Monetra®

Secure Implementation Guide

(Covering CISP, PCI and PABP requirements)

Revision: 1.3
September, 2006

Table of Contents

1 Payment Systems Security

1.1 Introduction

The landscape of merchants' electronic payment systems has changed radically over the past ten years. What was once mostly proprietary (closed loop) systems, has become a network of computers connecting to multiple partners and services (ASP's) via the public Internet.

In the past five years alone, Internet and computing technologies have grown at such a rate that many corporations today must provide substantial resources (policy and human-power) devoted solely to systems security. Keeping information such as customer and market data confidential has rapidly become one of the highest corporate priorities.

In April 2000, Visa began a proactive approach to payment security by announcing the Cardholder Information Security Program (CISP) as a standard for securing Visa cardholder data. Effective since June 2001, CISP compliance has been required for all entities that store, process or transmit Visa cardholder data.

Today the program has been embraced and expanded upon by other card associations and a new PCI standard defined to ensure multiple association compliance.

This document is designed to provide Main Street customers with instructions, notes and pointers on how to implement Monetra (v5.1.x and higher) into a CISP/PCI compliant system.

1.2 Visa CISP Overview

When customers offer their bankcard at the point of sale, over the Internet, on the phone, or through the mail, they want assurance that their account information is safe. That is why USA Visa has instituted the **Cardholder Information Security Program (CISP)**. Mandated since June 2001, the program is intended to protect Visa cardholder data—wherever it resides—ensuring that members, merchants and service providers maintain the highest information security standard.

1.3 The PCI Industry Standard

To achieve compliance with CISP, merchants and service providers must adhere to the Payment Card Industry (PCI) Data Security Standard, which offers a single approach to safeguarding sensitive data for all card brands. This Standard is a result of a collaboration between Visa and MasterCard and is designed to create common industry security requirements, incorporating the CISP requirements. Other card companies operating in the U.S. have also endorsed the PCI Data Security Standard within their respective programs.

1.4 Payment Application Best Practices (PABP)

Developed by Visa, the *Payment Application Best Practices* is a set of requirements and guidelines designed to address security and the risks associated when **full magnetic stripe data or CVV2 values are stored after authorization** by payment applications.

The Best Practices assist software vendors in creating secure payment applications that help ensure merchant CISP/PCI compliance.

Note: Currently, Visa only encourages, but does not require an application to provide for all 13 Best Practices. An application may be recognized on the approved list by not storing sensitive data and providing data encryption while reporting on which Best Practices are required to be handled by the user or integrator. Please check with your vendor for details (when evaluating in house or outsourced payment solutions) to

identify the roles and Best Practice responsibilities of your organization.

1.5 Payment Security Chain of Command

While the main job of systems security lies with the corporate CIO and/or the IT department, when implementing a payment system that will store, process, or transmit cardholder data, association rules will apply to the system. Much like the HIPAA initiatives within the health industry, the payment requirements such as CISP/PCI strictly deal with securing sensitive/private data. Below is a list of the formal Payment Security chain of command, with respective roles discussed.

1.5.1 Card associations

These are the actual associations like Visa and Mastercard who define and ultimately enforce the rules.

As quoted from Visa's website: " The Visa USA Operating Regulations govern the activities of member financial institutions and, by extension, merchants and service providers as participants in the Visa payment system. The simplified requirements (located at http://www.visa.com/cisp/) should help clarify the intent of the more formal regulations."

1.5.2 Acquirers

These are the merchant Acquiring banks that underwrite merchant accounts onto the respective payment networks. This is either done in-house or via a network of Merchant Service Providers who act as agents/members for the Acquiring bank.

As quoted from Visa's website: "Members are responsible for ensuring the CISP compliance of their merchants, service providers, and their merchants' service providers. Although there may not be a direct contractual relationship between merchant service providers and acquiring members, all members remain responsible for any liability that may occur as a result of CISP non-compliance. Acquirers must include a CISP compliance provision in all contracts with merchants and non-member agents."

1.5.3 Merchants

The end company receiving payment for goods and/or services. While the Acquirer or Merchant service provider should have explained (and be actively working you through) CISP/PCI, it is ultimately the Merchant's responsibility for verification and compliance.

1.5.4 Merchant Applications

As defined in the visa PABP, Software vendors are the developers of applications specifically for credit card transactions. Examples are point-of-sale (POS) and shopping cart products.

Note: Monetra is a purpose-built payment application, designed to integrate into a POS or online system to provide for the core payment functions inside the CISP/PCI environment.

2 Merchant Requirements for Compliance

Depending on annual transaction volume, CISP requirements range from completing a self-assessment questionnaire to engaging an independent security assessor for conducting annual on-site security audits. See www.visa.com/cisp and contact your bank, processor, or acquirer for more information.

Notes on fines: As quoted from Visa's website "If a merchant or service provider does not comply with the security requirements or fails to rectify a security issue, Visa may:

- Fine the acquiring member
- Impose restrictions on the merchant or its agent
- Permanently prohibit the merchant or its agent from participating in Visa programs

Members receive protection from fines for merchants or service providers that have been compromised but found to be CISP-compliant at the time of the security breach. Members are subject to fines up to **$500,000 per incident** for any merchant or service provider that is compromised and not CISP-compliant at the time of the incident."

Note: The CISP requirements for your systems do not change, and must be validated, no matter if you use an in-house product like Monetra, or a Visa approved online service provider such as Verisign®.

For example, the requirement for unique usernames and strong passwords does not change and is even a missing feature on some of the CISP listed Internet gateways. Before being validated, you must ask your staff if the entire system is conforming to the requirements or just the service provider themselves.

