

OLUBUNMI ELIZABETH ALBERT-ABU

FOCUS	DATA SECURITY
DATE	30 TH JANUARY, 2024
DECLARATION I certify that this report is entirely my original work and I have fully referenced and correctly cited the work of others, where required.	

TABLE OF CONTENTS

EXECUTIVE SUMMARY	3
DATA SECURITY IN MODERN TECHNOLOGICAL APPLICATIONS	3
TASK 1	4
1.0 INTRODUCTION.....	4
1.1 PASSWORD VALIDATION FOR USER INPUT	4
1.2 PASSWORD ENCRYPTION AND DECRYPTION STRATEGY	5
1.3 MULTIFACTOR AUTHENTICATION	9
1.4 PRESENTATION AND ANALYSIS OF DATA SECURITY CONCEPTS	11
1.5 RECOMMENDATION	12
1.6 CONCLUSION ON TASK 1	12
TASK 2	14
2.0 INTRODUCTION.....	14
2.1 ISO 27001 – PART OF THE EXPANDING ISO/IEC 27000 STANDARDS.....	14
2.2 PCI DSS – PAYMENT CARD INDUSTRY STANDARD	15
2.3 INTEGRATION OF ISO 27001 AND PCI DSS	15
2.4 PAYMENT SYSTEMS - DATA MANAGEMENT FRAMEWORKS, METHODS AND APPROACHES	16
2.5 REFLECTION AND CRITICAL ANALYSIS.....	17
2.6 RECOMMENDATION	17
2.7 CONCLUSION FOR TASK 2.....	18
REFERENCES	19
APPENDIX	22



EXECUTIVE SUMMARY

As opined by Summers (2004), data security means protecting digital data, like those stored in databases, from unauthorised access, corruption or theft. There now exists diverse technological frameworks which has made it imperative for data security and system integrity to stand paramount in ensuring robust operational structure for modern enterprises.

This report showcases a comprehensive exploration of data security within contemporary technological landscapes while basing practical applications on theoretical concepts. This endeavour is facilitated through a detailed analysis of two distinctive yet interconnected use cases: the development of a secure online hotel booking system and the dynamic domain of Point-of-Sale (POS) systems' data management. Each use case serves as a practical canvas for dissecting inherent data security intricacies, implementing solutions, and culminating in critical evaluations and recommendations.

DATA SECURITY IN MODERN TECHNOLOGICAL APPLICATIONS

In 2018, Limor Wainstein expressed in her article on ‘Data Security in Hospitality’ that from the viewpoint of cybercriminals, hospitality seem to offer an ideal target for orchestrating crimes such as identity theft or credit card fraud due to the existence of numerous databases and devices containing both Payment Card Information (PCI) and Personal Identifiable Information (PII).



Fig 1: Data Security in Hospitality
Source: Managed Outsource Solution

Hotel booking systems contains a wealth of sensitive information such as customers personal details, travel itineraries, payment information. A secure design safeguards this data from unauthorised access, mitigating against the risk of data theft and breaches. Some factors to consider when developing such systems are:

- Financial security
- Business continuity
- Trust and reputation
- Protection of sensitive information
- Compliance and legal obligation

TASK 1

ONLINE HOTEL BOOKING SYSTEM – SOFTWARE DESIGN AND SECURITY DEMONSTRATION

1.0 INTRODUCTION

This use case requires the conceptualization of essential data security functionalities for the eventual development of a software design for an online hotel booking system. It necessitates the application of python scripts to develop and test practical implementation of security measures. According to IT Business Edge (2017), ‘you have to know your data to protect your data’; this will enable you ensure efficacy in mitigating potential threats such as cyberattack or data breach.

The focus of this task is on password validation, encryption-decryption strategies and multifactor authentication methodology within the context of user login mechanisms. Each aspect of this task, from password validation protocols to encryption strategies, is critically evaluated in the context of data security risks and vulnerabilities. Jupyter notebook within Anaconda Navigator workspace was used for this task.

1.1 PASSWORD VALIDATION FOR USER INPUT

Password validation is the process of analysing the strength, authenticity and compliance of passwords used to access systems or accounts.

Below is an exploration of this process through the use of Python programming language given the criteria below:

- The password should include capital and lowercase letters, a number and 8 to 12 characters.
- The password should have at least 2 capital or lowercase letters.
- The password should not include any spaces.

PYTHON SCRIPTS IMPLEMENTATION

- First, I import my regular expression module, as seen in fig. 2 below, which checks if the password meets set benchmark. It also counts the number of uppercase and lowercase letters and ensures there are at least 2 of each. The ‘validate password’ function is based on Python’s ‘re’ module for pattern matching.

```
In [7]: import re

def validate_password(password):

    # Checking Length, uppercase, lowercase, and numbers
    if re.match(r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[A-Za-z\d]{8,12}$', password) and ' ' not in password:

        # Counting uppercase and lowercase letters
        count_uppercase = sum(1 for c in password if c.isupper())
        count_lowercase = sum(1 for c in password if c.islower())

        # Validating at least 2 uppercase or 2 lowercase letters
        if count_uppercase >= 2 or count_lowercase >= 2:
            return True
        return False
```

Fig. 2

- Next, I applied password validation to a sample users' input to login:

```
In [8]: # Let's apply password validation for a user input to login with sample user password 'SuccesS100'

password_input = "SuccesS100"
if validate_password(password_input):
    print("Password is valid.")
else:
    print("Password does not meet the criteria.")

Password is valid.
```

Fig. 3

The script above depicts that the user inputted the right password, hence the prompt 'Password is valid' in fig. 3.

