Ubersmith 3.x

PA-DSS 2.0 Implementation Guide

Version 1.3

April 18, 2016

Document Owners

Michael Styne

VP Operations

Ubersmith, Inc.

Table of Contents

Notice

THE INFORMATION IN THIS DOCUMENT IS FOR INFORMATIONAL PURPOSES ONLY. UBERSMITH, INC. MAKES NO REPRESENTATION OR WARRANTY AS TO THE ACCURACY OR THE COMPLETENESS OF THE INFORMATION CONTAINED HEREIN. YOU ACKNOWLEDGE AND AGREE THAT THIS INFORMATION IS PROVIDED TO YOU ON THE CONDITION THAT NEITHER Ubersmith, Inc. NOR ANY OF ITS AFFILIATES OR REPRESENTATIVES WILL HAVE ANY LIABILITY IN RESPECT OF, OR AS A RESULT OF, THE USE OF THIS INFORMATION. IN ADDITION, YOU ACKNOWLEDGE AND AGREE THAT YOU ARE SOLELY RESPONSIBLE FOR MAKING YOUR OWN DECISIONS BASED ON THE INFORMATION HEREIN.

Nothing herein shall be construed as limiting or reducing your obligations to comply with any applicable laws, regulations or industry standards relating to security or otherwise including, but not limited to, PA-DSS and DSS.

The retailer may undertake activities that may affect compliance. For this reason, Ubersmith, Inc. is required to be specific to only the standard software provided by it.

About this Document

This document describes the steps that must be followed in order for your Ubersmith installation to comply with Payment Application – Data Security Standards (PA-DSS). The information in this document is based on PCI Security Standards Council Payment Application Data Security Standards program (version 2.0 dated October, 2010).

Ubersmith, Inc. instructs and advises its customers to deploy Ubersmith, Inc. applications in a manner that adheres to the PCI Data Security Standard (v2.0). Subsequent to this, best practices and hardening methods, such as those referenced by the Center for Internet Security (CIS) and their various "Benchmarks", should be followed in order to enhance system logging, reduce the chance of intrusion and increase the ability to detect intrusion, as well as other general recommendations to secure networking environments. Such methods include, but are not limited to, enabling operating system auditing subsystems, system logging of individual servers to a centralized logging server, the disabling of infrequently-used or frequently vulnerable networking protocols and the implementation of certificate-based protocols for access to servers by users and vendors.

You must follow the steps outlined in this *Implementation Guide* in order for your Ubersmith installation to support your PCI DSS compliance efforts.

Revision Information

Name	Title	Date of Update	Summary of Changes
Michael Styne	VP Operations	11/19/2013	Document Creation
Michael Styne	VP Operations	08/21/2015	Updates for 3.3 Release
Michael Styne	VP Operations	04/15/2016	Updates for 3.4 Release
Michael Styne	VP Operations	04/18/2016	Updates for 3.5 Release

Note: This PA-DSS Implementation 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. Ubersmith, Inc. will distribute the IG to new customers via our website, http://www.ubersmith.com, and within the Ubersmith source distribution.

Executive Summary

Ubersmith version 3.5 has been PA-DSS (Payment Application Data Security Standard) certified, with PA-DSS Version 2.0. For the PA-DSS assessment, we worked with the following PCI SSC approved Payment Application Qualified Security Assessor (PAQSA):

Coalfire Systems, Inc. 361 Centennial Parkway Suite 150 Louisville, CO 80027	Coalfire Systems, Inc. 1633 Westlake Avenue N. Suite 100 Seattle, WA 98109

This document also explains the Payment Card Industry (PCI) initiative and the Payment Application Data Security Standard (PA-DSS) guidelines. The document then provides specific installation, configuration, and ongoing management best practices for using Ubersmith as a PA-DSS validated Application operating in a PCI Compliant environment.

PCI Security Standards Council Reference Documents

The following documents provide additional detail surrounding the PCI SSC and related security programs (PA-DSS, PCI DSS, etc):

- Payment Applications Data Security Standard (PA-DSS)
  https://www.pcisecuritystandards.org/security_standards/pa_dss.shtml
- Payment Card Industry Data Security Standard (PCI DSS)
  https://www.pcisecuritystandards.org/security_standards/pci_dss.shtml
- Open Web Application Security Project (OWASP)
  http://www.owasp.org

Application Summary

Payment Application Name:	Ubersmith
Payment Application Version:	3.5
Application Description:	Ubersmith is a recurring billing, customer support, and Data Center Infrastructure Management (DCIM) platform. An account with an acquiring bank or Payment Service Provider is required to take advantage of Ubersmith's recurring billing capabilities.
Application Target Clientele:	Internet Hosting providers who manage Data Center facilities
Components of Application Suite (i.e. POS, Back Office, etc.)	Within the Ubersmith source bz2ball, the the directory structure contains the following components: admin/ - Administrative interface

