50.020 Security Mid-Term 2019

Date of examination: 7th, Mar, 2019

Time: 1:20 pm

Duration: 90 min

Full Name :

Student ID:

General Remarks

- This is a closed book exam. Do not use any digital or external material.
- This exam consists of 5 questions on 4 double-printed pages.
- The last 2 double-printed pages are empty, and can be used for (ungraded) notes.
- Each question has an assigned weight in points. There are 100 points in total.
- Write your answer in the answer sheets provided.
- Answer all questions as thoroughly as you can.

Question	Points	Score
OS Security +malware	10	
Hash Application: HMAC	28	
Web Security	22	
Basic Encryption/Decryption	16	
Hash and rainbow table	24	

Question 1 (OS Security, Malware: 10 points)

Alice received an email from someone who claimed to be an IT administrator from Alice's company. The email content was stated as "Urgent and Important!! You immediate response is needed!! We identified a potential cyber security offense to your account, during our regular IT security assessment. To best prevent any further bad consequence, we need your cooperation to update your account as soon as possible. Please refer to the attachment for the instructions on how to update your account". Alice felt very anxious and thus quickly downloaded the attachment and followed the instructions to update her account. Alice was asked to restart her computer. After restarting, Alice tried to open an excel file whereas an alert box popped out, notifying her that the file has been encrypted and some compulsory momentary donation is required for recovering the file. She tried to open other excel files in her hard disk, but was greeted by the same alert boxes. Finally Alice realized that the attachment she downloaded contained a malware. We call the malware that was sent to Alice as Alice's malware.

(a) Please identify the three types of malware that Alice's malware belongs to. For each of the three types, please justify why (Alice's malware belongs to this type). Please only consider malware types we discussed in class. (9 points)

Solution: any three from the following four types:

virus, ransomware, trojan, rootkit.

Why virus: For spreading the malware, Alice was asked to download the attachment (and open it).

Why ransomware: some compulsory momentary donation is required for recovering the file

Why trojan: the program appeared friendly, but essentially it's malicious.

Why rootkit: the malware tried to gain root privilege by restarting, and still active (hidden) after restarting .

(b) Which type of attack is used by the attacker that makes Alice easily download the malware? (1 point)

Solution: social engineering.

Question 2 (Hash Application: 28 points)

Sammy designed a custom hash function HS(). The hash function operates on t=(t_0 ,t_1 ,… ,t_n), a list of integers (each element is smaller than 1024). The hash function computes the hash as follows: HS(t) = $t_0 \oplus t_1 \oplus \ldots \oplus t_n$, with $\oplus$ being XOR.

(a) Please start by listing and briefly explaining the four required properties of a cryptographic hash function. (8 points)

Solution: The properties of cryptographic hash functions are:

Assume hash function is f().

1) preimage resistance: given hash value of that given hash value y = f(x), it is hard to find the input x for f to produce y.
2) second preimage resistance: given x and f, it is hard to find an input x' for f such that f (x') = f (x)
3) collision resistance: given f, it is hard to find any two inputs x, x' for f such that f (x) = f (x')
4) random oracle: for two correlated inputs x and x', the output of f is completely uncorrelated.

Which of these properties are fulfilled by HS()? (8 points)

Solution: None of the properties actually hold.

It is trivial to generate a pre-image, for example, H(m)=m.

it's easy to generate a second-preimage, for example, if given x=(m1,m2), then x'=(m2,m1).

There is a strong correlation between input and output (so no random oracle).

There is no collision resistance, for example, any pair of (m1,m2) and (m2,m1) making HS((m1,m2)) = H((m2,m1))).

(b) Based on the hash function HS(), the following message authentication code is constructed using a secret k shared between Mike and Tom:

HMAC(t, k) = HS(t||k) (with || as concatenation)

Mike then sends (t, HMAC(t, k)) with t = (222, 222, 222) to Tom (message has three blocks of value 222 each). Charles intercepted the whole message of (t, HMAC(t, k)). Charles also knows how the hash function HS () works in general. Charles can modify the message t to another message r ≠t (up to Charles' choice). Is Charles possible to generate a valid HMAC for r, using the intercepted

(t, HMAC(t, k))? Please justify your answer with details on why/how the attack is possible (or not). (12 points)

Solution: yes, possible to generate a valid HMAC for r, due to second-preimage attack.

Charles can choose any r=(r_0 ,r_1 ,… ,r_n) with $r_0 \oplus r_1 \oplus \ldots \oplus r_{n-1}=0$ and $r_n=222$. One simple example is r= (333, 333,222) . Since HS(r)=HS(t), then HMAC(r,k)=HMAC(t,k).

Question 3 (Web Security: 22 points)

For this question we only consider the web security attacks discussed in class. For all the web applications and servers mentioned in this question, we assume there is no countermeasures in place against the web security attacks.
If you need any assumption for answering the questions, please state the assumption clearly. For questions (a) and (b), if your answer is "yes", you only need to provide one attack (even if you think multiple attacks are possible)

(a) Is the following html &php code snippet vulnerable to any web security attack? Justify why if your answer is "no". Otherwise please state which attack is possible, provide a possible example to conduct the attack, provide one possible countermeasure against this attack. (7 points)

```
// The "accountID" parameter is assigned to $account variable
$accountID = $_GET['account'];
// The $ accountID parameter is passed as part of the query
$query = "SELECT * FROM accountTable WHERE accoutnID = $accountID ";
```

Solution: vulnerable to SQL injection..

