Magento Payment Bridge ver. 1.0.0.0

Administration Guide

Version 1.0.0.0

Approval Date: 07/02/2010

Table of Contents

Notice

About this Document

This document describes command line tools for the Magento Payment Bridge administration and maintenance. These console tools allow for performing all operations necessary to create and configure merchant accounts, and to manage the encryption and security settings of Magento Payment Bridge.

Revision History

Revision	Date	Name	Title	Summary of Changes
Revision 1.0	03/20/2010	Michael Bessolov	Director of Technology	Initial Release
Revision 1.1	06/04/2010	Michael Bessolov	Director of Technology	Minor changes
Revision 1.2	06/09/2010	Michael Bessolov	Director of Technology	Minor changes
Revision 1.3	06/17/2010	Michael Bessolov	Director of Technology	Implementing Encrypted Storage section added
Revision 1.4	06/22/2010	Michael Bessolov	Director of Technology	How to Apply Patch section added
Revision 1.5	06/25/2010	Michael Bessolov	Director of Technology	Minor changes
Revision 1.6	07/02/2010	Michael Bessolov	Director of Technology	Application Setup Tool section added

Note: This Administration Guide must be reviewed on a yearly basis, whenever the underlying application changes or whenever the PA-DSS requirements change. Updates should be tracked and reasonable accommodations should be made to distribute or make the updated guide available to users. Magento Inc. will distribute the Administration Guide to new customers via the Magento Inc. support website (https://support.varien.com/).

Implementing Encrypted Storage

Encrypted Storage Initial Setup

Prior to start configuring an encrypted storage, a clean partition of disk to be encrypted is required.

By default, RHEL already has encryption tools in its default setup.

You need to create a file of necessary size that will be connected as a separate disk.

```
# dd if=/dev/zero of=/data/crypto bs=1G count=0 seek=10
# losetup /dev/loop0 /data/crypto
# echo "losetup /dev/loop0 /data/crypto" >> /etc/init.d/rc.local
# cryptsetup -y create cryptedDrive /dev/loop0
Enter passphrase: my_passphrase <-- enter the desired password that will be used to
encrypt the storage
Verify passphrase: my_passphrase
# mke2fs -j /dev/mapper/cryptedDrive
# mkdir /mnt/crypto
# mount /dev/mapper/cryptedDrive /mnt/crypto
```

You also need to encrypt the swap so that our data cannot be read in cleartext in the swap.

Swap Partition Encryption

Since you do not need to store useful information in the swap, you can use some random generated password to encrypt a swap partition each time at boot.

To achieve this, you need to create a file `/etc/crypttab` and provide the configuration of what partition is mapped to the encrypted storage and what type it has in this file.

And, add a mapped encrypted partition to your /etc/fstab file.

```
# echo "cryptedSwap        /dev/sdb        /dev/urandom        swap" >
/etc/crypttab
```

Now, you need to open the /etc/fstab file in a text editor and change the device used for swapping to '/dev/mapper/cryptedSwap'.

Note: In order to have an encrypted swap, you need to reboot the server.

Encrypted Partition Configuration After Server Boot

Currently, you have an encrypted swap partition with a random password that is added during the boot time and another encrypted partition that must be mounted after the server boot.

The reason for doing so is as follows – if you add another partition encrypted with a non-random password, the system will ask you to enter a password before starting the opensshd server. And to continue the system boot, you will need to connect to your server console to provide the password for the encrypted storage.

Note that there may be a problem when your server is in a collocation or when you use a dedicated server located in another place.

This can be done the other way. You have already created the encrypted storage and a new file system in it. You have not added this storage to the */etc/crypttab* file, so our server is able to boot and start main services like opensshd and others. After it, you need to log on to the server through ssh using the system administrator account, create a mapped storage device, mount it to its mount point, and start all services that depend on this storage.

Note: As you are using `sudosh` for the Magento Payment Bridge administrator accounts, administrators will not be able to log on as sudosh logs are stored in the encrypted storage and are not mounted yet.

In any case, you need to leave the system administrator account with the **bash** shell.

