Achieving PCI Compliance with Postgres

Denish Patel
Database Architect

Who am I ?

- Lead Database Architect @ OmniTI
- Email: denish@omniti.com ; Twitter: @DenishPatel
- http://www.pateldenish.com
- Providing Solutions for business problems to deliver
  - Scalability
  - Reliability
  - High Availability
  - Consistency
  - **Security**

We are hiring!! Apply @ l42.org/lg

Agendum

- PCI Introduction
- Merchant levels
- PCI compliance requirements
- How to achieve
- Common Myths
- QA

Introduction & History

- PCI DSS is a result of the collaboration between all major credit card companies (including Visa, MasterCard, JCB, American Express, and Discover) that designed the PCI DSS to establish industry-wide security requirements.
- The Payment Card Industry (PCI) Data Security Standard (DSS) is a set of specific credit card holder **protection regulations** and **guidance** to combat **identity theft**.
- PCI DSS v1.1 introduced in Sept 2006. PCI DSS v2.0 effective until December 31st, 2014
- **PCI DSS Version 3** published on Nov 2013 & effective since Jan 1st, 2014

PCI DSS Applicability

If a **Primary Account Number** (PAN) is stored, processed, or transmitted.

	Data Element	**Storage Permitted**	**Render data unreadable**
Card holder data	Primary Account Number (PAN)	Yes	Yes (1)
	Cardholder Name	Yes	No
	Service Code	Yes	No
	Expiration Date	Yes	No
Sensitive authentication data (2)	Full Magnetic Stripe Data	No	N/A
	CAV2/CVC2/CVV2/CID	No	N/A
	PIN/PIN Block	No	N/A

(1) PCI DSS requirements 3.3 and 3.4 apply only to PAN. *(2) Sensitive authentication data must not be stored after authorization (even if encrypted).*

Merchant Levels

DISCOVER NETWORK MasterCard VISA	Level 1	Level 2	Level 3	Level 4
TXN Processed Annually	>6 M	1-6 M	>20,000 & < 1 M	< 20,000
Site Audit	Annual	No	No	No
Network Scan	Quarterly	Quarterly	Quarterly	Quarterly
SAQ	Annual	Annual	Annual	Annual

* Reference : http://usa.visa.com/merchants/protect-your-business/cisp/merchant-pci-dss-compliance.jsp

Why Postgres?

“By default PostgreSQL is probably the most security-aware database available”

– David Litchfield (The Database Hackers Handbook)

PostgreSQL offers **encryption** at several levels, and provides flexibility in protecting data from disclosure due to database server theft, unscrupulous administrators, and insecure networks.

SE-PostgreSQL project SE-Linux will provide row level security features on the par with Oracle – Oracle Label Security and Virtual Private Database

PCI DSS v3 : Objectives

Build and Maintain a Secure Network

Requirement 1: Install and maintain a firewall configuration to protect cardholder data.

Requirement 2: Do not use vendor-supplied defaults for system passwords and other security parameters.

Protect Cardholder Data

Requirement 3: Protect stored cardholder data.

Requirement 4: Encrypt transmission of cardholder data across open, public networks.

PCI DSS v3 : Objectives

Maintain Vulnerability Management Program

Requirement 5: Use and regularly update anti-virus software.

Requirement 6: Develop and maintain secure systems and applications.

Implement Strong Access Control Measures

Requirement 7: Restrict access to cardholder data by business need-to-know.

Requirement 8: Assign a unique ID to each person with computer access.

Requirement 9: Restrict physical access to cardholder data.

