Problem 1

Assumptions

1. phone has a guessable password to gain access to the phone
2. recorded files are not individually password protected
3. the recordings are stored locally on the phone

Scenario
The attacker steals the phone and the user somehow gets it back. Theft of personal objects is realistic. Although it may be uncommon, it can happen. Additonally, it is possible for the item to be returned.

1. Interception

   Since the interviews are recorded on the phone getting access to that phone could mean exposure of the assets. An attacker could listen to the recordings and pull all the information. This is an attack on confidentiality.

2. Interruption

   Similar to the last point if some were to get a hold of the phone and break into it, then the user could delete the recordings. This would make the assets unavailable for the user. Additionally, if the phone is never returned then the user would never be able to recover the data. This is an attack on availability.

3. Modification

   The attacker could also tamper with the data if they really wanted to. Most phones have basic editing features such as cropping a video or recording. This would be less likely to happen though, as there is not real gain to this. This is an attack on Integrity and confidentiality.

4. Fabrication

   Fabrication of data could also occur. The attacker could create any data they want using the phone. For instance, they could create malicious data or processes on the phone. An attacker could download

any application or software they want. They could download a software that allows them to gain remote access. This would be an attack on integrity.

a) Confidentiality is compromised. The user does not know another research team is recording the session, thus the interview is not confidential.

b) Integrity is compromised. The entire interview is ruined due to the zoom-bombing, this makes the integrity of the recording almost useless.

c) Availability is compromised. The vital segments of the recordings are deleted and no longer available or accessible assets for the user.

Problem 2

a) Prevent

Lock up the smaller items. This will prevent the attackers from stealing the items without using force.

b) Deter

Hire another employee to help around the store. This would make the attack harder to perform as there are more possible observers.

c) Deflect

Install signs that say you have cameras. This would decrease the chances of attacks happening because attackers do not want to be caught on camera.

d) Detect

Install anti-shoplifting alarms. If an attacker leaves without paying the alarm would go off, which alerts you that something is being stolen.

e) Recover

Get insurance. This will allow you to cover for the lost items, finance wise. Another way to recover would be to have more items in the back so you can restore the lost items. However this does not help financially, this just ensures the quantity of assets is recovered.

Problem 3

a) Sobig This was a Worm. This worm spreads through networksand emails. The worm will appear in an electronic email with certain subject such as "Re: Approved" or "Your details". Along with the email it will contain an attachment and when this attachment is opened the host is infected. This can also be classified as a trojan as it disguises itself as non malicious data. Once the user's devices were infected it wold begin replicating itself. It searched for other emails in order to spread itself further to other's devices. This process could interrupt performance. Other than this it did not harm the computer much.

b) WannaCry This is a Ransomware. This ransomware attacked Windows operating system and encrypt host's data. It would then demand payment in order to decrypt the host data. The payment was usually in cryptocurrency. It spread through an exploit called EternalBlue which was a exploit for earlier versions of Windows. Users who did not update were particularly vulnerable to the attack since their security was not up to date.

c) Zeus This is a Trojan. This was a Trojan that targets Windows operating system. This was spread through spam messages and drive-by downloads. For spam messages, these can be found on media. Once the user clicks on the link containing the Trojan it directs them to a website that automatically downloads the virus. For drive-by downloads, attackers would implement malicious code on trusted websites. Once the user visits that particular website then the Trojan will infect the host. Zeus can do many things. One malicious use case was to steal baking credential through monitoring and logging. It monitors the websites users visit and starts keylogging users on banking websites.

d) CIH This is a Logic Bomb. It had a payload trigger date of April 26 (any year). It spread through the Portable Execution (PE) file format. When a user executes an infected file, the host then becomes infected and 2 payloads are released. The first payload affects contents of the partition table which can cause the user's machine to halt or even cause the blue screen of death. The second payload attempts

to write to Flash BIOS. If the second payload is successful, then the computer will fail to start. The user needs to replace the Flash BIOS chip.

References

https://en.wikipedia.org/wiki/Sobig
https://www.f-secure.com/v-descs/sobig.shtml
https://www.computerworld.com/article/2579931/sobig-worm-getting-bigger.html
https://en.wikipedia.org/wiki/WannaCry_ransomware_attack
https://www.kaspersky.com/resource-center/threats/ransomware-wannacry
https://en.wikipedia.org/wiki/EternalBlue
https://en.wikipedia.org/wiki/Zeus_(malware)
https://usa.kaspersky.com/resource-center/threats/zeus-virus
https://en.wikipedia.org/wiki/CIH_(computer_virus)
https://malwiki.org/index.php?title=CIH

Exploit 2

a) The identified vulnerability is a buffer overflow in the *print_usage* function. This buffer overflow exists with the $snprintf$. Here the BUFF_SZ is larger then the given buffer by 512 bytes (BUFF_SZ = 1024, buffer = 512). This extra space allows for a buffer overflow exploit.

b) My program exploits the vulnerability by taking advantage of the buffer overflow. I run the *pwgen* program using *execve* in my *sploit*2 program. With *execve* we can replace $arg[0]$ and let $arg[1]$ be -h. This will enter in the else statement of the *print_usage* function and let us exploit the $snprintf$ statement. I replaced $arg[0]$ with a series of NOPS, SHELLCODE, then a return address, in that order. This is all one large string that I pass as $arg[0]$. An example of this would be:

$$NOPNOPNOP...SHELLCODE...ADDRADDR$$

I have multiple return addresses in order to maximize the chances of pointer pointing to my malicious return address. My malicious return address points to an address within the NOPs and the NOPs then create a NOP sled. The NOP sled will keep going to the next NOP untill it reaches my SHELLCODE. Once the program reaches my SHELLCODE it will create a shell, and because of the environment of pwgen the shell has root access.

c) You could do multiple things to fix this vulnerability

   (a) Change the buffer, in *print_usage*, to fit the BUFF_SZ. This way the string will get properly truncated and the buffer overflow could not occur.

   (b) Change the BUFF_SZ to fit the buffer in *print_usage*. This would fix the buffer overflow vulnerability, however this could cause other issues in the program because BUFF_SZ is used in other places.

   (c) Assuming that *pwgen* is the only file that generates passwords, you could remove the $arg[0]$ argument. You could change the

formatting to just have a "pwgen" string in the text rather than formatting with $arg[0]$.