# BISM3205 - Assignment 2

**Part A:**

The Treasure is hidden at [E, 8]

**Part B:**

The Vigenere Cipher was used to decipher the message.

**Part C:**

The key used was **ALEXBISM**.

Regarding the number, it appeared as a 64-length hexadecimal string. This was copied into crackstation.net which had a match on the encrypted letter ‘E’.

He last sentence decoded to *mod(your\_student\_number, 10)*. Since my student number is 45857278, the number modulo 10 is 8. Hence, [E.8] would be the hidden treasure location.

**Part D:**

The hash given in the question is:

694430bed946b0330e4d15e9bc3931123c122166da6d353bad32d4c09da3788c

Using Crackstation.net, this is a known encryption of **universityofqueensland**.

Ordinarily, we would not be able to decrypt a hashed result besides using a brute force technique or looking up hashes in a repository, as we have done.

**Part E:**

The database administrator/security leader at the firm should consider Salting and Peppering the passwords stored in the database. Whilst hashing is secure, salting is the process of addition a random string to it before hashing it.

As we have found in the above example, a simple password and its corresponding hash may already exist in a hash-lookup table, however, a string of random characters appended to a password likely is not. It is highly unlikely that a salted password has been identified in stored in a hash-lookup table.

Peppering passwords is the process of storing the password salt in an object of file storage, instead of in the same database. In the circumstance that a hacker compromises your database, they would be missing the password salt used in combination to originally hash the password.

Besides this, the security leader could consider checking passwords for length and complexity, or checking against dictionaries and deny lists.

**Part F:**

The numbers written in hexadecimal were transformed to base 10 and then looked up on the ASCII table provided in the week 6 tutorial to decipher the question:

*What is it called when 2 strings have the same hash digest?*

This phenomenon is called a collision. As stated, this is when two *different* strings (or any serializable objects) generate the same hash value

G)

F5 CA 4F 93 5D 44 B8 5C 43 1A 8B F7 88 C0 EA CA

- 32 hexadecimal characters, each representative of 4 bits meaning a this image represents a 64-bit stream of characters.

- DES encryption uses a 64-bit block size

- This could be indicative of a 64-bit

H) and I)

MD5 hash of the image plane: 253dd04e87492e4fc3471de5e776bc3d

MD5 hash of the image ship: 253dd04e87492e4fc3471de5e776bc3d

- Talk about collisions

Part J)

This is a trick question

# Q2:

**Part A and B:**

What is the two-digit integer number in decimal notation that is greater than 0?

First, I used a substitution cipher to substitute characters in the initial message as it looked like a website link.

I could decode most of it, except for the two capital letters UV. I assumed it would be QR (for QRcode) but to be certain I wrote a script that every permutation of left over characters I have not substituted. The code snippet and output is provided in Appendix 1. Only <https://alexpudmenzky.com/BISM3205/QRcode.zip> gave me a result.

I downloaded the zip file, extracted it, and found the QR code within it (Figure 1)

Figure 1: QR code found in QRcode.zip

This QR code navigated me to <https://alexpudmenzky.com/BISM3205/sad.jpg>.

Sad face emoji consists of a colon ‘:’ and an opening parenthesis ‘(‘ which on the ASCII table correspond to decimal numbers 58 and 40, respectively.

Combined, this number would be 98, which is a single two-digit number, grater than 0, in decimal notation.

**Part C:**

An IP address is segmented into host and network portions via the use of a subnet mask, which is a series of bits which differentiate which section of the IP address represent the network and which part represent the host.

Bits in the subnet mask that are 1 represent the network portion, so the subnet mask 255.255.128, represented in binary form, i.e.

255.255.128.0 = 11111111.11111111.10000000.00000000 (binary)

Indicating that the first 17 bits represent the network portion of the IP address. A bitwise AND operation can be used to yield the relevant network bits.

11000000.10101000.00000101.00000010 (IP Address) AND

11111111.11111111.10000000.00000000 (Subnet mask)

11000000.10101000.00000000.00000000 (Network portion) = 192.168.0.0

Similarly, a bitwise NAND operation can be used to determine the computer (host) portion of the IP address:

11000000.10101000.00000101.00000010 (IP Address) NAND

11111111.11111111.10000000.00000000 (Subnet mask)

00000000.00000000.00000101.00000010 (Network portion) = 0.0.5.2

**Part D:**

This is more likely an encrypted message than it is a salted and hashed password. Some reasons are outlined below.

