Summary of PCI-DSS Requirement 2: "No Defaults"

This requirement focuses on the security principle of **not using vendor-supplied defaults** for system passwords and security parameters. It emphasizes the need to eliminate **known vulnerabilities** by removing defaults and ensuring that configurations are secure and compliant with security standards.

Key Components of Requirement 2

1. Changing Default Settings (2.1):

- **Why?**: Vendor-supplied default settings, such as default passwords, account names (e.g., "admin"), encryption keys, and connection strings, are publicly known and vulnerable to attackers.
- **What to Do?**: Change all default passwords, account names, encryption keys, and other parameters during the initial configuration or deployment of systems.
- **Best Practice**: Disable unused or unnecessary accounts (e.g., "guest").

2. Configuration Standards (2.2):

- **Why?**: Standards must be set for how systems are configured, ensuring that vulnerabilities are addressed during installation.
- **What to Do?**:
    - Apply a configuration standard for each deployed system.
    - Remove or disable unnecessary services and components.
    - Ensure that functions are isolated by security levels (e.g., don’t mix high- and low-security functions on the same server).

3. Encrypt All Admin Access (2.3):

- **Why?**: Admin credentials are a critical target for attackers. If an attacker breaches the network and admin access is not encrypted, they can easily capture credentials.
- **What to Do?**:
    - Ensure all admin access is encrypted using **strong cryptography** (NIST, OWASP recommendations).
    - Avoid using plaintext protocols like **telnet** or **HTTP** for admin access.

4. Asset Inventory (2.4):

- **Why?**: You cannot protect what you don’t know you have. Maintaining an up-to-date inventory of all **PCI-DSS scope assets** is crucial for security.
- **What to Do?**:
    - Create and maintain an inventory of assets that are in scope for PCI-DSS compliance.
    - Use automated tools to regularly scan and validate your asset inventory.
    - Update this inventory frequently with changes from **configuration management (CM)**.

5. Policies and Procedures (2.5):

- **Why?**: Documenting and enforcing security policies ensures that all team members understand and adhere to these security practices.
- **What to Do?**:
    - Document security policies that cover password management, system configurations, and encryption standards.
    - Ensure all employees are trained and follow these policies.

6. Isolate Multitenant Clients (2.6):

- **Why?**: In **shared hosting environments**, it's essential to ensure tenants do not have access to each other's data.
- **What to Do?**:
    - Isolate environments and ensure tenants do not share systems, services, or applications that could allow cross-tenant data access.
    - Follow PCI-DSS Appendix A1 guidelines for securing multitenant environments.

Implementation Steps for Each Role in the Organization

1. DevOps Team

- **Responsibilities**:
    - Change all default settings during system deployments.
    - Set up encryption protocols for admin access.
    - Isolate security-sensitive functions to dedicated servers or containers.
- **Tools to Use**:
    - **Encryption Tools**: SSL/TLS configurations.
    - **Automated Configuration Tools**: Chef, Ansible for applying configuration standards.
    - **Asset Inventory**: Use tools like **CMDB** (Configuration Management Database) to maintain inventory.
- **Documentation**:
    - Maintain logs of all changes to defaults and encryption settings.
    - Ensure security standards are met and documented.

2. System Administrators

- **Responsibilities**:
    - Ensure all admin access uses **strong encryption** and remove plaintext protocols.
    - Regularly audit systems for any remaining default settings.
    - Maintain access control lists and enforce policies.
- **Tools to Use**:
    - **Audit and Monitoring Tools**: To track admin logins and system configurations.
    - **Password Management Tools**: Use enterprise password managers to securely store admin credentials.

3. IT Security Team

- **Responsibilities**:
  - Create and update security policies that reflect PCI-DSS requirements.
  - Regularly audit systems for compliance with these security policies.
  - Enforce multitenant isolation if applicable.
- **Tools to Use**:
  - **Security Information and Event Management (SIEM)** tools for monitoring.
  - **Vulnerability Scanners**: Regular scans for outdated configurations and defaults.
- **Documentation**:
  - Maintain reports on security audits and any vulnerabilities found during scans.
  - Create training materials for employees on the importance of removing defaults.