PCI DSS v3 : Objectives

Regularly Monitor and Test Networks

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

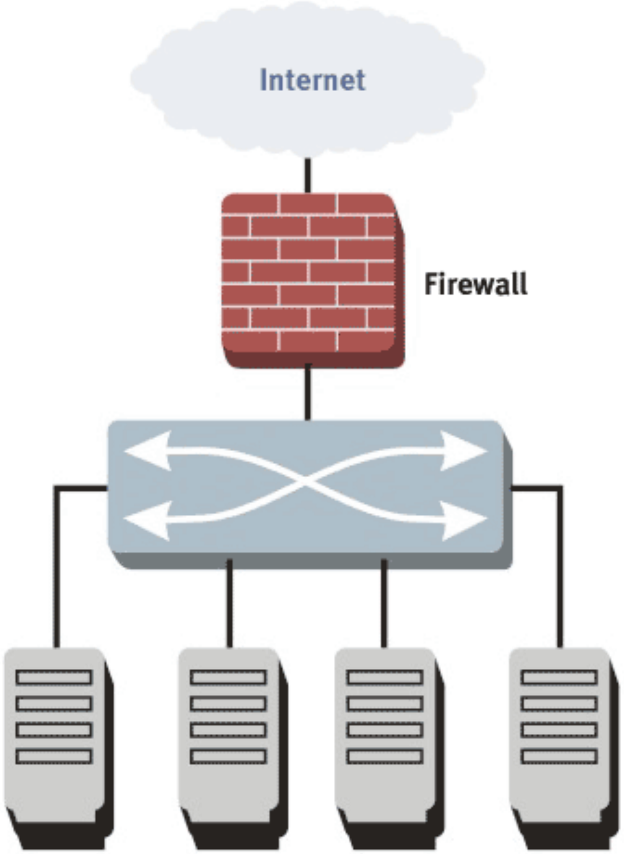
Requirement 11: Regularly test security systems and processes.

Maintain an Information Security Policy

Requirement 12: Maintain a policy that addresses information security.

Requirement # 1

Install and maintain a firewall configuration to protect cardholder data.



Requirement # 1

1.2 "Build a firewall configuration that denies all traffic from "untrusted" networks and hosts, except for protocols necessary for the cardholder data environment."

PostgreSQL helps to achieve this requirement:

- **pg_hba.conf** built-in security feature helps to deny or limit database access from IP address ranges that are deemed "untrusted."
- **log_connections/disconnections** settings helps to centralize database connection logs to be kept at centralized place

Requirement # 2

Do not use vendor-supplied defaults for system passwords and other security parameters.

- Use a password for default “postgres” user
- Don’t allow trust authentication from any host/user/database in pg_hba.conf file. Not even from localhost and postgres user.
- Revoke remote login access on template1 and postgres default databases
- Grant limited permissions to monitoring user.
- Revoke objects privileges from PUBLIC
- Be careful about GRANTS
- Security Definer functions and Views are your friends

Example: Monitoring User

```
CREATE ROLE circonus
       WITH NOSUPERUSER   NOCREATEROLE
      NOCREATEDB  LOGIN PASSWORD 'XXX';

ALTER ROLE circonus SET search_path TO
      secure_check_postgres, pg_catalog;

SET search_path to secure_check_postgres;

CREATE FUNCTION pg_ls_dir(text) RETURNS SETOF text
       AS $$begin
       return query(select pg_catalog.pg_ls_dir('pg_xlog'));
       end$$    LANGUAGE plpgsql SECURITY DEFINER;
```

Requirement # 3

Protect stored cardholder data.

“Protection methods such as encryption, truncation, masking, and hashing are critical components of cardholder data protection. If an intruder circumvents other network security controls and gains access to encrypted data, without the proper cryptographic keys, the data is unreadable and unusable to that person.....”

3.1 a Implement data retention policy (Quarterly)

3.1 b Re-encrypt and rehash data whenever an employee with prior access to the system leaves the company

3.3 Mask PAN when displayed (the first six and last four digits are the maximum number of digits to be displayed).

Requirement # 3

3.4 Render PAN unreadable anywhere it is stored (including on portable digital media, backup media, and in logs) by using any of the following approaches:

- One-way hashes based on strong cryptography (hash must be of the entire PAN)
- Truncation (hashing cannot be used to replace the truncated segment of PAN) ,
- Index tokens and pads (pads must be securely stored) Strong cryptography with associated key-management processes and procedures

Requirement # 3

3.5.1 Restrict access on encrypted keys to limited number of custodians

3.6.6 Verify that key-management procedures are implemented to require **split knowledge and dual control** of keys (for example, requiring two or three people, each knowing only their own part of the key, to reconstruct the whole key).

Requirement # 3

- http://www.postgresql.org/docs/9.3/static/pgcrypto.html
  - Example:
    - pgp_pub_decrypt(msg bytea, key bytea [, psw text [, options text]]) returns text
    - pgp_pub_decrypt_bytea(msg bytea, key bytea [, psw text [, options text]]) returns bytea
- Never hash a card without a salt, and preferably use variable salts
- Don't use md5, use something better like AES or SHA-256
- Column level Encryption using pgcrypto

