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 PCI 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 3.2 dated June 2016)[1].

Ubersmith, Inc. instructs and advises its customers to deploy Ubersmith, Inc. applications in a manner that adheres to the PCI Data Security Standard (v3.2). 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, disabling infrequently-used or frequently vulnerable networking protocols and implementing 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.

[1] PCI PA-DSS 3.1 can be downloaded from the PCI SSC Document Library.

Revision Information

Name	Title	Date of Update	Summary of Changes
Michael Styne	Dir. of Operations	05/16/2016	Initial Version

Note: This PA-DSS Implementation Guide (IG) 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 documentation website, https://docs.ubersmith.com, and our repository on github.com.

Executive Summary

Ubersmith 4.1.0 has been payment application - data security standard (PA-DSS) validated, in accordance with PA-DSS Version 3.2. For the PA-DSS assessment, we worked with the following PCI SSC approved Payment Application Qualified Security Assessor (PA-QSA):

Coalfire Systems, Inc. 11000 Westmoor Circle, Suite 450, Westminster, CO 80021	Coalfire Systems, Inc. 1633 Westlake Ave N #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, Inc.'s Ubersmith Version 4.1.0 as a PA-DSS validated application operating in a PCI-DSS 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 Card Industry Payment Applications - Data Security Standard (PCI PA-DSS)
  https://www.pcisecuritystandards.org/security_standards/index.php
- Payment Card Industry Data Security Standard (PCI DSS)
  https://www.pcisecuritystandards.org/security_standards/pci_dss.shtmlhttps://www.pcisecuritystandards.org/security_standards/index.php
- Open Web Application Security Project (OWASP)
  https://www.owasp.org/
- Center for Internet Security (CIS) Benchmarks (used for OS Hardening)
  https://benchmarks.cisecurity.org/downloads/multiform/

Application Summary

<table>
<tr><td>Payment Application Name</td><td>Ubersmith</td><td>Payment Application Version</td><td>4.1.0</td></tr>
<tr><td>Application Description</td><td colspan="3">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.</td></tr>
<tr><td>Typical Role of Application</td><td colspan="3">Ubersmith is used primarily as a platform to manage recurring billing of clients.</td></tr>
<tr><td>Target Market for Payment Application</td><td colspan="3">
<table>
<tr><td colspan="6">Target Market for Payment Application (check all that apply):</td></tr>
<tr><td></td><td>Retail</td><td></td><td>Processors</td><td></td><td>Gas/Oil</td></tr>
<tr><td>X</td><td>e-Commerce</td><td></td><td>Small/medium merchants</td><td></td><td></td></tr>
<tr><td></td><td colspan="5">Others (please specify): Internet hosting providers who manage data center facilities</td></tr>
</table>
</td></tr>
<tr><td rowspan="3">Stored Cardholder Data</td><td colspan="3">The following is a brief description of files and tables that store cardholder data:</td></tr>
<tr><td>File or Table Name</td><td colspan="2">Description of Stored Cardholder Data</td></tr>
<tr><td>cc_accounts

billing_info</td><td colspan="2">PAN, Expiry</td></tr>
<tr><td></td><td colspan="3">Individual access to cardholder data is logged as follows:
Access to clear text PAN data is not permitted.</td></tr>
<tr><td rowspan="2">Components of the Payment Application</td><td colspan="3">The following are the application-vendor-developed components which comprise the payment application:</td></tr>
<tr><td colspan="3">Within the Ubersmith source the directory structure contains the following components:
admin/ - Administrative interface
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</td></tr>
</table>