One example of user input is: 123 OR '1'=='1'; DROP Table accountable; --

Countermeasure (any valid countermeasure is fine):

Use prepared statements with parameterized values.

Sanitise user input, for example, only allow user inputs with valid data types, for example, integers etc.

(b) Is the following javascript code snippet vulnerable to any web security attack? Justify why if your answer is "no". Otherwise please state which attack, provide a possible example to conduct the attack, provide one possible countermeasure against this attack. (7 points)

```
//set($url = $!request.getParameter('url'))
```

```
<script type="text/javascript">
    window.location = $url;
</script>
```

Solution: possible to XSS attack. redirect the victims' browser to a malicious url designed by the attacker.

One example of malicious url : http://attackerserver.com/ test.php?val=<script>alert('You are vulnerable to XSS.

Countermeasure: sanitising user input, for example, whitelisting/backlisting possible urls etc, or any other valid countermeasure.

(c) A visitor to the website is able to check the network connection via the following input box:

Ping: [ ]

Conduct a command injection attack by writing a bash script to the input box. Once the script is executed (successfully), the following functionalities are achieved: listing the folder and file names in the current working directory, deleting a file named good.txt (assume it exists), creating a new file named bad.txt into the current working directory. (6 points)

Provide a possible countermeasure against the command injection attack as mentioned in this question. (2 points)

Solution:

; ls ; rm good.txt ; touch bad.txt

Countermeasure (any valid countermeasure is fine):

Sanitising user input, for example, special symbols like ";"

Restricting the format only to be a.b.c.d where a,b,c,d are numbers.

Question 4 (Basic Encryption/Decryption: 16 points)

Given the following plaintext and its plaintext:

Plaintext	**happygirl**
Ciphertext	**dbusjoksl**

(a) Identify the encryption scheme used. Is it transposition cipher only? Or substitution cipher only (you do not need to consider Vigenere cipher), or a combination of both? Please justify why. (8 points)

Solution: A combination of both. (2 points)

Not possible only transposition since characters are changed. (3 points)

Not possible substitution ciphers. We only need to consider Caeser cipher where all characters are shifted same positions, but, that is not that case, for example, 'h' to 'd' takes 22 positions while 'a' to 'b' takes 1 position only. (3 points)

(b) We used the same encryption scheme as your answer to (a) above, but with different key(s), to encrypt another plaintext PT2. The ciphertext for PT2 is: **gvgjcujgc**. What is PT2? Show you decryption with detail. (8 points)

Solution: according to (a) above, a combination of transposition and Caesar cipher is used.

Try Caeser cipher, try frequency analysis, the letter with highest frequency is Letter 'g', so we try substitution key='G'-'E'=2. By reversing (decrypting) substitution with key=2, we have the following intermediary text: etehashea. We then try transposition decryption. There are 9 letters in total, so the table size should be 3x3 (not possible 1 x9, or 9x1). We have to try possible key (actually possible permutations of 1,2,3), in the end, by using the key 213, we retrieve PT2=hehatesea

Question 5 (Hash and Rainbow Table: 24 points)

Assume we are given a cryptographically secure hash-function (i.e. all four properties are fulfilled) to convert a set of passwords into their hash values. It's known to Charles that a password is always 9-character with each character being a digit decimal number or chosen from {'*', '$', '&','q', 'j', 'z'}. The length of the hash value is 128 bit = 16 Byte. For all the following questions, we assume the passwords are uniformly distributed, i.e. they are all equally likely to occur.

(a) Charles is considering to find the preimage of a given hash value, without using any pre-computed hash table or rainbow table. What is the expected number of hashing attempts for this? Please justify with details. (6 points)

Solution: The expected number of hashing attempts= 16^9

The length (#positions) of password is 9, and each position has 10+6=16 possibilities. Therefore in total there are 16^9 possible passwords.

(b) To speed up future preimage searches, Charles decides to make use of a pre-computed hash table which consists of all the possible (password, hash value) pairs. The hash table is stored in a hard disk. What is the minimum size (in bytes) of hard disk space needed for storing this hash table? Please justify your answer with details. (8 points)

Solution: The minimum size required is $16^9 *(9+16)=25 *16^9$ bytes.

A pair of (password, hash) is 9+16=25 bytes. There are 16^9 pairs required o store in the hard disk.

(c) The available size of Charles' hard disk is 2GB (only). Charles decides to utilize rainbow-table to trade storage against computational costs.
We consider a full utilization of the 2GB in the hard disk. For computational cost, we only consider hashing operation (comparison cost is relatively much lower, and thus ignored).
Answer the following two questions and show your working in details:

(1) How many hashing operations are needed for each chain in the rainbow table? (Hint: consider the factor of trading space against hashing operations as defined above) (5 points)

(2) How many hash chains are needed in total? (5 points)

Hints for Question (c): To get the number of hash values in a chain and the number of chains needed, we define a factor: the **factor of trading space against hashing operations** as the number of hashing operations needed (in the worst case) for finding a hash value match within a hash chain. You are given a hint on how to calculate this factor: the factor of trading space against hashing operations is equivalent to the ratio of hash table size to the rainbow table size. For example, if the ratio of hash table size to the rainbow table size is 10, then the factor of trading space against hashing operations is 10.

Solution:

(1) The factor of trading space against hashing operations=$25 *16^9/ 2G= (25 *2^{36}) / 2^{31} =50 *2^4=800$. This means 800 hashing operations are needed for each chain in the rainbow table.

(2) The number of hash chains are needed in total= #hash values/ #hash operations in each chain= $16^9 /800=2/25 * 16^7$