Assignment 1

Name: **Tanmayi Jandhyala** Waterloo ID: **20877641**

WhatIAM id: tjandhya@uwaterloo.ca

Answer **1(a)**: Terminology: The attacker is named Eve.

A possible approach to the Man in the middle attack in the organisation:

Assuming that the company does not follow best security practices, Eve performs an SSL stripping attack between a customer's system and the server of the company, by IP spoofing. Eve could have done this using a Windows or a Linux machine, more conveniently if they were in the same LAN connection. Whenever a customer uses the website of the company to perform transactions, Eve sits in between the visitor of the site (the customer) and the server. The customer's browser connects to Eve's machine (for example, the browser will connect to port 1000 instead of port 80) and waits for a secure HTTP request from the server. When the server sends it, Eve "strips" the secure connection at the socket layer, rendering any data from the customer to arrive as unencrypted. This means that all of their personally identifiable information (PIIs) such as passwords, name, bank account number and the transaction information is obtained by Eve.

This type of attack violates the Confidentiality and Integrity security properies. If Eve is to use the transaction data to change the banking information like the account number (to their own) and the amount that is being transferred, the Integrity security property is compromised.

1(b) A possible approach to a Side Channel Attack:

Cache-timing vulnerability:

This is a kind of trace-driven attack that Eve could have performed, which can work when they try to exploit the scalar multiplication routine that happens in cryptographic algorithms used for storage (for example, the OpenSSL). Computational phases are loaded onto caches when a user tries to access a server and load its content. Eve could use a spy process to snoop over this cache and observe the addition of points to the cache memory. The spy process continuously loads data by itself onto the cache and calculate the time that it takes to read data from all cache lines in a set. When a user prompts loading of some cache data, this forces out some of Eve's data and then by checking which parts of Eve's data remain, they deduce what part of the user's memory were accessed. Eve can then get access to that memory and obtain information.

This violates the Integrity property, because Eve can make use of the timers to rewrite cache data.

1(c) Assuming that the company uses a password hash function for their identity and access management system, and that an average user (customer) does not change their password as frequently as they should, and also that they do not have Multi-factor Authentication (MFA), Eve could make a pass-the-hash attack by initially gaining access into their clientbase systems through using Social Engineering techniques. Whether the customers are using a Windows or Linux system, the password hashes are stored in the accounts manager (typically for a Windows system) and in other active memory locations. All Eve could have done is extract password hashes from the process memory, pass themselves as an authenticated user, and eventually perform lateral access to carry out the data breach. For this, Eve does not even have to decrypt the hash as the password hashes remain static until the password is changed.

This type of attack compromises the Confidentiality, Integrity and Availability properties.

<u>Answer 2.</u> The following formal methods can be adopted:

The **Code Review** can be done in the following steps:

i. The programmers could present a well-prepared documentation on their application including the general study and business logic, and provide a (strictly adhered to) list of what the application should be doing. This could be labelled the 'Introduction' phase.

ii. There could then be a discussion where senior developers as reviewers can go over the code and identify any vulnerabilities. Specifically, the code can be moderated in well-defined modules and the set of actions that a module is required to perform.

iii. The developers can then understand where to create pathes to fix these vulnerabilities based on comments.

Testing can be carried out once when the application code is compiled and the developers think the functionality is implemented. All of the below steps need to be followed to ensure the code does not contain avoidable security flaws.

i. Automated Regression and KPI Testing: The code could go through a series of batch suites consisting of various test cases that testers must identify beforehand and write scripts for. These will test the scenarios of if the code performs the necessary functionalities that it needs to.

ii. Manual Testing(including functional and non-functional testing): Although it is more time taking, manual testing (more specifically in that, black box testing) can be done to ensure that the code is not doing anything that it should not be doing. This is important in finding out about any back doors unintentionally left by the developers or of any vulnerabilities that new patches might create.

Code maintenance: As systems get frequent upgrades, the application codes could develop new security flaws that did not exist before. It can be practiced by having a timely review of the source code by a team of reviewers, not just to ensure that the code still performs the actions it is supposed to, but also to figure out how it can do it better. This could mean that new plugins could be added to make the code more efficient and up-to-speed with protecting itself from new vulnerabilities that it might be exposed to as and when new patches or versions of the code are released. The documentation really comes in handly at this point.

