Symantec VIP Quick Start Guide

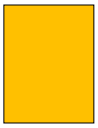
UNIX and Linux authentication

Version 1.0

Author

Maren Peasley

Symantec



Special Thanks to:

Jeff Cavanaugh for help with all things *NIX.

Travis Harmon and **William Beene** for their review of early drafts of this guide.

Introduction

Remote access services on Unix and Linux such as ssh, local login processes, and even privilege escalation can all leverage a common system to provide authentication, authorization, and related services called **Pluggable Authentication Modules** or PAM. Symantec VIP integrates with PAM to provide two factor authentication for various services on Unix and Linux platforms.

This guide is separated into two Sections:

- **Section 1:** **Installing the Symantec VIP PAM integration module**
  This section provides some additional background to augment the integration guide available with the integration component download. Additionally, a new installer is available for both interactive and non-interactive (unattended) installation.

- **Section 2:** **Background on the PAM subsystem**
  This section provides background on the PAM subsystem itself and is an abbreviated primer.

Section 1

Installing the Symantec VIP

PAM integration module

Section 1 Contents

Installation options 5
Choosing a Validation Server 5
Supported Systems 6
Installing on Linux 7
VIP PAM integration module arguments 9
Configuring PUSH authentication with PAM 10

Installation options

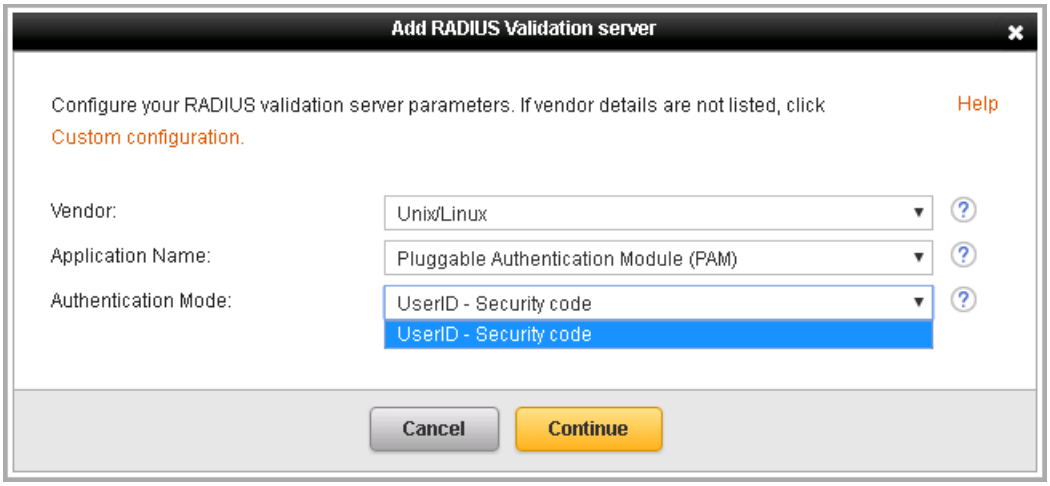
The installation details are described in Symantec's "Integration Guide for Pluggable Authentication Modules (PAM)" and can be summed up as:

- Configure the associated Validation Server on VIP Enterprise Gateway
- Move three binaries to two folders
- Update the PAM configuration
- Update other configuration if necessary (such as sshd_config)

These steps are unordered – all need to be completed, but they work slightly better in the order listed above.

Choosing a Validation Server

Symantec VIP Enterprise Gateway offers only one validation server to support VIP's integration component (displayed below), but other ones work.



To enable PUSH authentication, a custom one must be selected. This is covered in the Configuring PUSH authentication with PAM section, along with a picture of the validation server.

Supported Systems

This author successfully tested Symantec's VIP integration components on the following systems listed below. However, what may test successfully and what is supported by any organization can be different.

> **PLEASE NOTE**: The systems tested may or may no longer be supported by the vendor and the Symantec VIP integration component may or may not be supported by Symantec. Check with the vendor and Symantec's current recommendations for complete details. The systems listed below are for **informational purposes only**.

