Data Security Report

Riccardo Berni - Topic 7

November 2024

Contents

1 Introduction

This report outlines the security requirements that the system must meet to ensure robust data protection. The system is required to comply with industry standards, safeguarding information within the database as well as during data transfer and storage.

Organizations tipically refer to **security policies**, documents that define rules, guidelines, and responsibilities essential to ensuring security. These policies usually focus on three main areas: access control, compliance, and incident response. They are designed to uphold the CIA principles (Confidentiality, Integrity, and Availability), which form the foundation of a secure information system:

1. **Confidentiality**: ensures that information is accessible only to authorized users. Measures like encryption, access controls, and user authentication prevent unauthorized access.

2. **Integrity**: protects information from unauthorized modification, maintaining accuracy and trustworthiness. Techniques such as checksums, hashing, and version control help detect and prevent data tampering.

3. **Availability**: ensures that information and resources are accessible to authorized users when needed, through measures like redundancy, load balancing, and disaster recovery plans to prevent downtime.

The report outlines the key data security principles that the system must follow to remain compliant with current regulations, with **NIS2** [1] as the primary reference, along with the resulting requirements. Additionally, it provides recommendations for methodologies and tools (such as software, libraries, etc.) that support effective implementation of these security requirements.

These recommendations are not mandatory; each topic may select tools that best meet their needs, provided they align with the requirements. Compliance will subsequently be verified, and any identified deficiencies will be communicated to the responsible team for resolution.

2 NIS2

The NIS2 Directive (Network and Information Security Directive 2) [1] is an updated EU cybersecurity directive aimed at strengthening the cybersecurity frameworks across EU member states. It replaces the original NIS Directive (2016), expanding its scope and establishing more stringent requirements.

NIS 2 applies to essential and important entities. Companies are considered essential if they operate in one of the highly critical sectors (including manufacturing) and have more than 250 employees or an annual turnover exceeding 50 million euros.

Although not exclusively focused on data security, NIS 2 requires companies to adopt measures to protect data integrity and availability in essential sectors.

It includes requirements such as vulnerability management, network monitoring, and incident notifications in cases where corporate data may be at risk.

The NIS 2 directive introduces new requirements and obligations for organizations in four key areas: Risk Management and Security Measures, Corporate Responsibility, Reporting Obligations, Business Continuity.

2.1 Risk Management and Security Measures

Organizations must implement measures to minimize cybersecurity risks, such as incident management, enhanced supply chain security, improved network security, more effective access control, and encryption.

These measures are closely aligned with the **Principle of Least Privilege**[2], which states that users, applications, and systems should only have the minimum level of access necessary to perform their tasks. This principle reduces security risks by limiting access to sensitive resources and minimizing the potential impact of a breach.

2.2 Corporate Responsibility

NIS 2 mandates that corporate management is directly responsible for overseeing and approving cybersecurity measures within the organizations. This includes ensuring that appropriate policies, procedures, and measures are in place to manage cybersecurity risks. Additionally, management must ensure that employees are adequately trained and that a clear strategy for addressing cybersecurity risks is implemented.

If an organization fails to comply with these requirements, management can be held accountable. Penalties for non-compliance may include fines, as well as temporary bans from holding leadership positions, depending on the severity of the violation. This approach emphasizes the importance of leadership in maintaining robust cybersecurity practices within an organization.

2.3 Reporting Obligations

Entities must have processes in place for promptly reporting security incidents that significantly impact their service delivery or recipients. NIS 2 establishes specific deadlines for notifications: companies have 24 hours to submit an initial alert to the CSIRT or the relevant national authority. The official notification must also be provided within 72 hours of the cybersecurity incident. (Might require an additional non-functional requirement).

2.4 Business Continuity

Companies must have plans in place to ensure business continuity in the event of serious cybersecurity incidents. This plan should include considerations for system recovery, emergency procedures, and the establishment of a crisis response team.

3 Security Requirements

Based on the analysis of NIS 2 measures, the following requirements were identified, each addressing a critical aspect of the project.