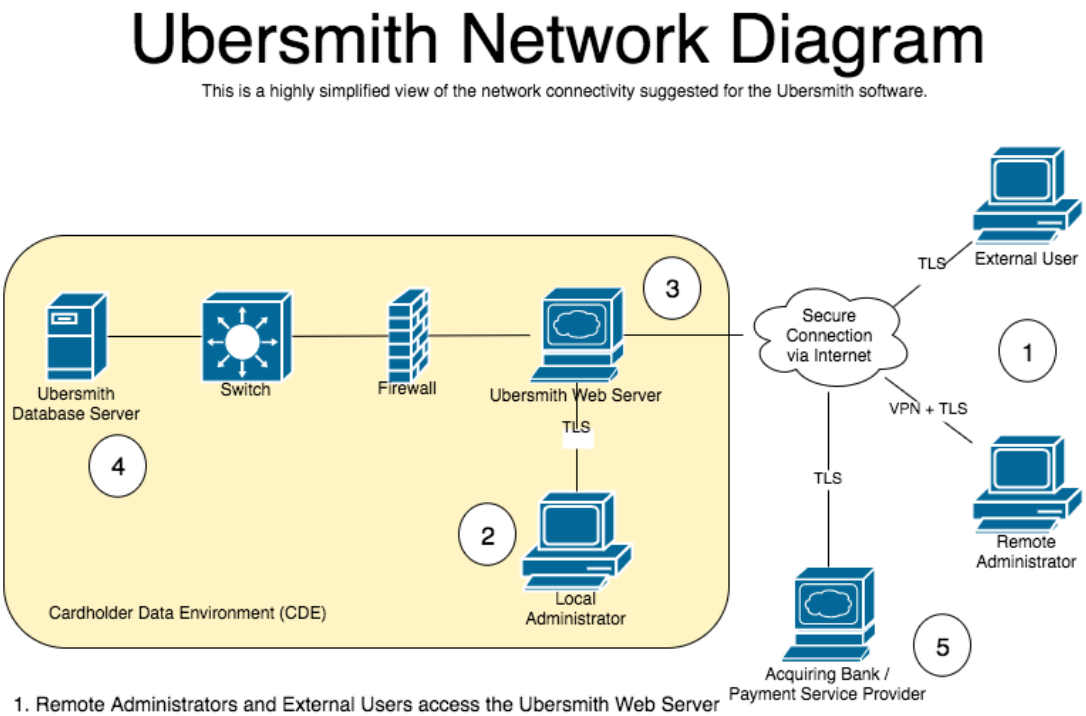
<table>
<tr><td></td><td>• Within the web root of the Ubersmith source distribution are various files required for Ubersmith to function properly.</td></tr>
<tr><td rowspan="2">Required Third Party Payment Application Software</td><td>The following are additional third party payment application components required by the payment application:</td></tr>
<tr><td>No third party payment application software is required, however 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 documentation at:

https://docs.ubersmith.com/</td></tr>
<tr><td rowspan="2">Database Software Supported</td><td>The following are database management systems supported by the payment application:</td></tr>
<tr><td>Percona Server 5.6</td></tr>
<tr><td rowspan="2">Other Required Third Party Software</td><td>The following are other required third party software components required by the payment application:</td></tr>
<tr><td>Apache HTTP Server 2.4 (https://httpd.apache.org/)
Postfix 2.6 (https://www.postfix.org/)
PHP 5.6.x (https://php.net/)
IonCube Loaders 5.1.1 or newer (https://www.ioncube.com/)</td></tr>
<tr><td rowspan="2">Operating System(s) Supported</td><td>The following are Operating Systems supported or required by the payment application:</td></tr>
<tr><td>• CentOS 7
• Debian 8
• Ubuntu 16.04 LTS</td></tr>
<tr><td>Application Authentication</td><td>To access the Ubersmith administrative or client interface, a username and password are required. Ubersmith stores users' passwords in its database using the bcrypt 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.</td></tr>
<tr><td>Application Encryption</td><td>Ubersmith administrator passwords are hashed using the bcrypt algorithm.