RedHat Enterprise Linux

RHEL 5	RHEL 6	RHEL 7
5.1	6.0	7.0
5.2	6.1	7.1
5.3	6.2	7.2
5.4	6.3	
5.5	6.4	
5.6	6.5	
5.7	6.6	
5.8	6.7	
5.9	6.8	
5.10		
5.11		

Ubuntu Desktop	**Ubuntu Server**
10.04 (64-bit only)	10.0.4.3 (64-bit only)
12.04 (64-bit only)	11.10 (64-bit only)
12.04.2 (32- and 64-bit)	14.04 (64-bit only)
12.04.3 (32- and 64-bit)	16.04 (64-bit only)
14.04 (32- and 64-bit)	
16.04 (32- and 64-bit)	

SUSE (64-bit only)	**OEL**	**CentOS**
10.1	5.5	5.8
10.2	5.7	6.2 (32- and 64-bit)
10.3	5.8	6.3
10.4	5.9	6.4
11	5.10	6.5
11 SP1	6.3	6.6
11 SP2	6.4	7.0
11 SP3	6.5	
11 SP4		
12 Server		
12 SP1 Server		

Installing on Linux

To simply the installation process, a short installation script provides an interactive configuration experience and a non-interactive experience (for deployment to multiple systems). There are two "parts" to the configuration: the raw installation of the executables plus the associated configuration and then a sample integration with various PAM-aware applications.

> **PLEASE NOTE:** The PAM application integration is merely to test that the VIP integration module is functional. Embedding this within any particular system's configuration will require careful planning and consideration. See Section 2 for background on reasoning about PAM syntax.

Interactive installation is available with superuser permission (generally as `root` or a member of the `sudoers` file). Start the install as follows:

```
./symc_pam_install.sh
```

Installing this on multiple systems at one time will still require planning, but moving the files to the correct location is a nuisance that this script may be able to assist with.

A "quick" method with the "secret=password" flag can test the syntax, but a secure implementation will rely upon a secret file accessible when the script is ran. One way to create this file is to use the included camouflage utility as follows:

```
./camouflage password > mysecretfile
```

But securely distributing the password in an automation script still needs to be done.

An alternative method involves pre-generating a series of these files on a secure computer and distributing each "installation package" uniquely to each system.

Install syntax for three non-interactive options follows:

```
./symc_pam_install.sh
        {ip=aa.bb.cc.dd} {port=ffff}
        {secret=password|secretfile=file} [timeout=gg]
        [retries=hh] [--ssh|--logon]

./symc_pam_install.sh
--force {ip=aa.bb.cc.dd} {port=ffff}
```

```
        {secret=password|secretfile=file} [timeout=gg]
        [retries=hh] {--ssh|--logon} [log=file]

./symc_pam_install.sh
--quiet {ip=aa.bb.cc.dd} {port=ffff}
        {secret=password|secretfile=file} [timeout=gg]
        [retries=hh] {--ssh|--logon} [log=file]
```

`ip`: The IP address of the VIP Enterprise Gateway server

`port`: The port of the configured Validation server on the VIP Enterprise Gateway server

`secret`: If the shared secret is specified on the command-line. Note that this is *before* the camouflage utility has been run against this shared secret. The more secure version of this is with secretfile.

`secretfile`: A pre-masked shared secret supplied in a file at that location.

`timeout`: The timeout, in seconds, to await a response from VIP Enterprise Gateway's validation server. Defaults to 10 if left unspecified.

`retries`: The number of times to attempt to communicate with VIP Enterprise Gateway's validation server before giving up. Defaults to 3 if left unspecified.

`--ssh`: To specify that a sample integration should be attempted with the ssh service.

`--logon`: To specify that a sample integration should be attemptd with the logon service

`--force`: Ignore questions and attempt the installation (will overwrite configuration files). Installation logs written to /tmp/symc_pam_install_log_DATESTAMP.log by default, unless overriden by the log= parameter.