Example- Encrypt Card Holder Data

```
CREATE OR REPLACE FUNCTION cc.insert_cc(p_cc_number text)
 RETURNS bigint
 LANGUAGE plpgsql
 SECURITY DEFINER
AS $function$
DECLARE

v_hashed_cc       bytea;
v_key_id          text;
v_newcardid       bigint;
v_oldcardid       bigint;
v_pubkey          bytea;
v_row             record;
v_salt            text;
v_salt_file       text;

BEGIN
IF p_cc_number IS NULL THEN
   RAISE EXCEPTION 'Cannot accept NULL for credit card number';
END IF;
FOR v_row IN SELECT salt_file FROM key.hash WHERE active = true ORDER BY
      key_timestamp DESC
```

Example- Encrypt Card Holder Data

```
LOOP
  v_salt := pg_read_file(v_row.salt_file);
  v_hashed_cc := pgcrypto.digest(v_salt || p_cc_number, 'sha512');
  SELECT cc_card_id INTO v_oldcardid FROM cc.creditcard WHERE
   salt_file = v_row.salt_file AND cc_hash = v_hashed_cc;
  IF v_oldcardid IS NOT NULL THEN
    EXIT;
  END IF;
END LOOP;

IF v_oldcardid IS NOT NULL THEN
  INSERT INTO audit.access_log (pid, role, hostname, ipaddr, port,
   access_time, activity_type, cc_card_id)
    SELECT pid, usename, client_hostname, client_addr, client_port,
   query_start, 'insert_cc: returned existing card_id', v_oldcardid
    FROM pg_stat_activity
  WHERE pid = pg_backend_pid();
  RETURN v_oldcardid;
ELSE
```

Example- Encrypt Card Holder Data

```
SELECT key_id, pubkey INTO v_key_id, v_pubkey FROM key.pgp WHERE active = true
      ORDER BY key_timestamp DESC LIMIT 1;
   SELECT salt_file INTO v_salt_file FROM key.hash WHERE active = true ORDER BY
      key_timestamp DESC LIMIT 1;
   v_salt := pg_read_file(v_salt_file);
   v_hashed_cc := pgcrypto.digest(v_salt || p_cc_number, 'sha512');

   INSERT INTO cc.creditcard (cc_number
      , cc_hash
      , pgp_key_id
      , salt_file)
      VALUES (pgcrypto.pgp_pub_encrypt(p_cc_number, v_pubkey)
      , v_hashed_cc
      , v_key_id
      , v_salt_file) RETURNING cc_card_id INTO v_newcardid;
   INSERT INTO audit.access_log (pid, role, hostname, ipaddr, port, access_time,
      activity_type, cc_card_id)
      SELECT pid, usename, client_hostname, client_addr, client_port, query_start,
      'insert_cc: successfully inserted new card_id', v_newcardid FROM pg_stat_activity
      WHERE pid = pg_backend_pid();
   RETURN v_newcardid;
   END IF;              END; $function$
```

Example- Decrypt Card Holder Data

```
CREATE OR REPLACE FUNCTION cc.get_cc_number(p_cc_card_id bigint)
 RETURNS text
 LANGUAGE plpgsql
 SECURITY DEFINER
AS $function$
DECLARE

v_activity_type     text;
v_key_id            text;
v_privkey           bytea;
v_privkey_file      text;
v_privkey_pwd       text;
v_privkey_store     text[][];
v_unencrypted_cc    text;
BEGIN

SELECT pgcrypto.pgp_key_id(cc_number) INTO v_key_id FROM cc.creditcard WHERE
      cc_card_id = p_cc_card_id;

SELECT privkey_file INTO v_privkey_file FROM key.pgp WHERE
      pgcrypto.pgp_key_id(pubkey) = v_key_id;
v_privkey := pg_read_binary_file(v_privkey_file);
```

Example- Decrypt Card Holder Data

```
FOR i IN 1..array_length(v_privkey_store, 1)::int LOOP
   IF v_key_id = v_privkey_store[i][1] THEN
      v_privkey_pwd = v_privkey_store[i][2];
      EXIT;
   END IF;
END LOOP;
SELECT encode(pgcrypto.pgp_pub_decrypt_bytea(cc_number, v_privkey,
      v_privkey_pwd), 'escape') INTO v_unencrypted_cc
FROM cc.creditcard
WHERE cc_card_id = p_cc_card_id;
IF v_unencrypted_cc IS NOT NULL THEN
   v_activity_type := 'get_cc_number: successfully returned unencrypted cc_number';
ELSE
   v_activity_type := 'get_cc_number: attempt to get non-existent cc_number';
END IF;
INSERT INTO audit.access_log (pid, role, hostname, ipaddr, port, access_time,
      activity_type, cc_card_id)
   SELECT pid, usename, client_hostname, client_addr, client_port, query_start,
      v_activity_type, p_cc_card_id FROM pg_stat_activity
   WHERE pid = pg_backend_pid();
RETURN v_unencrypted_cc;
   END $function$
```

Requirement # 4

Encrypt transmission of cardholder data across open, public networks.

