**INF226 - Eksamen Vår 2020**

**Exercise 1:**

1. True. Companies can be fined if a security breach affect their users in a negative way.
2. True
3. False
4. True
5. True
6. False
7. False
8. False
9. True
10. False
11. True
12. False
13. False
14. False
15. False
16. True
17. True/False. Harder for attacker to attack if code is hard to understand, but easier for dev to protect program if code is easy to understand.
18. False
19. True
20. True

**Exercise 2:**

1. I would require passwords to be atleast 8 characters long and meet NIST requirements. I would also make them fill out two passwords fields to make sure they match and they choose their preferred password. Password needs to have some special characters and numbers, it cant just be plain text.
2. Z
3. Every user should be registered with either a email or phone number. In the case a password is forgotten a form to reset the password should be sent to either the email or phone number. The email and phone number needs to be confirmed prior to this.
4. Two factor authentication is when you have an extra step that needs to be fulfilled in order to validate user. Instead of just having a username and password, a user also needs to go fetch a unique code sent to them either via email or phone number. You can also use authenticator apps that constantly and often have unique codes that are switched out.

**Exercise 3:**

1. SQL Injection
2. The attacker has been performing SQL Injection, the program has failed sometimes but some of his request has been successful and the attacker has most likely extracted some private data or performed actions to the database that he is not supposed to do.
3. Personal and sensitive data could be leaked, password and usernames could be used on other sites.
4. The operators of the site will be held responsible in a data leak. They can be fined and sued. They also have to notify their users what has happened.

**Exercise 4:**



**Exercise 5:**

1. This is a protection mechanism put in place to detect buffer overflow attacks. Stack smashing detected tells us that an illegal input was sent to the program.
2. To bypass stack smashing detected you first need to find the position in memory that we are interested in. You can use flat() from pwn for this. You then send in 1000 characters to fill the 1000 “allowed” bytes. The position converted to bytes then needs to be added to the end of the 1000 bytes already filled. The data we are interested in at the respective address would be returned.
3. You will get back the programs standard output seeing as stack smashing detected was bypassed and no error message should appear related to that.
4. You can mostly do the same thing we did on b), but this time we will inspect the binary file and find the position we are interested in + where to perform buffer overflow using objdump and grep (built in tools).
5. Address space randomization (ASLR), constantly moving the address space locations around and making it nearly impossible to perform buffer overflow, seeing as the attacker needs to know the locality of executable code.
6. An alternative way to perform buffer overflow, that also bypass ASLR, is to first find the address of a known function or object in the code, then compute an offset between that address and the target function or object. By adding the address and offset together we end up with the position of the function or object we are interested in targeting.
7. There might be stack protectors in place (stack canary values) that prevent your attack.