the field is part of one of the main PacketFence tables, but PacketFence is unaware of it. PacketFence won't consult the field and won't be able to modify it. A possible usage scenario would be a 3rd party application which maintains this field.

Since PacketFence doesn't have to know about the field, all you have to do is execute your SQL ALTER TABLE query and you are done.

Adding a field and giving PacketFence read-only access

In this case, PacketFence can show the contents of the table using both pfcmd and the Web Admin GUI, but won't be able to modify the contents of the field.

Start by modifying the database table using an SQL ALTER TABLE query.

Then, modify the Perl module having the same name as the table you have added the field to, i.e. If you added the field to the node table, then edit `/usr/local/pf/lib/pf/node.pm`. You'll have to modify the SQL SELECT queries at the beginning of the file to include your new field and, possibly the functions using these queries. If your new field should be used in reports, the dashboard or graphs, you'll also have to modify the queries in `/usr/local/pf/lib/pf/pfcmd/graph.pm`, `/usr/local/pf/lib/pf/pfcmd/report.pm` and `/usr/local/pf/lib/pf/pfcmd/dashboard.pm`.

Last, but not least, you'll have to modify the file `/usr/local/pf/conf/ui.conf`. In this file, you can also give a nice looking name to your field for showing up in the Web Admin GUI.

Adding a field and giving PacketFence read-write access

Start by creating the read-only field as described above.

Then, modify the SQL UPDATE and INSERT queries in the database tables Perl module, as well as the associated functions.

The last step is to make PacketFence's grammar aware of the new field. Modify /usr/local/pf/lib/pf/pfcmd/pfcmd.pm and then re-generate the precompiled grammar (which is used by the pfcmd CLI) with

```
cd /usr/local/pf

/usr/bin/perl -w -e 'use strict; use warnings; use diagnostics; use
Parse::RecDescent; use lib "/usr/local/pf/lib"; use
pf::pfcmd::pfcmd; Parse::RecDescent->Precompile($grammar,
"pfcmd_pregrammar");'

mv pfcmd_pregrammar.pm
/usr/local/pf/lib/pf/pfcmd/pfcmd_pregrammar.pm
```

VLAN assignment

The pfsetvlan daemon assigns by default a MAC to the VLAN which is saved in the VLAN field in its database entry. This VLAN field is, again by default, filled during registration with the normalVlan configuration setting, defined in /usr/local/pf/conf/switches.conf.

So, there are two different ways to change the VLAN a given node ends up in: by modifying the content which is saved in the VLAN field during registration and by modifying how pfsetvlan uses this information.

Modifying the VLAN assignment during registration

You can change the default behavior by modifying the following lines in /usr/local/pf/cgi-bin/register.cgi

```
#determine default VLAN if VLAN isolation is enabled

#and the vlan has not been set yet

if (isenabled($Config{'network'}{'vlan'})) {

  if (! defined($info{'vlan'})) {

    my %ConfigVlan;

    tie %ConfigVlan, 'Config::IniFiles', (-file =>
'/usr/local/pf/conf/switches.conf');

    $info{'vlan'}=$ConfigVlan{'default'}{'normalVlan'};

  }

}
```

Modifying how pfsetvlan calculates the VLAN for a node

pfsetvlan uses the `custom_getCorrectVlan` function defined in `/usr/local/pf/conf/pfsetvlan.pm` to determine a nodes VLAN. Here's the default function:

```
sub custom_getCorrectVlan {

   my ($switch_ip, $ifIndex, $mac, $status, $vlan, $pid) = @_;

   #$switch_ip is the ip of the switch the computer is connected to

   #$ifIndex is the ifIndex of the port the computer is connected to

   #$mac is the MAC connected

   #$status is the node's status in the database

   #$vlan is the vlan set for this node in the database

   #$pid is the owner of this node in the database

   my $logger = Log::Log4perl->get_logger();

   Log::Log4perl::MDC->put('tid', threads->self->tid());

   return $vlan;

}
```

As you can see, the function receives several parameters (such as the switch and the owner of the computer) which allow you to return the VLAN in a way that matches exactly your needs !

SNMP

Introduction

Good places to start reading about SNMP are http://en.wikipedia.org/wiki/SNMP and http://www.net-snmp.org/.

When working with SNMP, you'll sooner or later (in fact more sooner than later) be confronted with having to translate between OIDs and variable names. When the OIDs are part of the Cisco MIBs, you can use the following tool to do the translation: http://tools.cisco.com/Support/SNMP/public.jsp. Otherwise, you'll have to use snmptranslate for exemple and setup you own collection of MIBs, provided (hopefully) by the manufacturer of your network equipment. You might also want to get the following MIBs:

```
wget ftp://ftp.cisco.com/pub/mibs/v1/v1.tar.gz

wget ftp://ftp.cisco.com/pub/mibs/v2/v2.tar.gz
```

Obtaining switch and port information

Below are some example of how to obtain simple switch and port information using SNMP. We'll assume that your switch understands SNMP v2, has the read communnity `public` defined and is reachable at 192.168.1.10.

Switch Type

```
snmpwalk -v 2c -c public 192.168.1.10 sysDescr
```

Switchport indexes and descriptions

```
snmpwalk -v 2c -c public 192.168.1.10 ifDescr
```

Switchport types

```
snmpwalk -v 2c -c public 192.168.1.10 ifType
```

Switchport status

```
snmpwalk -v 2c -c public 192.168.1.10 ifAdminStatus

snmpwalk -v 2c -c public 192.168.1.10 ifOperStatus
```

Obtaining VLAN information on Cisco switches

Access VLAN on a switchport

```
snmpwalk -c public -m CISCO-VLAN-MEMBERSHIP-MIB -M /usr/local/share/
snmp/mibs:/usr/share/snmp/mibs -v 2c 192.168.1.10 vmVlan
```

Chapter 5

Supporting a new switch

PacketFence is designed to ease the addition of support for new switches. All supported switches are represented through Perl objects with an extensive use of inheritance. Adding support for a new product comes down to extending the pf::SNMP class (in `/usr/local/pf/lib/pf`).

The starting point to adding support for a new switch should be your switch's documentation ! First of all, you'll have to figure out the exact capabilities of the switch and how these capabilities will fit into PacketFence. Will you be able to use only link change traps ? Does your switch allow you to use MAC notification traps ? Port Security ?

Link change capabilities

You need to define a new class which inherits from pf::SNMP and defines at least the following functions:

- getMacAddrVlan
- getVersion
- getVlan
- getVlans
- isDefinedVlan
- parseTrap
- _getMacAtIfIndex
- _setVlan

The parseTrap function will need to return a hash with keys `trapType` and `trapIfIndex`. The associated values must be `up` or `down` for `trapType` and the traps ifIndex for `trapIfIndex`.

MAC notification capabilities

In addition to the functions mentioned for link change, you need to define the following function:

- isLearntTrapsEnabled

Also, your parseTrap function will need to be able to return a third value for the trapType key: `mac`. In this case, the hash also needs to contain `trapOperation`, `trapVlan` and `trapMac` keys.

Chapter 5

Port security capabilities

In addition to the functions mentioned for link change, you need to define de following functions:

- isPortSecurityEnabled
- authorizeMAC

In this case, the parseTrap function needs to be able to return `secureMacAddrViolation` for the `trapType` key.

Chapter 6

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

Chapter 7

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.

Chapter 8

GNU Free Documentation License

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