- Here, I tested if a user makes a wrong input:

```
In [9]: # Testing the compliance of the user password to set criteria

password_input = "cesS100"
if validate_password(password_input):
    print("Password is valid.")
else:
    print("Password does not meet the criteria.")

Password does not meet the criteria.
```

Fig. 4

From the instance used above in fig. 4, a user will receive a 'Password does not meet the criteria.' Prompt if they input a wrong password.

1.2 PASSWORD ENCRYPTION AND DECRYPTION STRATEGY

Encryption and decryption strategies are processes used to encode and decode information, usually data or messages, to ensure its confidentiality and security during transmission or storage (Clare Stouffer, 2023). As depicted in fig. 5, to encrypt means to convert normal readable messages, that is plaintext, into incoherent jargons called ciphertext. The science of this process is called cryptography and uses algorithms to

generate unique keys used to achieve this transformation. The security of encrypted data relies on the strength of the encryption algorithm and secrecy of the keys.

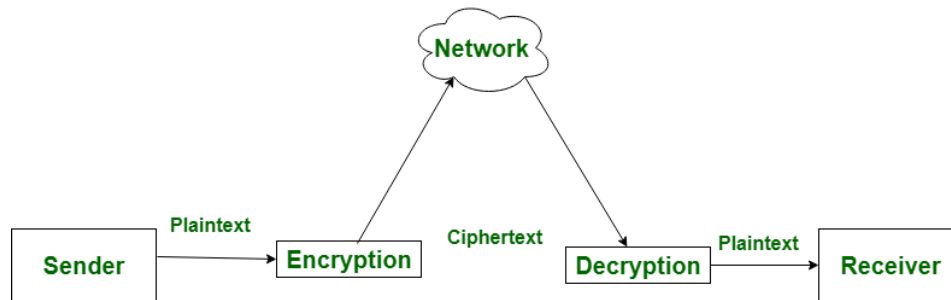


Fig. 5: Process of Cryptography
Source: Geeks for Geeks

PYTHON SCRIPTS IMPLEMENTATION

To explore the functionalities of this data security measure, cryptography hashing will be used below to demonstrate encryption and decryption of a username and password and to simulate successful and unsuccessful login attempts:

- First, import the ‘cryptography’ library and/or the ‘bcrypt’ function as shown in fig. 6. The cryptography library generates a unique key used for the encryption/decryption of password and username while the ‘bcrypt’ function uses the ‘hashing’ method to do the same.

```

In [5]: #Password encryption and decryption

import bcrypt

# Function to encrypt (hash) the password
def encrypt_password(password):
    return bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())

# Function to check if entered password matches the encrypted password
def check_password(entered_password, encrypted_password):
    return bcrypt.checkpw(entered_password.encode('utf-8'), encrypted_password)
  
```

Fig. 6

OR

- I chose the key ‘CAB’ for this data security measure as seen in fig. 7.

```

In [*]: #Password encryption-decryption

from cryptography.fernet import Fernet

# Generate a key for encryption/decryption
CAB = Fernet.generate_key()
cipher_suite = Fernet(CAB)
  
```

Fig. 7

- Next, simulate the registration process as one would on any system or service intended for use. The ‘encrypted_password’ command will encrypt and store the password on the systems’ database (fig. 8).

```
In [13]: # Simulate registration process (encrypt the password)
username = "Greatest"
password = "SuccesS100"

encrypted_password = encrypt_password(password)
```

Fig. 8

- Define the encryption-decryption path for the sample username and password with the code below (fig. 9):

```
In [*]: #Define the encryption-decryption path

def encrypt_password(password):
    return cipher_suite.encrypt(password.encode())

def decrypt_password(encrypted_password):
    return cipher_suite.decrypt(encrypted_password).decode()

def encrypt_username(username):
    return cipher_suite.encrypt(username.encode())

def decrypt_username(encrypted_username):
    return cipher_suite.decrypt(encrypted_username).decode()
```

Fig. 9

- The code below prompts the plaintext username and password variables for input with an associated command to encrypt the password only. See the ciphertext in fig. 10:

```
In [12]: # Example data input to encrypt password

username = input("Enter username: ")
password = input("Enter password: ")

encrypted_pass = encrypt_password(password)
print(f"Encrypted Password: {encrypted_pass}")

Enter username: Greatest
Enter password: SuccesS100
Encrypted Password: b'gAAAAABlj3YCTyEEDbXUj3b2wHkD7ikIerIbmQ4pKEBuX9JwieLy7M5HcXswP7zqaFPqgONQWPE5pMBaN_1fbwir8Fu3OKDgA=='
```

Fig. 10

- To decode the encrypted password, the prompt for ‘password’ should have the ciphertext as its input. This will generate the original plaintext, that is ‘SuccesS100’ (fig. 11).

```
In [13]: # Example data input to decrypt password

username = input("Enter username: ")
password = input("Enter password: ")

decrypted_pass = decrypt_password(encrypted_pass)
print(f"Decrypted Password: {decrypted_pass}")

Enter username: Greatest
Enter password: b'gAAAAABlj3cfQ7tjipn-nBcZXU5agfiqvE_9Gc1d4Pp1CZT6q-20qdPiiXUqCgAelgcgvo01dKYZZuYxXSw9oKX7WP7r6E3TIg=='
Decrypted Password: SuccesS100
```

