NTNU

Department of (Computer and Information Science)

Examination paper for (TDT4237) (Software Security)

Academic contact during examination: Jingyue Li

Phone: 9189 7446

Examination date: 24-May-2018

Examination time (from-to): 15.00-17.00

Permitted examination support material: D

Other information:

Language: English

Number of pages (front page excluded): 7

Number of pages enclosed: 1

Informasjon om trykking av eksamensoppgave
Originalen er:
1-sidig X **2-sidig ☐**
sort/hvit ☐ **farger ☐**
skal ha flervalgskjema ☐

Checked by: Per Håkon Meland

_____

Date Signature

Students will find the examination results in Studentweb. Please contact the department if you have questions about your results. The Examinations Office will not be able to answer this.

Introduction

In this course, the written exam will count 70% of the final grade and the remaining 30% of the final grade comes from the compulsory exercises. So, your final grade of this course will be:
(Points you get from this written exam) * 70% + your grade of compulsory exercises.

If you feel that any of the problems require information that you do not find in the text, then you should

- Document the necessary assumptions
- Explain why you need them

Your answers should be brief and to the point.

Problem 1 – (40 points)

1) (5 points) Explain what heap overflow is, and list at least three methods/strategies to defend against heap overflow.
   Answer:
   (2/5 points) Heap overflow attack is that hackers use malformed inputs to write its malicious code to overrun a heap's boundary and to overwrite adjacent memory locations. The malicious code in the overwritten location can later be jumped into and be executed.
   (3/5 points) Heap overflow countermeasures (to get the 3 points, you need to list at least three out of the following options):
   - Always use safe functions
   - Leverage defenses in compilers, e.g. GCC (-fstack-protector), Windows Visual studio (/GS option, /SAFESEH option, /SEHOP option)
   - Check length when read/write buffer
   - Use tools to audit source code (E.g. Coverity)
   - Static analysis and code review
   - Rewrite software in type safe language

2) (5 points) Explain the Vigenère method to encrypt and decrypt string, and explain how to crack the Vigenère method.
(2/5 points) Key is a string (e.g. "*cafe*"), not a single letter. Encrypt: shift each character in the plaintext by the amount dictated by the character of the key (with wraparound). Decrypt: does the reverse
(3/5 points) Crack the Vigenère method has two steps.
   - Step one is to brute force the length of the key.
     First, you guess a key length. Then, for each guessed key length, you extract ciphertext sequences. For each ciphertext sequence, you calculate similarities of the "letter of frequencies" of the extracted sequences and "letter of frequencies" of English. Then, you choose the length with the highest similarity.

   - Step two is to guess each character of the key.
     We assume you have found out the key length through step one. First, you extract the ciphertext sequences based on the key length. Then, you find out the most common character in the sequence, e.g., 'm'. Most likely, 'm' is encrypted from letter 'e'. So, you can guess key to encrypt the most common character is 'm' – 'e'.

3) (3 points) Explain how confidentiality and integrity are combined in SSL/TLS, IPSec, and SSH.
(1/3 point) SSL/TLS: First, you calculate MAC from plaintext. Then, you encrypt plaintext and MAC.

(1/3 point) IPSec: First, you encrypt the plaintext to get ciphertext. Then, you calculate MAC from ciphertext.

(1/3 point) SSH: First, you encrypt the plaintext to get ciphertext. Then, you calculate MAC from plaintext.

4) (2 points) Explain the SSL/TLS hand shake process.
(1/2 points) Preparation for shaking process: First, Public Keys(PK) of Certification Authority (CA), i.e., PK-CA, is embedded in all browsers and web servers when browser and servers are shipped. The server needs to generate its Secret Key (SK-Server) and Public key,

i.e. PK-Server. The PK-Server needs to be digitally signed by CA using CA's private key (SK-CA) to generate the certificate of the server. So, the certificate contains information of the server's Public Key.

(1/2 points) Shaking process:

- Step 1: the web browser sends request for hand shaking.
- Step 2: the server replies the request by sending back its certificate. From the certificate, the web browser can get information of the PK-Server.
- Step 3: the web browser generates the "shared key". The "shared key" is encrypted using PK-Server, and is sent back to the server.
- Step 4: the server decrypts the "shared key" using its SK-Server.