3 Monetra Security Validation (PABP)

Since its inception, Main Street has incorporated leading edge security practices and technologies directly into its software offerings.

The following sections outline the validation used against Monetra. It also outlines configuration and developer notes associated with secure implementation as defined by the Visa *Payment Application Best Practices*.

Note: This document originally referenced the Visa PABP document v1.1 as posted on their website date: June 01 2005. Version 1.3 of this guide (Secure Implementation) now references the Visa PABP document version 1.3 released May 08, 2006

3.1 Do Not Store Magnetic Stripe, CVVS/CVC2 or PinBlock (PVV) Data

One of the main goals of CISP/PCI/PABP is to prevent the risks associated when full magnetic stripe data or CVV2 values are stored **after authorization by payment applications**. Monetra does not store Magnetic stripe, CVV2 (security codes) or PinBlock(PVV) data post-authorization anywhere within the application.

PABP references (1.1, 1.1.1, 1.1.2, 1.1.3)

Validated Architecture:

- ✔ Incoming transaction data: Once Monetra receives sensitive authentication data, it is forwarded to the EFT processor and erased. See Integrator notes below.
- ✔ Transaction logs: Monetra does not store sensitive authentication data in transaction logs.
- ✔ History files: Monetra does not store sensitive authentication data in history files.
- ✔ Debug logs: Under production settings, Monetra does not output sensitive authentication data in debug logs. See configuration and integrator notes.
- ✔ Audit logs: Monetra does not store sensitive authentication data in audit logs.
- ✔ Database schema's and tables: Monetra does not store sensitive authentication data inside the database.

Configuration Notes:

- The monetra.log (debug) output level is configurable. To remain compliant with PCI, this must be set to 2 (or lower) in a production system.

Integrator Notes:

- The integrator is responsible for securely passing all sensitive authentication data (i.e. magstripe/CV/pin block) to the Monetra application and receiving and destroying any sensitive information returned (i.e. account numbers, exp. dates) by the Monetra application.

  When integrating an application with Monetra, there are several communication methods that can be used with both secure (over public transport) and non-secure (over private transport) to choose from. Currently Monetra provides both Key/Value pair and XML based protocols that can be communicated to the Monetra engine via TCP/IP, SSL, HTTP/HTTPS and Drop-File.

  When you are communicating with Monetra from over a public network (i.e. the Internet), you should always enable and use a secured connection either via our built in SSL or HTTPS facilities or a secure link such as VPN.

  NOTE REGARDING DROP FILE INTEGRATIONS: The use of drop file communications with the Monetra Engine should be considered the least secure method. While drop files provide an easy way for quick administration, legacy application integration and system testing, the use of this method should be limited to testing and should only be used in a production environment in which you cannot use one of the more secure methods. Developers should highly consider upgrading any drop file integration to an IP based method, and all new integrations should be considering either IP or HTTP (dependent on

protocol/toolkits used).

Note: Please review section 7 “Alternate file system security”.

If you use drop files, the following requirements will apply:

1. When a file is written into the designated *trans* directory, the Monetra application will remove it within a configurable amount of time. When the response file is written back, it is the application's responsibility to destroy or encrypt that file.
2. Care should be taken to apply proper security via permissions on any shared folder/directory. For example, only give the Monetra User and the Integrated Application read/write permissions on the folder.

3.2 Stored Data Protection

PABP reference (2.1)

Validated Architecture:

- ✔ **Mask displayed account numbers:** Monetra provides the ability to mask transaction report data based on the security level of the application user. For example, the account manager might need access to pre-settlement data, whereas a day-to-day clerk would not.

PABP reference (2.2)

Validated Architecture:

- ✔ **Encrypt stored sensitive data:** Monetra's encryption policies happen at the application level and are user-configurable. Both the encryption algorithm (cipher) used, as well as a key length(strength) may be specified. Currently, Monetra can utilize Blowfish, AES, RC4, RC5, and CAST5 ciphers. The allowable size (strength) of the encryption key varies depending on the capabilities of the cipher itself, ranging between 8 bits and 2048 bits. The recommended *minimum key length* for all algorithms is 128 bits, the recommended length is 256 bits. The default algorithm is AES, with a default key length of 256 bits.

  Note: As of version 5.3.1, Monetra has the ability to run all cryptographic modules and procedures in a FIPS-140-2 compliant manner. See http://csrc.nist.gov/cryptval/ for more information regarding FIPS 140 (*Security Requirements for Cryptographic Modules*)

  Note: see configuration notes.

PABP reference (2.3)

Validated Architecture:

- ✔ **Protect encryption keys:** One of the most important aspects of any cryptographic system is the creation, usage, maintenance and support of encryption keys within applications. When Monetra starts up, a series of events are triggered for secure cryptographic key support.
  - ➢ Information is retrieved from the Monetra configuration file (main.conf) for an encryption cipher, key length and key file location.
  - ➢ The key file is read into memory and decrypted using the private key, which is stored within the Monetra executable, which itself is encrypted by a symmetric key generated internally.
  - ➢ Once the database encryption key itself is decrypted, a key length is checked to ensure it matches the length specified in the configuration.
  - ➢ Finally, Monetra attempts to decrypt a single known-value in the database, utilizing the expected good key in order to verify its accuracy prior to startup.

PABP reference (2.4)

Validated Architecture:

- ✔ **Implement key management processes and procedures:** Monetra's encryption policies happen at the application level and are user-configurable. Both the encryption

algorithm (cipher) used, as well as a key length(strength) may be specified. Currently, Monetra can utilize Blowfish, AES, RC4, RC5, and CAST5 ciphers. The allowable key length (strength) is dependent on the Cipher used.