After the administrator logs on to the system, he needs to connect and mount encrypted partition in the following way:

```
# /sbin/cryptsetup create cryptedDrive /dev/loop0
Enter passphrase: my_passphrase
# /bin/mount /dev/mapper/cryptedDrive /mnt/crypto
```

Preparing to Use Magento Payment Bridge Administration Tools

Prior to start working with the administration tools described below, the administrator must change the current working directory in the following way:

```
$ cd %APP_BASE_DIR%
```

where `%APP_BASE_DIR%` is an absolute path to the Magento Payment Bridge application base directory.

Application Setup Tool

Magento Payment Bridge application setup tool is used to set the encryption key used for the database data encryption/decryption and database access password in the configuration file.

Note:

This tool must be used during the initial application installation process.

```
$ php tools/setup.php

Application setup tool
Setting up

Enter new encryption key PART 1:
Enter new encryption key PART 2:
Enter database connection password:

Encrypting DB connection password... DONE
Writing values to the config... SUCCESS
```

Merchant Configuration Tool

Magento Payment Bridge merchant configuration tool provides many options that allow performing all necessary operations for creating and deleting merchants, configuring their parameters, assigning payment gateways and payment services to their configurations.

```
$ php tools/merchant.php

Merchant Configuration Tool

Usage: merchant.php [-a] | [-r merchant-code] | [-i merchant-code] |
[-k merchant-code [new-key]] | [-m merchant-code] | [--ek merchant-code] |
[--psc merchant-code] | [--pgc merchant-code] | [--cgc [merchant-code]]

Options:
  -a                             Add new merchant.
  -r merchant-code               Remove a merchant.
  -i merchant-code               Display the merchant data and configuration.
  -k merchant-code [new-key]     Change the merchant Key.
  -m merchant-code               Change the merchant E-mail.
  --ek merchant-code             Change the merchant Data Transfer Key.
  --psc merchant-code            Change the merchant payment services
                                 configuration.
  --pgc merchant-code            Change the merchant payment gateways
                                 configuration.
  --cgc [merchant-code]          Check payment gateways configuration for
                                 specified merchants or all of them.
```

Creating Merchant

The administrator starts with creating a merchant. To create a new merchant entry in the Magento Payment Bridge database, the **-a** command line key is used.

```
$ php tools/merchant.php -a

Merchant Configuration Tool
Add new merchant.
Enter Merchant Code: somemerchant
Enter Data Transfer Key (will be generated if not specified):
Enter new E-mail: somemerchant@gmail.com

New merchant has successfully been added.
Data Transfer Key generated: cfd3e40fa5e5b23ce2fab78a7d9fbadc

Merchant Key generated:
f5dd8311dfbd368b3b9891ae338b88c2a93c236731de0aa2c98b676963eea96b
Use 'merchant.php -pgc somemerchant' to setup payment gateways configuration for
this merchant.
```

Changing Merchant Configuration Parameters

To change one of the merchant parameters, any of the **-k**, **-m**, or **--ek** keys are used.

In particular, to change the merchant key, the **-k** key is used as shown in the following sample:

```
$ php tools/merchant.php -k somemerchant NEW_MERCHANT_KEY

Merchant Configuration Tool
Change the merchant Key.

Key has been changed successfully.
New key: 41d1f934d4527bfbeccac6569af57e80baacb7d71007a87b4ba7cc9d2cadb75c
```

To change the data transfer key, the **--ek** key is used.

```
$ php tools/merchant.php --ek somemerchant

Merchant Configuration Tool
Change the merchant Data Transfer Key.
Enter new Data Transfer Key: NEW_DATA_TRANSFER_KEY

Data Transfer Key has been changed successfully.
```

In case neither a merchant key nor a data transfer key is specified, the script will request the administrator to enter the new key value.

To change the merchant email address, the **-m** key is used.

```
$ php tools/merchant.php -m somemerchant

Merchant Configuration Tool
Change the merchant E-mail.
Enter new E-mail: newemail@gmail.com

E-mail has been changed successfully.
```

Configuring Merchant Payment Gateways

To configure payment gateways for the merchant, the **--pgc** key is used.

```
$ php tools/merchant.php --pgc somemerchant

Merchant Configuration Tool
Change the merchant payment gateways configuration.
Select payment gateway (paypal_direct, paypaluk_direct, verisign, authorizenet):
authorizenet
Enter allowed credit cards types [default: AE,VI,MC,DI]:
Default value 'AE,VI,MC,DI' has been used.

Use credit card verification number (y|n) [default: y]:
Default value 'y' has been used.

E-mail customer (y|n) [default: n]: y
```

```
Enter login: anet_login
Enter merchant e-mail (optional):
Is account using for testing (y|n) [default: n]: y
Enter data transfer key: some_data_transfer_key
Enter gateway URL [default: https://test.authorize.net/gateway/transact.dll]:
Default value 'https://test.authorize.net/gateway/transact.dll' has been used.


Payment gateway configuration has been saved successfully.
```

Repeat this operation for each payment gateway which is required to be configured for the merchant.