3.1 Login requirements

1. The system must encrypt personal data before storing it in the database, using AES [3] as the encryption standard.
2. Passwords must be hashed upon user registration.
3. Ensure that sessions are secure by using HTTP-only and secure cookies (consider implementing session expiration to minimize risk in cases of inactive sessions).
4. Implement logic to verify credentials (consider even implementing two-factor authentication (2FA) [4]).
5. Decrypt data after retrieving it from the database, ensuring that the decryption key is securely managed and not exposed in the code or logs.
6. Implement rate limiting to prevent brute-force attacks and lock out accounts after a certain number of failed attempts to deter attackers.
7. Keep an audit log of login attempts (successful and failed), which can be useful for identifying and analyzing suspicious activities.

3.1.1 How to

Several Python libraries enable AES implementation. AES requires a secret key for both encryption and decryption; as a symmetric algorithm, the same key must be used for both processes.

- **Libraries for Encryption and Lockout Mechanisms**
  - **PyCryptodome**[5]: library for cryptographic operations. It provides tools for encryption, decryption, and secure key management, and supports a wide range of algorithms, including AES, RSA, and SHA, making it versatile for implementing encryption and digital signatures.
  - **cryptography**[6]: library which offers a robust set of tools for cryptographic needs, including symmetric and asymmetric encryption, key generation, and data hashing. It is known for its ease of use and high-level functions, such as *Fernet*, which simplify secure data encryption and decryption.
  - **bcrypt**[7]: library for hashing passwords. It is based on the bcrypt hashing algorithm, which is slow and resistant to brute-force attacks.

  - **Flask** or **Django**: frameworks to ensure secure session configurations and setting session expiration.
  - **pyotp**: library for 2FA, using Time-based One-Time Passwords (TOTP) and HMAC-based One-Time Passwords (HOTP).
  - **ratelimit** or **redis**: libraries for rate limiting and lockout mechanisms.
  - **logging**: module to create audit losg, storing login attemps, IP addresses, timestamps, and user IDs.

To prevent encryption keys from being exposed, environment variables or secure key management services can be used to handle keys safely. Secure key storage minimizes the risk of unauthorized access and helps maintain compliance with security standards..

- **Tools for Secure Key Management**
  - **AWS KMS** or **Azure Key Vault**: these are cloud-based solutions, which provide secure storage and management for encryption keys, offering built-in access control, logging, and automatic key rotation.
  - **dotenv**: python library which loads environment variables from a ".env" file into the application's environment. Keys stored in ".env" files are kept outside of the main application code, which helps to prevent accidental exposure and provides an extra layer of security. To access an encryption key, the application reads the key as an environment variable without hardcoding it in the code.

Note: while external authentication protocols (e.g., OAuth 2.0, OpenID Connect) are available, our system may rely on internal authentication protocols to manage access securely.

3.2 Database encryption requirements

1. Provide *Encryption at Rest*, which protects data when stored in the database, safeguarding it from unauthorized access in the event of a storage system breach.
2. Define access control to enforce the principle of least privilege and implement it within the application.
3. After access control, as for the login, decrypt the data when retrieving it from the database before using it, ensuring that the decryption key is securely managed and not exposed in the code or logs.
4. Define recovery protocols for compromised data to support effective incident response and recovery.

3.2.1 How to

Most modern databases, such as PostgreSQL, MySQL, and SQL Server, offer built-in options for encrypting data at rest, which can be enabled directly through the database configuration.

- **Libraries for encryption at rest**
  - **PyCryptodome** [5]
  - **SQLAlchemy** [8]: python SQL toolkit and Object-Relational Mapping (ORM) library. It allows developers to interact with databases using Python objects instead of writing raw SQL queries.
  - **dotenv**, **AWS KMS** or **Azure Key Vault**: for key management (see login tools).

Implementing fine-grained access control within the application enforces the Principle of Least Privilege, ensuring users only have the necessary access to perform their tasks.

Role-based access control (RBAC [9]) is one common solution: it's a security approach that restricts system access based on users' roles within an organization. Rather than assigning permissions to individual users, RBAC associates permissions with specific roles (e.g., "admin," "editor," "viewer"). Users are assigned to roles based on their responsibilities, and they inherit the permissions tied to their role, ensuring they only access resources necessary for their tasks.