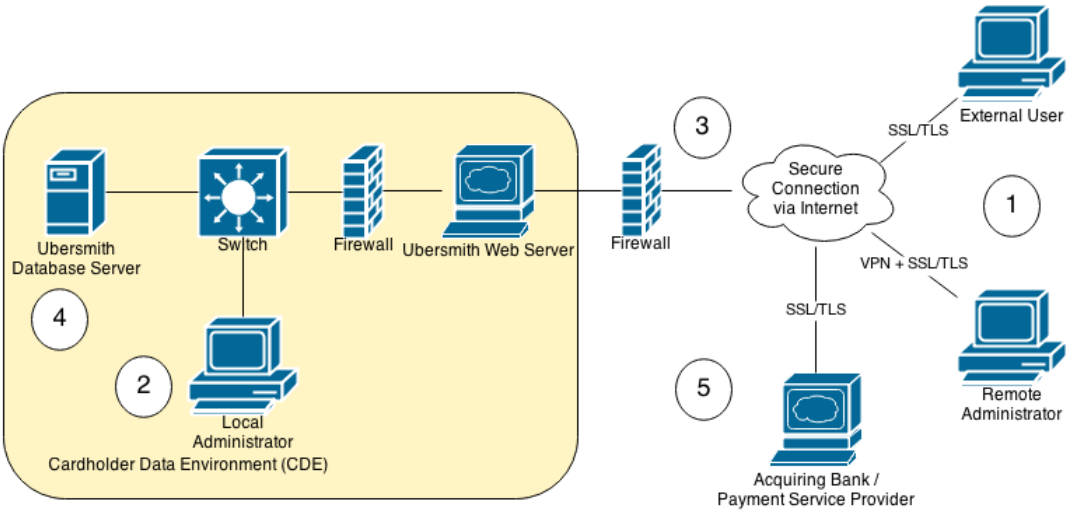
<table>
<tr><td></td><td>api/ - Application Programming Interface (API)
app/ - Graphical User Interface support files
client/ - Client (end user) Interface
cron/ - Automated task support files
css/ - Cascading Style Sheet (CSS) support files
fonts/ - TrueType font files
images/ - Image support files
include/ - Core Ubersmith support files
ipn/ - Instant Payment Notification (IPN) and related support files
js/ - JavaScript (JS) support files
locale/ - Localization (i18n) support files
order/ - Order Manager support files
quote/ - Sales Manager quoting tool support files
rssgen/ - Rich Site Summary support files (deprecated)
setup/ - Setup, Configuration, and Installation support files
solr/ - Apache Solr search engine support files

Within the web root of the Ubersmith source distribution are various files required for Ubersmith to function properly.</td></tr>
<tr><td>Required Third Party Payment Application Software:</td><td>An account with an acquiring bank or Payment Service Provider is required to take advantage of Ubersmith’s recurring billing capabilities. Ubersmith supports many Payment Service Providers, a current list is always available within the Ubersmith Knowledge Base, at:

http://www.ubersmith.com/kbase</td></tr>
<tr><td>Database Software Supported:</td><td>MySQL 5.5+ is required.</td></tr>
<tr><td>Other Required Third Party Software:</td><td>Apache HTTP Server 2.4 or newer (http://httpd.apache.org/)
Postfix 2.6 or newer (http://www.postfix.org/)
PHP 5.6.x or newer (http://php.net/)
IonCube Loaders 5.1.1 or newer (http://www.ioncube.com/)</td></tr>
<tr><td>Operating System(s) Supported:</td><td>CentOS 7.x+
Debian 8.x+ “jessie”
Ubuntu 14.04 LTS “Trusty Tahr”</td></tr>
<tr><td>Application Functionality Supported</td><td>Select one or more from the following list:
<table>
<tr><td>☐</td><td>POS Suite</td><td>☐</td><td>POS Admin</td><td>☐</td><td>Shopping Cart & Store Front</td></tr>
<tr><td>☐</td><td>POS Face-To-Face</td><td>☐</td><td>Payment Middleware</td><td>☐</td><td rowspan="3">Others (Please Specify):
Card-Not-Present</td></tr>
<tr><td>☐</td><td>POS Kiosk</td><td>☐</td><td colspan="2">Payment Back Office</td></tr>
<tr><td>☐</td><td>POS Specialized</td><td>☐</td><td colspan="2">Payment Gateway/Switch</td></tr>
</table></td></tr>
<tr><td>Payment Processing Connections:</td><td>When a customer submits their Credit Card Primary Account Number (PAN) and CVV2 code via an Ubersmith API call or order form, it is transmitted via Secure Sockets Layer / Transport Layer Security (SSL/TLS) to the Ubersmith Web Server. The Web Server then initiates another SSL/TLS to a configured Acquiring Bank / Payment Service Provider. The PAN is encrypted and stored in the Ubersmith database, and a response is provided via the Ubersmith User Interface (UI). If the card details are rejected, an error message is returned via the Ubersmith UI.</td></tr>
</table>