Fig. 11

- The same cryptography method is used to encrypt and decrypt the username as shown in the scripts below:

```
In [14]: # Example data input to encrypt username

username = input("Enter username: ")
password = input("Enter password: ")

encrypted_user = encrypt_username(username)
print(f"Encrypted Username: {encrypted_user}")

Enter username: Greatest
Enter password: SuccesS100
Encrypted Username: b'gAAAAABlj4fwkRd0kpQqigs0fqQx3PifJgyVVIuC9XV5aemRlRefU6HGZtbr6SLvgHZgSdqpAme2cf5aCjnL992F24bEa4UAAQ=='
```

Fig. 12

```
In [15]: # Example data input to decrypt username

username = input("Enter username: ")
password = input("Enter password: ")

decrypted_user = decrypt_username(encrypted_user)
print(f"Decrypted Username: {decrypted_user}")

Enter username: b'gAAAAABlj4fwkRd0kpQqigs0fqQx3PifJgyVVIuC9XV5aemRlRefU6HGZtbr6SLvgHZgSdqpAme2cf5aCjnL992F24bEa4UAAQ=='
Enter password: SuccesS100
Decrypted Username: Greatest
```

Fig. 13

- Simulating login attempts would inform us of the systems' compliance with set security parameters. Inputting the correct and incorrect passwords and username generated successful and unsuccessful prompts as seen in the script snippets in fig. 14-16 below:

```
In [14]: # Simulate Login attempt with correct password
entered_username = "Greatest"
entered_password = "SuccesS100"

# Check login attempt
if entered_username == username and check_password(entered_password, encrypted_password):
    print("Login Successful")
else:
    print("Login Failed - Incorrect username or password")

Login Successful
```

Fig. 14

```
In [16]: # Simulate Login attempt with incorrect password
entered_username = "Greatest"
entered_password = "Password123"

# Check login attempt
if entered_username == username and check_password(entered_password, encrypted_password):
    print("Login Successful")
else:
    print("Login Failed - Password is incorrect!")

Login Failed - Password is incorrect!
```

Fig. 15


```
In [17]: # Simulate Login attempt with incorrect username
entered_username = "Great"
entered_password = "Success100"

# Check Login attempt
if entered_username == username and check_password(entered_password, encrypted_password):
    print("Login Successful")
else:
    print("Login Failed - Username is incorrect!")

Login Failed - Username is incorrect!
```

Fig. 16

1.3 MULTIFACTOR AUTHENTICATION

Multifactor Authentication, popularly known as MFA or with similar terms like 2-Factor Authentication (2FA) is an authentication method that grants a user access to a digital resource after successfully submitting 2 or more identity proofs. Brian Barrett (2018), a Cyber Security expert, promoted the use of MFA across all digital accounts, especially with the use of an authenticator app.

PYTHON SCRIPTS IMPLEMENTATION

The scripts below demonstrate multifactor authentication using the ‘PyOTP’ library to simulate a Time-Based One-Time-Password (TOTP) scenario. It showcases the following:

- Generation of a random secret key
 - Creation of a TOTP object
 - Generation and verification of a one-time pin
 - Verification after 2 different time delays
- First, install the PyOTP library as shown in fig. 17:

```
In [21]: #Install the PyOTP Library

import sys
import subprocess

# Check if PyOTP is installed
try:
    import pyotp
    print("PyOTP is already installed.")
except ImportError:
    # Install PyOTP using subprocess
    subprocess.check_call([sys.executable, "-m", "pip", "install", "pyotp"])
    print("PyOTP has been successfully installed.")

PyOTP has been successfully installed.
```

Fig. 17

- Next, import the ‘pyotp’ module, generate a random secret key, create a TOTP object, generate an OTP, a one-time pin and verify the user using the codes in fig. 18-20 below:

```
In [18]: # Import the pyotp module

import pyotp

# Generate a random secret key
secret = pyotp.random_base32()

# Create a TOTP object
totp = pyotp.TOTP(secret)

# Generate a time-based one-time password (OTP)
otp = totp.now()
print(f"One-Time Password (OTP): {otp}")
```

One-Time Password (OTP): 276941

Fig. 18

```
In [19]: # Simulate multifactor authentication

pin = totp.now() # Simulate a new one-time pin
print(f"One-Time PIN: {pin}")
```

One-Time PIN: 276941

Fig. 19

```
In [20]: # Verify the user

user_input = input("Enter the OTP: ")
if totp.verify(user_input):
    print("User verified!")
else:
    print("Verification failed!")
```

Enter the OTP: 276941
User verified!

Fig. 20

- A time delay of 30 seconds is then used to accurately test the authenticity of the OTP. The function 'time' was used for this purpose. The verification failed because the user had successfully used the OTP previously as seen in fig. 21 below.

```
In [22]: import time

# Wait for 30 seconds
time.sleep(30)

# Verify again after waiting
user_input = input("Enter the OTP again: ")
if totp.verify(user_input):
    print("User verified!")
else:
    print("Verification failed!")
```

Enter the OTP again: 276941
Verification failed!

Fig. 21

- In fig. 22-23 below, a shortened period interval of 25 seconds was tested after a new OTP had been generated and used. Once again, the authenticity test of the OTP was successful.

```

In [25]: # Generate a time-based one-time password (OTP)
otp = totp.now()
print(f"One-Time Password (OTP): {otp}")

# Verify the user again

user_input = input("Enter the OTP: ")
if totp.verify(user_input):
    print("User verified!")
else:
    print("Verification failed!")

One-Time Password (OTP): 838510
Enter the OTP: 838510
User verified!

