CSci-454/554 Computer & Network Security
Final Exam – May 11, 2021

Please include the following statement into your document:
I understand that violations such as submitting work that isn't my own, sharing questions from this exam, asking for assistance from anyone other then the instructor, copy-pasting answers from outside resources may result in permanent failure of the course and/or any other sanctions in accordance with the William & Mary Honor Code. I understand that the contents of this exam are required to keep secret and after completing of the course such materials are required to be deleted including from any cloud resources such as Google Drive.

The exam is open book. You can use slides, book, videos, etc. You can use tools like calculator or python to solve some problems. The scores will be curved. If you see an error, please contact me via Discord or send me an email.

Student name: Daniel Quiroga

Question 1 (12 Pt)

Alice is sending a sequence of messages M_1, M_2, M_3 to Bob using a public network. An adversary Eve can listen and intercept/send any message on the network. Assume Alice and Bob **securely** shared a password **p="Fj#k39_38"** before starting communicating on the network. They also agreed on using a secure hash function **H()** and a symmetric block cipher that provides encryption function **E_K()** and decryption function **D_K()**.

Below is an example of Alice messages for unencrypted communication:

Message 1	Message 2	Message 3
M_1	M_2	M_3

a) Now Alice wants to add **confidentiality** to her messages (based on the shared password). Write down what she needs to compute and what she needs to send. **(4 pt)**

Alice's computation:

$h_n = H(p)$ and then $E_k(h_n\|m_n) = e_n,\ n \in \{1,2,3\}$ is what she would send

Messages sent:

Message 1	Message 2	Message 3
e_1	e_2	e_3

b) Now Alice wants to ensure message **integrity** i.e. Bob will be able to tell if message was manipulated by Eve or if Eve injected a message (even meaningless) Alice never sent. Assume k is a securely obtained symmetric key. Listed below is how Alice sends each message M_n for n ∈ {1..3}. Symbol "**|**" means simple concatenation. Mark with "**+**" all message formats that ensure **integrity**. Note that we care **only** about **integrity**, so ignore message confidentiality. Points will be subtracted for any format not marked or marked incorrectly. **(10 pt)**

$E_k(M_n)$ +	$H(M_n)$	M_n | $E_k(M_n)$ +	M_n | $H(p)$	M_n | n
M_n | p	M_n | $H(n)$	M_n | $H(M_n)$ +	M_n | $E_k(M_n)$ +	M_n | $H(M_n$ | $p)$ +
M_n | $E_k(H(M_n))$ +	$E_k(M_n$ | $H(M_n))$ +	$E_k(M_n)$ | $H(E(M_n))$ +	$E_k(M_n)$ | $H(M_n$ | $p)$ +	$E_k(M_n)$ | $E_k(H(M_n))$ +

c) Even if confidentiality and integrity are provided. Eve may try to violate ordering of the messages, e.g. by delaying message M_1 so it arrives after M_2 and M_3. Using format similar to the ones in previous part of the question, propose a mechanism that prevents Eve from reordering messages: **(2 pt)**

Each message sent with the message number then concatenated: $E_k(n)|E_k(h_n)$ that way they will be numbered once they are decrypted using the secured key and password

Question 2 (9 Pt)

```
echo(){
      char user_str[16];
      // .................. //
      scanf("%s", user_str);
      printf(user_str);}
```

Suppose user_str is located at address `0x7327ffd8` (on stack), while its address is read by **printf** function at address `0x7327ffc0` (on stack).

a. What does an attacker need to provide as input (for `user_str`) to read memory (string) at address `0xcafef00d` using the format string attack? Assume 32-bit little-endian x86 platform. (5 pt)