Application Authentication	To access the Ubersmith Administrative or Client interface, a username and password are required. Ubersmith stores users' passwords in its database using a salted SHA-1 hashing algorithm. Direct access to the Ubersmith database itself is likewise restricted using a username and password; MySQL hashes user passwords using its own hashing mechanism.
Description of Versioning Methodology:	Ubersmith versioning has four levels, Major, Minor, Build, and Patch: X.X.X.X. Major changes include significant changes to the application and would have an impact on PA-DSS requirements. Minor changes include small changes such as minor enhancements and may or may not have an impact on PA-DSS requirements. Build and Patch changes include bug fixes and would have no negative impact on PA-DSS requirements.
List of Resellers/Integrators (If Applicable):	None

Typical Network Implementation



1. Remote Administrators and External Users access the Ubersmith Web Server via SSL/TLS, with Remote Administrators additionally connecting to the local network via VPN

2. Local Administrators access the Ubersmith Web Server via SSL/TLS, via connectivity provided by the internal network.

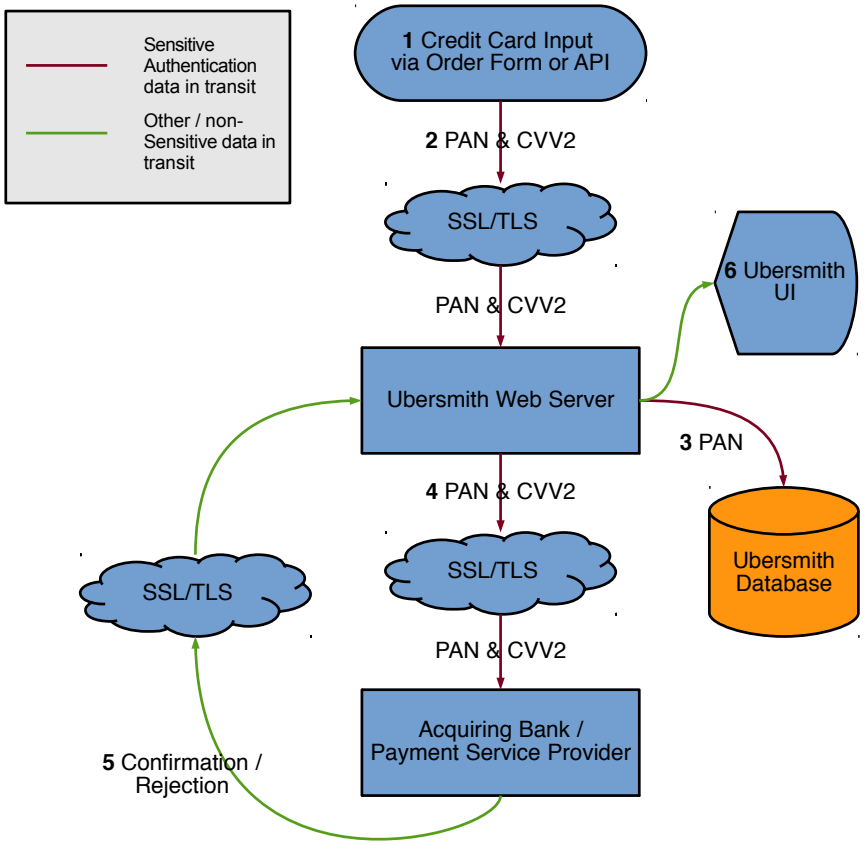
3. The Ubersmith Web Server is accessible via the Internet, however the configured firewall (software or hardware) only allows remote access to allowed ports (HTTP, HTTPS, etc.).

4. The Ubersmith Database Server hosts the Ubersmith database, and stores encrypted PAN and other client data. It, along with the local network, is protected by another firewall.

5. Credit cards transactions are processed by the Ubersmith Web Server using an SSL/TLS enabled connection to the Acquiring Bank / Payment Service Provider.

Dataflow Diagram

Ubersmith Data Flow Diagram



1 Credit Card PAN and CVV2 is entered into Ubersmith via Order Form or API

2 PAN and CVV2 are transmitted via SSL/TLS to the Ubersmith Web Server

3 PAN is encrypted and stored in the Ubersmith database for future use

4 PAN and CVV2 are transmitted to the acquiring bank / payment service provider utilizing SSL/TLS.

5 Authorization response is sent back to the Ubersmith Web Server via SSL/TLS

6 Transaction result is displayed to the user / administrator via the User Interface

Difference between PCI Compliance and PA-DSS Validation

As a software vendor, our responsibility is to be "PA-DSS Validated."

We have performed an assessment and certification compliance review with our independent assessment firm, to ensure that our platform does conform to industry best practices when handling, managing and storing payment related information.

PA-DSS is the standard against which Ubersmith has been tested, assessed, and validated.