`--quiet`: Will attempt the installation, but will fail if interaction is necessary. Installation logs written to /tmp/symc_pam_install_log_DATESTAMP.log by default, unless overriden by the log= parameter.

`--log`: Will store installation logs at a different location, when using `--quiet` or `--force`.

VIP PAM integration module arguments

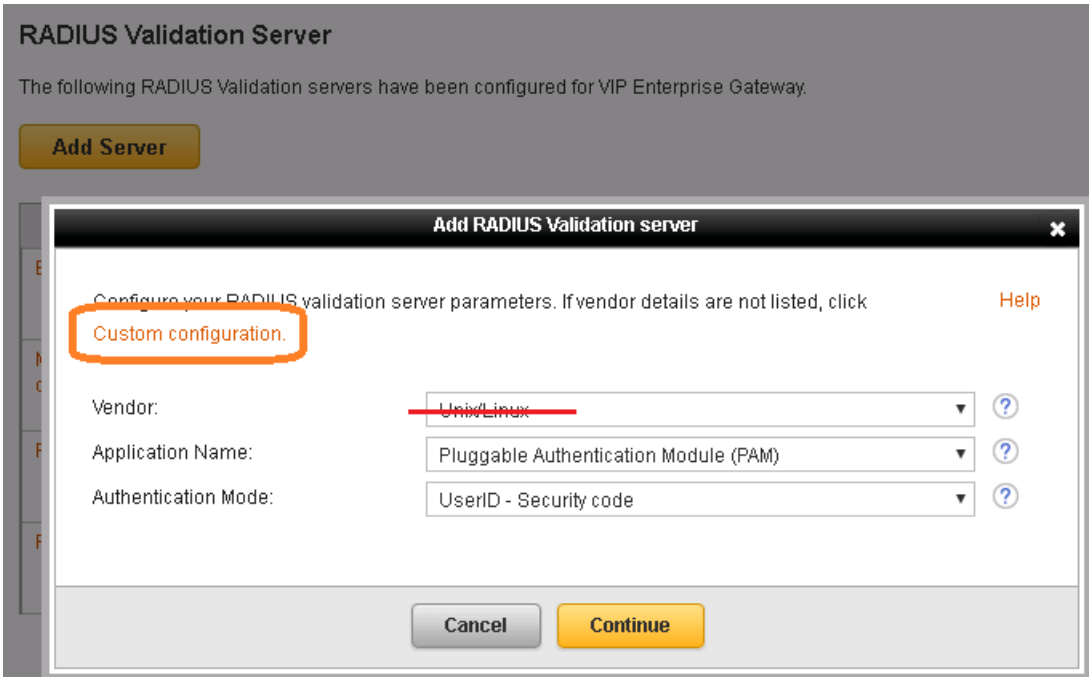
Symantec's VIP PAM integration supports several arguments that can be specified in the PAM configuration file:

Argument	Meaning
`no2fa group1:group2:…`	The defined local groups "group1", "group2" are not prompted for a Security Code.
`prompt=<prompt_string>`	Change the prompt offered to users that need to provide a security code.
`conf=<filename>`	Changes the configuration file that identifies where the VIP Enterprise Gateway validation server is located: IP, port, shared secret, timeout and number of retries are stored in there. The default location is: /etc/raddb/vrsn_otp
`split_password`	The last six characters are peeled off of the password and sent as the Security Code. The remaining characters are sent to the next PAM module for authentication.
`local_ip=aa.bb.cc.dd`	Useful if this system has multiple IP addresses and used to select an adapter to send authentication requests to VIP Enterprise Gateway.
`debug`	Provides additional logs to the system log.

For additional information, see "Advanced Configuration of PAM Files" in the "Integration Guide for Pluggable Authentication Modules (PAM)", included with the download for the integration component.