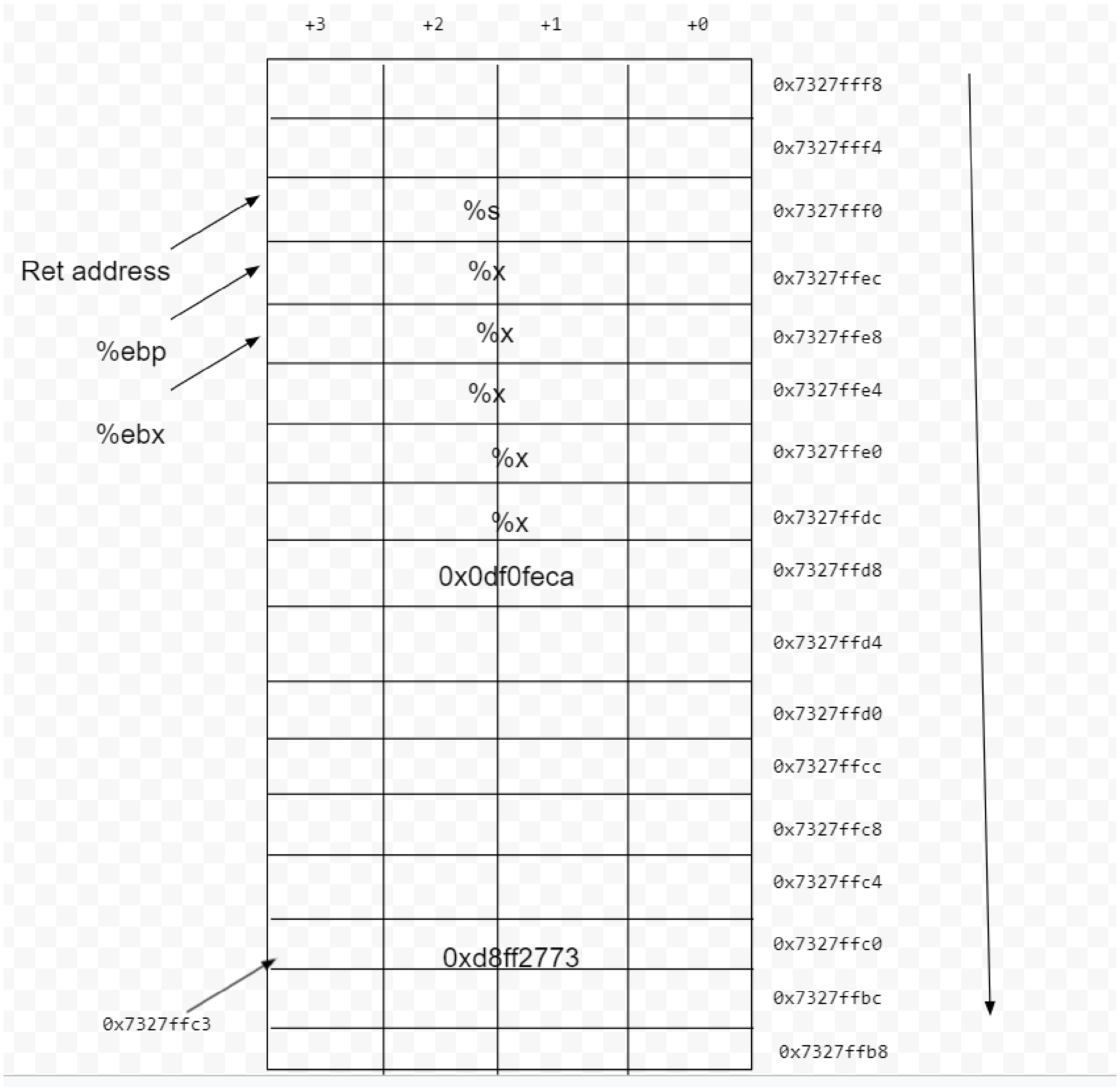
   "\x0d\xf0\xfe\xca%x%x%x%x%x%x%s" would allow us to read memory at 0xcafef00d

b. Using the template below demonstrate following things at the moment when printf is executed and user provided malicious inputs from (a) (10 pt):
   i. Stack growth direction (1 pt)
   ii. Memory contents of `0x7327ffc0 -- 0x7327ffc3` (2 pt)
   iii. **Each byte** (!) of the `user_str` string at the addresses where it is located (3 pt)
   iv. In addition to format string attack, demonstrated code is also vulnerable to buffer overflow attack capable of overwriting the return address and redirecting execution to arbitrary place. Show where such return address can be located on stack. Note depending on compiler, etc. return address can be at different addresses, you need to pick just one possible address. (1 pt)
   v. For this return address, show input data attacker needs to provide in order to redirect execution to `0xaabbccdd` (2 pt)

      Input data would be: 0xffffffff ffffffff ffffffff ffffffff ffffffff ddccbbaa

   vi. When is execution redirected to `0xaabbccdd`? (1 pt)
      1. When program enters `echo()` Yes __ No X
      2. When program leaves `echo()` Yes X No __
      3. When program enters `scanf()` Yes __ No X
      4. When program leaves `scanf()` Yes __ No X
      5. When program enters `printf()` Yes __ No X
      6. When program leaves `printf()` Yes __ No X