PCI Compliance is then later obtained by the merchant, and is an assessment of your actual server (or hosting) environment.

Obtaining "PCI Compliance" is the responsibility of the merchant and your hosting provider, working together, using PCI compliant server architecture with proper hardware & software configurations and access control procedures.

The PA-DSS Validation is intended to ensure that Ubersmith will help you achieve and maintain PCI Compliance with respect to how Ubersmith handles user accounts, passwords, encryption, and other payment data related information.

The Payment Card Industry (PCI) has developed security standards for handling cardholder information in a published standard called the PCI Data Security Standard (DSS). The security requirements defined in the DSS apply to all members, merchants, and service providers that store, process or transmit cardholder data.

The PCI DSS requirements apply to all system components within the payment application environment which is defined as any network device, host, or application included in, or connected to, a network segment where cardholder data is stored, processed or transmitted.

The 12 Requirements of the PCI DSS:

Build and Maintain a Secure Network

1. *Install and maintain a firewall configuration to protect data*
2. *Do not use vendor-supplied defaults for system passwords and other security parameters*

Protect Cardholder Data

3. *Protect Stored Data*
4. *Encrypt transmission of cardholder data and sensitive information across public networks*

Maintain a Vulnerability Management Program

5. *Use and regularly update anti-virus software*
6. *Develop and maintain secure systems and applications*

Implement Strong Access Control Measures

7. *Restrict access to data by business need-to-know*
8. *Assign a unique ID to each person with computer access*
9. *Restrict physical access to cardholder data*

Regularly Monitor and Test Networks

10. *Track and monitor all access to network resources and cardholder data*

11. Regularly test security systems and processes

Maintain an Information Security Policy

12. Maintain a policy that addresses information security

Considerations for the Implementation of Ubersmith in a PCI-Compliant Environment

The following areas must be considered for proper implementation in a PCI-Compliant environment.

- Sensitive Authentication Data requires special handling
- Remove Historical Cardholder Data
- Set up Good Access Controls
- Properly Train and Monitor Admin Personnel
- Key Management Roles & Responsibilities
- PCI-Compliant Remote Access
- Use SSH, VPN, or SSLV3/TLS 1.0 or higher for encryption of administrative access
- Log settings must be compliant
- PCI-Compliant Wireless settings
- Data Transport Encryption
- PCI-Compliant Use of Email
- Network Segmentation
- Never store cardholder data on internet-accessible systems
- Use SSLV3 for Secure Data Transmission
- Delivery of Updates in a PCI Compliant Fashion

Remove Historical Sensitive Authentication Data (PA-DSS 1.1.4.a)

The following previous versions of Ubersmith stored card validation (CVV2) codes by default:

Ubersmith DE 1.8.0 and older

- Historical data must be securely deleted (magnetic stripe data, card validation codes, PINs, or PIN blocks stored by previous versions of the software) – removal is absolutely necessary for PCI compliance

Secure Removal of Sensitive Authentication Data

Ubersmith has developed a script to remove previously stored CVV2 data. Please contact Ubersmith Support for assistance in removing this data from your installation of Ubersmith.

Sensitive Authentication Data requires special handling (PA-DSS 1.1.5.c)

Ubersmith, Inc. does not store Sensitive Authentication data for any reason, and we strongly recommend that you do not do this either. However, if for any reason you should do so, the following guidelines must be followed when dealing with sensitive authentication data (swipe data, validation values or codes, PIN or PIN block data):

- Collect sensitive authentication data only when needed to solve a specific problem
- Store such data only in specific, known locations with limited access
- Collect only the limited amount of data needed to solve a specific problem
- Encrypt sensitive authentication data while stored
- Securely delete such data immediately after use

Purging of Cardholder Data (PA-DSS 2.1)

The following guidelines must be followed when dealing with cardholder data (PAN alone or with any of the following: expiry date, cardholder name or service code):

- A customer defined retention period must be defined with a business justification.
- Cardholder data exceeding the customer-defined retention period must be purged.
- To purge the cardholder data you must do the following:
    - In Ubersmith's Client Manager interface, you must delete the credit card from the client's account. This will purge the encrypted PAN data from the Ubersmith database.

As Ubersmith credit card processing routines and debug logging do not store PAN or CVV data in plain text, no further configuration is necessary to prevent inadvertent capture or retention of unencrypted cardholder data. However, if Ubersmith's API is being used to submit cardholder data, any logging performed by the software communicating with the API must be written or configured in such a way that it does not inadvertently capture or retain cardholder data.

Cardholder Data Encryption Key Management (PA-DSS 2.5.c and 2.6.a)

Ubersmith automatically handles the management and expiration of credit card encryption keys and key encryption keys (KEKs). Encryption keys and KEKs are stored in the Ubersmith database. Ensuring access to the database is limited to authorized parties and software only is of the utmost importance.