Ubersmith encrypts customer PAN data using AES compliant methods. Encryption keys and Key Encryption Keys are rotated automatically by the Ubersmith software, but can be manually rotated if necessary.</td></tr>
<tr><td>Application Functionality Supported</td><td>Payment Application Functionality (check only one):
<table>
<tr><td></td><td>Automated Fuel Dispenser</td><td></td><td>POS Kiosk</td><td></td><td>Payment Gateway/Switch</td></tr>
<tr><td>X</td><td>Card-Not-Present</td><td></td><td>POS Specialized</td><td></td><td>Payment Middleware</td></tr>
<tr><td></td><td>POS Admin</td><td></td><td>POS Suite/General</td><td></td><td>Payment Module</td></tr>
<tr><td></td><td>POS Face-to-Face/POI</td><td></td><td>Payment Back Office</td><td></td><td>Shopping Cart & Store Front</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 transport layer security to the Ubersmith web server. The web server then initiates another TLS</td></tr>
</table>

	connection 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.
Description of Listing Versioning Methodology	Ubersmith versioning has three levels, Major, Minor, and Build: **<Major>.<Minor>.<Build>** • **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** changes include bug fixes or rollups and would have no negative impact on PA-DSS requirements and are indicated by the WILDCARD (X). Based on the above versioning methodology the application version being listed with the PCI SSC is: 4.1.X.

Typical Network Implementation



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

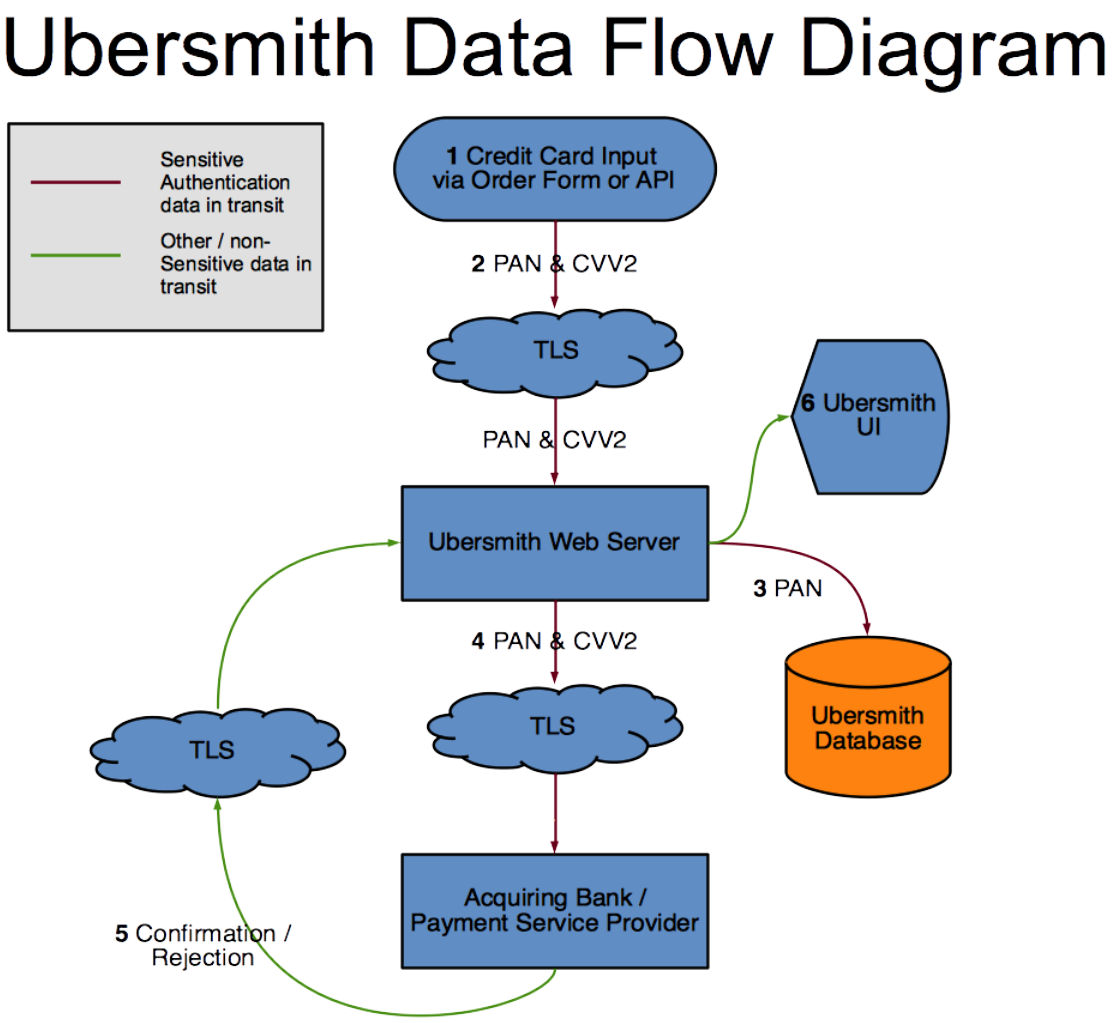
2. Local Administrators access the Ubersmith Web Server via TLS.