Template:


+3
+2
+1
+0
0x7327fff8
0x7327fff4
%s
0x7327fff0
Ret address
%x
0x7327ffec
%ebp
%x
0x7327ffe8
%ebx
%x
0x7327ffe4
%x
0x7327ffe0
%x
0x7327ffdc
0x0df0feca
0x7327ffd8
0x7327ffd4
0x7327ffd0
0x7327ffcc
0x7327ffc8
0x7327ffc4
0xd8ff2773
0x7327ffc0
0x7327ffbc
0x7327ffc3
0x7327ffb8

Question 3 (11 Pt)

Consider an encryption scheme where each character is upper case english letter or a space character. The message is encrypted using pad K consisting of same type of characters. The pad must be either same length as message or longer. The message is encrypted by:

$(M_i + K_i) \bmod 27 = C_i$

Decryption is done by:

$(C_i - K_i) \bmod 27 = M_i$

Where, M is message, K is pad and C is resulting ciphertext. Before encrypting/decrypting, characters are translated into numbers using following dictionary:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Note that the number for space character is 26

You may find helpful precomputed encryption/decryption matrices:
https://docs.google.com/document/d/1KciT7vjX0Nn9Rud6d1j50rNcMUUxVDIkGYoRmHsfup8/edit?usp=sharing

a. Write down your last name using capital letters (if you have special characters in your last name please omit them). Encrypt your last name using the following pad. `K="DFJJDFUHNSMLSNDUFHUDHFLNSJDNFVJJLKDNFUH".` Show the plaintext and ciphertext. (3 pt)
DFJJDFU – 03 05 09 09 03 05 20
QUIROGA - 16 20 08 17 14 06 00
19 25 17 26 17 11 20 -> **TZR RLA**

b. Show how you encrypted the **first** character (1 pt)

D – 03 Q – 16
We do 3+16 mod 27 = 19 and see that 19 = T so that would be the first character

Alice and Bob used encryption scheme described above. They sent two messages M1 and M2 encrypted with the same unknown pad using public channel. Turns out attacker tricked Alice into sensing a known message M1.

c. Assuming attacker intercepted both C1 and C2, explain how what attacker needs to do to recover M2. (2 pt)

The attacker would have to do $(C_i - M_i) mod\ 27 = K_i$ this is how the attacker would recover the key and then he would just preform the decryption from before with the C2 to get the M2 message.

d. How many characters of M2 attacker can recover? (1 pt)

The attacker would be able to recover at most |M1| characters since that's the highest amount of the key the attacker would be able to retrieve

e. If M1 = "TO BE OR NOT TO BE", C1 = "OV UDRVETYRSVXNCPD", C2 = "WSLSSZHGTROHOXDUFZD SUNTZJEEQ", find out as many characters of K and M2 as you can. Bonus 1 pt if you can figure out entire M2. (4+1 pt)

K = WHAT SHOULD WE DO
M2 = "ALL THAT GLITTERS "

Question 4 (7 Pt)

Alice and Bob want to exchange a session key for their communication using the Diffie Hellman (DH) key exchange. Assume, Alice generated private integer **a**. Bob generated private integer **b**. The value of generator **g** and modulo **n** are common and publicly known.

1. (4 pt) Assume a=212, b=321, g = 3, n=17. Demonstrate the entire procedure of key exchange. Show what value they arrive to. what integers Alice and Bob need to compute and send and how they arrive to the same shared integer.

Alice computes: $3^{212} mod\ 17$	Bob computes: $3^{321} mod\ 17$
Alice sends: 13	Bob sends: 3
Alice computes: $(3)^{212} mod\ 17$ = 13	Bob computes: $(13)^{321}\ mod\ 17$ = 13

Secretly shared common integer:

13

2. (2 pt) Basic implementation of DH is vulnerable to the man-in-the-middle (MITM) attack. Assume Alice and Bob trust the same certificate authority (CA). The certificate authority can sign data using its own private key with function $S_{CA}()$. What data do Alice and Bob have to ask the CA to sign to prevent MITM?