Configuration Notes:

- Please review the information contained within this guide and associated documentation for information on creating and maintaining strong encryption keys.

Integrator Notes:

- It is the integrator's responsibility to ensure all initial encryption keys are created and ongoing encryption key maintenance is supported.

3.3 Provide Secure Password Features

PABP reference (3.1)

Validated Architecture:

- ✔ **Require unique/strong usernames and passwords for administrative access:** Monetra provides multi-level username and password facilities for both administrative and general application access.

PABP reference (3.2)

Require a unique username and complex password for access to PC's, Servers, and databases where payment applications reside.

Note: Monetra is provided as a single stand-alone software application. Requirement 3.2 should be implemented and enforced at the systems admin level.

PABP reference (3.3)

Validated Architecture:

- ✔ **Encrypt application passwords:** Monetra uses strong encryption to store application passwords. See PABP ref 2.2 above for more details.

PABP reference (3.4)

Validated Architecture:

- ✔ **Allow complex passwords:** Monetra provides for strong usernames and passwords across the virtual user subsystem. Monetra is considered a 'stateless' application that must be supplied a username and password with every transaction. Monetra can mandate/regulate the use of strong passwords as defined in the PCI standard 8.5.8 – 8.5.15 .

Configuration Notes:

- Strong password management is configurable and can be handled at the Monetra level, or integrated at the application level. If you disable this feature in Monetra, the developer must provide strong password administration inside the integrated application. Please review the configuration guide for information on creating and maintaining strong password management.

Integrator Notes:

- Monetra was designed from the start as a 'stateless' (non persistent/near real-time) application. In short, all requests (functions) into and out of Monetra must be verified (via approved security policy) and should never be considered to act in a persistent (sessioned) state.

3.4 Log Application Activity

PABP reference (4.1)

Validated Architecture:

- ✔ **Log access by individual users:** Monetra provides extensive logging for security audits at both the internal and external level. External and internal Security logging is set to ON by default.

PABP reference (4.2)

Validated Architecture:

- ✔ **Implement an automated audit trail:** Monetra provides extensive logging both at the connection and transaction level. Security logging is set to ON by default.

Configuration Notes:

- Security logging is a configurable parameter. These features must be enabled for PCI compliance and are set to ON by default. Please reference the Monetra configuration guide for more information.

Integrator Notes:

- The integrator is responsible for logging and audit trails outside of the Monetra payment application. Example: A POS developer must log all activity inside the order entry application. Once the POS application sends Monetra a transaction, it is logged from that connection level forward.

3.5 Develop Secure Applications

PABP reference (5.1)

Develop web (Internet-based) software and web applications based on secure coding guidelines.

Note: Monetra is provided as a single stand-alone ANSI C software application running as a Daemon or Service process. Web application developers integrate their applications with Monetra. The web integrator is therefore responsible for the particular OWASP guidelines. Monetra does provide extensive security features to assist in Complaint integrations such as direct SSL socket communications, connection level certificates and an application level firewall. See the Monetra configuration guide and associated API guides for specific instructions on using these security technologies.

PABP reference (5.2)

Develop software applications based on industry Best Practices and include information security throughout the software development life cycle.

Note: Main Street software products provide some of the most advanced security features available on the market. All of our flagship products take full (native) advantage of the latest operating platforms. Examples: Monetra is the only product to run NATIVE (i.e. no Java or special runtime libraries required) on Linux, FreeBSD, IBM AIX, Sun Solaris, SCO Unix, Mac OSX and Microsoft Windows.

PABP reference (5.2.1)

Validated Architecture:

- ✔ **All changes/patches are tested via QA:** Main Street tests all application changes internally via set QA procedures prior to releasing any code into a beta or production release.

PABP reference (5.2.2)

Validated Architecture:

- ✔ **Non essential application accounts, usernames and passwords are removed prior to release:** Monetra is a self contained application that will build a default (empty) data structure. See integrator notes below for more instructions.

PABP reference (5.2.3)

Validated Architecture:

- ✔ **Removal of unnecessary and insecure services, applications and protocols:** Monetra is designed and applied in ANSI C as a self-contained, fully multi-threaded application. Monetra provides a small security footprint and requires minimal external libraries to function properly (see CIR below) . Example: Monetra does not require you to install vulnerable applications like Internet Explorer or Internet Information Server(IIS) to operate on the windows platform. We also do not require any special runtimes such as Java, which can add to the security footprint.

Current Internal Requirements: Monetra relies only on 3 primary components: OpenSSL (http://www.openssl.org), Zlib (http://www.gzip.org/zlib/) and the system-specific C library (including threading and math components) provided by the manufacturer of the OS on which Monetra resides. Both OpenSSL and Zlib are statically linked, so the version residing on the system does not inhibit system security. Other specific add-on modules may make use of other libraries, especially those which require communication with external SQL databases.

PABP reference (5.2.4)

Validated Architecture:

- ✔ **Custom code review:** As per documented coding policy and procedures, all software developed by Main Street complies with industry best practices and standards.

Integrator Notes:

- The integrator is responsible developing external applications that adhere to the standards as outlined in PABP ref 5.2. If the integrator first installs Monetra into a test environment and then deploys to a live environment, the integrator will be responsible for the removal of all non-essential application accounts as per PCI standard 6.3.5 and 6.3.6

3.6 Protect Wireless Transmissions

PABP references (6.1, 6.2)

Monetra provides several methods for securing communications. Many network topologies including wireless may take advantage of the provided features. Please refer to the section on secure remote application access (PABP ref 2.11 below) for more details.