3. The Ubersmith Web Server is accessible via the Internet.

4. The Ubersmith Database Server hosts the Ubersmith database, and stores encrypted PAN and other client data.

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

Credit/Debit Cardholder Dataflow Diagram



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

2 PAN and CVV2 are transmitted via 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 TLS.

5 Authorization response is sent back to the Ubersmith Web Server via 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 who develops payment applications, our responsibility is to be PA-DSS Validated. We have performed an assessment and payment application validation review with our independent assessment firm (PA-QSA), to ensure that our platform conforms to industry best practices when handling, managing and storing payment related information.

PA-DSS Version 3.2 is the standard against which payment application 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 called the cardholder data environment (CDE).

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

The PA-DSS validation is intended to ensure that Ubersmith will help you facilitate and maintain PCI compliance with respect to how the payment application 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 and Systems

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

Protect Cardholder Data

3. *Protect stored cardholder data*
4. *Encrypt transmission of cardholder data across open, public networks*

Maintain a Vulnerability Management Program

5. *Protect all systems against malware and regularly update anti-virus software or programs*
6. *Develop and maintain secure systems and applications*

Implement Strong Access Control Measures

7. *Restrict access to cardholder data by business need-to-know*
8. *Identify and authenticate access to system components*
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 for all personnel

Considerations for the Implementation of Payment Application in a PCI-Compliant Environment

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

Removing Historical Sensitive Authentication Data (PA-DSS 1.1.4)

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

Ubersmith DE 1.8.0 and older

- Historical sensitive authentication data (SAD) 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

Sensitive Authentication Data[2] includes security-related information (including but not limited to card validation codes/values, full track data (from the magnetic stripe or equivalent on a chip), PINs, and PIN blocks) used to authenticate cardholders and/or authorize payment card transactions.

Historical SAD stored by previous versions of Ubersmith must be securely deleted and removal is absolutely necessary for PCI DSS compliance. Ubersmith, Inc. provides a secure deletion tool which includes capabilities to securely delete historical SAD as follows:

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.

Handling Sensitive Authentication Data (PA-DSS 1.1.5)

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 used for pre-authorization (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.

Securely Deleting Cardholder Data (PA-DSS 2.1)

The following guidelines must be followed when dealing with cardholder data (primary account number (PAN); cardholder name; expiration date; or service code):

- A customer defined retention period must be defined with a business justification.

---

[2] Sensitive Authentication Data as defined in the PCI SSC's Glossary of Terms, Abbreviations, and Acronyms

- Cardholder data exceeding the customer-defined retention period or when no longer required for legal, regulatory, or business purposes must be securely deleted.
- To securely delete 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.
- All underlying software (this includes operating systems and/or database systems) must be configured to prevent the inadvertent capture of PAN. Instructions for configuring the underlying operating systems and/or databases can be found in Appendix A: Addressing Inadvertent Capture of PAN.
- 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.

Masking all PAN by Default (PA-DSS 2.2)

Ubersmith masks all PAN by default in all locations that display PAN (screens, reports, etc.) by masking all but the last four digits. The payment application displays PAN in the following locations:

- Client Manager -> Billing / Invoicing -> Payment Methods

Ubersmith does not have the ability to display full PAN for any reason and therefore there is no configuration details to be provided as required for PA-DSS v3.2.

Encrypting Cardholder Data & Managing Keys (PA-DSS 2.3, 2.4, and 2.5)

Ubersmith does store cardholder data and does not have the ability to output PAN data for storage outside of the payment application. All PAN must be rendered unreadable anywhere it is stored (including data on portable digital media, backup media, and in logs). The payment application uses an encryption methodology with dynamically generated keys to automatically encrypt all locations/methods where cardholder data is stored.