They would have to ask the CA to verify that $3^{212}\ and\ 3^{321}$belong to alice and bob, respectively

3. (1 pt) If Alice and Bob want to use AES to encrypt their communication, what should they do with the shared integer they obtained?

   They should hash the shared integer and use those bits as their key with a certain length

Question 5 (5 Pt)

Hotel owner decided to use hash chain to enable safe door lock mechanism. The mechanism works as following. First the owner picks a secret integer s. Then a hash chain $h_0..h_n$ is computed using secure hash function H().
Specifically $h_0 = H(s)$, $h_1 = H(h_0)$, $h_2 = H(h_1)$, …, $h_n = H(h_{n-1})$.
Then, the owner loads h_n into the doorlock device.
For first client, the owner loads h_n into the key card. Then the doorlock would check the key card in the following way:

```
# x is current valid key value
# H() is hash function
def check_key(key):
  if key == x:
    open()
```

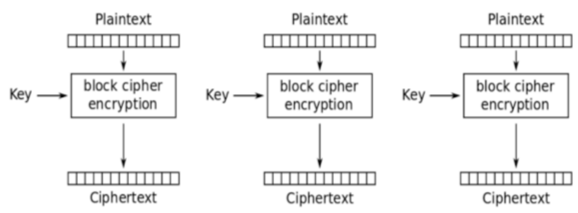
a. Assume the first client checks out and leaves. A new client arrives. The hash chain allows the owner to avoid manually resetting the key at the door lock device for each client. What value should the owner put in the key card for the this new client? (2 pt)

The owner should continue the chain and place the key $h_{n+1} = H(h_n)$

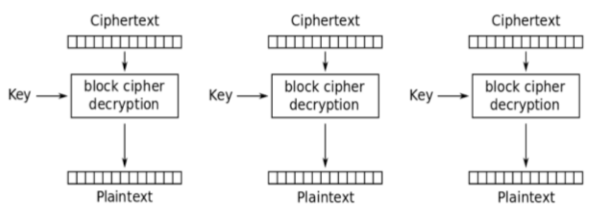
b. Complete the implementation of the check_key() function that enables the lock device to accept accept not only the currently stored key but also the key for future guest and when it does so, the old key should become invalid. (3 pt)

```
# x is current valid key value
# H() is hash function
def check_key(key):
  if key == x:
    open()
  else:
     check_key(H(key))
```

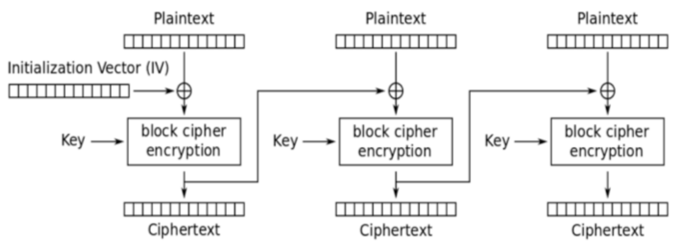
Q6. [12 pt] Assuming ciphertext blocks $c_1,c_2,...,c_n$ and plaintext blocks $m_1,m_2,...,m_n$, fill the table



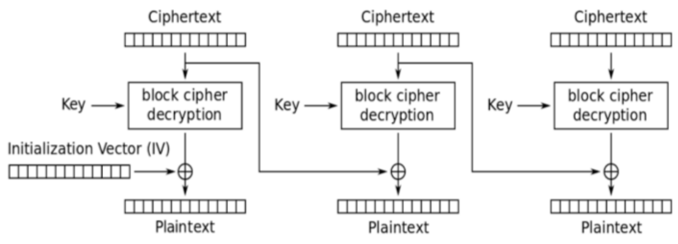
Electronic Codebook (ECB) mode encryption



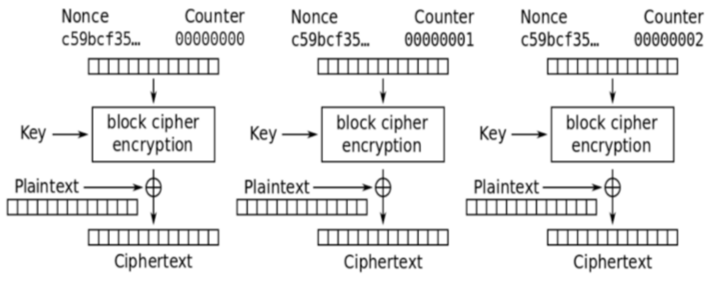
Electronic Codebook (ECB) mode decryption