4. Project Managers

- **Responsibilities**:
  - Oversee the implementation of PCI-DSS requirements in all new projects.
  - Ensure that the project includes budget and timelines for applying security standards.
  - Communicate security needs across teams.
- **Tools to Use**:
  - **Project Management Tools**: Jira, Asana for task assignment and tracking.
  - **Compliance Checklists**: Ensure that PCI-DSS requirements are met at each phase of deployment.
- **Documentation**:
  - Ensure that all project documentation reflects security measures and best practices for removing defaults.

Example: How a Company Can Apply Requirement 2

Scenario: New Server Deployment for an E-commerce Site

1. **Step 1**: The **DevOps team** spins up a new instance for the **web server**. They must immediately:
   - Change the default credentials (passwords, usernames like “admin”).
   - Disable any unused default accounts (e.g., "guest").
   - Set up a **secure configuration standard** that isolates security functions.
2. **Step 2**: The **System Administrators** ensure that **admin access** to this server is encrypted, and they configure secure protocols (e.g., SSL/TLS).
   - All access is monitored via an audit tool that logs admin login attempts.
   - Ensure all remote access tools (e.g., SSH) use strong encryption and not plaintext.
3. **Step 3**: The **IT Security Team** updates the inventory to include this new server.
   - They run a port scan to identify which services are running and ensure that no unnecessary ports are open.
   - Ensure the server is added to the **Configuration Management System (CMDB)** for regular compliance checks.

4. **Step 4**: The **Project Manager** ensures that all tasks related to PCI-DSS compliance are completed and documented.
    - They create tickets for all necessary security steps (changing defaults, encryption, isolation) and ensure they are tracked in the project management tool (e.g., Jira).

Conclusion and Best Practices for the Company

- **Change Defaults**: All systems must have default credentials, account names, and encryption keys changed upon deployment.
- **Encrypt Admin Access**: Never allow unencrypted admin access, and always use strong cryptography.
- **Inventory Management**: Ensure a real-time inventory of PCI-DSS-scope assets is maintained and updated regularly.
- **Isolate Critical Functions**: If you are a multitenant provider, ensure tenants' data is isolated and protected.
- **Document and Enforce**: Policies for securing systems must be documented and enforced across the organization, with regular audits for compliance.

Key Tools:

- **Encryption tools** (SSL/TLS)
- **Configuration Management tools** (Chef, Ansible)
- **Password Management tools**
- **Inventory and Audit tools**

By applying these practices, a company can **minimize vulnerabilities**, meet **PCI-DSS compliance** standards, and ensure a **secure IT infrastructure**.

Here are real-world examples for each of the tasks related to **Configuration Standards (2.2)**, along with specific guidelines for implementing them in a company:

---

1. Apply a Configuration Standard for Each Deployed System

Example:

In a company deploying a new **web server**, the system administrator should apply predefined configuration standards to ensure that the server is secure from the outset.

- **What to Do**:
    - Use predefined templates, such as **CIS (Center for Internet Security) benchmarks** for web servers, databases, and operating systems.
    - Ensure that each new server follows these benchmarks during installation.

Real-World Scenario:

The company launches a new **web application** on an **Nginx web server**. Instead of using default settings, the DevOps team applies a **CIS-compliant configuration**. This ensures that unnecessary features, such as directory browsing or weak encryption protocols, are disabled.

Tools:

- **Ansible** or **Chef** configuration management tools can automatically apply secure configurations during deployment.
- Use **Security Content Automation Protocol (SCAP)** to check and enforce configuration standards.

---

2. Remove or Disable Unnecessary Services and Components

Example:

When setting up a new **Linux server** for hosting a database, the system administrator should disable any **unused services** (e.g., FTP, HTTP if not needed) and uninstall unnecessary software packages.

- **What to Do**:
    - Disable services such as **remote desktop** if it's not required for server management.
    - Uninstall **default packages** that aren't needed, such as **Apache** or **PostgreSQL**, on a server that only requires **MySQL**.

Real-World Scenario:

In a financial institution, the server hosts a **database** that only communicates via **MySQL**. However, the default installation also enables services such as **FTP** and **Telnet**. The system administrator disables these services to prevent unnecessary exposure.