The hand shaking process finishes. Both the web browser and the server have the "shared key"

5) (5 points) Explain the possible vulnerability of DAC (Discretionary access control), and explain why Bell-LaPadula model can help defend against the vulnerability.
(3/5 points) DAC does not distinguish user and process. It is therefore vulnerable from process executing malicious programs (Trojan Horse) exploiting the authorization of the user. A scenario could be that the hacker creates a file, e.g., steal.txt, and give CEO authorization to write a file, without CEO's knowledge. Two hidden operations (Trojan horse), i.e., reading the secret file and writing to steal.txt, are added to a CEO's app. The CEO is lured to run the app. App executes on behalf of CEO (access control checks only the user, not process). So, reading the secret file and writing the information of the secret file to steal.txt is allowed.

(2/5 points) In Bell-LaPadula model, the rules are "no write down" and "no read up". For the same scenario above, when the hacker creates the steal.txt file, the hacker can only classify the file into a low class, e.g., "unclassified", because we assume the hacker does not have authority to assign a higher class for the file he/she creates. When the CEO is lured to run the app with Trojan horse operations, if the CEO's app runs with Secret class, the app cannot write to

secret.txt, because of "no write down". If the CEO's app runs with unclassified class, the app cannot read the secret file, because of "no read up".

6) (5 points) Explain what the "first one wins" principle of Android is and why such a principle can be vulnerable.
(2/5 points) Android app can define new permission types. First app to define a permission also sets the permission's attribute (e.g. protection level) regardless of an app that may define the same permission after that.

(3/5 points) A legitimate app recording heart beat has a permission type, e.g., read_heartbeat_sensor, the permission should be given "dangerous" protection level. Attacker wants to use his app to read heartbeat sensor data.

- Step 1: Attacker lures a user to install a malicious app and the app defines a new permission type, i.e., read_heartbeat_sensor, and gives it "normal" protection level.
- Step 2: The user later installs the legitimate app and the app wants to give read_heartbeat_sensor "dangerous" protection level. However, the definition of the protection level from the legitimate app will not succeed, because of "first one wins" principle. Thus, the attacker can easily read the heartbeat sensor data, because the protection level is "normal".

7) (5 points) Explain what web application firewall is in Azure, and explain why SQL injection, session fixation, session hijacking, and cross-site scripting can be fully or not fully defended by using the web application firewall.
(1/5 point) The function of web application firewall in Azure is to filter out malicious input and output of the web applications deployed in the cloud.
(1/5 point) SQL injection is very well covered by the web application firewall by setting rules to whitelist and blacklist malicious inputs and outputs.
(1/5 point) session fixation can only partially covered the web application firewall. The web application needs to issue a new token when the user is logged in.

(1/5 point) session hijacking is hard to be defended by the web application firewall, especially if the session token is stolen via the insure communication channel.
(1/5 point) cross-site scripting can partially be defended by web application firewall. Mostly, reflected XSS can be detected and prevented. The persistent XSS cannot be detected.

8) (2 points) Explain what BSIMM (Building Security in Maturity Model) is, and propose how can a software company use such a model to improve security of own product.
(1/2 point) BSIMM is a prescriptive model that describes what companies are doing to improve their software security.

(1/2 point) Companies can use BSIMM as measurements to compare with peers, to compare business units, and to compare themselves over time.

9) (5 points) Explain what vulnerability the XML External Entities Attack exploits, how an attacker can exploit the vulnerability, and how to defend against such an attack.
(1/5 point) Vulnerability: an application that parses XML input without disabling XML external entity and DTD processing.
(2/5 points) Attack: Using untrusted XML input containing a reference to an external entity as input to the application. The content of the external entity is processed by a weakly configured XML parser.
(2/5 points) How to defend: Disable XML external entity and DTD processing, or using input sanitization.

10) (3 points) Explain what password salting is, what kinds of attack password salting can defend against and what kinds of attack it cannot defend against.
(1/3 point) Salting is a defend to dictionary attack. It includes additional info. in hash, and the hash password is concatenated with salt (a random number).
(1/3 point) Can defend against dictionary attack against arbitrary users.