Configuration Notes:

- The Network Administrator should provide security policy and settings when related to wireless technologies (such as access points, bridges and routers).

Integrator Notes:

- When integrated into a wireless environment, Main Street recommends the use of SSL connectivity. See configuration guide for more details.

3.7 Test for Application Vulnerabilities

PABP reference (7.1)

Validated Architecture:

- ✔ **Test for application vulnerabilities: Removal of unnecessary and insecure services, applications and protocols:** Main Street has been validated to comply with known development processes including, but not limited to:
  - Monitoring and using information from outside security sources for vulnerability assessment and policy.
  - Testing Monetra against new 'identified' vulnerabilities.
  - The ability to timely develop, test and deploy a patch for the primary application security, within a known chain of trust.

Configuration Notes:

- The system administrator must insure any application (service or software) is implemented into, updated and tested within the target environment as accepted policy dictates.

Integrator Notes:

- The integrator is responsible for testing the associated network applications for vulnerabilities outside of the Monetra payment application. Example: A POS integrator must ensure the operating system from which Monetra is executing has the approved security updates/patches in place.

3.8 Facilitate Secure Network Implementations

PABP reference (8.1)

Validated Architecture:

- ✔ **Facilitate secure network implementations:** Monetra provides the following tools for secure network deployment.
  - Direct SSL socket connection method.
  - Direct XML HTTPS connection method.
  - Digital certificate validation (per connection).
  - Digital certificate validation (per merchant/user).
  - Internal firewall for defined (extended) application access validation.

Configuration Notes:

- Monetra connection parameters are a configurable option. Please reference the most recent Monetra configuration guide for more information.

3.9 Never Store Cardholder Data on a Public-Facing Internet Connection

PABP reference (9.1)

Validated Architecture:

- ✔ **Provide payment applications and data separation facilities:** Monetra provides the ability to be fully separated from both the application and the database. For example Monetra does not require the client (requesting POS) application or the database (required for storage and parameters) to be located on the same system as the payment server.

Configuration Notes:

- The Monetra database (used for system logs and parameter storage) is a configurable parameter. Please reference the most recent Monetra configuration guide for more information. To remain compliant with PCI standards, the cardholder data must never reside on Internet-accessible systems (DMZ). The installation must be configured such that Monetra and the database reside on secure(firewalled) network segments (inside the DMZ [De-Militarized-Zone]).

Integrator Notes:

- The integrator is responsible for proper Monetra network configuration, including secure communication channels to and from the Monetra application data storage mechanism(s).

3.10 Facilitate Secure Remote Software Updates

PABP reference (10.1)

Validated Architecture:

- ✔ **Provide Secure software updates:** Monetra supports 'live update' features via a “subscribe/pull” method, where instead of Main Street needing access into a customer system, the customer simply downloads a new release from a known Main Street Server via a provided update utility.

  Example: Very much like the anti-virus software update utilities in use today, Main Street provides an installer/updater utility for simplified (push button) software updates.

Integrator Notes:

- The integrator is responsible for securely implementing the required system update and maintenance policy regarding the Monetra software within a validated system. If the production system uses VPN and/or broad band “always-on” connections then it is recommended you use a personal firewall product as per PCI ref 1.3.10

3.11 Facilitate Secure Remote Application Access

PABP reference (11.1)

Secure remote administration may be required from time to time by the merchant and the use of SSL or HTTPS communications should be used.

Note: Main Street Softworks makes it a policy to NEVER connect into remote systems for any reason (installation, configuration upgrades etc.).

Validated Architecture:

- ✔ **Provide Secure remote application access:** Monetra provides a native SSL socket connection that can also authenticate against a user certificate. Please reference the appropriate documentation regarding certificate procedures.
  Note: Additionally, Monetra provides an internal IP ruleset (firewall) to protect against unauthorized application access.

Configuration Notes:

- All Monetra connection methods are configurable. Please reference the most current Monetra Configuration Guide for more details.

Integrator Notes:

- The integrator is responsible for the secure configuration of any Monetra application, in regards to remote access. When customers access Monetra remotely, it is a requirement to use two factor authentication.

  Two factor examples:
  1. App-Username/Password, SSL with certificate(s).
  2. App-Username/Password, VPN with certificate(s).
  3. User-Radius, tokens.

3.12 Encrypt Traffic Over Public Networks

PABP reference (12.1, 12.2)

Validated Architecture:

- ✔ **Provide Strong encryption for data transmission over public networks:** Monetra provides built-in SSL connectivity methods, that are approved for use across the public Internet.
- ✔ **Never communicate sensitive data via unencrypted e-mail:** Monetra does not communicate any sensitive data via e-mail.

Configuration Notes:

- All Monetra connection methods are configurable. Please reference the most current Monetra Configuration Guide for more details on PCI compliant settings.

Integrator Notes:

- The integrator is responsible for securely configuring the Monetra application to communicate securely over a public network.

3.13 Encrypt All Non-Console Administrative Access

PABP reference (13.1)

If you are not using a secure console (such as ssh or putty) for remote Administrative access, you must use either the Monetra provided SSL or HTTPS facilities, or an externally configured secure channel (such as VPN) for application access.

Validated Architecture:

- ✔ **Encrypt all non-console administrative access:** Monetra allows non-console application access to administrative features via SSL connection methods.

Configuration Notes:

- All Monetra connection methods are configurable. Please reference the most current Monetra configuration guide for more details on PCI compliant settings.

Integrator Notes:

- The integrator is responsible for configuring the Monetra application to communicate securely while providing administrative access.

4 Monetra Virtual User SubSystem

4.1 Introduction

The Monetra virtual subsystem has been deployed for years to administer, secure and report within the payment application. The base system works as follows.