To check the merchant for the configured payment gateways, the **--cgc** command line key is used.

```
$ php tools/merchant.php --cgc somemerchant

Merchant Configuration Tool
Check payment gateways configuration for specified merchants or all of them.
Checking the merchant somemerchant...
    Checking gateway 'paypal_direct'...not configured
    Checking gateway 'paypaluk_direct'...not configured
    Checking gateway 'verisign'...not configured
    Checking gateway 'authorizenet'...OK
```

Configuring Merchant Payment Services

Currently, only the Centinel payment service support has been implemented in Magento Payment Bridge. To configure the payment service for the merchant, the **-psc** command line key is used as shown in the following sample:

```
$ php tools/merchant.php --psc somemerchant

Merchant Configuration Tool
Change the merchant payment services configuration.
Sole payment service 'centinel' has been selected for the merchant configuration.
Enter processor id: 999-09
Enter merchant id: merchantTEST
Enter password: merchantPASS
Test mode flag (y|n) [default: n]: y

Payment service configuration has been saved successfully.
```

Please note: *Currently, the administrator cannot delete any particular payment gateway or payment service configuration from the merchant entry using console tools.*

Displaying Merchant Info

When the administrator needs to check some merchant's configuration parameters, this can be done by using the **-i** key, as shown in the following code sample.

```
$ php tools/merchant.php -i somemerchant

Merchant Configuration Tool
Display the merchant data and configuration.
```

```
Merchant data:
    Merchant ID:5
    Merchant Code:somemerchant
    Merchant Key:f5dd8311dfbd368b3b9891ae338b88c2a93c236731de0aa2c98b676963eea96b
    Merchant E-mail:somemerchant@gmail.com
    Data Transfer Key:cfd3e40fa5e5b23ce2fab78a7d9fbadc

Merchant configuration:
    payment_gateways/authorizenet/cctypes: AE,VI,MC,DI
    payment_gateways/authorizenet/useccv: 1
    payment_gateways/authorizenet/email_customer: 1
    payment_gateways/authorizenet/login: anet_login
    payment_gateways/authorizenet/merchant_email:
    payment_gateways/authorizenet/test: 1
    payment_gateways/authorizenet/trans_key: some_data_transfer_key
    payment_gateways/authorizenet/cgi_url:
https://test.authorize.net/gateway/transact.dll
    payment_services/centinel/processor_id: 999-09
    payment_services/centinel/merchant_id: merchantTEST
    payment_services/centinel/password: merchantPASS
    payment_services/centinel/test_mode: 1
```

Deleting Merchant

To delete the merchant, the **-r** key is used. Once the merchant is deleted, all his/her data is removed from the database, and all related configuration parameters are deleted as well.

```
$ php tools/merchant.php -r somemerchant

Merchant Configuration Tool
Remove a merchant.
Do you really want to delete the merchant? [y/n] y

Merchant has been successfully deleted.
```

Encryption/Decryption Tool

Magento Payment Bridge encryption tool provides the following options:

```
$ php tools/crypt.php

Encryption tool

Usage: crypt.php [-r] | [-d] | [-e]

Options:
  -r             Re-encrypts all encrypted fields in DB using a new key and
                 stores the key to config. The password to access DB from config
                 is also re-encrypted using the new key.
  -d             Decrypts data interactively by requesting a data and a key
                 value (URL-encoded data is decoded automatically).
  -e             Encrypts data interactively by requesting a data and a key
                 value.
```

Re-Encrypting Data in the Database

To create a new data encryption key and to apply it to all already encrypted data in the database, the **-r** key is used. It can either use a specified key or generate a random one if no key has been specified.