Answer 3. (extra) **Prevent** Pass-the-hash attacks:

To adopt a better password management system: In that, the passwords for privileged systems can be set to be auto-changed after every session so that the hash will be of no use after a privileged session closes. Also, having a 2FA mechanism can ensure that a password hash in itself cannot give privileged access to a malicious user, hence thwarting the attack entirely.

(a) **Deflect** from a side-channel attack:

A side channel attack can be deflected, for in case of relections, by mitigating the reflective screens around a potential victim. For cache-based attacks, the defences have to be in the policies of cache replacement. One way to do this is to not have user arguments make the code decide on how much secretive information can the attacker get. That way even if the attack happens, not much secretive information is let out. Further, the bandwidth of the side channels can also be decreased by having any timestamp APIs for shared memory buffers removed so that the attackers cannot create any timers.

(b) **Detect** an MITM attack:

A man-in-the-middle attack can be detected by having an inexpensive system make an attacker-anticipated connection to the browser server that has the attacker perform SSL stripping on, and then check the webpage for any details that make it look illegitimate. Another way to detect this is to have the browser redirections to be reported and checked on, and also set up a host detection mechanism to find out which system might the attacker be communicating with the browser from.

(c) **Recover** from a Privilege escalation attack:

The immediate step into recovering from a privilege escalation attack is to have the user's password changed. This will end the privilege session and also generate a new hash. But more importantly, since the integrity of the user's data might also be compromised, a complete and encrypted back-up of the original user data can help with recovery. The finance company could employ a security mechanism to have this backup isolated so that it does not sync with any changes made to the original data. Further, automation could be used for quicker recovery and this way, the availability security property is reestablished. The system can then undergo more testing in the back end to check if any unauthorised users still have privileged access and can delink them.

(d) **Deter** a Man-in-the-middle attack:

In case of SSL Stripping mentioned in the answer 1, a way to deter the attack is to have any redirections in the browser to be blocked. Further, for every few minutes of being in the session, the user could be auto-logged off at a pre-defined idle time and a pop-up could be displayed asking the user to close all browsers.

Answer **4. (a)** The possible problems that the password authentication system at the company can have are listed below:

1. There is no way for the password mechanism to check for the authorisation of the user, because it does not use methods like 2FA and relies on a single password.
2. There don't seem to be any data recovery mechanisms in case the device gets locked and there is a threat to the data that can be accessed with it.
3. An attacker can intercept the password check from the company's database and essentially exploit a TOCTOU vulnerability, since the action of verifying if the password is correct is not atomic.
4. If an attacker gets access to the database and it is not encrypted or hashed, the passwords will be available in plain text.
5. If a user forgets their password, there does not seem to be a way to recover or reset it, and even if they were, a secure way of doing that is difficult to monitor.
6. It still is a physical device with probable hardware vulnerabilities(that Morty might steal again), like the possibility of a shoulder spoofing attack or attempting a brute force attack by finding out about the memory information and exploiting it at his convenience.

4(b) The below ways could be better secure mechanisms for the portal:

1. It is good practice to salt the passwords and have them hashed with strong hash functions. That way, even if many users use the same or similar passwords, the attacker would not be able to guess that just by looking at leaked passwords from the portal (because encrypted passwords are of no use to the attacker).
2. Having users get One-Time Passwords (OTPs) is a good way to ensure dual authentication.
3. It is also helpful to have secure account recovery options like an encrypted backup of the users' data if they get locked out and also a secure password-recovery mechanism like secondary email authentication with some security questions.
4. An unconventional idea is that along with the password, another authentication on top of it could be to through pictures that the users can better remember and recognise. This way, recovery in case of lost or stolen passwords would not be a huge problem.

The potential problems with these methods are that:

1. Using social engineering techniques is still a credible way to get customer data and get access to their passwords.
2. Hashes can be exploited with methods like buffer overflows and the system can still be exploited.
3. If the picture authentication is used, it is easy to perform shoulder spoofing or side channel attacks to find out the password.

Answer 5:

(a) CryptoLocker - This can be categorised as a kind of Trojan Horse and a Ransomware.