“Sensitive information must be encrypted during transmission over networks that are easily accessed by malicious individuals….”

The traffic between datacenters is **encrypted at the network layer** (secure VPN, for example)

The **pg_hba.conf** file allows administrators to specify which hosts can use non-encrypted connections (host) and which require SSL-encrypted connections (hostssl).

Encrypt applicable data **before** insert into database

Requirement # 5

Use and regularly update anti-virus software or programs.

"Anti-virus software must be used on all systems **commonly affected** by malware to protect systems from current and evolving malicious software threats...."

Requirement # 6

Develop and maintain secure systems and applications.

> “All critical systems must have **the most recently released, appropriate software patches** to protect against exploitation and compromise of cardholder data by malicious individuals and malicious software.”

- Apply all new security patches **within one month.**
- PostgreSQL Security releases : http://www.postgresql.org/support/security.html

Requirement # 6

6.4 Follow change control procedures for all changes to system components.

- Documentation of impact
- Management sign-off for change
- Testing of operational functionality , results
- Rollback procedures

Requirement # 7

Restrict access to cardholder data by business need-to-know.

- Since PostgreSQL 8.4 provides **column level permissions**
- Use separate schema and revoke all permissions from schema
- Easy to revoke permissions on schema using PostgreSQL 9.0 schema level permissions feature
- Use Group Roles with NOLOGIN to avoid group login i.e “devs”
- pg_hba.conf : Fine Grained Access Control

Requirement # 8

Assign a unique ID to each person with computer access.

- Assign unique ID for all users who have access to card holder data and systems related to it
- Ensure proper highly secure password policies in place for the systems storing credit card
- **Use Two-factor authentication** (for example, Duo, token devices, smart cards, biometrics, or public keys) authentication method.

Requirement # 9

Restrict physical access to cardholder data.

"Any physical access to data or systems that house cardholder data provides the opportunity for individuals to access devices or data and to remove systems or hardcopies, and should be appropriately restricted…"

Requirement # 10

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

- Install **pg_stat_statements** extension to monitor all queries (SELECT, INSERT, UPDATE, DELETE)
- Setup monitor to find out suspicious access on PAN holding table
- Enable connection/disconnection logging
- Enable Web Server access logs
- Monitor Postgres logs for unsuccessful login attempts
- Automated log analysis & Access Monitoring using Alerts
- Keep archive audit and log history for at least one year and for last 3 months ready available for analysis

Requirement # 11

Regularly test security systems and processes.

“System components, processes, and custom software should be **tested frequently to ensure security** controls continue to reflect a changing environment. “

Experience Consulting Companies can provide best practices around security policy, monitoring, and testing.