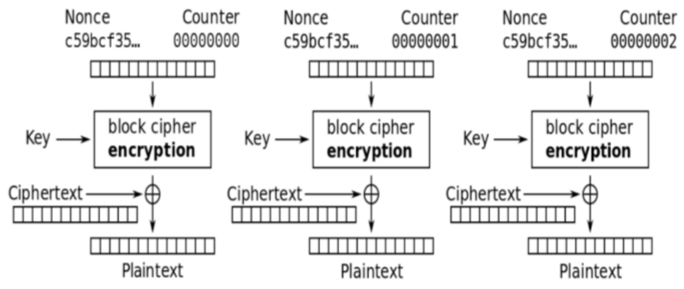
Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption



Counter (CTR) mode encryption



Counter (CTR) mode decryption

	ECB	CBC	CTR
Encrypt in parallel (Y/N)	Yes	No	Yes
Decrypt in parallel (Y/N)	Yes	Yes	Yes
Random block encryption (Y/N)	Yes	No	Yes
Random block decryption (Y/N)	Yes	Yes	Yes
Offers data pattern confidentiality?	No	Yes	Yes
If plaintext block m_1 is corrupted which ciphertext blocks will be damaged after encryption?	C1	C1, C2, C3	C1
If ciphertext block c_1 corrupted which plaintext blocks will be damaged after decryption?	P1	P1	P1

If initialization vector (IV) is not correct which ciphertext and plaintext block will be damaged?		All of the ciphertexts will be damaged which would mean all of the plaintext as well since the ciphertext is what is used to decrypt the messages	

Question 7 (20 Pt) -cd

In RSA we generate public **(e, n)** and private (**d**) key pair, (where **e** is public exponent, **d** is private exponent and **n** is public modulo) by starting from finding two large prime numbers **(p,q)**. Using Euler totient function (**φ**) show how each parameter is computed/picked:

a. (8 pt)

Assuming p = 31, q = 71

	How calculated	Value
n	$n = pq$	$n = 31 * 71 = 2201$
φ	$\varphi(n) = (p-1)(q-1)$	$\varphi(n) = (31-1)(71-1) = 30 * 70 = 2100$
e	We would find a value that is a co-prime with 2100 and is less than 2100	11
d	We need a value that when multiplied by e gives 1 mod 2100, here I would find the value that when multiplied by 11 gives me multiple of 2101	191

b. (5 pt)

Assume **m = 10** is an integer that we want to encrypt/decrypt using RSA. Show how this is computed:

Encrypt with **public** key formula:	$m^e = 10^{11} \equiv 1507 \, mod \, 2201$
Result:	1507

How it is decrypted formula:	$1507^{191} \equiv m \, mod \, 2201 \, and \, m \, would \, be \, 10$

c. (5 pt) Now assume we have a primitive hash function: **H(x) = x % 7**. Demonstrate how we can sign integer **t = 1948572** using keys you derived above.

Formula:	$H(t) = h, h^e \equiv c \, mod \, n$, we care about c which would be a sign
Result:	1067
Formula how signature will be verified:	$1067^{191} \equiv m \, mod \, 2201$ would give us m and using our H() function we would plug t into H() and compare the two values if they match we have verified the signature

d. (2 pt)

RSA is known to have issues with providing confidentiality when encrypting integers with certain properties. Show one such integer with a calculations highlighting the lack of confidentiality.

t = 3 would give me the same hash value at the original t. Meaning that we could have several different options. And the attacker would just have to try all numbers in whatever range until he finds a matching hash.

Question 8 (8 Pt)

Alice wants to send an email with signature of the email body attached. For recipients who want to verify that the cryptographic key used for signing really belongs to her and not to someone else, she also adds the public key certificate obtained from a trusted Certificate Authority. The structure of the resulting email looks like this:

"Text" | S | C, Where S is the signature and C is the certificate

Show how **S** and **C** are computed by giving **formulas** like **S = …..** *Indicate what party computes S and what party computes **C**.* Assume Alice has **$\text{Pub}_{\text{Alice}}$**, **$\text{Priv}_{\text{Alice}}$** keys, CA has **$\text{Pub}_{\text{CA}}$**, **$\text{Priv}_{\text{CA}}$** keys, also commonly known **H()** hash function and **E(Key, Data)** asymmetric encryption function. *Your explanation must be limited to this function and must not include any Sign() or GetCert() or similar functions.*

H(M) = h, then using that h, alice uses E($\text{Priv}_{\text{Alice},}$, h) = S this is all computed by Alice and she then places this S into her message

Using the certificate, alice would call E(Pub_{CA}, H(M)) = C, the computation would be done in the certificate authority and then just return a signature which users would be able to use the decryption for.