Configuring PUSH authentication with PAM

Symantec VIP's integration module is capable of authenticating via VIP's PUSH method. In VIP Enterprise Gateway, a validation server that supports "Access Challenge" will allow validation via PUSH. In order to configure this, select "custom configuration" instead of **Vendor** "Unix/Linux" at the initial Validation Server configuration screen in VIP Enterprise Gateway:



Two methods are available here:

- To have Enterprise Gateway perform username + password validation and PUSH validation, select:

  `UserID – LDAP Password – Security Code`

- To have Enterprise Gateway only perform PUSH validation, select:

  `UserID – Security Code`

Section 2

Background on the

PAM subsystem

Section 2 Contents

Understanding the PAM Subsystem 13
Advanced Concepts 14
Substacks 14
Includes 14
Custom actions in Linux 14
Arguments 16
Types of PAM module 16

Understanding the PAM Subsystem

PAM is the method used by multiple services on a *NIX system to centralize authentication and authorization logic. System Administrators program how authentication and authorization should work in different circumstances for different services. Each service calling PAM is governed by the associated PAM configuration file, centrally stored on most systems in /etc/pam.d/. For instance: /etc/pam.d/sshd

It is important to know that any given PAM ruleset is processed from the top, down. This will become important later.

Services configured to rely on PAM will call PAM in certain circumstances, such as when needing to authenticate a user. PAM, in turn, interprets the administrator-configured ruleset that calls a series of **modules**. These **modules** are called with a specific context (known equivalently as **type** or **facility**) that handles different aspects of interacting with the user, interacting with the system, or dynamically moving through the ruleset.

Each **rule** brings together the specific PAM **module**, the specific context it is called in (**type**/**facility**) and gives the administrator the ability to do different things depending upon the feedback from that module in a **control statement**. One typical configuration line (or **rule**) of "sshd" might look like this:

```
account          required    pam_nologin.so
```

In the above rule, sshd is the **service** that called PAM to invoke this **ruleset** in this configuration file, pam_sepermit.so is the **module** that is called by the rule above, auth is the **facility** – the particular section of pam_nologin.so that is invoked, and required is the **control** statement – what to do with the results that pam_nologin.so passes back when it is done.

Each **module** is called with one of the **types** below.

`auth` – Performs user authentication related tasks.

`account` – Performs user authorization related tasks.

`password` – Special handling for passwords.

`session` – Handles session initiation activities, if any.

Each **module** does different things with each of these **types**. The **module** might interact with the user (to prompt the user for their username or password) or interact with the system (assigning user ids,

launching a shell, etc.) **Modules** might not implement all four of these – see Appendix B for tables depicting popular modules and what **types** they support.

All the rules for a given **service** and **type** together are generally referred to as a "**stack**" and are generally organized together in the configuration file.

A **stack** groups together common logic with the idea that a set of rules need to be executed together, such as when a user is attempting to switch user via the "su" command.

So, when sshd needs a user to reauthenticate a session, PAM will invoke the **auth** stack of the sshd PAM configuration file and execute the **modules** within that stack as the logic indicates.

Advanced Concepts

Substacks

PAM can collect advanced processing together into a **substack** so that multiple rulesets can refer to this same logic. The entire processing of the **substack** is treated as a rule in the main flow of PAM processing, which can be important for setting up more complex logic. For instance, many systems have a "system-auth" file that allows other services to process common logic. Doing this abstracts complex logic that can be used in a variety of services and maintained separate from the service-specific configuration.

Several advanced notes: a substack can call a substack out to a depth of 16 (in Linux). Each "layer" keeps its own state independent of the others (eventually, though, a substack will finish and then return a value to the layer it came from).

Includes

To save space in the main configuration file for any given service, generic groups of **rules** can be grouped into common rulesets and "brought in" to any given configuration file via the **include** keyword. There are subtle shades of meaning between **include** and **substack** better explained elsewhere that articulate exactly how one vs. the other works.