Note: On a properly licensed Monetra server, an unlimited number of MONETRA USER and SUBUSER accounts may be installed and supported, dependent on production hardware platform and available resources.

A Virtual User is defined by a unique identifier (username) and password combination. When setting user passwords, please use the following guidelines:

- Passwords should be at least 7 alpha-numeric characters long (and include both characters (such as the *) and letters).
- Passwords should be changed every 90 days.
- Passwords should not be repeated/re-used in up to 4 changes.

4.1.1 Administrative User(s)

The main application administrative account has master privileges for administering any and all user/subuser accounts within the system. This user is often referred to as MADMIN.

EXAMPLE:

1. Upon a fresh install, the system administrator will log into the Monetra server using username=MADMIN and password=password.
2. Once connected, the Administrator should change the MADMIN password to something strong(see note above).
3. At this point the first system user (merchant profile) can be added. Lets say we add an account for Jane that connects to vital and we call this user 'JaneVital' and give her a password of 'test-123'.

4.1.2 System User(s)

The Monetra Administrative account has the power to add a user account. (i.e. This represents any single MID/TID/TERMINAL combination routing to any number of processors, on a per industry profile). Each user account is responsible for managing its own subusers.

EXAMPLE:

1. The master user account for Jane was created by MADMIN and is called 'JaneVital'.
2. JaneVital has the administrative role for the JaneVital account, and all subusers.

4.1.3 System SubUser(s)

These are administered via the master user account to create a non-privileged sub user account.

EXAMPLE:

1. JaneVital logs in and creates a sub-user called 'Jim' that only has access to the 'Sale' function for the JaneVital master account. Jim gets a password of 'test-9876'
2. When Jim logs into monetra with [username=JaneVital:Jim password=test-9876] he can only run a sale transaction.

5 Monetra Encryption Keys

5.1 Introduction to Encryption Keys

As associated with PABP ref(2.2), Monetra provides facilities for strong application level encryption. One of the most important aspects of this system is the creation, distribution and maintenance of the encryption keying infrastructure.

While the following reference outlines the basic steps for key administration, we recommend you refer to the most current Monetra Configuration Guide.

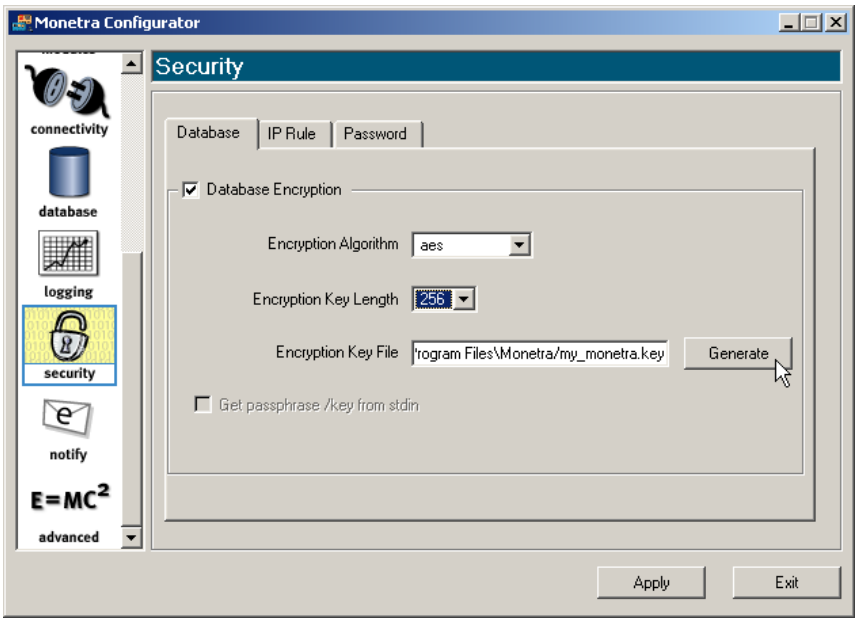
5.1.1 Encryption Key Creation

Depending on the base operating platform from which Monetra is deployed, the steps may vary. Please reference the most recent Installation and Configuration guides for more details on how to use Monetra key generation tools.

On the Linux operating system, an encryption key might be created/generated for use with Monetra as follows.

To generate a key, use the following shell command:
monetra_keygen keylen outfile [egd pool]
Example:
$ /usr/local/monetra/bin/monetra_keygen 256 /usr/local/monetra/my_monetra.key

Note: On Microsoft Windows or Mac OS X operating systems, it is recommended to use the Configuration utility to build all encryption keys (see image below).



5.1.2 Encryption Key Maintenance

As with most security systems and policies, it is required to change out encryption keys from time to time or as often as needed. Please review the most recent Installation and Configuration guides for more details.

Below is an example of the steps required to replace an encryption key on the Linux Operating system:
1. Settle all transactions.
2. Export your current Monetra datafile.
3. Stop Monetra.
4. Remove the contents of your data directory.
5. Build a new Monetra encryption key.
6. Re-start Monetra
7. Import your data.

Note: When you upgrade or re-install any Monetra engine via the Monetra Installer Utility and it performs an export/import, the encryption key will be replaced as part of the process.

6 Monetra Client Certificates

6.1 Introduction to Client Certificates

In order to address issues of secure client verification, Monetra has the ability to only allow secure SSL and XML_HTTPS connections from authorized clients. By utilizing a Certificate Authority and client-side SSL certificates, only clients who present an SSL certificate signed by a CA that the administrator has configured Monetra to recognize, are allowed to connect. This document describes the actions required to set up a simple Certificate Authority using OpenSSL and the steps required to integrate the Certificate Authority's signing certificate and client-side SSL certificate restrictions into Monetra.