```

Fig. 22

```

In [26]: # Wait for 25 seconds
time.sleep(25)

# Verify again after waiting
user_input = input("Enter the OTP again: ")
if totp.verify(user_input):
    print("User verified!")
else:
    print("Verification failed!")

Enter the OTP again: 838510
Verification failed!

```

Fig. 23

1.4 PRESENTATION AND ANALYSIS OF DATA SECURITY CONCEPTS

According to a Trustwave cybersecurity publication by Marcus Law (2023), nearly 31% of hospitality companies have formally announced a data breach in their organisation's history with the average cost of a breach amounting to approximately £2.7 million. As reported by Cornell University and FreedomPay in the same year, 89% of the aforementioned companies have been attacked more than once annually. Some of the reported tactics used for these cybercrimes are:

- HTML file attachments made up 50% of email-borne malware.
- 26% of reported exploits aimed at obtaining credential access were executed through brute force attacks.
- Cyber attackers utilised MOVEit RCE's zero-day vulnerability to perpetrate Clon ransomware attacks on 150+ victims within the hospitality sector.

Analysing data security concepts such as those explored in this task for an online hotel booking system and others widely used, calls for deliberate focus on the design, development and operation of such technology. Robust data security measures and strategies are quintessential in fostering customer trust and protecting sensitive information in the hospitality industry and beyond. There are inexhaustive issues that can

arise when developing a software such as this and it calls for stringent attention to detail, comprehensive software testing and expertise of the developers.

1.5 RECOMMENDATION

The following data security recommendations are made for any company within the hospitality industry aiming to strengthen their current digital protection systems:

- Adhere to data protection regulations.
- Address the criticality of securing payment transactions and financial data in compliance with relevant standards.
- Implement technology such as SSL/ TLS protocols, EMV standards and tokenization methods that ensures data integrity and confidentiality during online transactions.
- Implement multifactor authentication for user accounts, adopting secure payment APIs and integrating advance decryption algorithms, such as AES, for data protection.
- Maintain system integrity through increased risk assessments/audits, continuous monitoring and mitigation of security vulnerabilities by using vulnerability assessment tools.
- Ensure secure coding practices and manage data access control.
- Incorporate privacy-enhancing features adhering to GDPR principles.
- Safeguard sensitive data by using secure database storages and database management systems (DBMS) for user information and transactional data. Adhere to ISO/IEC 27002 for secure storage and handling of data.

These measures will collectively ensure the booking system's resilience against cyber threats which will elevate users' trust in the platform as it exemplifies the hotel's commitment to data security and user privacy.

1.6 CONCLUSION ON TASK 1

The significance of the data security aspects covered in this report are pivotal to fostering customer loyalty, mitigating risks associated with data breaches and ensuring compliance to data principles, industry standards and regulation. A fortified security infrastructure solidifies the digital platform's reliability and hotel's credibility among users and

stakeholders. This is integral for the sustainable and trustworthy functioning of an online hotel booking system in today's digitally reliant world.

TASK 2

2.0 INTRODUCTION

ISO 27001 AND PCI DSS – DATA SECURITY FOR MODERN IT COMPANIES

Cybersecurity Standards or Information Security Standards are published techniques aimed at protecting the cyber environment of an organization or a user (Willie May, 2014). The key aim of these standards is to reduce all risks associated with cyberattacks. They generally emerged in the 1990s as a result of the collaborative work of users and providers at the Stanford Consortium of Research and Innovation on Information Security and Policies (Michael M. May and David Elliot, 1998).

In the context of payment management, many standards have arisen to regulate all payment methods and a case in point to support this is the Point-Of-Sale (POS) credit card processing which has been a major concern for modern IT companies. As technology advances, the handling and management of payment data has necessitated robust frameworks, methods and standards to ensure confidentiality, availability and integrity. In this domain, two prominent standards play significant roles in fortifying data security namely ISO 27001 and PCI DSS.

2.1 ISO 27001 – PART OF THE EXPANDING ISO/IEC 27000 STANDARDS

ISO 27001 is an internationally recognized standard for Information Security Management System (ISMS) established by an independent non-governmental body known as the International Organization for Standardization (IOS) and International Electrotechnical Commission (IEC). It provides a comprehensive framework for establishing, maintaining, implementing and continually improving information security within an organization (ISO 2022). Its relevance to modern IT companies, especially those involved in POS credit card processing, lies in its systematic approach to managing sensitive data.

ISO 27001 aids IT companies by offering a structured framework to identify, assess and manage risks associated with payment data processing. For instance, it emphasizes risk assessment methodologies that enables organization to identify vulnerabilities in POS system and insecure credit card data processing, thereby allowing for pre-emptive measures against potential breaches or data compromise.

2.2 PCI DSS – PAYMENT CARD INDUSTRY STANDARD

The Payment Card Industry Data Security Standard (PCI DSS) is an information security standard administered by Payment Card Industry Security Standard Council which was formed by a group of major card brands namely American Express, Visa Inc., JCB International, Mastercard and Discover Financial Services on September 7, 2006. It provides a set of security standards designed to ensure that all systems that processes, stores or transmits cardholder data maintain a secure environment. Its importance to POS credit card processing lies in its focus on securing payment card data throughout the transaction lifecycle.

PCI DSS offers a set of specific requirements (fig. 24) and best practices for securing cardholder data. For instance, it mandates the use of encryption, secure authentication and regular vulnerability assessments to process payment data. PCI DSS emphasis on encryption aligns with the evolution of POS systems towards supporting EMV chip technology, contactless payment and NFC. These technologies are integral to meeting PCI DSS requirements, protecting cardholder data and ensuring secure transactions.