Custom actions in Linux

The PAM **module** will exit with a specific **return code** that is vital to understand for the adminstrator when customizing PAM logic. Linux, Solaris, AIX all differ when it comes to the finer details, covered below for Linux. See Appendix A: Additional Resources and Guides for links to Solaris, AIX, and HP-UX PAM documentation.

By default, there are four **control** keywords that summarize common pairings between **return values** and **actions**:

`required`: [success=ok new_authtok_reqd=ok ignore=ignore default=bad]

`requisite`: [success=ok new_authtok_reqd=ok ignore=ignore default=die]

`sufficient`: [success=done new_authtok_reqd=done default=ignore]

`optional`: [success=ok new_authtok_reqd=ok default=ignore]

So the following two lines are equivalent:

```
account         required    pam_nologin.so
account         [success=ok new_authtok_reqd=ok ignore=ignore
default=bad]    pam_nologin.so
```

The general format is [**return value**=action]. There are a lot of possible **return values** that could be used (see Appendix A for links to more), but only a few "common" ones:

`default`

`success`

`new_authtok_reqd`

`ignore`

Links to additional return values are in Appendix A.

For some modules, you may need to handle one of these return values with a custom **action** in order to fine-tune authentication logic. There are seven **action** keywords. The **action** keyword essentially allows/disallows continued processing in this substack (or to stop processing them) and whether they should leave the current PAM stack's return value alone or update it. A table provides some clarity, but the logic is still difficult (this summarizes a few concepts – interested readers are encouraged to read the source code):

	Continue processing?	Update the **return value**?
bad	Yes	Yes (mark as PAM_PERM_DENIED)
die	No	Yes (mark as PAM_PERM_DENIED)
ok	Yes	If the current return value is success (PAM_SUCCESS), update it with this module's value. If the current return value is not successful, leave it alone.
done	**Yes**: If the current return value appears to be a failure	If the current return value is success (PAM_SUCCESS), update it with this module's value.

	No: If the current return value was successful.	If the current return value is not successful, leave it alone.
N	Yes	Skip next N modules in the current stack or substack
ignore	Yes	No
reset	Yes	Clear memory, stop processing this line, and continue processing.

Arguments

Modules can be called with arguments in some cases – see the definition for each module for what arguments it accepts. For instance, "debug" is defined for most of them.

Types of PAM module

PAM modules can be grouped depending upon their function. Below is one attempt at categorizing the available modules and what facilities they support.

Logic: These modules principally offer control logic when executing a given PAM configuration file. Used in conjunction with **actions** to be taken on particular **return values**.

> Modules: pam_succeed_if, pam_deny, pam_faildelay, pam_permit

General: These modules should be available for use in most situations.

> Modules: pam_access, pam_debug, pam_echo, pam_env, pam_exec, pam_group, pam_limits, pam_listfile, pam_loginuid, pam_warn

Authentication: Provide various controls on authentication attempts such as making sure the user is or isn't root, requiring that the user be in the local database, allowing login based on timestamp, and more.

> Modules: pam_cracklib, pam_localuser, pam_kerberos, pam_nologin, pam_pwhistory, pam_rootok, pam_securetty, pam_shells, pam_tally, pam_tally2, pam_time, pam_timestamp, pam_userdb, pam_wheel

Console: PAM modules that are principally useful when using an interactive (shell) session.

> Modules: pam_filter, pam_issue, pam_laslog, pam_mkhomedir, pam_motd, pam_rhosts, pam_umask, pam_unix

Other: Other modules for special situations.

Modules: pam_ftp, pam_keyinit, mail, pam_namespace, pam_selinux, pam_xauth

Troubleshooting tips

Is the user actually on the system?

Are the files present and in the right place?

Are the right ones present?

- 64-bit vs. 32-bit?
- Unix vs. Linux

Check the logs:

- VIP Enterprise Gateway (with the validation server set in “Debug” mode)
- Local on-box logs (sometimes /var/log/secure, /var/log/messages, /var/adm/authlog, /var/log/auth.log)

Try using vsradiusclient_test to authenticate.