(1/3 point) It is ineffective to defend against dictionary attack targeting at a particular account or user.

Problem 2 – (30 points in total)

For each of the code snippets listed below, your task is to:

- Identify all security vulnerability in the code (Note: you may find more than one vulnerabilities in one code snippet. You need to list and identify all of them.)
- Explain why these are security vulnerabilities/issues
- Fix the code (You may use pseudo-code for this. Remember to explain your solution).

Code snippet 1

Source: CWE-384

```
1.  <?php
2.  $SessionID = md5($UserName);
3.  if (empty($_COOKIE["SESSION_ID"]))
4.          setcookie("SESSION_ID",$SessionID);
5.  if ($_COOKIE["SESSION_ID"] == $SessionID):
6.          echo "Hello ".$UserName;
7.  else:
8.          echo "Please, enter your credentials";
9.  endif;
10. ?>
```

(2 /30 points) Vulnerability identified: line 2: MD5 is an outdated encryption

(2 /30 points) Vulnerability fix: line 2, change MD5 to SHA1 or SHA2

(2 /30 points) Vulnerability identified: line 2: predictable session token solely based on username. An attacker, who knows username of the victim, can forge the cookie and successfully authenticate against the web application.

(2 /30 points) Vulnerability fix: line 2, add other information to create the user session token.

Code snippet 2

```
1.  <form action="changeAddress.php" method="POST">
2.  <p><input type="text" name="newAddress" /></p>
```

```
3. <p><input type="submit" value="Change Address" /></p>
4. </form>
```

changeAddress.php

```
1.  <?php
2.      session_start();
3.
4.      if (isset($_REQUEST['newAddress'])) {
5.        change_address($_REQUEST['newAddress']);
6.      }
7.      echo "<p>Your address has been changed to $newaddress </p>";
8.  ?>
```

Note: change_address () is a user defined function to store the new address into the database. We assume that this function is secure.

(2 /30 points) Vulnerability identified: changeAddress.php line 5. Vulnerable to XSS attack, because there is no input sanitization.
(2 /30 points) Vulnerability fix: changeAddress.php line 5. Sanitize the inputs using blacklist or whitelist.

(2 /30 points) Vulnerability identified: changeAddress.php line 5. Vulnerable to CSRF attack, because there is no code to check if the request comes from a legitimate user or not.
(4 /30 points) Vulnerability identified: changeAddress.php line 5. Store a randomly generated token for each authenticated user. Add security tokens to transaction pages using hidden field. Verify that server-side and client-side tokens match.

Code snippet 3

Source: CWE-613

```
1.  <?php
2.  if (empty($_COOKIE["SESSION_ID"])):
3.          $SessionID = GenerateSecureToken();
4.          setcookie("SESSION_ID",$SessionID, time()*3600);
5.  elseif (ValidateSession($_COOKIE["SESSION_ID"])):
6.          echo "Hello ".$UserLogin;
7.  else:
8.          echo "Please, enter your credentials";
9.  endif;
10. ?>
```

(2 /30 points) Vulnerability identified: Line 4, Insufficient session

expiration. The session has been last for too long time.
(2 /30 points) Vulnerability fix: Line 4, shorten the duration of the session.

Code snippet 4

Source: https://www.acunetix.com/blog/articles

```
1. <html>
2. <head>
3. <title>Custom Dashboard </title>
4.     ...
5. </head> Main Dashboard for
6.
7. <script>
8.   var pos=document.URL.indexOf("context=")+8;
9.   document.write(document.URL.substring(pos,document.URL.length));
10. </script>
11.    ...
12. </html>
```

Note: This is a web page http://www.example.com/userdashboard.html. The result of http://www.example.com/userdashboard.html?context=Mary would be a customized dashboard for Mary, containing the string “Main Dashboard for Mary” at the top.

(2 /30 points) Vulnerability identified: Line 8 and 9. The vulnerability is DOM XSS. The malicious script can be embedded in the URL as follows
http://www.example.com/userdashboard.html?context=<script>SomeFunction(somevariable)
(2 /30 points) Vulnerability fix: Any one of the following will be regarded as sufficient to get full point.