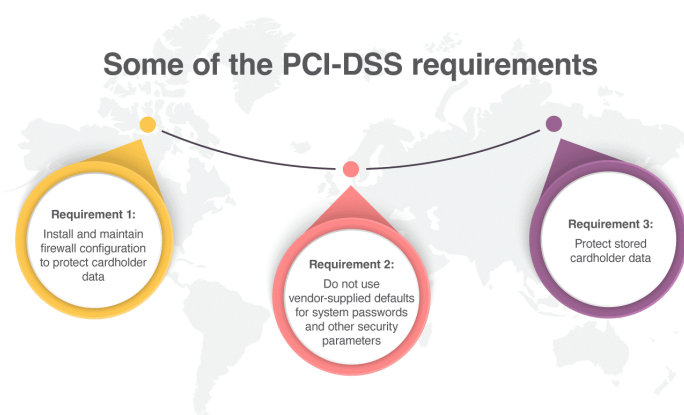


Fig 24: PCI DSS Requirements

Source: Advisera

2.3 INTEGRATION OF ISO 27001 AND PCI DSS

Modern IT companies often integrate ISO 27001 ISMS's principles with PCI DSS compliance to create a holistic approach to data security (Antonio Jose Segovia, 2022).

This integration ensures a comprehensive framework that covers not only the management aspects but also the specific requirements for protecting payment data.

2.4 PAYMENT SYSTEMS - DATA MANAGEMENT FRAMEWORKS, METHODS AND APPROACHES

In the context of payment systems, especially as it relates to Point-Of-Sale (POS) credit card processing, various data management frameworks, methods and approaches plays crucial roles in ensuring the security, integrity and efficiency of handling sensitive payment data.

DATA MANAGEMENT FRAMEWORKS

Below are some key data management frameworks that guides the development of payment systems:

1. GDPR – General Data Protection Regulation

The principles emphasizes data protection, consent and user rights, thereby affecting how payment data is collected, processed and stored.

2. Secure Sockets Layer (SSL) and Transport Layer Security (TLS):

SSL and its successor TLS are cryptographic protocols that provides secure communication over networks. They encrypt data transmitted between servers and browsers, thereby ensuring secure payment transactions.

3. Point-To-Point Encryption (P2PE):

P2PE encrypts payment card data from the point of interaction to the payment processor, safeguarding it against interception.

4. EMV (Europay, Mastercard, Visa) Standards:

EMV standards focuses on secure payment processing via chip-enabled payment cards. These standards reduce fraud by generating unique transaction data for each transaction.

DATA MANAGEMENT METHODS AND APPROACHES

1. Tokenization:

This protects sensitive payment data with unique tokens that are generated at each point of payment. It has greatly reduced the risks associated with storing cardholder information.

2. Data Masking:

This involves obscuring specific data elements within payment information, thereby protecting sensitive data. For instance, displaying only the last four digits of a credit card number.

3. Secure Encryption Protocols:

Encryption is vital in safeguarding payment data. Advanced encryption standards ensure that data is unintelligible to unauthorised parties. This enhances data security during transmission and storage.

2.5 REFLECTION AND CRITICAL ANALYSIS

The use case reveals potential gaps in security such as inadequate encryption method, lack of secure data transmission channels or insufficient protection for mobile payment data. It also highlights the need for secure data storage methods, especially concerning payment card information. Concepts such as encryption, tokenization, and data masking are essential for secure storage in POS systems.

Creating POS systems software involves incorporating stringent security measures to safeguard payment data. This includes encryption algorithms, secure data handling during transactions and protection against vulnerabilities in credit card terminals.

2.6 RECOMMENDATION

Considering the analysis made, the following recommendations are key to ensuring data security for payment systems is sustained at cyberattack-proof levels:

- ISO 27001 should be implemented to establish an Information Security Management System (ISMS). Use it as a framework for systematic risk assessments, control implementation and continual improvement in security practice.
- Ensure alignment to PCI DSS standards, especially when dealing with payment card information.
- For companies yet to do this, combine and implement ISO 27001 standards with PCI DSS compliance to create a comprehensive security framework.
- Conduct regular training sessions and awareness programs to educate employees on ISO 27001 and PCI DSS standards.
- Periodically evaluate the effectiveness of security measures through internal and external audits.

2.7 CONCLUSION FOR TASK 2

The implementation of data security principles within POS systems for modern IT companies is essential. Adhering to established frameworks like ISO/IEC 27001 and PCI DSS, while continuously evolving to counter emerging threats, is critical. These principles underscore the need for a holistic approach, encompassing encryption, compliance, vigilance and user awareness to fortify POS systems against threats and ensure secure payment processing.