The following key management functions must be performed per PCI DSS:

- Generation of strong cryptographic keys.
- Secure cryptographic key distribution.
- Secure cryptographic key storage.
- Cryptographic key changes for keys that reached the end of their cryptoperiod (for example, after a defined period of time has passed and/or after a certain amount of ciphertext has been produced by a given key), as defined by the associated application vendor or key owner, and based on industry best practices and guidelines (for example, NIST Special Publication 800-57.
- Retire keys when the integrity of the key has been weakened.
- Replace known or suspected compromised keys.
- If retired or replaced cryptographic keys are retained, the application cannot use these keys for encryption operations.
- Manual clear-text key-management procedures require split knowledge and dual control of keys.
- Prevention of unauthorized substitution of cryptographic keys.

Ubersmith automatically handles these requirements. No action is required by an Ubersmith administrator to satisfy these requirements. In the case of a known or suspected compromised key, please contact Ubersmith support for assistance in replacing such keys ahead of their scheduled rotation period.

Removal of Cryptographic material (PA-DSS 2.7.a)

All versions of Ubersmith encrypt cardholder data. Ubersmith automatically handles the removal of old cryptographic material during the software upgrade process. Old encryption keys and key encryption keys are securely removed when they are overwritten during the upgrade process.

Addressing Inadvertent Capture of PAN on Linux

Clear swap space on your system

The process to clear the swap file on Linux is as follows:

Execute these commands in this order:

- `free` (to review swap usage)
- `swapoff –a` (requires elevated privs)
- `swapon –a` (requires elevated privs)
- `free` (should show that swap has been cleaned out)

OR

Disable swap space

(NOTE: Disabling swap space can be risky to the operation of your system. With no swap space, the Linux/Unix operating system will automatically kill processes if the amount physical RAM needed for all running processes is exceeded):

- Comment out the swap entry in `/etc/fstab`.

Encrypt the swap space on your system

This section covers encrypting swap through the use of `dm-crypt`. This requires you to run a 2.6 kernel. For the purposes of this HOWTO, say the swap partition will be `/dev/VolGroup00/LogVol01`. This is the default swap partition for RedHat systems. (Please note, the swap partition does not need to be part of an LVM. `dm-crypt` can encrypt disk partitions (`/dev/hda2`) or whole disks (`/dev/hda`). Be sure to change the commands to fit your swap partition accordingly.)

When encrypting your swap partition, you will need to temporarily turn off swap. This means you need to shut all unnecessary applications to free up memory. If this memory is not freed, you will be unable to turn off the swap space. The best way to handle this is to boot the system into single user mode. This shuts down most services with the exception of a single root shell. To boot the system into single user mode, run the following command.

```
# /sbin/telinit s
```

Turn off the swap space by running the following command.

```
# swapoff -a
```

To ensure a completely clean and sterile swap space, you must overwrite swap partition with random data. This will help prevent the recovery of any data written to swap before the encryption process. The `shred` command overwrites the specified file or device with random data.

```
# shred -v /dev/VolGroup00/LogVol01
```

Next, create a file named `/etc/crypttab`. The man page for `crypttab` covers the particulars of this application. The below example

1) Creates a encrypted block device named swap at `/dev/mapper` (first field)
2) Specifies `/dev/VolGroup00/LogVol01` as the underlying block device (second field)

3) Specifies `/dev/random` as the encryption password. (third field)

4) Specifies the encrypted device as a swap device with an encryption cypher with AES encryption and unpredictable IV values (fourth field)

```
swap    /dev/VolGroup00/LogVol01    /dev/random    swap,cipher=aes-cbc-
essiv:sha256
```

Next, you need to edit `/etc/fstab` to point to the encrypted block device, `/dev/mapper/swap` as opposed to `/dev/VolGroup00/LogVol01`. The current file should resemble this.

```
/dev/VolGroup00/LogVol01  swap  swap  defaults  0 0
```

Change the file to resemble this.

```
/dev/mapper/swap  swap  swap  defaults  0 0
```

Now, reboot your system to create the encrypted swap space with following command.

```
# reboot -n
```

If you do not wish to reboot, you may create the encrypted swap swap partition using the commands below.

```
# cryptsetup -d /dev/random create swap /dev/VolGroup00/LogVol01
# mkswap /dev/mapper/swap
# swapon -a
```

Before running the above commands, make sure you understand what the commands do. For the first command, the options passed do the following:

- "-d" specifies cryptsetup to use `/dev/random` as the key file

"create" creates a mapping with the name, swap backed by the device, `/dev/VolGroup00/LogVol01`

Set up Strong Access Controls (3.1.a and 3.2)

The PCI DSS requires that access to all systems in the payment processing environment be protected through use of unique users and complex passwords. Unique user accounts indicate that every account used is associated with an individual user and/or process with no use of generic group accounts used by more than one user or process.

3.1.a: You must assign strong passwords to any default accounts (even if they won't be used), and then disable or do not use the accounts.