- **Tools for RBAC**
  - **Flask**: to add role-based access control, user authentication, and permission management.
  - **Django**: provides role and built-in permission management for applications using Django (see data transmission tools), allowing to restrict access based on user roles.

For effective incident response, it is essential to regularly back up data using automated tools to ensure quick restoration in case of compromise. Additionally, conducting periodic recovery tests verifies that backups are reliable and that the restoration process is clear and efficient.

- **Tools for automated back up**
  - **azure-storage**: it is the Azure SDK tool for Python, which enables you to interact with Azure Backup and manage backups in Azure Storage, Virtual Machines, and SQL databases.
  - **pg_dump**: a utility for PostgreSQL that can be automated using Python to regularly back up databases.
  - **google-cloud-storage**: it's the Google Cloud Storage python client library.

3.3 Data Transmission requirements

1. Ensure data transmission follows the CIA principles.
2. Implement end-to-end encryption for all data transmissions between system components.
3. Use SHA-2 [10] for data hashing.
4. Provide incident response mechanisms.
5. Periodically test the transmission infrastructure (encryption protocols, key management).

3.3.1 How to

End-to-end encryption ensures data is encrypted on the sender's side and decrypted only on the receiver's side. Using TLS (Transport Layer Security) with HTTPS for web applications or SSL/TLS for API communications is standard practice.

- **Tools for End-to-End encryption**
  - **ssl module** [11]: in python for TLS encryption over sockets.
  - **HTTPS/TLS** [12]: for secure data transmission in web applications.
  - **Let's Encrypt**: free, automated, and open certificate authority (CA) that provides SSL/TLS certificates for websites, enabling HTTPS.

To ensure data integrity during transmission, use SHA-2 hashing (e.g., SHA-256) to create a unique hash of the data before sending it. The receiving system can then hash the received data and compare it with the original hash to detect any alterations.

- **Tools and libraries for SHA-2**
  - **hashlib** [13]: python built-in library which supports SHA-2 algorithms, including SHA-256 and SHA-512.
  - **Django**: open-source web framework for Python, it includes built-in tools and practices to ensure secure communication between clients and servers.

To implement incident response during data transmission, there should be a defined process in place to identify, report, and mitigate any incidents in the event of a transmission breach.

- **Tools and libraries for incident response**
  - **Splunk**: provides real-time monitoring, logging, and incident response.

- **Snort**: widely-used open-source intrusion detection and prevention system. It can be integrated in python by the wrapper **PySnort**.
- **Scapy**: python tool for network traffic analysis, can be used to detect malicious traffic patterns.
- **Suricata**: an open-source Intrusion Detection/Prevention Systems that detects anomalies in network traffic and alerts the system on suspicious events.

To maintain infrastructure security, it is beneficial to conduct regular penetration and encryption testing. Additionally, routine log audits help ensure that security events are recorded and reviewed. Implementing an automated notification process for the security team when suspicious activity is detected further enhances the system's responsiveness to potential threats.

- **Tools and libraries for testing**
  - **OWASP ZAP** [14]: a penetration testing tool focused on finding vulnerabilities in web applications.
  - **Testssl.sh**: an open-source tool to test the configuration of SSL/TLS protocols on servers, verifying encryption strength.
  - **PyCryptoDome**[5]: can be used to test the strength and implementation of encryption.
  - **Pytest**: python testing framework, useful for running automated tests on your security code, such as checking for vulnerabilities in encryption.
  - **Bandit**: a python tool for security linting, which checks your code for security vulnerabilities, including unsafe encryption or poor key management.

4 Other EU Regulations

4.1 Cybersecurity Act

The Cybersecurity Act [15] of the European Union, introduced in 2019, establishes a European framework for cybersecurity certification. Its goal is to enhance user trust by ensuring that products, services, and processes meet adequate security standards. Through a unified certification system, the EU aims to improve cyber resilience and create a single market for digital security.

4.1.1 Requirements

The proposed requirements and corresponding solutions for vulnerability management and continuous security monitoring align well with the directives of the Cybersecurity Act.