REFERENCES

- Antonio Jose Segovia 2022. PCI DSS vs. ISO 27001: Similarities, differences, implementation, and certification. Available at:
<https://advisera.com/27001academy/knowledgebase/pci-dss/> Accessed on: 30th December, 2023.
- Barrett, Brian (2018). "How to Secure Your Accounts With Better Two-Factor Authentication", Available at: <https://www.wired.com/story/two-factor-authentication-apps-authy-google-authenticator/> Accessed on: 22nd December, 2023.
- Business Process Outsourcing, Data Entry Services: Best Practices to Adopt for Data Security in Hospitality by Managed Outsource Solutions (2023). Available at:
<https://www.managedoutsource.com/blog/best-practices-adopt-data-security-hospitality/>
 Accessed on: 30th December, 2023.
- Christopher J Hodson, 2019. Cyber Risk Management Prioritize Threats, Identify Vulnerabilities and Apply Controls. London: Kogan Page.
- Clare Stouffer, 2023. What is encryption? How it works + types of encryptions. Available at: <https://us.norton.com/blog/privacy/what-is-encryption> Accessed on: 22nd December, 2023.
- David Alexander, Amanda Finch, David Sutton, Andy Taylor / Andy Taylor, 2020. Information Security Management Principles. Swindon: BCS Learning & Development Limited.
- Difference between Encryption and Decryption (2023). Available at:
<https://www.geeksforgeeks.org/difference-between-encryption-and-decryption/>
 Accessed on: 30th December, 2023.
- Encryption, decryption, and cracking, 2023. Available at:
<https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:online-data-security/xcae6f4a7ff015e7d:data-encryption-techniques/a/encryption-decryption-and-code-cracking> Accessed on: 22nd December, 2023.

- Hospitality Sector Threat Landscape (2023): Trustwave Threat Intelligence Briefing and Mitigation Strategies,' Available at: <https://www.trustwave.com/en-us/resources/library/documents/2023-hospitality-sector-threat-landscape-trustwave-threat-intelligence-briefing-and-mitigation-strategies/> Accessed on: 30th December, 2023.
- IBM (2023). What is data security? Available at: <https://www.ibm.com/topics/data-security#:~:text=Data%20security%20is%20the%20practice%20of%20protecting%20digital,access%2C%20corruption%20or%20theft%20throughout%20its%20entire%20lifecycle> Accessed on: 30th December, 2023.
- ISO: the International Organization for Standardization, 2022. Available at: <https://www.iso.org/home.html> Accessed on: 22nd December, 2023.
- IT Governance: An International Guide to Data Security and ISO 27001/ISO 27002, 2019. 7th edition. London: Kogan Page.
- Jacomme, Charlie; Kremer, Steve (2021). An Extensive Formal Analysis of Multi-factor Authentication Protocols. Available at: <https://dl.acm.org/doi/10.1145/3440712> Accessed on: 22nd December, 2023.
- Jessica Barker, 2023. Confident Cyber Security. 2nd edition. London: Kogan Page.
- Limor Wainstein (2018). Data Security in Hospitality: Risks and Best Practices. Available at: <https://hospitalityinsights.ehl.edu/data-security-in-hospitality-best-practices> Accessed on: 30th December, 2023.
- Marcus Law (2023). Trustwave report on hospitality industry security threats. Available at: <https://cybermagazine.com/articles/trustwave-report-on-hospitality-industry-security-threats> Accessed on: 30th December, 2023.
- Michael M. May and David Elliot, 1998. "FSI - Consortium for Research on Information Security and Policy". Stanford: Freeman Spogli Institute for International studies. Available at: https://fsi.stanford.edu/research/consortium_for_research_on_information_security_and_policy Accessed on: 22nd December, 2023.

NIST Cybersecurity Framework: National Institute of Standards and Technology. Available at: <https://www.nist.gov/cyberframework> Accessed on: 2nd January, 2024.

Official PCI Security Standards Council Site - Verify PCI Compliance, Download Data Security and Credit Card Security Standards. Available at: <https://www.pcisecuritystandards.org/> Accessed on: 2nd December, 2023.

Sue Poemba (2017). IT Business Edge: "Knowing Your Data to Protect Your Data". Available at: <https://www.itbusinessedge.com/it-management/knowning-your-data-to-protect-your-data/> Accessed on: 30th December, 2023.

Summers, G. (2004). Data and databases. In: Koehne, H Developing Databases with Access: Nelson Australia Pty Limited. p4-5.

Willie May, 2014. "Guidelines for Smart Grid Cyber Security" (PDF). Volume 1 - Smart Grid Cybersecurity Strategy, Architecture, and High-Level Requirements. National Institute of Standards and Technology. September 2014. <http://dx.doi.org/10.6028/NIST.IR.7628r1> Accessed on: 22nd December, 2023.

APPENDIX

APPENDIX 1: PORTFOLIO DOCUMENT


BRIAN BARRETT SECURITY JUL 22, 2018 7:00 AM

How to Secure Your Accounts With Better Two-Factor Authentication

Two-factor authentication is a must, but don't settle for the SMS version. Use a more secure authenticator app instead.

HOPEFULLY BY NOW you've heeded the repeated warnings from your friends and loved ones (and friendly, beloved internet writers) to use two-factor authentication to [secure your digital accounts](#). That's where access to [Facebook](#) or [Twitter](#) or your online bank—anything that supports it, really—requires not just a password but also a special code. Not all two-factor is created equal, however. For better protection, you're going to want an authenticator app.

TRENDING NOW



Article on 2FA - Barrett, Brian (2018). "How to Secure Your Accounts With Better Two-Factor Authentication". Available at: <https://www.wired.com/story/two-factor-authentication-apps-authy-google-authenticator/> Accessed on: 22nd December, 2023.

MY VIEWPOINT ON THIS:

Authenticator apps are a more secured way of verifying one's identity especially because scammers have been able to manipulate security gaps that comes with SMS version of 2FA. For instance, banks now use 'open banking' authentication method to approve credit card and other transactions.

Use Encryption When....