All authentication credentials are provided by the application. For both the completion of the initial installation and for any subsequent changes (for example, any changes that result in user accounts reverting to default settings, any changes to existing account settings, or changes that generate new accounts or recreate existing accounts), the following 10 points must be followed per PCI 8.1, 8.2, and 8.5.8-15:

1. The application must assign unique IDs for user accounts. (8.1)
2. The application must provide at least one of the following three methods to authenticate users: (8.2)
    a. Something you know, such as a password or passphrase
    b. Something you have, such as a token device or smart card
    c. Something you are, such as a biometric
3. The application must NOT require or use any group, shared, or generic accounts or passwords.(8.5.8
4. The application requires passwords to be changed at least every 90 days (8.5.9)
5. The application requires passwords must to be at least 7 characters (8.5.10)
6. The application requires passwords to include both numeric and alphabetic characters (8.5.11)
7. The application keeps password history and requires that a new password is different than any of the last four passwords used. (8.5.12)
8. The application limits repeated access attempts by locking out the user account after not more than six logon attempts. (8.5.13)
9. The application sets the lockout duration to a minimum of 30 minutes or until an administrator enables the user ID. (8.5.14)
10. The application requires the user to re-authenticate to re-activate the session if the application session has been idle for more than 15 minutes.

These same account and password criteria from the above 10 requirements must also be applied to any applications or databases included in payment processing to be PCI compliant. Ubersmith, as tested in our PA-DSS audit, meets, or exceeds these requirements for the following additional required applications or databases:

- Ubersmith MySQL Database

[Note: These password controls are not intended to apply to employees who only have access to one card number at a time to facilitate a single transaction. These controls are applicable for access by employees with administrative capabilities, for access to servers with cardholder data, and for access controlled by the application.]

3.2: Control access, via unique username and PCI DSS-compliant complex passwords, to any PCs or servers with payment applications and to databases storing cardholder data.

Properly Train and Monitor Admin Personnel

It is your responsibility to institute proper personnel management techniques for allowing admin user access to cardholder data, site data, etc. You can control whether each individual admin user can see credit card PAN (or only last 4).

In most systems, a security breach is the result of unethical personnel. So pay special attention to whom you trust into your admin site and who you allow to view full decrypted and unmasked payment information.

Log settings must be compliant (PA-DSS 4.1.b, 4.4.b)

4.1.b: Ubersmith has PA-DSS compliant logging enabled by default. This logging is not configurable and may not be disabled. Disabling or subverting the logging function of Ubersmith in any way will result in non-compliance with PCI DSS.

Ubersmith's Event Log and other logs can be viewed in the 'reports & stats' section of the main Ubersmith administrator user interface.

Implement automated assessment trails for all system components to reconstruct the following events:

10.2.1 All individual user accesses to cardholder data
10.2.2 All actions taken by any individual with root or administrative privileges
10.2.3 Access to application audit trails managed by or within the application
10.2.4 Invalid logical access attempts
10.2 5 Use of the application's identification and authentication mechanisms
10.2.6 Initialization of the application audit logs
10.2.7 Creation and deletion of system-level objects within or by the application

Record at least the following assessment trail entries for all system components for each event from 10.2.x above:

10.3.1 User identification
10.3.2 Type of event
10.3.3 Date and time
10.3.4 Success or failure indication
10.3.5 Origination of event
10.3.6 Identity or name of affected data, system component, or resource.

Disabling or subverting the logging function of Ubersmith in any way will result in non-compliance with PCI DSS.

4.4.b: Ubersmith facilitates centralized logging by way of the underlying operating system's 'syslog' facility. The syslog output from the Ubersmith web server can then be directed to a remote syslog host for centralized review.

Services and Protocols (PA-DSS 5.4.c)

Ubersmith does not require the use of any insecure services or protocols. Here are the services and protocols that Ubersmith does require:

- Ubersmith Instance/Web Host (Frontend)

- 22/tcp Secure Shell (optional) – OpenSSH (http://openssh.com/)
- 25/tcp SMTP – Postfix (http://www.postfix.org/)
- 80/tcp HTTP (optional) – Apache HTTP Server (http://httpd.apache.org/)
- 443/tcp HTTPS – Apache HTTP Server (http://httpd.apache.org/)
- 4321/tcp RWhois (optional) - Ubersmith RWhois

- Ubersmith Appliance Host (Backend)
  - 22/tcp Secure Shell (optional) – OpenSSH (http://openssh.com/)
  - 80/tcp HTTP (optional) – Apache HTTP Server (http://httpd.apache.org/)
  - 443/tcp HTTPS – Apache HTTP Server (http://httpd.apache.org/)

PCI-Compliant Wireless settings (PA-DSS 6.1.f and 6.2.b)