Can you reach the VIP Enterprise Gateway server from the target server?

Appendix A: Additional Resources and Guides

Symantec VIP Quick Start Guides

Symantec VIP Resources

Symantec VIP Integration Guides

Linux PAM links:

- The Linux-PAM System Administrators' Guide
  - PAM Modules
  - PAM Actions
- The Linux-PAM Application Writers' Guide
- The Linux-PAM Module Writers' Guide

PAM manual page (and return codes)

More PAM return codes

Non-Linux PAM links:

- Oracle Solaris 11 – PAM Configuration
- Pluggable Authentication Modules – AIX 6.1
- PAM Administration - SUN

Appendix B: Tables

Linux PAM modules – Alphabetically

	auth	account	password	session
access	Y	Y	Y	Y
cracklib			Y	
debug	Y	Y	Y	Y
deny	Y	Y	Y	Y
echo	Y	Y	Y	Y
env	Y			Y
exec	Y	Y	Y	Y
faildelay	Y			
filter	Y	Y	Y	Y
ftp	Y			
group	Y			
issue	Y			
keyinit				Y
lastlog	Y	Y		Y
limits				Y
listfile	Y	Y	Y	Y
localuser	Y	Y	Y	Y
loginuid				Y
mail	Y			Y
mkhomedir				Y
motd				Y
namespace				Y
nologin	Y	Y		
permit	Y	Y	Y	
pwhistory			Y	
rhosts	Y			
rootok	Y	Y	Y	
securetty	Y			
selinux				Y
shells	Y	Y		
succeed_if	Y	Y	Y	Y
tally	Y	Y		
tally2	Y	Y		
time		Y		
timestamp	Y			Y
umask				Y
unix	Y	Y	Y	Y
userdb	Y	Y		
warn	Y	Y	Y	Y
wheel	Y	Y		
xauth				Y

Linux PAM modules – By Facility

	auth	account	password	session
access	Y	Y	Y	Y
debug	Y	Y	Y	Y
deny	Y	Y	Y	Y
echo	Y	Y	Y	Y
exec	Y	Y	Y	Y
filter	Y	Y	Y	Y
listfile	Y	Y	Y	Y
localuser	Y	Y	Y	Y
succeed_if	Y	Y	Y	Y
unix	Y	Y	Y	Y
warn	Y	Y	Y	Y
permit	Y	Y	Y	
rootok	Y	Y	Y	
lastlog	Y	Y		Y
nologin	Y	Y		
shells	Y	Y		
tally	Y	Y		
tally2	Y	Y		
userdb	Y	Y		
wheel	Y	Y		
env	Y			Y
mail	Y			Y
timestamp	Y			Y
faildelay	Y			
ftp	Y			
group	Y			
issue	Y			
rhosts	Y			
securetty	Y			
time		Y		
cracklib			Y	
pwhistory			Y	
keyinit				Y
limits				Y
loginuid				Y
mkhomedir				Y
motd				Y
namespace				Y
selinux				Y
umask				Y
xauth				Y

About Symantec

Symantec Corporation (NASDAQ: SYMC), the world's leading cyber security company, helps organizations, governments and people secure their most important data wherever it lives. Organizations across the world look to Symantec for strategic, integrated solutions to defend against sophisticated attacks across endpoints, cloud and infrastructure. Likewise, a global community of more than 50 million people and families rely on Symantec's Norton and LifeLock product suites to protect their digital lives at home and across their devices. Symantec operates one of the world's largest civilian cyber intelligence networks, allowing it to see and protect against the most advanced threats. For additional information, please visit www.symantec.com or connect with us on Facebook, Twitter, and LinkedIn.

For specific country offices and contact numbers, please visit our website.

Symantec World Headquarters 350 Ellis St.

Mountain View, CA 94043
USA+1 (650) 527 8000

1 (800) 721 3934

www.symantec.com