Making a purchase online



Securing information on your laptop if it's stolen



Emailing confidential data



Storing sensitive information in the cloud



Your business requires it

Blogpost by Norton - Clare Stouffer, 2023. What is encryption? How it works + types of encryptions. Available at: <https://us.norton.com/blog/privacy/what-is-encryption> Accessed on: 22nd December, 2023.

MY VIEWPOINT ON THIS:

As long as you are utilising your personal information on the digital space or just merely using any technological resource such as the online hotel booking system or POS terminals, you must ensure encryption is enabled.



2023 Hospitality Sector

RESEARCH REPORT

2023 Hospitality Sector Threat Landscape: Trustwave Threat Intelligence Briefing and Mitigation Strategies

The Trustwave report, *2023 Hospitality Sector Threat Landscape: Trustwave Threat Intelligence Briefing and Mitigation Strategies*, explores the specific threats and risks that hospitality organizations face, along with practical insights and mitigations to strengthen their defenses.

Through research, Trustwave SpiderLabs has documented the attack flow utilized by threat groups, exposing their tactics, techniques, and procedures. From brute forcing to exploiting known vulnerabilities to attacking exposed open ports, these persistent threats pose significant risks to the hospitality industry.

Activate Windows
Go to Settings to activate Windows.

Research report on Hospitality Sector Threat Landscape (2023): Trustwave Threat Intelligence Briefing and Mitigation Strategies,' Available at: <https://www.trustwave.com/en-us/resources/library/documents/2023-hospitality-sector-threat-landscape-trustwave-threat-intelligence-briefing-and-mitigation-strategies/> Accessed on: 30th December, 2023.

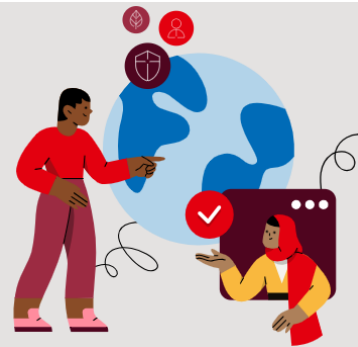
MY VIEWPOINT:

It is reported that overtime, nearly 31% of hospitality companies have formally announced a data breach in their organisation's history with the average cost of a breach amounting to approximately £2.7 million. Such losses can be devastating for the company and recovery period, if the business can survive the hit, will be long and expensive. This financial and reputational strain can be avoided with a sophisticated security system in place.

ISO: Global standards for trusted goods and services

Standards define **what great looks like**, setting consistent benchmarks for businesses and consumers alike — ensuring **reliability, building trust, and simplifying choices**.

Making lives easier, safer and better.



ISO: the International Organization for Standardization, 2022. Available at: <https://www.iso.org/home.html>
Accessed on: 22nd December, 2023.

MY VIEWPOINT:

ISO global standards has consistently delivered on data security aspects for over a decade now. The widespread adoption of its principles has clearly increased uniformity in data security applications and has made replications and upgrades to security systems easier.

PCI Security Standards Council Hosts 2023 Europe Community Meeting

Payment Security Industry Leaders Collaborate to Help Protect Payment Data and Discuss the Modern Threat Landscape

DUBLIN, 26 October 2023 – More than 700 in person and online stakeholders from Europe and around the world convened this week in Dublin for the Payment Card Industry Security Standards Council (PCI SSC) [Europe Community Meeting](#). The multi-day event focused on updates in payment security standards and programs, and provided industry stakeholders with opportunities to learn, share, network, and discuss the current state of payment security.

Official PCI Security Standards Council Site - Verify PCI Compliance, Download Data Security and Credit Card Security Standards. Available at: <https://www.pcisecuritystandards.org/> Accessed on: 2nd December, 2023.

MY VIEWPOINT ON THIS:

The attention given to payment security standards and programs is well worth it, as modern IT companies are constantly dealing with the ever-evolving nature of technology and its respective attackers. Reviews at such meetings eventually brings about some form of payment system advancement which the hotel and other businesses will benefit from.



What is PCI DSS ? | Merits & Demerits | Who Manage PCI DSS ?

Youtube video by Iconic Smart Tech – Available at: <https://youtu.be/A5xGgt4BM0w>. Accessed on 4th December, 2023.

MY VIEW ON THIS:

While PCI DSS may have some demerits when compared to other standards, its benefits still outweigh any lapses it might have. Research is tirelessly being done to ensure development to payment systems.

APPENDIX 2: PYTHON SCRIPT

PASSWORD VALIDATION:

```
In [7]: import re

def validate_password(password):

    # Checking length, uppercase, lowercase, and numbers
    if re.match(r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[A-Za-z\d]{8,12}$', password) and ' ' not in password:

        # Counting uppercase and lowercase letters
        count_uppercase = sum(1 for c in password if c.isupper())
        count_lowercase = sum(1 for c in password if c.islower())

        # Validating at least 2 uppercase or 2 lowercase letters
        if count_uppercase >= 2 or count_lowercase >= 2:
            return True
        return False
```

```
In [8]: # Let's apply password validation for a user input to login with sample user password 'Success100'

password_input = "Success100"
if validate_password(password_input):
    print("Password is valid.")
else:
    print("Password does not meet the criteria.")

Password is valid.
```

```
In [9]: # Testing the compliance of the user password to set criteria

password_input = "ces100"
if validate_password(password_input):
    print("Password is valid.")
else:
    print("Password does not meet the criteria.")

Password does not meet the criteria.
```