As a result, all data previously encrypted with an old key will end up encrypted with the new data encryption key, so no data will be lost.

```
$ php tools/crypt.php -r

Encryption tool
Re-encryption

Do you really want to re-encrypt the database? [y/n] y
Enter new encryption key PART 1:
Enter new encryption key PART 2:
Application is being locked... SUCCESS
Mapper 'Varien_Mapper_Merchant_Config' re-encryption... SUCCESS
Mapper 'Varien_Mapper_Payment_Gateway_Transaction' re-encryption... SUCCESS
Mapper 'Varien_Mapper_Payment_Transfer' re-encryption... SUCCESS
Mapper 'Varien_Mapper_Token' re-encryption... SUCCESS
Mapper 'Varien_Mapper_Merchant' re-encryption... SUCCESS
Re-encrypting DB connection password... DONE
Writing new key to the config... SUCCESS
Application is being unlocked... SUCCESS
```

Encrypting Data

To decrypt the arbitrary data, the **-e** key is used. The script will request an encryption key value and textual data to encrypt. Then, it will return the same data encrypted.

```
$ php tools/crypt.php -e

Encryption tool
Encryption
```

```
Enter encryption key: SOME_DB_KEY
Enter data to encypt: Some arbitrary text info

Encrypted data:

f0260b4fed830704:Wai3SC0Iq1eMyFzs0yYR5alh9mGT2ZfgRW08pfKyPc0=
```

Decrypting Data

To reverse the process, the **-d** key is used to decrypt some previously encrypted textual data. Similarly, the script will request an encryption key value and previously encrypted data, and then will return the same data decrypted.

Please note, that the decryption will finish successfully only if the same key is used for it as has been used for the encryption.

```
$ php tools/crypt.php -d

Encryption tool
Decryption

Enter decryption key: SOME_DB_KEY
Enter data for decryption:
f0260b4fed830704:Wai3SC0Iq1eMyFzs0yYR5alh9mGT2ZfgRW08pfKyPc0=

Data is not URL encoded. Decoding skipped.

Decrypted data:

Some arbitrary text info
```

IP Restriction Management Tool

An approved list management tool is used as one of the measures to prevent unauthorized access to the Magento Payment Bridge service. It allows the administrator to restrict all incoming requests to the Magento Payment Bridge service for a limited list of IP addresses.

```
$ php tools/ipfilter.php

IP filter tool

Usage: ipfilter.php [-a ip-address] | [-r ip-address] | [-s]

Options:
  -a ip-address  Add new IP to allowed list.
  -r ip-address  Remove IP from allowed list.
  -s             Show current allowed IP list.
```

Whitelisting IP-address

To add a new IP address to the list, the **-a** command line key is used.

```
$ php tools/ipfilter.php -a 192.168.0.1

IP filter tool
Add new IP to allowed list.
IP address has been successfully added to the approved list.
```

Deleting IP-address

To remove the IP address from the approved list, the **-r** key is used.

```
$ php tools/ipfilter.php -r 192.168.0.1

IP filter tool
Remove IP from allowed list.
Do you really want to delete IP? [y/n] y
IP address has been successfully removed from the approved list.
```

Displaying IP-address restrictions

Finally, to display the whole approved list, the **-s** key is used.

```
$ php tools/ipfilter.php -s

IP filter tool
Show current allowed IP list.
192.168.0.77
192.168.0.123
192.168.0.231
192.168.0.155
192.168.0.122
192.168.0.12
192.168.0.2
192.168.0.3
192.168.0.4
```

```
192.168.0.5
```

Database Password Management

Storing Password

An encrypted password for connecting to the database is stored in the `resource/options/password` section of the `cfg/config.php` file. A key for its encryption is stored in the `crypt_key_db` section of the `cfg/config.php` file (this key is also used for encrypting data in the database).

Setting/Changing Password for Connecting to Database

Prior to setting or changing a password, it must be encrypted with the **-e** key of the `tools/crypt.php` tool.

```
php tools/crypt.php -e
Encryption tool
Encryption
Enter encryption key: crypt_key_db <-- enter your current encryption key
Enter data to encrypt: my_db_password <-- enter the desired password to connect to
the database
Encrypted data:
2d385ae5490a9d53:tzczA7+G2rtWhA6iJIn/Pw== <-- copy the entire encrypted string and
paste it to the resource/options/password section of the cfg/config.php file
```

Note: When changing a key for encrypting data in the database with the **-r** key of the `tools/crypt.php` tool, the password for connecting to the database is automatically re-encrypted with a new key; the administrator does not need to take any additional actions.

