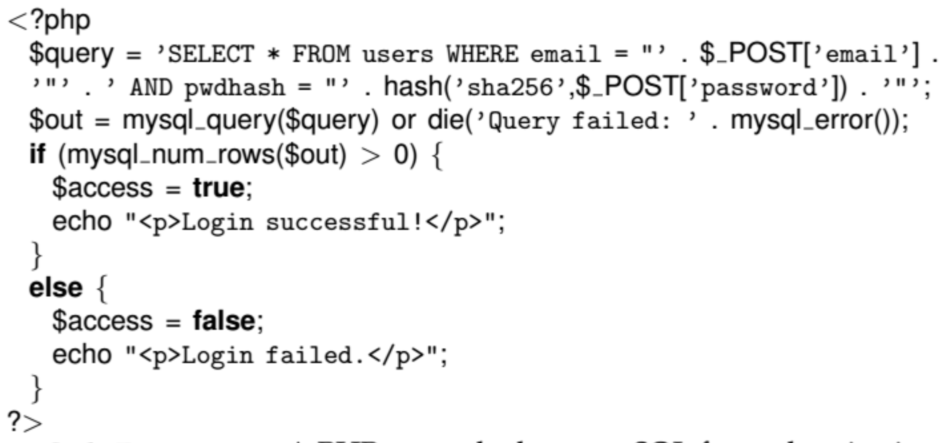
Overview: Question 1
These are short-answer questions.

1. Can an attacker steal Alice's cookies for www.example.com by exploiting a buffer overflow vulnerability in Alice's browser? Justify.

   Yes. If the browser code is altered via buffer overflow, he can make it do whatever he wants including stealing cookies.

2. In the SQL attack covered in class that bypasses authentication (see code below), if the email is set to " ; DROP TABLE users --
   Is there any harm done? What is the attacker achieving? Can he login? Assume there are syntax errors.



```
<?php
  $query = 'SELECT * FROM users WHERE email = "' . $_POST['email'] .
  '"' . ' AND pwdhash = "' . hash('sha256',$_POST['password']) . '"';
  $out = mysql_query($query) or die('Query failed: ' . mysql_error());
  if (mysql_num_rows($out) > 0) {
    $access = true;
    echo "<p>Login successful!</p>";
  }
  else {
    $access = false;
    echo "<p>Login failed.</p>";
  }
?>
```

Code Fragment 19: A PHP example that uses SQL for authentication.

   Yes, the attacker can delete the table user. But he cannot login since the query will return null.

3. A student says that a computer virus ate his homework, which was saved as a Word document. What kind of virus is the most likely culprit?

   A macrovirus.

Question 2: Web security
Part-A

Suppose we could deploy a mechanism (details irrelevant) that would ensure IP source addresses always corresponds to the actual sender of a packet—in other words, it is impossible for an attacker to spoof source addresses. For each of the following threats, explain whether (and briefly why) the mechanism would either:

- Completely eliminate the threat.
- Eliminate some instances of the threat, but not all of them.
- Have no impact on the threat.

(a) Non-persistent cross-site scripting (XSS) attacks.

Have no impact. *Non-persistent XSS attacks are conducted over HTTP, which uses an established TCP connection. The attack does not require spoofing in any form.*

(b) DNS cache poisoning attack

Completely eliminates the threat. *The local DNS server is expecting a reply from the proper authoritative server. An attacker cannot spoof the reply by using the IP address of the proper authoritative server.*

(c) DoS "amplification" where the attacker sends traffic to an Internet "broadcast" address.

Completely eliminates the threat. *This attack works by spoofing traffic seemingly from the victim and sent to the broadcast address of an Internet subnet.*

(d) Clickjacking.

Have no impact. *Attacks involving confusing the user regarding with which UI elements do not require spoofing of IP addresses in any form.*

Part-B:

Suppose a user turns Javascript completely off in their browser. For each of the following threats, indicate whether (and briefly explain why) doing so would either:

- Completely eliminate the threat.
- Eliminate some instances of the threat, but not all of them.
- Have no impact on the threat.