Due to the large number of possible configurations and implementations, this guide only describes the minimal steps required to enable client-side certificates. It is up to the administrator to ensure that file permissions, locations, security, etc. are implemented and controlled according to local security guidelines.

6.1.1 Certificate Authority Setup and Use

Create the CA base directory:
$ mkdir ~/myCA

Copy CA.pl from your OpenSSL distribution into the CA base directory:
$ cd ~/myCA
$ cp .../CA.pl .

Edit the CA.pl script, ensuring the basic settings are correct:
- verify number of days certificate is valid ($DAYS)
- set CA base directory ($CATOP) relative to the location of CA.pl
- fix dirmode as appropriate ($DIRMODE), 0777 is *not* safe

Edit the system openssl.cnf and change 'dir' to match $CATOP

Create the CA layout. Enter a CA private key pass phrase when prompted and fill in the relevant information for your use:

```
$ ./CA.pl -newca
 A certificate filename (or enter to create)
 <press ENTER here>
 Making CA certificate ...
 Generating a 1024 bit RSA private key
 ................++++++
 ..++++++
 writing new private key to 'CA/private/cakey.pem'
 Enter PEM pass phrase:
 Verifying - Enter PEM pass phrase:
 -----
 You are about to be asked to enter information that will be incorporated
 into your certificate request.
 What you are about to enter is what is called a Distinguished Name or a DN.
 There are quite a few fields, but you can leave some blank
 For some fields, there will be a default value,
 If you enter '.', the field will be left blank.
 -----
 Country Name (2 letter code) [AU]:US
 State or Province Name (full name) [Some-State]:Florida
 Locality Name (ie. city) []:Gainesville
```

```
  Organization Name (eg, company) [Internet Widgits Pty Ltd]:Main Street                Softworks, Inc.
  Organizational Unit Name (ie. section) []:
  Common Name (ie. YOUR name) []:
  Email Address []:support@monetra.com
$
```

Fix permissions on the generated files as required (files are generated using the user's umask):

$ chmod -R o-rx .

Verify that the Certificate Authority layout includes both the Certificate Authority Cert and private key:

```
$ find . | sort
  .
  ./CA
  ./CA.pl
  ./CA/cacert.pem   <-- CA Cert
  ./CA/certs
  ./CA/crl
  ./CA/index.txt
  ./CA/newcerts
  ./CA/private
  ./CA/private/cakey.pem  <-- CA private key
  ./CA/serial
```

cacert.pem should look like:

```
-----BEGIN CERTIFICATE-----
MIIDYzCCAsygAwIBAgIJAKcV7BfxXP68MA0GCSqGSIb3DQEBBAUAMH8xCzAJBgNVBAYTAlVTMRAwDg
YDVQQIEwdGbG9yaWRhMRQwEgYDVQQHEwtHYWluZXN2aWxsZTEkMCIGA1UEChMbTWFpbiBTdHJlZ
XQgU29mdHdvcmtzLCBJbmMuMSIwIAYJKoZIhvcNAQkBFhNzdXBwb3J0QG1vbmV0cmEuY29tMB4XDT
A1MDYwOTEzNTMyMVoXDTA2MDYwOTEzNTMyMVowfzELMAkGA1UEBhMCVVMxEDAOBgNVBAgTB0
Zsb3JpZGExFDASBgNVBAcTC0dhaW5lc3ZpbGxlMSQwIgYDVQQKExtNYWluIFN0cmVldCBTb2Z0d29ya3
MsEluYy4xIjAgBgkqhkiG9w0BCQEWE3N1cHBvcnRAbW9uZXRyYS5jb20wgZ8wDQYJKoZIhvcNAQEBBQA
DgY0AMIGJAoGBAL1xNVuiUpYu4lRxWVdcAv/Fd1JVujLhHAgMo1BA1w8br2QJ7++eK7BXEU3mCN8jmu
Uyn2R/nE2U87X+RdY0KH9FQ7Abfj1b2vOYcmVHBvgm8FGBeueew8900M3xi/Gwz8AINOFCEg/+/0TuLk
CUDg4RFuJDOyZQaGHzMSeGreZrAgMBAAGjgeYwgeMwHQYDVR0OBBYEFCGABWx3AnXYZysdlhtL/5q
BZxDbMIGzKIjblieJRLILJDlivliEJfIDlfildIFJLIdfhqBZxDboYGEpIGBMH8xCzAJBgNVBAYTAlVTMRAwDg
YDVQQIEwdGbG9yaWRhMRQwEgYDVQQHEwtHYWluZXN2aWxsZTEkMCIGA1UEChMbTWFpbiBTdHJlZ
XQgU29mdHdvcmtzLCBJbmMuMSIwIAYJKoZIhvcNAQkBFhNzdXBwb3J0QG1vbmV0cmEuY29tggkApxXs
F/Fc/rwwDAYDVR0TBAUwAwEB/zANBgkqhkiG9w0BAQQFAAOBgQBVwrhdYFnkoISAwivmKbs6Mf7Ogm
vTm5Cg+hjG7VBwPbJf991tsxqirbEd9W3tqGJ58RKJchnstrcihpfdMptFfKDaNDawG6xWiHUYvmwzkaAI+ci
XBm/DpvDZvm9A09SCOHz+aJNILazyhz9q38MAtK04vwT7o5Cac/LCh5Yr6w==
-----END CERTIFICATE-----
```

and private/cakey.pem should look like:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,2505C24E04098DEE