Question 9 (5 Pt)

A website needs to distinguish between admins and regular users. To do so, the `login.py` script that is responsible for user authentication does the following: 1) checks if provided (via form) username and password are correct 2) checks if the user belongs to administrator group, if so, the servers sets the cookie `user_admin=1`, otherwise the cookie `user_admin=0` is set.

1. (2 pt) Explain why this is not a secure way to distinguish between regular users and admins. Briefly describe how this website can be attacked.

We see that the admin is being set through cookies, a user would be able to set cookies using the Cookie module and hence gain access to the system as an admin user even if they are not admin users.

2. (3 pt) What would be a right way to distinguish between admins and regular users but still use cookies for authentication/session management?

A better way to distinguish between admins and regular users would be to user server-side secret which would be unknown to the browser. This would check if an account had admin privileges when preforming actions and then go through with the request if so, else it would require admin access.

Question 10 (6 Pt)

Consider following implementation of online calculator written in python.

```
#!/usr/bin/env python2.7

import cgi, cgitb, sys, os

form = cgi.FieldStorage()
exp = form.getvalue("exp")

html_header = """
<html>
<head><title>Calculator</title></head>
<body>
<div style="text-align:center;">
"""

html_footer = """
</div>
</body>
</html>
"""

def GenHTML(i):
    print "Content-Type: text/html"     # HTML is following
    print                               # blank line, end of headers
    print  html_header + i + html_footer

error_message = '<h1><b>Error: </b>Input is wrong!</h1>'
result_message_format = '<h3><b>Result: </b> {} </h3>'


if not exp:
    GenHTML(error_message)
    exit()

result = eval(exp)
GenHTML(result_message_format.format(result))
```

a. (3 pt)
Code is vulnerable to cross site request forgery attack Yes _X_ No __
Code is vulnerable to cross site scripting attack Yes _X_ No __
Code is vulnerable to code injection attack Yes _X_ No __

b. (2 pt) Analyze the code and **describe** the most impactful attack from your point of view the attacker can perform

I would say the CRSF attack would be the most impactful since the attack would be able to do both a XSS and code injection attack with some input BUT also be unknown to the server and take on an identify of someone else, making it that much more impactful when trying to figure out how the attack was preformed and who exactly did it, since he technically could be able to mimic that of an admin user.

Question 11 (3 Pt)

Assume the following HTML with PHP hosted at **http://foo.com/site/script.html**:

```
<HTML>    <TITLE> Search Results </TITLE>
<BODY>
Results for <?php echo $_GET[term] ?> :
. . .
</BODY>   </HTML>
```

1. (3 pt) Explain How can attacker utilize the vulnerable code demonstrated above to perform a cross-site request forgery (CSRF) attack on another website **bar.com**? Assume the person who clicks on the malicious link has an active session at **bar.com** and **bar.com** does not have CSRF countermeasures.

The attacker would just have to add in an invisible image to this webpage where the image sends a request of the form bar.com?total_amount=1000&item=3245 and hence would cause the user to spend x amount of money on item y which may belong to that of the attacker and he wants to sell worthless things to vulnerable people.

Question 12 (3 Pt)

Tom checked out his `/etc/shadow` file and discovered that his Linux system stores each password using a secure hash function where each hash is computed with a unique salt (which is stored in the same file). He noticed that the salt is long, consisting of 20 characters (upper and lowercase letters and numbers). He knows that passwords are hashed in the following way: H(password | salt). He concluded: *"Since the salt is very long and is added to my password, a weak password of 4 characters becomes secure password of 24 characters."*. Is Tom's logic correct? Why?

The password itself does not become 24 characters. The salt just gives an extra set of characters when hashed to make it harder to figure out what the hash is of the password. Since the password is only of length 4, the attacker could very easily try every combination with length 4 and eventually figure out the password that was used, hence making the logic flawed since the attacker will only need to figure out 4 characters rather than the 24 characters.