(a) Cross-site request forgery (CSRF).

Have no impact. *In general, CSRF attacks do not have a Javascript component. An attacker conducts them by manipulating a victim into fetching a URL that has side effects beneficial to the attacker.*

(more sophisticated attacks can be launched via java script, details need to be provided)

(b) SQL injection.

Have no impact*. SQL injection attacks target web servers, not browsers.*

(c) Non-persistent cross-site scripting (XSS).

Completely eliminate the thread. *The whole goal of a non-persistent XSS attack is to get a victim's browser to execute malicious Javascript.*

(d) Persistent cross-site scripting (XSS).

Completely eliminate the threat*. Similarly, the whole goal of a stored XSS attack is to get a victim's browser to execute malicious Javascript.*

(e) Phishing.

Has no impact*. Phishing in general does not need Javascript. Such attacks simply need to present a user with a convincing impersonation of another party that the user trusts. The impersonation could involve Javascript in order to render a suitably convincing web page for the user, details needed in this case.*

Question 3: Worms

A worm malware can use various strategies for target discovery to decide on who to infect next. Three such strategies are:

- Random: Target is chosen randomly
- Hit List: Target is from a pre-computed list present in the software
- Topological: Information on the currently infected machine is used to find the target

For each of these strategies, name one key advantage (over other techniques) and one key disadvantage (over other techniques)

Random:

- *no particular 'preparatory' work by the hacker required; the worm can spread to 'new' parts/networks of the Internet*
- *actual propagation/attack may be slow, as many of the 'probed'machines may not be vulnerable; small chance that the same machine may be probed multiple times*

Hit List:

- *each worm uses a part of that list allows for fast spread once the attack is launched, as only (truly) vulnerable machines will be 'probed'*
- *time consuming to identify a large number of potentially vulnerable machines (more prep work)*

Topological:

- *high likelihood of successful spread as machines that are topologically connected tend to have higher level of mutual trust*
- *the spread may be limited in terms of its reach*

Question 4: Memory Safety

Consider the below code:

```
int CheckPassword(char* pwd)
 {
   int i;
  int userAuthenticated = 0;
 char password[16];


 for (i=0;i<=16;i++)
   if (pwd[i] !=0) password[i]=pwd[i];

 if ((strncmp('secret',password,7) == 0)
 userAuthenticated = 1;

 return userAuthenticated;
 }
```

(Note: *strncmp(x1, x2, n) compares up to n characters between strings x1 and x2 and returns 0 if they are equal and non-zero otherwise.)*

1. What does this function do?

   This function is intended to check if the user password provided is 'secret' and if not return a 0 and otherwise return 1.

2. Can this function be used to hijack control of the program and run malicious code? Why or why not?

No. The "for" loop handles only 17 characters. To overflow the password buffer and get to the return address to overwrite it to hijack control, the for loop should run longer.

3. If this function safe under any input? If yes prove it. If not, explain what input would subvert it.

   No, it is not safe. One can however overwrite the lower address byte of the userAuthenticated variable (of the 4 bytes corresponding to int) with 1 and gain access.

Question 5: Format string Attacks

1. You observe an attacker sending the following string to a program you wrote.
   `“\xc4\xe4\xff\x9fAAAA\xc6\xe4\xff\x9f%08x%08x%08x%135u%n\n”`
   You suspect that the attacker is exploiting a format string vulnerability to overwrite a pointer in your program.

   a. At what address does the attacker think the pointer is located? Give the address in hex.

      0x9fffe4c4 (assuming little endian)

   b. What value is the attacker overwriting the pointer with? Give the value in hex.

   Each of the \xC4 count as 1 character. So it is 4(for\x..)+4(forAAAA)+4(for\x..)+8(for the %08x) +8(for the %08x) + 8(for the %08x) + 135 = 171 = 0xab