ENCRYPTION-DECRYPTION

```
In [5]: #Password encryption and decryption

import bcrypt

# Function to encrypt (hash) the password
def encrypt_password(password):
    return bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())

# Function to check if entered password matches the encrypted password
def check_password(entered_password, encrypted_password):
    return bcrypt.checkpw(entered_password.encode('utf-8'), encrypted_password)
```

```
In [13]: # Simulate registration process (encrypt the password)

username = "Greatest"
password = "Success100"

encrypted_password = encrypt_password(password)
```



```
In [*]: #Password encryption-decryption

from cryptography.fernet import Fernet

# Generate a key for encryption/decryption
CAB = Fernet.generate_key()
cipher_suite = Fernet(CAB)
```

```
In [*]: #Define the encryption-decryption path

def encrypt_password(password):
    return cipher_suite.encrypt(password.encode())

def decrypt_password(encrypted_password):
    return cipher_suite.decrypt(encrypted_password).decode()

def encrypt_username(username):
    return cipher_suite.encrypt(username.encode())

def decrypt_username(encrypted_username):
    return cipher_suite.decrypt(encrypted_username).decode()
```

```
In [12]: # Example data input to encrypt password

username = input("Enter username: ")
password = input("Enter password: ")

encrypted_pass = encrypt_password(password)
print(f"Encrypted Password: {encrypted_pass}")

Enter username: Greatest
Enter password: SuccesS100
Encrypted Password: b'gAAAAABlj3YCTyEEDbXUjJb2wHkD7ikIErIbmQ4pKEBuX9JwieLy7M5HcXsWP7zqaFPqgONQMWPE5pMBaL_1fbwir8Fu3OKDgA=='
```

```
In [13]: # Example data input to decrypt password

username = input("Enter username: ")
password = input("Enter password: ")

decrypted_pass = decrypt_password(encrypted_pass)
print(f"Decrypted Password: {decrypted_pass}")

Enter username: Greatest
Enter password: b'gAAAAABlj3cfQ7tjipn-nBcZXU5agfiqvE_9Gc1d4Pp1CZT6q-20qdPiixUqCgAelgcgvo01dKYZZuYxXSw9oKX7WP7r6E3TIg=='
Decrypted Password: SuccesS100
```

```
In [14]: # Example data input to encrypt username

username = input("Enter username: ")
password = input("Enter password: ")

encrypted_user = encrypt_username(username)
print(f"Encrypted Username: {encrypted_user}")

Enter username: Greatest
Enter password: SuccesS100
Encrypted Username: b'gAAAAABlj4fwkRdOkpQqigs0fqQx3PifJgyVVIuc9XVsaemRlRefU6HGZtbr6SLvgHZgSdqoAme2cf5aCjnL992F24bEa4UAAQ=='
```

```
In [15]: # Example data input to decrypt username

username = input("Enter username: ")
password = input("Enter password: ")

decrypted_user = decrypt_username(encrypted_user)
print(f"Decrypted Username: {decrypted_user}")

Enter username: b'gAAAAABlj4fwkRdOkpQqigs0fqQx3PifJgyVVIuc9XVsaemRlRefU6HGZtbr6SLvgHZgSdqoAme2cf5aCjnL992F24bEa4UAAQ=='
Enter password: SuccesS100
Decrypted Username: Greatest
```

```
In [14]: # Simulate Login attempt with correct password
entered_username = "Greatest"
entered_password = "SuccesS100"

# Check Login attempt
if entered_username == username and check_password(entered_password, encrypted_password):
    print("Login Successful")
else:
    print("Login Failed - Incorrect username or password")

Login Successful
```

```
In [16]: # Simulate Login attempt with incorrect password
entered_username = "Greatest"
entered_password = "Password123"

# Check Login attempt
if entered_username == username and check_password(entered_password, encrypted_password):
    print("Login Successful")
else:
    print("Login Failed - Password is incorrect!")

Login Failed - Password is incorrect!
```

```
In [17]: # Simulate Login attempt with incorrect username
entered_username = "Great"
entered_password = "SuccesS100"

# Check Login attempt
if entered_username == username and check_password(entered_password, encrypted_password):
    print("Login Successful")
else:
    print("Login Failed - Username is incorrect!")

Login Failed - Username is incorrect!
```

MULTIFACTOR AUTHENTICATION

```
In [18]: # Import the pyotp module

import pyotp

# Generate a random secret key
secret = pyotp.random_base32()

# Create a TOTP object
totp = pyotp.TOTP(secret)

# Generate a time-based one-time password (OTP)
otp = totp.now()
print(f"One-Time Password (OTP): {otp}")
```

One-Time Password (OTP): 276941

```
In [19]: # Simulate multifactor authentication

pin = totp.now() # Simulate a new one-time pin
print(f"One-Time PIN: {pin}")
```

One-Time PIN: 276941

```
In [20]: # Verify the user

user_input = input("Enter the OTP: ")
if totp.verify(user_input):
    print("User verified!")
else:
    print("Verification failed!")
```

Enter the OTP: 276941
User verified!

```
In [22]: import time

# Wait for 30 seconds
time.sleep(30)

# Verify again after waiting
user_input = input("Enter the OTP again: ")
if totp.verify(user_input):
    print("User verified!")
else:
    print("Verification failed!")
```

Enter the OTP again: 276941
Verification failed!

```

In [25]: # Generate a time-based one-time password (OTP)
otp = otp.now()
print(f"One-Time Password (OTP): {otp}")

# Verify the user again

user_input = input("Enter the OTP: ")
if otp.verify(user_input):
    print("User verified!")
else:
    print("Verification failed!")

One-Time Password (OTP): 838510
Enter the OTP: 838510
User verified!

```