The payment application does not output PAN for use or storage in a merchant's environment for any reason therefore there are no location or configuration details to provide as required by PA-DSS v3.2.

Ubersmith does have a debugging mode, but it will not write PAN to debugging logs.

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

- You must restrict access to encryption keys to the fewest number of custodians necessary.
- You must store encryption keys securely in the fewest possible locations and forms.

- A sample Key Custodian form has been provided in Appendix A: Addressing Inadvertent Capture of PAN for key custodians to acknowledge that they understand and accept their key custodian responsibilities.

The following key management functions are performed automatically using AES 256 dynamic encryption key methodology and there are no key custodians or interventions required by customers or resellers/integrators.

- Generation of strong cryptographic keys.
- Secure cryptographic key distribution.
- Secure cryptographic key storage.
- Cryptographic key changes for keys that have reached the end of their cryptoperiod.
- Retire or replace keys when the integrity of the key has been weakened and/or when known or suspected compromise. If retired or replaced cryptographic keys are retained, the application cannot use these keys for encryption operations.
- Manual clear-text cryptographic key-management procedures require split knowledge and dual control of keys.
- Prevention of unauthorized substitution of cryptographic keys.

Removing Historical Cryptographic Material (PA-DSS 2.6)

All previous version of Ubersmith encrypted cardholder data.

Ubersmith uses previously validated encryption algorithms that are PCI compliant. Therefore, there is no need to render historical cryptographic keys or cryptograms irretrievable as they are still in use by the payment application.

Setting up Strong Access Controls (3.1 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.

All authentication credentials are generated and managed *by the application*. Secure authentication is enforced automatically by the payment application for all credentials *by 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). To maintain PCI DSS compliance the following 11 points must be followed per the PCI DSS:

1. The payment application must not use or require the use of default administrative accounts for other necessary or required software (for example, database default administrative accounts) (PCI DSS 2.1 / PA-DSS 3.1.1).
2. The payment application must enforce the changing of all default application passwords for all accounts that are generated or managed by the application, by the completion of installation and for subsequent changes after the installation (this applies to all accounts, including user accounts, application and service accounts, and accounts used by Ubersmith, Inc. for support purposes) (PCI DSS 2.1 / PA-DSS 3.1.2).

3. The payment application must assign unique IDs for all user accounts. (PCI DSS 8.1.1 / PA-DSS 3.1.3).
4. The payment application must provide at least one of the following three methods to authenticate users: (PCI DSS 8.2 / PA-DSS 3.1.4).
    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
5. The payment application must NOT require or use any group, shared, or generic accounts and passwords (PCI DSS 8.5 / PA-DSS 3.1.5).
6. The payment application requires passwords must to be at least seven characters and includes both numeric and alphabetic characters (PCI DSS 8.2.3 / PA-DSS 3.1.6).
7. The payment application requires passwords to be changed at least every 90 days (PCI DSS 8.2.4 / PA-DSS 3.1.7).
8. The payment application keeps password history and requires that a new password is different than any of the last four passwords used (PCI DSS 8.2.5 / PA-DSS 3.1.8).
9. The payment application limits repeated access attempts by locking out the user account after not more than six logon attempts (PCI DSS 8.1.6 / PA-DSS 3.1.9).
10. The payment application sets the lockout duration to a minimum of 30 minutes or until an administrator enables the user ID (PCI DSS 8.1.7 / PA-DSS 3.1.10).
11. The payment application requires the user to re-authenticate to re-activate the session if the application session has been idle for more than 15 minutes (PCI DSS 8.1.8 / PA-DSS 3.1.11).

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.

These same account and password criteria from the above 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 validation, meets, or exceeds these requirements for the following additional required applications or databases:

- Percona Server

[**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 systems with cardholder data, and for access controlled by the application.

The requirements apply to the payment application and all associated tools used to view or access cardholder data.]

PA-DSS 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 Training and Monitoring 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 the last four).

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 fully decrypted and unmasked payment information.

