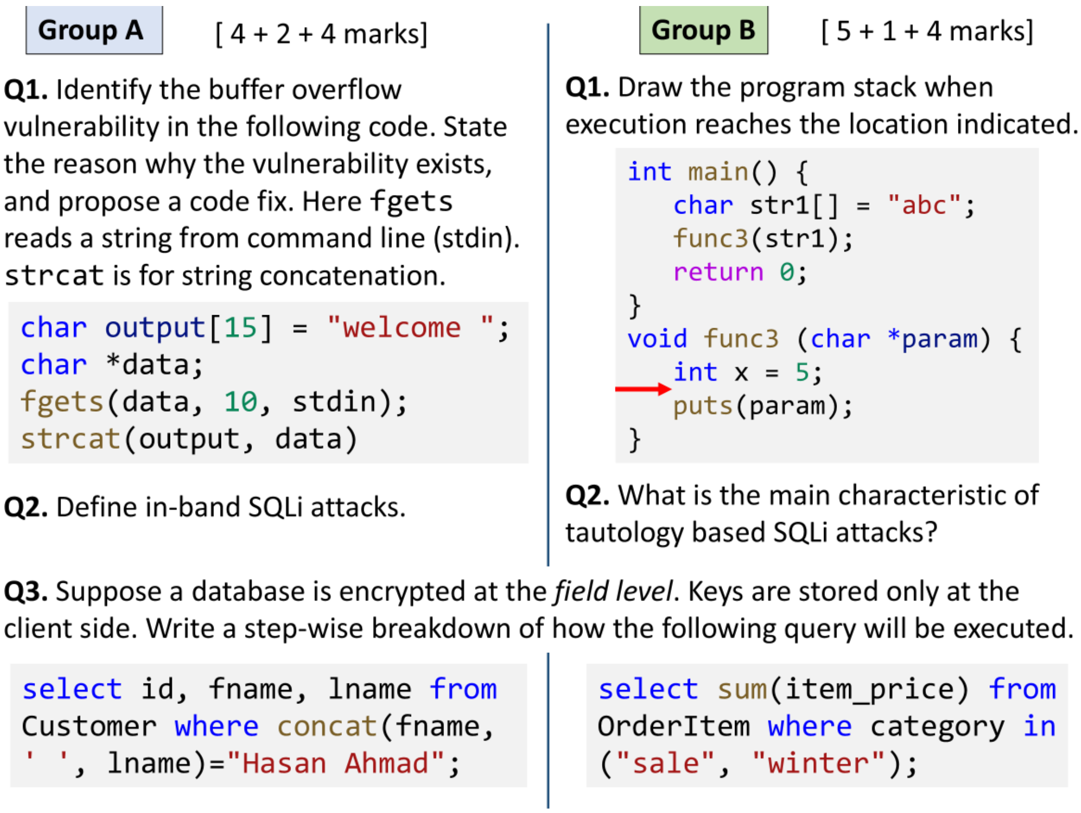


Group A [4 + 2 + 4 marks]

Q1. Identify the buffer overflow vulnerability in the following code. State the reason why the vulnerability exists, and propose a code fix. Here `fgets` reads a string from command line (stdin). `strcat` is for string concatenation.

```
char output[15] = "welcome ";
char *data;
fgets(data, 10, stdin);
strcat(output, data)
```

Q2. Define in-band SQLi attacks.

Group B [5 + 1 + 4 marks]

Q1. Draw the program stack when execution reaches the location indicated.

```
int main() {
   char str1[] = "abc";
   func3(str1);
   return 0;
}
void func3 (char *param) {
   int x = 5;
   puts(param);
}
```

Q2. What is the main characteristic of tautology based SQLi attacks?

Q3. Suppose a database is encrypted at the *field level*. Keys are stored only at the client side. Write a step-wise breakdown of how the following query will be executed.

```
select id, fname, lname from
Customer where concat(fname,
' ', lname)="Hasan Ahmad";
```

```
select sum(item_price) from
OrderItem where category in
("sale", "winter");
```

Group A

Q1

`output` array has capcity of 15 chars, initially 9 occupied (including null terminator). `data` string could be be at most 10 characters (including null). So their concatenation can create a string of 18 characters, which exceeds capacity. Hence `strcat()` call is vulnerable.

To remove vulnerability, replace `strcat` with `strncat` which also takes in a size argument. That way, programmer can pass a size of 15, and output buffer will not overflow.

Q2

Lec 14 slides 15.

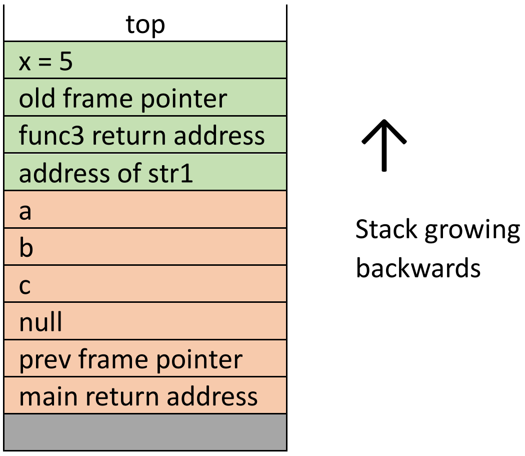
Q3

The WHERE clause is complex, and server can not filter the rows in encrypted form. So the server returns id, fname, lname columns of the whole Customer table. At client side, the encrypted columns are decrypted, WHERE clause is then applied to filter the rows.

Alternate answer: query processor on client side simplifies the WHERE clause as `fname="Hasan"` AND `lname="Ahmad"`. Then it encrypts "Hasan" to, say "xyz", and "Ahmad to "lmn". A transformed query is sent to server, which matches a small number of rows with fname "xyz" and lname "lmn". For those rows, id, fname, lname columns are returned to client in encrypted form, where they are decrypted and shown to user.

Group B

Q1



top
x = 5
old frame pointer
func3 return address
address of str1
a
b
c
null
prev frame pointer
main return address

Stack growing backwards

Students can also draw it inverted, growing downwards. Order of the items inserted is important.

Q2

Lec 14 slides 15.

Q3

Query processor on client side encrypts “sale” to, say “abcd”, and “Ahmad to “wxyz”. A transformed query is sent to server, which matches a small number of rows with category either “abcd” or “wxyz”. For those rows, item_price column is returned to client in encrypted form, where it is decrypted. The sum operation happens on client side after decryption.