How to Set Up Logging and User Policy

The following instructions are given as an example for CentOS 5.4. As CentOS is based on Red Hat Enterprise Linux (RHEL), the following instructions will also work in RHEL. Instructions may slightly vary on other Linux distributives.

Logging Setup

Bouncing Idle Users

To make idle users log off, execute the following commands as root:

```
# echo "readonly TMOUT=900" >> /etc/profile.d/os-security.sh
# echo "readonly HISTFILE" >> /etc/profile.d/os-security.sh
# chmod +x /etc/profile.d/os-security.sh
```

Note: TMOUT=900 - sets session timeout to 900 seconds, i.e. 15 minutes.

This will cause all users whose sessions are idle for more than 900 seconds to be automatically logged off.

Using sudo to Get Root Permissions

1. To install `sudo`, run the following command:

```
# yum install sudo
```

2. To configure `sudo` to allow it gaining root permissions only for the people in wheel group, run visudo as root

```
Cmnd_Alias NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping, /sbin/dhclient,
/usr/bin/net, /sbin/iptables, /usr/bin/rfcomm, /usr/bin/wvdial, /sbin/iwconfig,
/sbin/mii-tool
Cmnd_Alias SOFTWARE = /bin/rpm, /usr/bin/up2date, /usr/bin/yum
Cmnd_Alias SERVICES = /sbin/service, /sbin/chkconfig
Cmnd_Alias LOCATE = /usr/bin/updatedb
Cmnd_Alias STORAGE = /sbin/fdisk, /sbin/sfdisk, /sbin/parted, /sbin/partprobe,
/bin/mount, /bin/umount
Cmnd_Alias DELEGATING = /usr/sbin/visudo, /bin/chown, /bin/chmod, /bin/chgrp
Cmnd_Alias PROCESSES = /bin/nice, /bin/kill, /usr/bin/kill, /usr/bin/killall
Cmnd_Alias DRIVERS = /sbin/modprobe
Defaults requiretty
Defaults env_reset
Defaults env_keep = "COLORS DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR \
LS_COLORS MAIL PS1 PS2 QTDIR USERNAME \
LANG LC_ADDRESS LC_CTYPE LC_COLLATE LC_IDENTIFICATION \
LC_MEASUREMENT LC_MESSAGES LC_MONETARY LC_NAME LC_NUMERIC \
LC_PAPER LC_TELEPHONE LC_TIME LC_ALL LANGUAGE LINGUAS \
_XKB_CHARSET XAUTHORITY"
root ALL=(ALL) ALL
%wheel ALL=(ALL) ALL
```

User Actions Logging

1. To install `sudosh`, install the rpmforge repo first.

```
# rpm -Uvh http://apt.sw.be/redhat/el5/en/x86_64/rpmforge/RPMS/rpmforge-release-
0.5.1-1.el5.rf.x86_64.rpm
```

2. Then, install `sudosh` in the following way:

```
# yum install sudosh
```

3. Create a log directory for user logs and set permissions for the directory.

```
# mkdir -p /mnt/crypto/sudosh
# chmod 733 /mnt/crypto/sudosh
```

4. Configure `sudosh` - *edit /etc/sudosh.conf.*

```
logdir = /mnt/crypto/sudosh
default shell = /bin/bash
delimiter = -
syslog.priority = LOG_INFO
syslog.facility = LOG_LOCAL2
-c arg allow = scp
-c arg allow = rsync
```

Warning: *By default, the log directory is* /var/log/sudosh.

Critical: *The config file in the current sudosh version must not contain comments or blank lines. Otherwise, sudosh terminates with* ***segmentation fault*** *error code.*

Note: *Keep in mind the following options explanations:*

- *logdir - where user's logs will be written*
- *default shell - default shell that the user will be running*
- *-c arg allow = command - command that will be allowed to run as shell argument string*

5. To prevent users from changing their shell scripts, the `suid` bit from `chsh` command must be removed in the following way:

```
# chmod 711 /usr/bin/chsh
```

The same can be done by forbidding users to run `chsh` command by setting its permissions to 700

```
# chmod 700 /usr/bin/chsh
```

Creating Magento Payment Bridge Administrator Account

To create a Magento Payment Bridge administrator account, use the following command:

```
# useradd %USERNAME% -m -s /usr/bin/sudosh
```

***Warning:** Please note that once the partition, where logs for `sudosh` are located, is full, users will not be able to log in.*

***Note:** Keep in mind the following options explanations:*

- *-m - create user home dir if it doesn't exist yet*
- *-s - set user login shell to given value*
- *%USERNAME% - user's login name*

Important:

After the Magento Payment Bridge application is installed, you must use only the Magento Payment Bridge administrator accounts.