Using Compliant Logging Settings (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.* Logs may be accessed in the Reports section of Ubersmith:

Reports -> Global Stats -> Event Log

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

10.2.1 All individual user accesses to cardholder data from the application
10.2.2 All actions taken by any individual with administrative privileges in the application
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 (including but not limited to creation of new accounts, elevation of privileges, etc.) and all changes, additions, deletions to application accounts with root or administrative privileges
10.2.6 Initialization, stopping, or pausing 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. This type of logging is disabled by default.

PCI-Compliant Wireless Settings (PA-DSS 6.1.a 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:

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.

2.1.1: Change wireless vendor defaults per the following five 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.

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.

Services and Protocols (PA-DSS 8.2.c)

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

- Ubersmith Core Host (Frontend)
    - 443/tcp HTTPS – Apache HTTP Server (https://httpd.apache.org/)
- Ubersmith Appliance Host (Backend)
    - 443/tcp HTTPS – Apache HTTP Server (https://httpd.apache.org/)

Ubersmith can utilize other protocols (ex. SMTP) to extend the system's functionality, however these protocols are not required to utilize the system.

Never Store Cardholder Data on Internet-Accessible Systems (PA-DSS 9.1.c)

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 (10.1)

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. This 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 10.2.1.a, 7.2.3)

Ubersmith delivers patches and updates in a secure manner:

- New updates

  Ubersmith communicates the availability of new updates via our website and social media channels.

- Timely development and deployment of bugfixes and updates.

  Bugfixes are included in build-version releases, which are made available on a regular basis. Ubersmith does not provide one-off patch files to be applied to an installation.

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

  Updated Ubersmith container images are pushed to and pulled from our repository via HTTPS.

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

  Container images are not modified once downloaded.

- Integrity testing of patches or updates prior to installation.

  All Ubersmith releases, including build-version releases pass through a testing and QA process prior to release.

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 (https://nvd.nist.gov/)
- US-CERT (https://www.us-cert.gov/)
- Common Vulnerabilities and Exposures (https://cve.mitre.org)
- Packet Storm (https://packetstormsecurity.com/)
- Krebs on Security (https://krebsonsecurity.com/)
- Schneier on Security (https://www.schneier.com/)

Once we identify a relevant vulnerability, we work to develop and test an update, or generate new container images that help protect Ubersmith against a specific, new vulnerability. We attempt to publish a patch within 10 days of identifying the vulnerability. We then contact vendors and dealers 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.

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 (10.2.3.a)

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 payment application 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 two-factor authentication requirement for users connecting from outside the payment processing environment). 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 when an employee with access to the keys leaves the company.

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 TLS 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 transport layer security (TLS 1.1 / TLS 1.2) 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 Credit/Debit Cardholder 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, email, instant messaging, and chat).

Non-console administration and Multi-Factor Authentication (PA-DSS 12.1, 12.2)

Ubersmith allows non-console administration, via direct database query. This method of modifying/interacting with Ubersmith is not supported or approved by Ubersmith. You must use SSH, VPN, or TLS 1.1 or higher for encryption of this non-console administrative access. Because Ubersmith allows such access, multi-factor authentication (at least two of something you know, something you have, something you are) must be utilized when accessing Ubersmith over these technologies.

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 one 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 Typical Network Implementation 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 payment application 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.

- OS: CentOS 7, Debian 8, Ubuntu 16.04 LTS
- RAM: 8GB
- Disk: 100GB

For more specific details, please visit: https://docs.ubersmith.com/x/bwCO

Payment Application Initial Setup & Configuration

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

https://docs.ubersmith.com/

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

If you need assistance installing a 128 bit TLS certificate, contact Ubersmith Support.

If you need assistance configuring your payment service provider account, view these documentation articles:

Supported Merchant Gateways: https://docs.ubersmith.com/x/JIFK

Configuring Merchant Accounts: https://docs.ubersmith.com/x/eYFK

or contact Ubersmith support.

Appendix A: 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:

1. `free` (to review swap usage)
2. `swapoff –a` (requires elevated privs)
3. `swapon –a` (requires elevated privs)
4. `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 of 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 running a 2.6 kernel. As an example, 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, 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. See the example below:

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, 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 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`