Requirement # 12

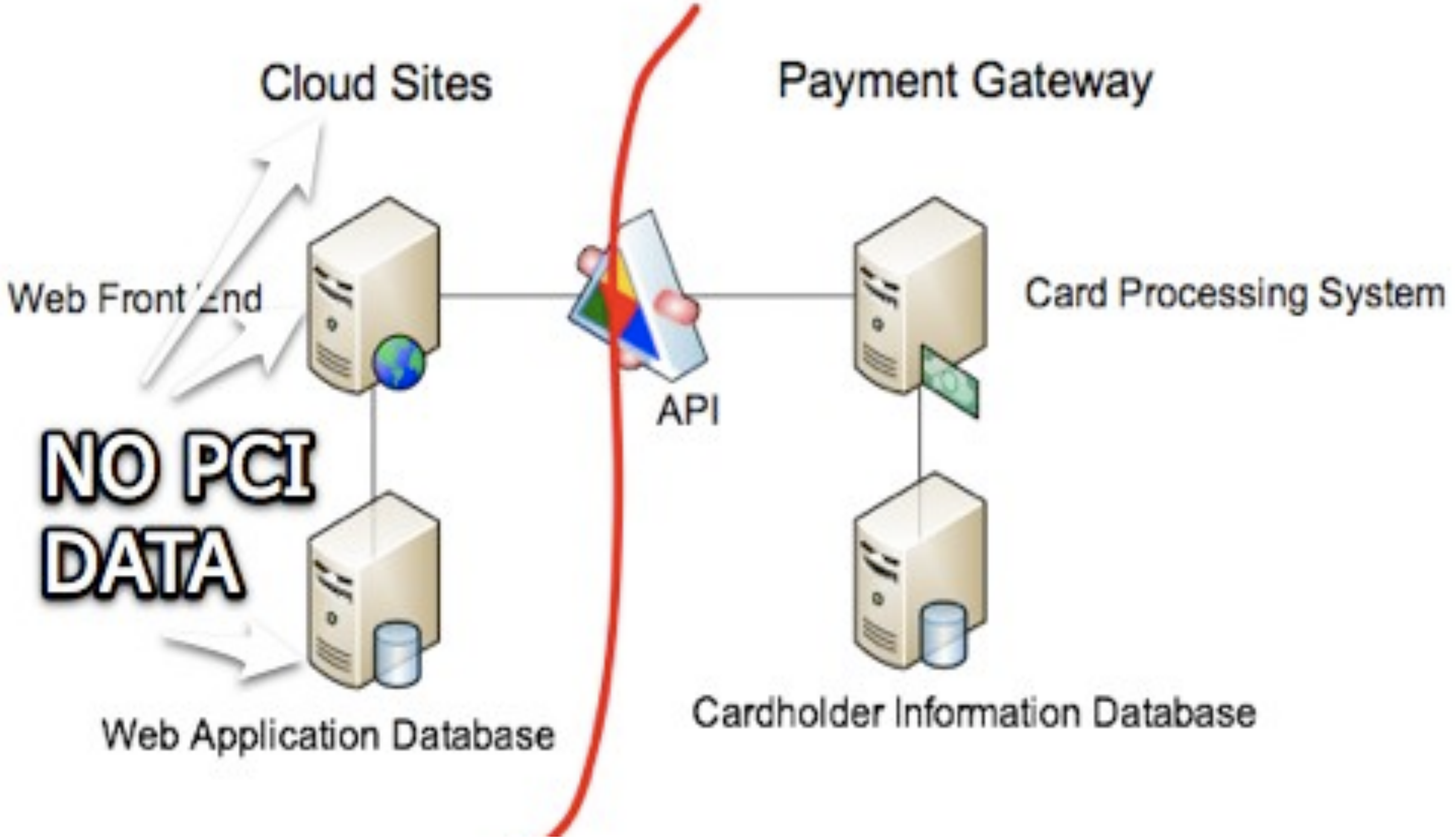
Maintain a policy that addresses information security.

> “A strong **security policy sets the security tone** for the whole company and informs employees what is expected of them. All **employees** should be **aware** of the sensitivity of data and their responsibilities for protecting it... “

Security = **80% of people and processes** + 20% technology

Technology
Process
People

Solution


Cloud Sites
Payment Gateway
Web Front End
Card Processing System
API
NO PCI DATA
Web Application Database
Cardholder Information Database