How it spreads: In Windows systems, the malware first infects the system and searches for files to encrypt, including the USB memory devices and any other file systems even on the cloud. It is planted through sending a malicious executable hidden in a *.pdf or a *.doc file sent via an email or clicked on in a malicious webpage. Cryptolocker does not reveal itself until it infects the command and control server. In Windows and linux machines, the malware copies itself to the %AppData% or the %LocalAppData% directories and then has the original executable file delete itself. This does not work on Mac directories, hence. It then encrypts all the files it can and locks them and reveals itself as a pop-up, demanding ransom from the user to get back their access. The files are encrypted using the RSA-2048 (asymmetric) key algorithm.

The resulting effect: The confidentiality and availability security properties are violated. The file system is compromised and decrypting it, along with full data recovery and securing the system again from further attack will cost money.

(b) Stuxnet - It was the world's first detected digitalweapon that was built to destroy uranium centrifuges in Iran. It is a worm or a Logic Bomb.

How it spreads: It specifically targetted systems that were using Programmable Logic Controllers built by Siemens. The Windows OS infrastructure, the targetted software on the Windows systems, and the embedded software in the PLCs are all attacked by Stuxnet. It acted as a rootkit to both the user and kernel mode, essentially altering the PLC's programming logic so that the centrifuges were operated in an incorrect way, hence destroying the process, while having the controller computer blissfully unaware of this. Four zero-day bugs were exploited in to make stuxnet work which included 2 privilege escalation attacks.

The resulting effect: A large-scale nuclear operational loss in one scenario. Typically, the programmable logics would be altered without having the control systems know.

(c) CIH - Also called Chernobyl, it is an example of a time bomb malware.

How it spreads: When a CIH file enters a Windows (95, 98 or ME) system, it looks to infect the executable files on the system. Although initially the size of the infected files might be the same as the original files, the CIH virus looks for empty spaces in the files and then breaks itself down to fill those spaces. It is known to have overwritten the hard drive with random data, along an infinite loop until the system crashed. It also corrupts the Flash BIOS stored data and renders the computer to not boot at start.

The resulting effect: The malware renders the computer system to not display a thing upon start. All information and data in it becomes inaccessible.

Exploit Descriptions for Programing Questions:

Exploit 1:

A buffer overflow vulnerability.
Where it is located:
In the function get_entropy(), the buffer is of size (BUFF_SZ) 1024 bytes. The function get_entropy() is called in fill_entropy(), so the effects in stack happen in the latter function. When the stack is filled, there is the 3 local variables and then the buffer, followed by the Frame Pointer and the Instruction

Pointer. The buffer takes up 7 spaces, and there is an additional NULL space appended to the end of the stack that makes the entire length to be 1025 bytes. So, the stack is overwritten by one byte.

In statistical theory, half of the stack is filled with NO-Operation instructions and the next half contains the address to the shell code. The shell code in itself is filled by calculating: (the size of the stack/2 - the size of the shell code - the length that is filled with shell code). In the function call of get_entropy, the Least Significant Bit gets overwritten and becomes the Instruction Pointer. So this vulnerability can be exploited into filling the buffer with the shell code.

Exploit 2:

A formal string vulnerability.
In the function get_reason(), the buffer is filled with stats_file() which is of size 1024, and then the reason buffer, which is of 512 bytes in size. The strcpy() function copies STATS_G or STATS_UG which is of 15 or 16 bytes and copies that to the stats_file, but in the conditional statement below, the reason is overwritten because the loop runs over to 1024 spaces, and the 15 or 16 bytes memory of stats_file() gets overwritten as well.. This is definitely a vulnerability that could be exploited, but the Instruction Pointer is still not overwritten in this case.

A further observation is that when stats_file is opened to write, the fprintf does not check the bounds and essentially the statement should give a segmentation fault.

Exploit 3:

A TOCTOU vulnerability.
In fill_entropy() functions, the get_entropy() function is called before the file FILENAME is opened. Circling back to check_perms(), where the FILENAME is unlinks and then the setuid values of the file are manipulated to essentially create the file as root, the function would either return a 0 or a -1 to the call in get_entropy(). And then back to fill_entropy(), the the fd writes to FILENAME *after* the get_entropy() function is called for buffer. This could be taken advantage of when there can be a symbolic link established (because get_entropy takes input from the user), and then when the fd writes in buffer, the content could be obtained from the user where they could insert their own values for /etc/passwd and not at the place where the file is intended to be written.
Ofcourse, to have this work, the input should be given after the symbolic link is established, but it is nevertheless still possible to play with this vulnerability.