Displaying Logged User Sessions

To display all logged user sessions, run sudosh-replay under root permissions. To display a specific session, run sudosh-replay ID, where ID is an identification of a session.

For more detail, see the following code sample:

```
# sudosh-replay
Date Duration From To ID
==== ======== ==== == ==
05/13/2010 15:04:14 1h andrew andrew andrew-andrew-1273752254-iOi1NJj3J7UpWbEH
05/13/2010 16:27:39 1m50s andrew andrew andrew-andrew-1273757259-sX2LOYh9KoARHN0J
05/13/2010 16:37:53 21m30s andrew andrew andrew-andrew-1273757873-KSWLCox3EK4mWVKv
05/13/2010 16:59:36 22m58s andrew andrew andrew-andrew-1273759176-x6xgsK7AKqB6nsw4

Usage: sudosh-replay ID [MULTIPLIER] [MAXWAIT]
See 'sudosh-replay -h' for more help.
Example: sudosh-replay andrew-andrew-1273759176-x6xgsK7AKqB6nsw4 1 2

# sudosh-replay andrew-andrew-1273759176-x6xgsK7AKqB6nsw4 2
```

To find out whether anybody has accessed specific files (e.g. security logs, etc.), the grep utility must be used.

For example, to see who has accessed the *%PBRIDGE%/var/log/* directory where transactions and other Magento Payment Bridge logs are stored, the following commands must be executed:

```
# cd /mnt/crypto/sudosh
# grep -rl '%PBRIDGE_BASE_DIR/var/log%' ./*-script-*
./%USERNAME%-%USERNAME%-script-%TIMESTAMP%-%ID%
```

In case the user %USERNAME% has accessed the %PBRIDGE_BASE_DIR%/var/log/ directory, the following command must be executed to see the list of actions the user has performed during the session:

```
# sudosh-replay %USERNAME%-%USERNAME%-%TIMESTAMP%-%ID%
```

User Policy Setup

1. To set up the minimum password length and the expiration time for new users, edit /etc/login.defs

```
PASS_MAX_DAYS 90
PASS_MIN_DAYS 0
PASS_MIN_LEN 7
PASS_WARN_AGE 7
```

Note: The specified parameters have the following definitions:

PASS_MAX_DAYS - sets maximum age for passwords

PASS_MIN_DAYS - sets minimum age for passwords

PASS_MIN_LEN - sets minimum length for passwords

PASS_WARN_AGE - sets warn age for passwords, i.e. - the system will warn you if your password is expiring in PASS_WARN_AGE days

2. To set up password strength policy, edit */etc/pam.d/system-auth*

```
auth required pam_env.so
auth required pam_tally2.so deny=6 unlock_time=1800
auth sufficient pam_unix.so nullok try_first_pass
auth requisite pam_succeed_if.so uid >= 500 quiet
auth required pam_deny.so

account required pam_unix.so
account sufficient pam_succeed_if.so uid < 500 quiet
account required pam_permit.so

password requisite pam_cracklib.so try_first_pass retry=3 minlen=7 dcredit=-1
ocredit=1 lcredit=-1 ucredit=1
password sufficient pam_unix.so sha512 shadow nullok try_first_pass use_authtok
remember=4
password required pam_deny.so

session optional pam_keyinit.so revoke
session required pam_limits.so
session [success=1 default=ignore] pam_succeed_if.so service in crond quiet use_uid
session required pam_unix.so
```

Note:

deny=6 – Sets the amount of failures after which an account is locked.

unlock_time=1800 – Sets the timeout for locked accounts to 1800 seconds.

This policy sets up the minimum password strength to 7 points:

lcredit=-1 – The minimum number of lowercase letters that must be met for a new password is 1.

ucredit=1 – The number of points to one uppercase letter.

dcredit=-1 – The minimum number of digits that must be met for a new password is 1.

ocredit=1 – The number of points for another character.

remember=4 – Sets the password history to remember 4 old passwords.