Ubersmith does not support wireless technologies. However, should the merchant implement wireless access within the cardholder data environment, the following guidelines for secure wireless settings must be followed per PCI Data Security Standard 1.2.3, 2.1.1 and 4.1.1:

2.1.1: Change wireless vendor defaults per the following 5 points:

1. Encryption keys must be changed from default at installation, and must be changed anytime anyone with knowledge of the keys leaves the company or changes positions
2. Default SNMP community strings on wireless devices must be changed
3. Default passwords/passphrases on access points must be changed
4. Firmware on wireless devices must be updated to support strong encryption for authentication and transmission over wireless networks
5. Other security-related wireless vendor defaults, if applicable, must be changed

1.2.3: Perimeter firewalls must be installed between any wireless networks and systems that store cardholder data, and these firewalls must deny or control (if such traffic is necessary for business purposes) any traffic from the wireless environment into the cardholder data environment.

4.1.1: Industry best practices (for example, IEEE 802.11.i) must be used to implement strong encryption for authentication and transmission of cardholder data.
Note: The use of WEP as a security control was prohibited as of June 30, 2010.

Never store cardholder data on internet-accessible systems (PA-DSS 9.1.b)

Never store cardholder data on Internet-accessible systems (e.g., web server and database server must not be on same server.)

PCI-Compliant Remote Access (10.2)

The PCI standard requires that if employees, administrators, or vendors are granted remote access to the payment processing environment; access should be authenticated using a two-factor authentication mechanism. The means two of the following three authentication methods must be used:

1. Something you know, such as a password or passphrase
2. Something you have, such as a token device or smart card
3. Something you are, such as a biometric

PCI-Compliant Delivery of Updates (PA-DSS 10.3.1)

Ubersmith delivers patches and updates in a secure manner:

- Timely development and deployment of patches and updates.

  Depending on the severity of the issue, Ubersmith can provide (and, if necessary, apply) security patches to clients within one business day.

- Delivery in a secure manner with a known chain-of-trust.

  Patches are uploaded using SCP or SFTP, with authentication provided by an SSH key installed when the Ubersmith software is initially deployed.

- Delivery in a manner that maintains the integrity of the deliverable.

  Hashes are generated for each patch, and the hash is verified after the patched file is copied to the remote host.

- Integrity testing of patches or updates prior to installation.

  All patches to a production Ubersmith host are built and installed by the same technician to ensure proper application. A PHP lint (syntax) check is performed to verify that the file is compatible with the environment it is being installed on.

As a development company, we keep abreast of the relevant security concerns and vulnerabilities in our area of development and expertise.

The following outside sources are monitored for security vulnerability information:

- National Vulnerability Database (http://nvd.nist.gov/home.cfm)
- US-CERT (http://www.us-cert.gov/)
- Common Vulnerabilities and Exposures (http://cve.mitre.org)

- Packet Storm (http://packetstormsecurity.com/)
- Krebs on Security (http://krebsonsecurity.com/)
- Schneier on Security (https://www.schneier.com/)

Once we identify a relevant vulnerability, we work to develop & test a patch that helps protect Ubersmith against the specific, new vulnerability. We attempt to publish a patch within 10 days of the identification of the vulnerability. We will then contact our clients to encourage them to install the patch. Typically, merchants are expected to respond quickly to and install available patches within 30 days.

We deliver software and/or updates via remote access to customer networks using key-based authentication via Secure Shell, specifically using the SCP or SFTP protocols provided by the OpenSSH software package.

For receiving updates via remote access, merchants must adhere to the following guidelines:

Secure remote access technology use, per PCI Data Security Standard 12.3.9:

> *12.3 Activation of remote access technologies for vendors only when needed by vendors, with immediate deactivation after use*

Use a personal firewall product if computer is connected via VPN or other high-speed connection, to secure these "always-on" connections, per PCI Data Security Standard 1.3.10.

PCI-Compliant Remote Access (10.3.2.b)

The PCI standard requires that if employees, administrators, or vendors are granted remote access to the payment processing environment; access should be authenticated using a two-factor authentication mechanism (username/ password and an additional authentication item such as a token or certificate).

In the case of vendor remote access accounts, in addition to the standard access controls, vendor accounts should only be active while access is required to provide service. Access rights should include only the access rights required for the service rendered, and should be robustly audited.

If users and hosts within the Ubersmith environment may need to use third-party remote access software such as OpenSSH to access other hosts within the payment processing environment, special care must be taken.

In order to be compliant, every such session must be encrypted with at least 128-bit encryption (in addition to satisfying the requirement for two-factor authentication required for users connecting from outside the payment processing environment). This requirement is met using a default OpenSSH configuration. Additionally, the PCI user account and password requirements will apply to these access methods as well.