Common Myths of PCI DSS

Myth 1 – One vendor and product will make us compliant

Myth 2 – Outsourcing card processing makes us compliant

Myth 3 – PCI compliance is an IT project

Myth 4 – PCI will make us secure

Myth 5 – PCI is unreasonable; it requires too much

Myth 6 – PCI requires us to hire a Qualified Security Assessor

Myth 7 – We don’t take enough credit cards to be compliant

Myth 8 – We completed a SAQ we’re compliant

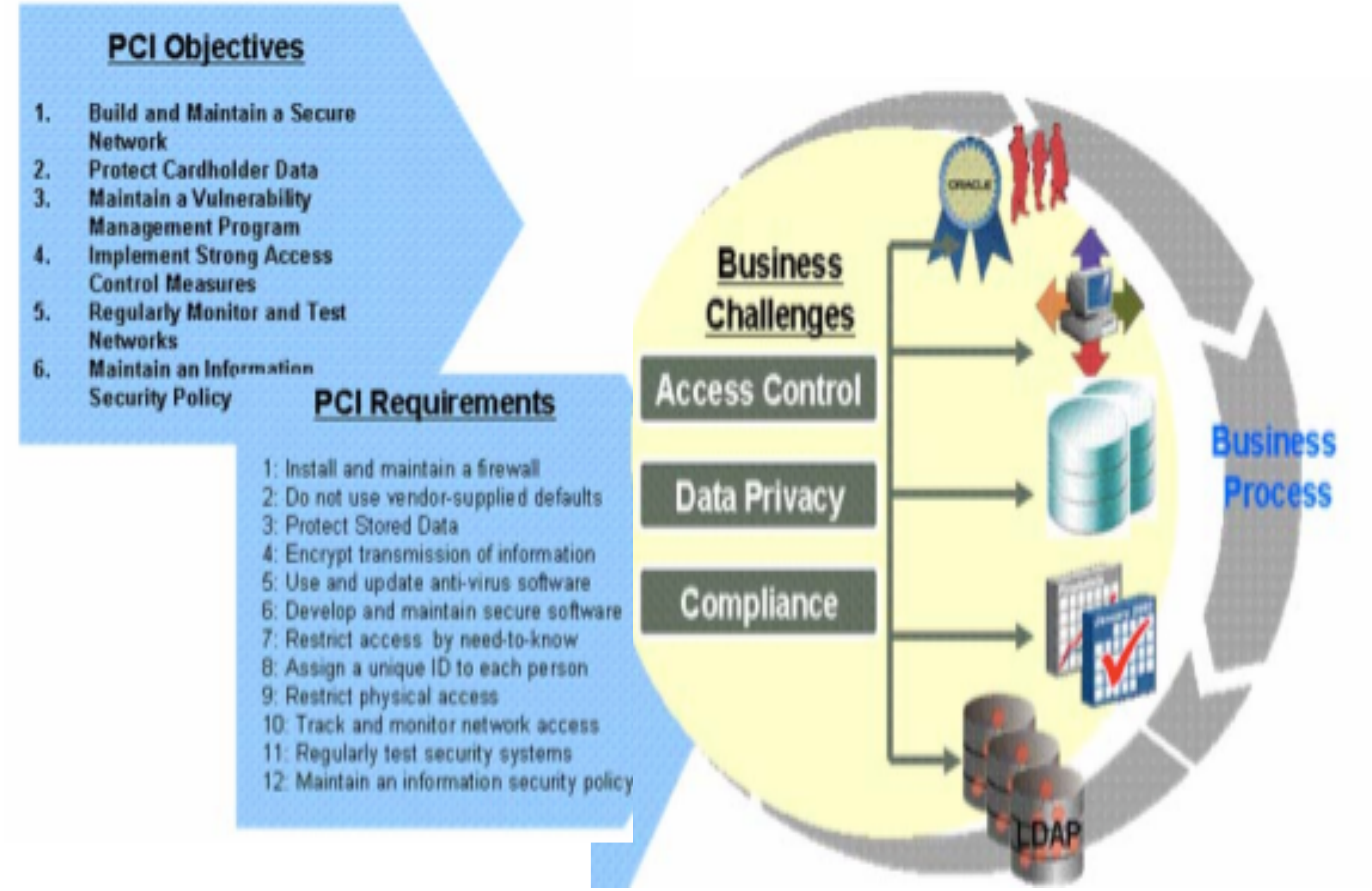
Myth 9 – PCI makes us store cardholder data

Myth 10 – PCI is too hard

Take away ....

- Security first, **Compliance is result**.
- Think beyond credit card data and **grow overall security**!!
- Develop "**Security and Risk**" mindset , not "compliance and audit" mindset.
- Security is your **goal**!!
- **Stop complaining** about it and start doing it!!
- PCI Compliance is **business requirement,** it's not an IT issue.

Conclusion



PCI Objectives

1. Build and Maintain a Secure Network
2. Protect Cardholder Data
3. Maintain a Vulnerability Management Program
4. Implement Strong Access Control Measures
5. Regularly Monitor and Test Networks
6. Maintain an Information Security Policy

PCI Requirements

1: Install and maintain a firewall
2: Do not use vendor-supplied defaults
3: Protect Stored Data
4: Encrypt transmission of information
5: Use and update anti-virus software
6: Develop and maintain secure software
7: Restrict access by need-to-know
8: Assign a unique ID to each person
9: Restrict physical access
10: Track and monitor network access
11: Regularly test security systems
12: Maintain an information security policy

Thanks

- PgConf NYC Conference Committee
- OmniTi
- You!!

We are hiring!! Apply @ l42.org/lg

References

- https://www.pcisecuritystandards.org/documents/PCI_DSS_v3.pdf
- https://www.pcisecuritystandards.org/documents/DSS_and_PA-DSS_Change_Highlights.pdf
- https://www.pcisecuritystandards.org/security_standards/pci_dss_supporting_docs.shtml
- https://www.pcisecuritystandards.org/pdfs/pciscc_ten_common_myths.pdf