1. 127 bits – odd length of a hash. Typically an index of 2 – e.g 64, 128, 256 (for SHA-256)
2. Password Hashes typically do not need to be this long – even a SHA-256 hash in base 64 (which this seems to be) would only be 64 bits long. 127 bits seems too much

# Q3

Part A:

Part D:

The best option is (3) – hand the USD stick to your IT department or security team.

1. USB Sticks Can Be a Security Threat: Plugging unknown USB devices into your computer is a significant risk because the USB stick may contain malicious software, such as viruses, ransomware, or malware, that can compromise your company's network and sensitive data.
2. IT or Security Teams Are Trained to Handle Such Situations: The IT or security team has the necessary tools and expertise to safely investigate the contents of the USB stick without putting the network or devices at risk. They can check if the USB contains any malware or determine if it's safe to access.

Why Not the Other Options?

* Option 1: Plugging the USB stick into your computer:
* Dangerous: This could lead to a security breach, data theft, or malware infection that could compromise sensitive company data.
* Option 2: Leaving the USB stick where you found it:
* Not Responsible: Someone else might pick it up and plug it into their computer, putting the network and systems at risk.
* Option 4: Taking the USB stick home to investigate:
* Risky: This option still carries the potential of infecting your home devices and, even worse, exposing sensitive company data if you access it from home.
* Option 5: Throwing the USB stick in the trash:
* Ineffective: The USB could still be found and potentially used by someone else, leading to the same security risks.

Part E)

An Intrusion Detection System (IDS) raises alerts when suspicious activity is detected. However, two types of errors can occur:

* False Positive:
* An alert is triggered for legitimate activity mistaken as a threat.
* Impact: A nuisance that wastes IT staff's time investigating non-threats, contributing to alert fatigue.
* False Negative:
* The IDS fails to detect a real attack.
* Impact: Serious, as actual threats go unnoticed, leaving the system vulnerable.
* Security Perspective: Least Desirable
* A false negative is far worse than a false positive. While false positives are inconvenient, false negatives mean real threats go undetected, potentially causing severe harm.
* Key Terminology:
* False Positive: A false alarm due to a false attack stimulus, adding to normal noise in network traffic.
* False Negative: A missed attack, which is very bad and must be avoided with high IDS accuracy.

# Appendix 1

Script used to find if there were any ‘hiding’ websites.

import requests

base\_url = "https://alexpudmenzky.com/BISM3205/"

letters\_left = ['F', 'G', 'J', 'Q', 'R', 'V', 'W', 'Z']

for letter1 in letters\_left:

    for letter2 in letters\_left:

        url = base\_url + letter1 + letter2 + 'code.zip'

        try:

            response = requests.get(url)

            # Check if the response status code is 200 (OK)

            if response.status\_code == 200:

                print(f"Valid route found: {url}")

            else:

                print(f"Route not found: {url} (Status code: {response.status\_code})")

        except requests.exceptions.RequestException as e:

            print(f"Error accessing {url}: {e}")

(base) PS C:\git\DanielCiccC.github.io\BISM3205\Assignments\A2> python routes.py

Route not found: https://alexpudmenzky.com/BISM3205/FFcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/FGcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/FJcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/FQcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/FRcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/FVcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/FWcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/FZcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/GFcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/GGcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/GJcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/GQcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/GRcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/GVcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/GWcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/GZcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/JFcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/JGcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/JJcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/JQcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/JRcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/JVcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/JWcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/JZcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/QFcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/QGcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/QJcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/QQcode.zip (Status code: 404)

Valid route found: https://alexpudmenzky.com/BISM3205/QRcode.zip

Route not found: https://alexpudmenzky.com/BISM3205/QVcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/QWcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/QZcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/RFcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/RGcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/RJcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/RQcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/RRcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/RVcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/RWcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/RZcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/VFcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/VGcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/VJcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/VQcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/VRcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/VVcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/VWcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/VZcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/WFcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/WGcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/WJcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/WQcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/WRcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/WVcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/WWcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/WZcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/ZFcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/ZGcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/ZJcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/ZQcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/ZRcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/ZVcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/ZWcode.zip (Status code: 404)

Route not found: https://alexpudmenzky.com/BISM3205/ZZcode.zip (Status code: 404)

# Appendix 2