A key requirement involves **security certification**, including the identification and evaluation of security risks to information, along with defining control measures to mitigate these risks.

4.2 ISO-IEC 27001

ISO/IEC 27001 [16] is an international standard that specifies the requirements for an Information Security Management System (ISMS). It provides a systematic approach to risk management, helping organizations protect the confidentiality, integrity, and availability of their information.

Although ISO/IEC 27001 is a standard rather than a legal directive like NIS 2, it is frequently used as a foundation for data security compliance. Organizations can seek certification to demonstrate the effectiveness and adequacy of their security measures, ensuring alignment with NIS 2 requirements.

5 Conclusion

This report outlines the essential security requirements and best practices needed for the system to achieve NIS2 compliance. Implementing these measures will strengthen data confidentiality, integrity, and availability, minimizing the risk of unauthorized access and data breaches.

Continual evaluation and adaptation of security measures are essential, as security threats evolve. Regular audits, vulnerability assessments, and user training will help ensure ongoing compliance with NIS2 and reinforce the system's resilience against cybersecurity threats.

References

[1] European Union Agency for Cybersecurity (ENISA). *NIS Directive 2*. 2024. URL: https://www.enisa.europa.eu/topics/cybersecurity-policy/nis-directive-new.

[2] OWASP. *Access control*. 2024. URL: https://owasp.org/www-community/Access_Control.

[3] National Institute of Standards and Technology (NIST). *Advanced Encryption Standard (AES) - FIPS 197*. 2001. URL: https://csrc.nist.rip/encryption/aes/aesfact.html#:~:text=The%20Advanced%20Encryption%20Standard%20%28AES%29%20is%20a%20Federal,U.S.%20Government%20organizations%20to%20protect%20sensitive%2C%20unclassified%20information..

[4] OWASP. *Two-Factor Authentication*. Accessed: 2024-11-10. 2024. URL: https://cheatsheetseries.owasp.org/cheatsheets/Multifactor_Authentication_Cheat_Sheet.html.

[5] PyCryptodome Contributors. *PyCryptodome GitHub Repository*. Accessed: 2024-11-10. 2024. URL: https://github.com/Legrandin/pycryptodome.

[6] OWASP Foundation. *Principles of Cryptography*. 2020. URL: https://owasp.org/www-project-developer-guide/draft/foundations/crypto_principles/.

[7] USENIX. *A Future-Adaptable Password Scheme: bcrypt*. 1999. URL: https://www.usenix.org/legacy/event/usenix99/provos/provos.pdf.

[8] SQLAlchemy Contributors. *SQLAlchemy GitHub Repository*. Accessed: 2024-11-10. 2024. URL: https://github.com/sqlalchemy/sqlalchemy.

[9] National Institute of Standards and Technology. *Role-Based Access Control*. Accessed: 2024-11-10. 2024. URL: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final.

[10] National Institute of Standards and Technology. *Secure Hash Standard (SHS) - SHA-2*. Accessed: 2024-11-10. 2024. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.

[11] Python Software Foundation. *ssl — TLS/SSL wrapper for socket objects*. Accessed: 2024-11-10. 2024. URL: https://docs.python.org/3/library/ssl.html.

[12] Internet Engineering Task Force (IETF). *Transport Layer Security (TLS) Protocol*. Accessed: 2024-11-10. 2024. URL: https://datatracker.ietf.org/doc/html/rfc8446.

[13] Python Software Foundation. *hashlib — Secure hashes and message digests*. Accessed: 2024-11-10. 2024. URL: https://docs.python.org/3/library/hashlib.html.

[14] OWASP Foundation. *ZAP" YOUR APP'S VULNERABILITIES*. 2023. URL: https://wiki.owasp.org/images/2/23/Owasp_ZAP_Final.pdf.

[15] European Parliament and Council of the European Union. *Cybersecurity Act: A Framework for European Cybersecurity Certification*. 2019. URL: https://eur-lex.europa.eu/eli/reg/2019/881/oj.

[16] International Organization for Standardization. *ISO/IEC 27001: Information Security Management Systems — Requirements*. 2013. URL: https://www.iso.org/standard/27001.