z1eWv6x/NNmVibFRUvi3G0bbLwv+RfbEPLAqNDzeHNS4Zm5ksTwkxaNNfg+X5wVjhchXuq4T0yt5USWk3
fCU2Dx+2vGm4j/rkEQJDUGGnRihoi5ZyrHi68fJOrO4G1xvCf37IMSb65LGcNKWtaEFo2YV5SnJfYtJU1vAU
h4LV4Zt5ZT6CWGTFcxVzmtEWaCjpDEvKliclbjLDIFj4idfldDO1xAoAaesVsD1V+KsxrSdqwek5T1TzB9478Tt
```

bodfRRzRlsE6bFVNDkdQFv3im/ZgM4rFhAHihRffQy7mW6XGkg19T1JbcZfIHXGUrzOi3iOb6mBJVANZcjZ
dwnbOsrbWcUPBEXgHcQUrsce7qhzx5LaEQUowEJHAhERQ5GxGRAmeHz8MeyJklolnH5chVkmEAKBRr
WMi4EsPHdfgXVyEG9uJ2StbHDuh7N6fkmCGXwrblbkpf1M6/j/i31e58FVOB616Oufa1EaSDzeaCQzm/uWII
iCz0RsxXY/hoGOZ9miyQjjZru0OiVFf8QNsF2MpD2uVraWSyYTcGgF0x8hKXvlcnsi3aNlzzMT+iGxOOEWdI
bFsIfD9YqQ35h9BAkn7weJQfDAlD2K6noktgBmfcYJlUqGDt46/a990nKek6jQH02rPngZMRNaO/97VgNbgi
Z1zkBq2p83FY397IhuSVRLqq5wW7S4dXNyu7J3+tIqN5LiUdePGqyix7ltOovFqmV5cWnKVquK02Vc/Y8KN
ZNFBREC4WT2m8663DThC2ocQrmleHcLP7YoGCWp4EAmBpxHrU0FQYCTX7tIXO5KKLRPjenrPQ==
-----END RSA PRIVATE KEY-----

6.1.2 Restricting SSL Connections with Certificates

Copy your CA's certificate (cacert.pem) to a location that Monetra can access:

$ cp CA/cacert.pem /etc/monetra/mycafile.pem

Edit prefs.conf, set:

enableSSL=yes
SSLCertRequired=yes
SSLCAFile=/etc/monetra/mycafile.pem

and restart Monetra.

Note: you will need to ensure you have a server-side SSL cert/key to allow SSL connections. This may be a certificate generated with the newly-created CA or an existing cert/key combination.

You can verify that these settings are being used by checking monetra.log and locating these lines:

SSLCAFile: /etc/monetra/mycafile.pem
SSLCertRequired: yes

At this point, only clients with valid SSL certificates signed by this certificate authority key are granted further access.

To create a signed client certificate, you must first create a certificate request. Using OpenSSL:

$ openssl req -new -nodes -out newreq.pem
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to 'privkey.pem'

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Florida
Locality Name (eg, city) []:Gainesville
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Main Street Softworks, Inc.
Organizational Unit Name (eg, section) []:Testing
Common Name (eg, YOUR name) []:testbox.monetra.com
Email Address []:support@monetra.com

Please enter the following 'extra' attributes
to be sent with your certificate request:

```
 A challenge password []:
 An optional company name []:
$
```

At this point, the Certificate Authority must sign the certificate:

```
 $ ./CA.pl -sign
 Using configuration from /etc/ssl/openssl.cnf
 Enter pass phrase for ./CA/private/cakey.pem:
 DEBUG[load_index]: unique_subject = "yes"
 Check that the request matches the signature
 Signature ok
 Certificate Details:
        Serial Number:
            a7:15:ec:17:f1:5c:fe:bf
        Validity
            Not Before: Jun 10 15:37:32 2005 GMT
            Not After : Jun 10 15:37:32 2006 GMT
        Subject:
            countryName             = US
            stateOrProvinceName     = Florida
            localityName            = Gainesville
            organizationName        = Main Street Softworks, Inc.
            organizationalUnitName  = Testing
            commonName              = testbox.monetra.com
            emailAddress            = support@monetra.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                AB:72:09:4E:22:65:8E:6F:79:CA:9A:AD:3E:C4:20:05:4C:8E:99:B0
            X509v3 Authority Key Identifier:
                keyid:21:80:05:6C:77:02:75:D8:67:2B:1D:96:1B:4B:FF:9A:81:67:10:DB
                DirName:/C=US/ST=Florida/L=Gainesville/O=Main Street Softworks,
Inc./emailAddress=support@monetra.com
                serial:A7:15:EC:17:F1:5C:FE:BC

 Certificate is to be certified until Jun 10 15:37:32 2006 GMT (365 days)
 Sign the certificate? [y/n]:y


 1 out of 1 certificate requests certified, commit? [y/n]y
 Write out database with 1 new entry
 Data Base Updated
 Signed certificate is in newcert.pem
$
```

The signed client certificate is now located in newcert.pem, and the private key is in privkey.pem.

On the client side, applications developed using the libmonetra API can use the M_SetSSL_CAfile() and M_SetSSL_Files() functions to load the CA's certificate along with the client private key and signed client certificate into the client application after calling M_SetSSL() but before calling M_Connect().

6.1.3 Restricting User Access with Certificates

The administrator can add per-user restrictions on which client certificate(s) are allowed to execute transactions. Monetra uses a cryptographically-secure digest (hash) of the client certificate to identify

individual client certificates.

The client certificate digest can be generated using the M_SSLCert_gen_hash(), giving the filename as the first argument. Alternatively, the digest can be generated using the X509_digest() digest in OpenSSL with the SHA1 digest algorithm or can be generated using the openssl(1) application:

