# Project Specification: Automated Monthly Cybersecurity Report Generation

## Objective

To automate the extraction of data from PDF reports generated by Datto, process and store the extracted data in a local SQL database, and generate a structured, user-editable report in Microsoft Word (.docx) format. The report will provide a comprehensive analysis of security, device performance, backup status, patch compliance, and recommendations for clients.

## Requirements

### 1. PDF Data Extraction

- Automated Scanning: The script will scan a designated folder for new PDFs.
- Text Extraction: Use pdfplumber to extract textual content from reports.
- Table Extraction: Use Camelot to extract structured data, such as patch compliance, device health, and storage statistics.
- Data Categorization: Separate extracted data into tables and standalone text, mapping them to the relevant report sections.

### 2. Database Storage

- Schema Design: Each PDF type (e.g., "Device Performance," "Backup Metrics") will have a corresponding schema.
- Table Storage: Tables extracted from PDFs will be stored as SQL tables.
- Text Storage: Non-tabular text will be stored in fields linked to appropriate report sections.
- Query Optimization: Structure the database for efficient retrieval of data for report generation.

### 3. Report Generation

- Structured Template: The report will follow a standardized format with key sections:
  - Executive Summary
  - Device and Endpoint Health
  - Backup and Continuity
  - Security Metrics
  - Incident Mitigation
  - Recommendations

    - Scorecard
- Content Insertion:
    - Populate tables from the SQL database into the report.
    - Insert extracted text into relevant sections (e.g., recommendations, incident reports).
- Formatting: Ensure tables and text are inserted correctly and formatted consistently.

**4. Report Template Design**

- Header & Footer: Include branding elements such as logos and company information.
- Dynamic Content:
    - Tables populated from extracted PDF data.
    - Text content dynamically inserted into predefined sections.
- Editable Output: The final report will be in .docx format for easy modifications.

**5. Template Management**

- Reusable Structure: The template will be stored separately for consistency across clients and reporting periods.
- Version Control: The script should fetch the latest template version for each report generation.

**Cybersecurity Measures**

To ensure data security and integrity throughout the process, the following cybersecurity best practices will be implemented:

**1. Logging & Monitoring**

- Implement detailed logging for all operations, including PDF processing, database interactions, and report generation.
- Store logs securely and enable monitoring for anomalies or failed operations.
- Set up alerts for suspicious activities or unauthorized access attempts.

**2. Access Control & Authentication**

- Restrict database access to authorized users with role-based permissions.
- Implement authentication and authorization mechanisms to prevent unauthorized modifications.
- Use environment variables to store database credentials securely instead of hardcoding them.

### 3. Data Encryption

- Encrypt sensitive data stored in the database to protect against unauthorized access.
- Use SSL/TLS encryption for data transmission to prevent interception.

### 4. Secure Storage & File Handling

- Store extracted data in a protected environment with proper access controls.
- Validate all PDF inputs to prevent processing of malicious files (e.g., through file integrity checks).

### 5. Regular Updates & Security Audits

- Keep dependencies (e.g., pdfplumber, Camelot, python-docx) up to date to mitigate vulnerabilities.
- Conduct periodic security audits to assess and improve the system's defenses.

## Implementation Steps

1. **PDF Extraction:**
    - Develop a Python script to scan a folder and extract text and tables from PDFs using pdfplumber and Camelot.
    - Categorize extracted data and prepare it for database insertion.
2. **Database Setup:**
    - Define SQL schema with separate tables for different data types (e.g., device health, patch compliance).
    - Normalize data storage to maintain relational integrity.
3. **Report Template Creation:**
    - Design a .docx template with placeholders for dynamic content.
    - Set up structured sections and formatting for consistency.
4. **Report Generation:**
    - Develop a Python script using python-docx to pull data from the database and insert it into the Word template.
    - Format tables and text correctly for readability and professionalism.
5. **Template Updates & Versioning:**
    - Allow easy updating of templates while maintaining consistency across reports.
6. **Testing & Validation:**
    - Use sample PDFs to validate accurate extraction, storage, and report generation.
    - Verify final reports for consistency, accuracy, and formatting.

## Deliverables

- Python Script for:
    - Extracting text and tables from PDFs.
    - Storing extracted data in an SQL database.
    - Generating a structured Word report from the stored data.
- Reusable Report Template ensuring consistent structure and branding.
- Instructions on:
    - Running the script for monthly report generation.
    - Updating the template for consistency across reporting periods.
- Documentation covering:
    - Database schema structure.
    - Template format and dynamic content integration.
    - Step-by-step report generation process.
    - Security best practices for data handling, logging, and access control.