3. To see a locked account, run the `pam_tally2` command with specific parameters:

```
# pam_tally2 -u %USERNAME%
Login Failures Latest failure From
%USERNAME% 1 %DATE_TIME% %IP.ADD.RE.SS%
```

4. To change the default hashing algorithm, run the following command:

```
# authconfig --passalgo=sha512 --update
```

User Permissions Management

General Notes

1. To create a user group

```
# groupadd %GROUPNAME%
```

2. To modify a user group

```
# usermod -aG %GROUPNAME% %USERNAME%
```

3. To set specific permissions for a file / directory, use chmod command

Note: *The chmod object operation permissions file, where:*

- *object is one of u,g,o,a ("u" means user, "g" means group, "o" means others, "a" means all)*
- *operation is - or ("-" means remove and "" means add)*
- *permission is one of r,w,x ("r" means read, "w" means write, "x" means execute*)

For example,

```
chmod ug+wx somefile.txt
```

…sets the write and execute permissions to the *somefile.txt* file for the user and the group. Additional options may be set, i.e. -R to apply permissions to the directory recursively.

For example,

```
chmod -R ug+rwx,o-w somedir
```

…recursively sets the read, write, and execution permissions to the *somedir* directory for the user and the group and recursively revokes the write permissions for all the others.

To change file owner you can use `chown` command

```
# chown %USERNAME% %FILE%
# chown -R %USERNAME% %DIRECTORY%
```

Application Base Directory Settings

To provide Magento Payment Bridge application security, a separate user group must be created. This group must contain the following users:

1. Magento Payment Bridge application local administrator
2. User the web server will be started from

Reading, writing, and executing permissions for the base application directory must only be granted for this user group.

How to Configure Apache HTTP Server

Apache HTTP Server must be compiled with the `mod_headers` module. Accordingly, Apache server configuration file must contain the following directives:

```
<Directory %APP_BASE_DIR%>
        AllowOverride All
</Directory>
```

where `%APP_BASE_DIR%` is an absolute path to the Magento Payment Bridge application base directory.

How to Configure MySQL Server

Prior to configuring the MySQL server, you need to install it first using the following command:

```
# yum install mysql-server
```

By default, the MySQL server writes all queries to the history file. However, there may occur a situation when a query for the user password setup or change will be logged to the MySQL history file in a clear text.

To avoid such situations, you must set the **MYSQL_HISTFILE** environment variable to */dev/null* before adding a new user with a password or modifying an existing one. After this operation stops, you must set the **MYSQL_HISTFILE** environment variable to `~/.mysql_history`.

You can see this operation in the following example of the Magento Payment Bridge database setup.

```
# export MYSQL_HISTFILE=/dev/null
# mysql_secure_installation
# mysql
mysql> create db payment_bridge;
mysql> grant all on payment_bridge.* to `pbuser`@`IP.ADD.RE.SS` identified by
'P@ssw0rd';
mysql> quit;
# export MYSQL_HISTFILE=~/.mysql_history
```

Note: Never add users with the `'mysqladmin -u user -ppassword'` command as in such case this command is written to the shell history file and other users that are logged into the system may see your command through `ps` or `top`.

How to Apply Patch

After you have been notified about a new patch, you need to take the following steps to apply it:

1. You need to log on to https://support.varien.com with your credentials received after registration.
2. Go to Downloads section, download the required patch (%PATCH.tgz%).
3. Upload the downloaded patch on the server where Magento Payment Bridge application is installed via the SCP.
4. Log on to the server where Magento Payment Bridge application is installed via the SSH and change the current working directory in the following way:

```
$ cd %APP_BASE_DIR%
```

Where `%APP_BASE_DIR%` is an absolute path to the Magento Payment Bridge application base directory.

5. Move the required patch to the current working directory.
6. Unpack the patch in the following way:

```
$ tar -xvf %PATCH.tgz%
```

As a result you will receive the %PATCH.patch% and md5.txt files.

7. Check the MD5 sum in the following way:

```
$ md5sum -c md5.txt
```

If verification has been successfully performed, you may proceed to the following step; in case it has not, please contact the Magento support team.

8. Apply the patch in the following way:

```
$ patch -p0 -i %PATCH.patch%
```

9. Delete the %PATCH.patch% and md5.txt files.