Tools:

- Use **Linux service management tools** like `systemctl` to disable unwanted services.
- Regularly run tools like **nmap** to check open ports and identify unnecessary services.

---

3. Ensure that Functions Are Isolated by Security Levels

Example:

When deploying a new system, the company must ensure that **critical functions** (e.g., payment processing) are isolated from **lower-security functions** (e.g., user registration).

- **What to Do**:
    - Separate the web application's **public-facing interface** from the **backend system** that handles sensitive data such as payments or user credentials.
    - Use **network segmentation** to create different security zones (e.g., DMZ for web traffic and internal zone for sensitive data).

Real-World Scenario:

In an e-commerce platform, the company separates its **user-facing web application** from the internal **payment gateway**. The **payment gateway** is placed in an internal network zone, protected by **firewalls**, and only accessible through authorized API calls from the web server.

Tools:

- **VLANs** and **firewall rules** can isolate network segments.
- Tools like **AWS Security Groups** or **Azure Network Security Groups** can be used in the cloud to enforce security isolation.

---

4. Encrypt All Admin Access

Example:

The system administrator must ensure that all remote administrative access is conducted over **encrypted channels** such as **SSH** instead of plaintext protocols like **Telnet**.

- **What to Do**:
    - Use **SSH** with **public/private key authentication** for remote administration.
    - Avoid using **HTTP** for management consoles; instead, use **HTTPS** with SSL/TLS certificates.

Real-World Scenario:

In a cloud environment, the system administrator disables **Telnet** and uses **SSH** to access remote servers. Additionally, they configure the **web server management console** to be accessible only via **HTTPS**, ensuring that administrative credentials are not exposed.

Tools:

- Use **SSH** with **key-based authentication**.
- Deploy **SSL/TLS certificates** for any admin-facing interfaces.

---

5. Asset Inventory

Example:

The IT security team must maintain an accurate and up-to-date inventory of all hardware, software, and network resources that are in scope for **PCI-DSS compliance**.

- **What to Do**:
    - Use an **automated inventory tool** to track all devices and software.
    - Regularly audit the inventory to ensure that it reflects current configurations.

Real-World Scenario:

A large retail company uses a tool like **ServiceNow CMDB** to track all devices that store, process, or transmit **cardholder data**. The IT team regularly updates this inventory whenever a new server is deployed or decommissioned.

Tools:

- Use tools like **CMDB**, **Asset Panda**, or **AWS Systems Manager** to track inventory.
- Run **automated discovery scans** to ensure no system is missed.

---

6. Isolate Multitenant Clients

Example:

In a **cloud hosting environment**, companies must ensure that **multitenant** systems properly isolate data and access between tenants.

- **What to Do**:
  - Use **virtualization** or **containerization** to ensure that tenants don't share the same data or services.
  - Implement strict **access control policies** to prevent cross-tenant data access.

Real-World Scenario:

A cloud service provider hosting multiple clients on shared infrastructure uses **Kubernetes namespaces** to isolate each client's application and data. Additionally, **AWS Identity and Access Management (IAM)** is configured to ensure that each client only has access to their resources.

Tools:

- Use **Docker** or **Kubernetes** to isolate tenant workloads.
- Implement **Role-Based Access Control (RBAC)** to limit what tenants can access.

Summary and Best Practices for the Company

1. **Apply Configuration Standards**: Use pre-defined standards like **CIS Benchmarks** to ensure secure configurations for all deployed systems.
2. **Remove Unnecessary Services**: Disable any unused services to reduce the attack surface.
3. **Function Isolation**: Separate critical functions (e.g., payments) from less sensitive ones (e.g., user accounts) using **network segmentation**.
4. **Encrypt Admin Access**: Always use **SSH** and **HTTPS** to secure administrative access to servers.
5. **Maintain an Asset Inventory**: Regularly update asset inventories using automated tools to ensure accurate tracking of all systems.
6. **Isolate Multitenant Clients**: Ensure that tenants are isolated in shared environments to prevent cross-tenant data breaches.

These steps help ensure that **PCI-DSS Requirement 2** is met, securing the company's systems from known vulnerabilities and minimizing the risk of a breach.