Attention: This exam has three sections and all of them count. Please write your name and student ID on each section. You have 80 minutes to complete the questions. As with any exam, you should read through the questions first and start with those that you are most comfortable with. Be sure to answer all parts of each question you answer. If you believe that you cannot answer a question without making some assumptions, state those assumptions in your answer. For partial credit, be sure to show how you arrived at your answer as well as the answer itself.

<table>
<thead>
<tr>
<th></th>
<th>/8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>/24</td>
</tr>
</tbody>
</table>
1. (8pts) Quickies

a) True or False (1pt each)

T  F  Alice has a message to send to Bob and she wants to sign the message using public-key cryptography so that Bob can verify that she sent it, she does so by using her private key.

T  F  Properly used, a MAC can provide both confidentiality and authenticity.

T  F  One advantage of public-key cryptography is that, when properly implemented, it is much faster than symmetric key cryptography.

T  F  The SSL protocol is used to both authenticate Web servers and provide confidentiality for client transactions with them.

b) Suppose Bob wishes to use public key cryptography to send a secure e-mail message, M, to Alice. Bob has public key Bk and private key bk while Alice has public key Ak and private key ak. Describe which keys Bob and Alice should use for encrypting and decrypting the message respectively.

**Bob will encrypt with Ak and Alice will decrypt with ak.**

c) Match the defense to what it accomplishes (draw lines) (1pt each)

i. Detects attempt to overflow local buffer into stack control data  
   Data Execution Prevention (DEP)

ii. Detects attempt to execute from stack data  
    Address Space Layout Randomization (ASLR)

iii. Makes it harder to transfer control to shellcode  
    Stack cookies

i. -> Stack cookies,  ii. -> DEP,  iii -> ASLR
2. Passwords (8pts)

On June 7th, 2012, LinkedIn confirmed that it had experienced a data breach that likely compromised the e-mail addresses and passwords of 6.5 million of its users. This confirmation followed the posting of the password hashes for these users in a public forum. One criticism of LinkedIn is that they used unsalted password hashes. In this question we’ll explore this criticism.

Assume that each stolen password record had two fields in it: [user_email, Hash(password)] and that a user login would be verified by looking up the appropriate record based on user_email, and then checking if the corresponding hashed password field matched the hash of the password inputted by the user trying to log in. By contrast, if LinkedIn had used a salted scheme, then each record would have had three fields: [user_email, salt, Hash(password+salt)] and login verification would similarly require looking up the salt and using it when matching hashes. Given this:

a) Suppose the attacker’s goal is to break your password via a dictionary attack. Does the lack of salting in LinkedIn’s scheme make this goal substantially easier?

No. Because even were there a salt, the attacker knows what salt is used for a given user. Thus, the time to create a dictionary of hashes is pretty much the same in both schemes.

b) Suppose the attacker’s goal is to break at least half of the passwords via a dictionary attack. Does the lack of salting in this scheme make this goal substantially easier?

Yes. Without salting one dictionary of hashes is sufficient for searching the entire set of users. With salting it will require a dictionary for each salt value seen.

c) Suppose you are contacted by the attacker and given a set of password hashes (that’s it, no user_name, no salt). Assuming the hash function is known, is there a measurement you could make in order to infer if the hashes are likely salted or not?

Yes. Recall that some passwords are much more popular than others. For example, the password 123456 is used by at least 0.1% of all accounts. Thus, if you hash such passwords and they appear disproportionately in the list then you might infer that the list is not salted. Similarly, even without doing a hash, if you sort the hashes by frequency, in an unsalted list you will expect that there is some hash that occurs with frequency ~ 0.1%, whereas in a salted list it will be ~0.1%/2^n where n is the size of the salt in bits.

d) It turns out that that roughly 20% of LinkedIn users with Yahoo Mail e-mail addresses used the same password at LinkedIn as Yahoo. You learn that, unlike LinkedIn, Yahoo salts its passwords. Should Yahoo be concerned about the LinkedIn breach or not?

Yes. For 20% of the Yahoo users in the LinkedIn breach, their user name and password is known to the attacker. Yahoo’s salting helps mitigates a breach of their password database, but doesn’t help at all in this case.
3. Crypto protocols and Software vulnerabilities (8pts, 4pts each for a and b)

a) Consider the hash-based one-time passwords we described in class. Under this approach, Alice generates a secret number \( w \), and then repeatedly applies one-way hash function \( H() \) \( t \) times to obtain \( H^t(w) \) (meaning \( H(H(H(H(H(H(...(H(H(w)))))))))) \) where \( H \) is applied \( t \) times). Alice then secretly gives Bob the value \( H^t(w) \) in a dark alley. Now to authenticate, Alice sends a message containing \( H^{t-i}(w) \) over a public channel to Bob. Bob then applies \( H \) to \( H^{t-i}(w) \) to obtain \( H^{t-i+1}(w) \) and verifies that this is the value expected (e.g., in the first round \( H(H^{t-1}(w)) = H^t(w) \), the value that Alice provided). If so then Alice is authenticated, and Bob updates the value he expected for future authentications to be \( H^{t-i}(w) \). Is this protocol subject to a man-in-the-middle attack? If not, explain why not. If so, describe the attack.

Yes. Adversary can intercept A’s message \( H^{t-i}(w) \) and relay it to B pretending to be A. (note this requires the adversary to be able to intercept and block/invalidate A’s message)

b) Consider the following code:

```c
1  int CheckPassword(char* pwd)
2  {
3     int userAuthenticated = 0;
4     char password[16];
5     int i;
6
7     for (i=0;i<=16;i++)
8        if (pwd[i] !=0) password[i]=pwd[i];
9
10    if ((strncmp('secret',password,7) == 0)
11       userAuthenticated = 1;
12
13   return userAuthenticated;
14 }
```

(Reminder: `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. Also, you may assume that the stack is organized as in class: with the stack growing from high addresses to low addresses and on a function call, the caller’s parameters are pushed first, then the return address, then the frame pointer, and then the callee’s locals, stored in the order they are declared in the callee)

This function is intended to check if the user password provided is ‘secret’ and if not return a 0 and otherwise return a non-zero value. Does this function do what it intends? If so, explain why it is safe under any input. If not, explain what input would subvert it. Be precise.

No. If the string pointed to by pwd contains at least 17 consecutive non-null bytes then the loop in lines 7 and 8 will overwrite the buffer password and modify the the value of userAuthenticated to make it non-zero (hence CheckPassword will return a non-zero value. Note, it is not possible to cause a control flow vulnerability here because the off-by-one overflow will never touch control data on the stack.)