```
$ openssl x509 -sha1 -in newcert.pem -noout -fingerprint
  fingerprint: 96:f8:ac:6b:76:8b:d5:f3:5f:bb:2d:0c:4e:9d:19:c4:b4:49:ad:36
$
```

To add a client certificate restriction to Monetra, a transaction such as this would be used:

```
username=madmin
password=password
action=admin
admin=restriction
restriction=add
restriction_user=loopr
restriction_type=ssl_cert
restriction_data=96:f8:ac:6b:76:8b:d5:f3:5f:bb:2d:0c:4e:9d:19:c4:b4:49:ad:36
```

To retrieve the client restrictions for a particular user, a transaction such as this would be used:

```
username=madmin
password=password
action=admin
admin=restriction
restriction=list
restriction_user=loopr
```

To remove a client restriction:

```
username=madmin
password=password
action=admin
admin=restriction
restriction=remove
restriction_num=1
```

7 Alternate file system security

7.1 Notes on File System Security, as applied to PABP

Since the earliest days of Unix, DOS and most versions of Windows, payment applications have relied heavily on on inter-application communication via textual based files that are written and then read from a computers hard disk. While simple and easy to implement, these legacy integrations (drop-files) have higher security implications.

The first risk identified is the ability of an attacker to gain root control over a machine and scan the shared directory for incoming and outgoing files. This risk must be mitigated via operating system and file system security measures.

The second risk identified is when a physical disk is examined outside of the operating system. Examples would be removing a hard drive and performing a forensic analysis, or booting the computer with a CDROM that contains a base OS and forensic tools. In theory an attacker or even a technical ebay shopper could peruse old files that were intentionally deleted on the physical disk.

Since payment applications can communicate sensitive data such as card numbers and CVV2 values, it becomes imperative to look at if, how and why any "disk based" communication happen. If disk based files are written and contain sensitive data then we highly recommend you look at alternate security measures that help improve the security posture of your disk based communication systems.

Some of the more modern alternatives would be to deploy an additional layer of security around your /trans directory such as an encrypted file system or implementing a temporary/memory-resident file system.

7.2 Encrypted File Systems

The advantage of an encrypted file system is that when a file is written to disk, no matter the file system or operating system used, it should be considered safe form of protection since the process does not depend on the integrity of the operating system after the encryption takes place.

For more information on encrypted file systems on linux, please visit.
http://www.linuxjournal.com/article/6481

For more information on encrypted file systems for Microsoft Windows, please visit.
http://www.microsoft.com/technet/prodtechnol/winxppro/deploy/cryptfs.mspx

7.3 Temporary File Systems

The advantages of a temporary file system (or RAM disk) is that while it mimics a physical disk drive, it maintains all the processes in "volatile" system memory (RAM).

If the computer shuts down, the memory is reset, and thus erased.

Note: A nice side effect of using a temp/RAM disk are noticeable performance improvements across all operating platforms.

For more information on temporary file systems, please visit:
http://www.wikipedia.org/wiki/TMPFS

8 Operating System Information

8.1 PCI/PABP compliance statement

The current version of Monetra is developed and deployed to support Multiple Operating Systems including Linux, Unix, Mac OSX, Microsoft Windows and others.

As per current PCI requirements, a validated application must execute from a System that is supported by the manufacturer, to include up-to-date security related patches and enhancements. Reference PCI 6.1

If you are unsure of which operating systems are valid, please reference the current Operating System Manufacturers support page.

Example for Microsoft:
http://www.microsoft.com/windows/lifecycle/default.mspx

Examples of unsupported Operating Systems':
- Windows 98
- Windows 98-SE
- Windows ME
- RedHat Linux 6,7, 8, 9
- IBM AIX 4.3

As per vendor notices:
http://www.microsoft.com/windows/support/endofsupport.mspx

http://www.redhat.com/security/updates/eol/

9 Resources on the World Wide Web

9.1 Card Association Links

Visa:
http://www.visa.com/cisp

Master Card:
http://www.mastercard.com
http://www.mastercardmerchant.com/datasecurity/data_protection.html

American Express:
http://home.americanexpress.com/homepage/merchant_ne.shtml

9.2 Security Organizations

CERT: http://www.cert.org/

Security Authorities: Bureau of Industry and Security:
http://www.bis.doc.gov/

FBI CyberSecurity: http://www.fbi.gov/cyberinvest/cyberhome.htm

U.S. Secret Service: http://www.secretservice.gov/financial_crimes.shtml

INTERPOL: http://www.interpol.int/Public/TechnologyCrime/default.asp

9.3 Operating Systems

Linux: http://www.kernel.org/ http://www.nsa.gov/selinux/

FreeBSD: http://www.freebsd.org/security/

Apple Mac OS X: http://www.apple.com/macosx/features/security/

IBM AIX: http://www-1.ibm.com/servers/aix/overview/security.html

Sun Microsystems Solaris: http://www.sun.com/software/security/

SCO Unix: http://www.sco.com/support/security/

Microsoft Windows: http://www.microsoft.com/security/default.mspx

9.4 Monetra Related Technologies

SSL: http://www.openssl.org/
ZLIB: http://www.zlib.net/

GNU C: http://www.gnu.org/software/libc/libc.html

10 References, Acknowledgments, License

10.1 References

This document references the following publications.

1. Visa CIPS/PCI version 1.0
2. Visa PABP version 1.3
3. Monetra Installation and Configuration Guides

10.2 Acknowledgments

Main Street's software products actively use, support and promote and use the Open Source SSL toolkit located at http://www.openssl.org .