Written Responses

1. Identify compromised CIA properties and possible attack methods.

    b. CIA Properties: Availability - Freezing of Jane's computer restricts her access to her data. Her own access to her computer's data was denied.
    Possible attack method: Ransomware - The attacker could have tricked Jane into downloading malicious software, which encrypts and locks her files upon installation. Then, the attacker can demand ransom (eg. cryptocurrency) to release them.

    c. CIA Properties: Confidentiality - If there is a location tracker installed on her phone, then her physical location is being monitored without her consent. This violates her privacy and the confidentiality of her location data.
    Possible attack method: Trojan - Jane could have accidentally downloaded some malicious app(malware) that seemed innocent from the app store, but it secretly uses user's location without asking for permission.

2. Jane's Android Automotive car
    a. Both.
        i. This is a security concern because if the multi-user isolation is not enforced properly, then unauthorised users can gain access to personal data stored in the car's memory. This violates Confidentiality and Integrity from compromised CIA properties.
        ii. It is also a privacy concern, because even though the sensitive data aren't accessed by other users, Jane's personal details can be prone to unintentional sharing, which violates Jane's privacy.
    b. What actions should Jane take?
        i. Set strict permissions for each user profile, so that we can ensure one user's data is completely separated from another user's data. Ensure that some data aren't accessible or modifiable by different users. This will prevent unintentional access.
        ii. Review and permanently clear data from time to time. Regularly clear out data stored in the car's system. Such data can be navigation history, outdated device connections, etc. This will ensure that not the entirety of information is at risk at all times.
3. Fighting against DoS attack
    a. Prevent it.
    Solution: Limit number of requests one user can send during a set time period. This prevents the attacker from flooding the system. (similar to limited # of attempts for entering passwords)
    Why & How it helps: The system can significantly limit the number of potential DoS attacks, because the number of requests are capped in the first place. It acts as a gatekeeper, preventing attacks to overwhelm the system.
    b. Deter it.
    Solution: Deploy a firewall. The firewall can inspect incoming traffic and filter out invalid requests based on company-defined rules. Also, additional rules can be applied dynamically, based on previous DoS attacks.

Why & How it helps: This solution acts as a discouragement factor for potential attacker. The barrier can dissuade them from continuing the attack to break the system.

   c. Deflect it.
   Solution: Traffic filtering based on geographical location of accessor (IP address). Block or limit traffic from atypical region of access with respect to the company's database, so that such access can be simply not reached.
   Why & How it helps: Deflects a portion of potential attack surface, thus makes the attacks more manageable. Server and network resources can be significantly conserved for non-malicious users.
   d. Detect it.
   Solution: Use traffic analysis tools. Monitor suspicious network traffic patterns in real-time, so that unusual spikes of access or patterns can indicate DoS attacks. Then, the administrators can be alerted in real-time if they actually detect a potential threat.
   Why & How it helps: This method supports early detection of malicious access and immediate response to mediate them. It ensures the company is ready to be alerted any time and minimize the impact of attacks.
   e. Recover from it.
   Solution: Backup to cloud, establish recovery plan for specifically DoS attacks. Regularly back up critical data to cloud, so that there is a full instance of the system ready to replace the attacked system.
   Why & How it helps: Having backup systems and a clear, precise recovery plan lets the company to restore essential services quickly. There will be little to no downtime and ensure smooth business continuity even under DoS attacks.

4. Classify malware.

   b. Conti
      i. Solution: Ransomware
      ii. How a computer becomes infected & How it spreads: This malware is delivered via phishing emails. It tricks recipients into downloading and installing the malware, or exploits vulnerabilities in remote desktop protocols and VPNs. Once Conti gets inside a network, it can possibly spread to other machines.
      iii. Resulting Effect: Encrypts the file of the infected computer, then demands for ransom (eg, cryptocurrency) in exchange for the decryption key. Also, it threatens to lead the stolen data, if the ransom is not paid to the attacker.
   c. Pegasus
      i. Solution: Spyware
      ii. How a computer becomes infected & How it spreads: Pegasus does not need to be clicked on to get it installed. This can be achieved through text messages or calls that specifically exploits unknown vulnerabilities.
      iii. Resulting Effect: Provides the spyer with spied content of the machine, such as messages, emails, camera, calls, contacts, etc - it ultimately turns the device into a surveillance tool.