When requesting support from a vendor, reseller, or integrator, customers are advised to take the following precautions:

- Change default settings (such as usernames and passwords) on remote access software (e.g. VNC)
- Allow connections only from specific IP and/or MAC addresses
- Use strong authentication and complex passwords for logins according to PA-DSS 3.1.1 – 3.1.10 and PCI DSS 8.1, 8.3, and 8.5.8-8.5.15
- Enable encrypted data transmission according to PA-DSS 12.1 and PCI DSS 4.1
- Enable account lockouts after a certain number of failed login attempts according to PA-DSS 3.1.8 and PCI DSS 8.5.13
- Require that remote access take place over a VPN via a firewall as opposed to allowing connections directly from the internet
- Enable logging for auditing purposes
- Restrict access to customer passwords to authorized reseller/integrator personnel.
- Establish customer passwords according to PA-DSS 3.1.1 – 3.1.10 and PCI DSS Requirements 8.1, 8.2, 8.4, and 8.5.

Ubersmith automatically rotates out the SSH keys used for remote access every 12 months.

Data Transport Encryption (PA-DSS 11.1.b)

The PCI DSS requires the use of strong cryptography and encryption techniques with at least a 128 bit encryption strength (either at the transport layer with SSLV3 or IPSEC; or at the data layer with algorithms such as RSA or Triple-DES) to safeguard cardholder data during transmission over public networks (this includes the Internet and Internet accessible DMZ network segments).

PCI DSS requirement 4.1: Use strong cryptography and security protocols such as secure sockets layer (SSLV3) / transport layer security (TLS 1.0 or higher) and Internet protocol security (IPSEC) to safeguard sensitive cardholder data during transmission over open, public networks.

Examples of open, public networks that are in scope of the PCI DSS are:

- The Internet
- Wireless technologies
- Global System for Mobile Communications (GSM)
- General Packet Radio Service (GPRS)

Refer to the Dataflow diagram for an understanding of the flow of encrypted data associated with Ubersmith.

PCI-Compliant Use of End User Messaging Technologies (PA-DSS 11.2.b)

Ubersmith does not allow or facilitate the sending of PANs via any end user messaging technology (for example, e-mail, instant messaging, and chat).

Non-console administration (PA-DSS 12.1)

Ubersmith allows non-console administration via direct database query. This method of modifying/interacting with Ubersmith is not supported or approved by Ubersmith, Inc. You must use SSH, VPN, or SSLV3/TLS 1.0 or higher for encryption of this non-console administrative access.

Network Segmentation

The PCI DSS requires that firewall services be used (with NAT or PAT) to segment network segments into logical security domains based on the environmental needs for internet access. Traditionally, this corresponds to the creation of at least a DMZ and a trusted network segment where only authorized, business-justified traffic from the DMZ is allowed to connect to the trusted segment. No direct incoming internet traffic to the trusted application environment can be allowed. Additionally, outbound internet access from the trusted segment must be limited to required and justified ports and services.

- Refer to the standardized Network diagram for an understanding of the flow of encrypted data associated with Ubersmith.

Maintain an Information Security Program

In addition to the preceding security recommendations, a comprehensive approach to assessing and maintaining the security compliance of the Ubersmith environment is necessary to protect the organization and sensitive cardholder data.

The following is a very basic plan every merchant/service provider should adopt in developing and implementing a security policy and program:

- Read the PCI DSS in full and perform a security gap analysis. Identify any gaps between existing practices in your organization and those outlined by the PCI requirements.
- Once the gaps are identified, determine the steps to close the gaps and protect cardholder data. Changes could mean adding new technologies to shore up firewall and perimeter controls, or increasing the logging and archiving procedures associated with transaction data.
- Create an action plan for on-going compliance and assessment.
- Implement, monitor and maintain the plan. Compliance is not a one-time event. Regardless of merchant or service provider level, all entities should complete annual self-assessments using the PCI Self Assessment Questionnaire.
- Call in outside experts as needed.

Application System Configuration

Below are the operating systems and dependent application patch levels and configurations supported and tested for continued PCI DSS compliance.

Ubersmith Frontend Server Minimum Requirements:

- OS: CentOS 6.x, Debian 7.x ‘wheezy’, Ubuntu 12.04 LTS ‘Precise Pangolin’

- Dedicated host or VPS
- 2.66GHz Xeon Processor (Dual Core Processor, 4MB Cache)
- 2GB DDR2 RAM
- Dual 80GB Hard Disks
- Hardware RAID1
- Gigabit Network Adapter

Payment Application Initial Setup & Configuration

For information on the installation, configuration, and upgrade procedures for Ubersmith, please see our Knowledge Base at:

http://ubersmith.com/kbase

In most scenarios, Ubersmith support will be performing the software installation and subsequent upgrades on your behalf.

If you need assistance obtaining or installing a 128 bit SSLV3 Certificate, contact Ubersmith support.

If you need assistance configuring your Payment Service Provider account, view this Knowledge Base article:

http://ubersmith.com/kbase/index.php?_m=knowledgebase&_a=viewarticle&kbarticleid=40

or contact Ubersmith support.