d. QakBot
    i. Solution: Trojan (also has Worm capabilities)
    ii. How a computer becomes infected & How it spreads: It is mainly spread through malicious email campaigns. They specifically exploit software vulnerabilities, and can also spread through network drives and/or removable drives.
    iii. Resulting effect: QakBot is specifically designed to steal banking credentials and personal data. These two combined, can be used for fraudulent transactions, as well as installing other malwares.

e. Stuxnet
    i. Solution: Worm (along with Trojan-like payloads)
    ii. How a computer becomes infected & How it spreads: Stuxnet mainly spreads via infected USB drives. Once this USB gets plugged into a device, then it specifically exploits Window OS's vulnerabilities to execute. This exploit can spread to other computers over local networks.
    iii. Resulting effect: Previously, Stuxnet targeted attack on nuclear facilities of Iran to damage industry equipment. It modifies the code and operations of PLC(Programmable Logic Controllers) to make the system non-deterministic and behave not as intended. Doing so, it also sends back normal logs and feedback to the facility's operators.

Programming Question

sploit2.c

- What vulnerability does sploit2.c address?
    - Incomplete mediation
    - In the get_uid() function of pwgen.c, notice:

      ```
      dir = getenv("HOME");
      ```
    - Here, a normal program will then store “/home/user” in dir. This get_uid() function is used as a helper function - get_username() and check_perms() to is used to check for access permissions.
    - However, if we change the value of “HOME” to “/root”, then the we can get the uid of the root file, the user name of the root uid, and write the root user password in /etc/shadow with some generated password.
- How does your program exploit it?
    1. Set “HOME” to “/root” by setenv().
    2. Run pwgen, and write the output of the program into some file “pwgen”.
    3. Using strtok(), we get the field of length 8, <password>.
    4. Write a string script so that the user can “su root” with <password>
    5. Execute the string script to log in as user root.
- How can the vulnerability be fixed?
    - Don’t write your own get_uid() in pwgen.c, but rather use the built-in getuid() from <unistd.h>.

sploit3.c

- What vulnerability does sploit3.c address?
    - Buffer overflow.
    - Notice that pwgen doesn’t necessarily need an argument. For this reason, in pwgen.c’s print_usage(), argv[0] can be appended into the buffer within this function.
- How does you program exploit it?
    - Using that fact that executing pwgen without any arguments, we can change argv[0] to a buffer we’ve defined using `exec -a buff pwgen`.
    - This buffer will be oversized, because it will be filled with NOPs and the provided shellcode. It ensures that when print_usage() returns, the return address is completely replaced with the addresses of a NOP.
    - Thus, when pwgen resumes, the instruction pointer will align with a NOP, then proceed to traverse down the stack until the end of this NOP sled, then execute the shellcode
- How can the vulnerability be fixed?
    - In line 303, the following code:

      ```
      strncat(buffer, argv[0], BUFF_SZ);
      ```

      specifically exposes the cause of buffer overflow attacks, because it lets argv[0] be exposed. To mitigate this issue, let some constant string variable be argv[0], and use this variable replace the usage of argv[0] in pwgen.c.
    - In addition, we can use a programming language that supports bound checking, or create some shadow stack so that it monitors the original return addresses for the instruction pointer.

sploit4.c

- What vulnerability does sploit3.c address?
    - String format vulnerability.
    - Notice that in line 269, we are taking the string buffer and printing it to stderr. Here, you can trigger a string format vulnerability.
- How does you program exploit it?
    - This can be exploited by triggering a failure from check_perms(), ie. by getting check_perms() to return -1. The only case that holds is when the program fails to unlink the preexisting /tmp/pwgen_random.
    - Here, since the program is a directory, not a file, it will fail to unlink.
    - Soon, check_persm() will return -1, which triggers line 268 and 269 to run.
- How can the vulnerability be fixed?
    - Use %s to print the buffer on stderr.