- Avoiding client-side sensitive actions such as rewriting or redirection, using client-side data;
- Sanitization of the client-side code by inspecting and securely handling references to DOM objects that pose a threat, such as url, location and referrer, especially in cases when the DOM may be modified;
- Using intrusion prevention systems which can inspect inbound URL parameters and prevent the inappropriate pages to be served.

Code snippet 5

Source: https://www.hackthis.co.uk

```
1. <?php
2.   $page = $_GET;
3.   $filename = "/pages/$page";
4.   $file_handler = fopen($filename, "r");
5.   $contents = fread($file_handler, filesize($file));
6.   fclose($file_handler);
7.   echo   $contents;
8. ?>
```

(2 /30 points) Vulnerability identified: Line 4 and 5. Directory traversal vulnerability. If the input is "view.php?page=../admin/login.php", then, instead of getting a file from the pages directory the ../ traverses to the parents directory so instead gets the file /admin/loin.php. This is very bad news as it will most likely contain admin or database credentials.
(2 /30 points) Vulnerability fix: Input sanitization. Using regex to remove all ../s but there are some nice functions built into PHP that will do a much better job. $page = basename(realpath($_GET)). More details can be seen at https://www.hackthis.co.uk/articles/common-php-attacks-directory-traversal.

Problem 3 – (30 points)

Case description:

Company A is developing an IoT (Internet of Things) – based remote rehabilitation consulting service.

A patient with rehabilitation needs will log each day's activity using sensors installed within a wearable device. Using Bluetooth, the wearable device continuously communicates with an app of company A running on the patient's mobile phone (the mobile phone runs on Android platform). The activity data are stored at an external SD (Secure Digital) card of the patient's mobile phone. When the patient wants to upload the activity data to the web server of company A, the patient needs to log in the server first and then send the data. After the data is uploaded to the server, the corresponding data in the SD card will be

deleted, to save space for new data.

When a therapist wants to read the activity data, the therapist needs also to log in to the server. Based on some statistical analysis, the therapist can advise the patient to do certain exercises more often. The advice will be sent to the patient using emails. The patient pays the therapist based on advice provided by the therapist.
To use such a service, the patient needs to register his or her personal information, such as username, password, email address, age, gender, and some medical record to inform the therapist about the symptoms and what kinds of advice he or she needs. In addition, the patient can store credit card information in the server of company A for one-click payment. If the patient does not want to store the credit card information, the patient needs to type in such information every time he or she pays.
The therapist also need to register, and fill in some information, such as username, password, email address, name, office address, a short CV, and a bank account to receive the payment.

Your task is to make a risk-based assessment of this application based on RMF (Risk Management Framework).
Your tasks include:

- Identify business goals, business assets, and business risks (5 points).

(2/30 points) Business goals: List minimum four of the following or other goals that are valid will get full points.

- The system should be available
- The system should be easy to use
- The payment system should be secure and fast
- The user information should be secure
- Privacy of the user should be protected according to GDPR

(2/30 points) Business assets: List minimum four of the following or other assets that are valid will get full points.

- Activity/medical data
- Patients, Therapists & Admins credentials

- Encryption keys
- Patients personal data (e-mail, age, gender...)
- Credit card and bank account information
- Therapists personal data

(1/30 points) Business risks: List minimum two out of the following or other risks that are valid will get full points.

- Legal issue and reputation damage due to bank info. or credit card numbers are stolen
- Lose business because the system is not available during a significant amount of time
- Legal and GDPR issue because medical and personal data are disclosed
- Lose business because the payment system does not work
- Legal issue and reputation damage because user or admin credentials are disclosed

- Identify at least 10 technical risks using threat modelling. The technical risks can be relevant to web server of company A and the mobile application of company A (10 points). (Note: You do not need to draw the threat modelling graphs. However, you need explain briefly how the threat modelling, e.g., misuse cases and attack trees, are applied to help you identify the technical risks.)

(10/30 points), The following are examples of technical risks that are valid. There are other technical risks that are also regarded as valid, but not are listed here. If you list 10 valid risks, you will get full points. Each valid risk will count one point.

- TR1: Brute force guessing of passwords.
- TR2: The attacker can perform SQL injection attack when filling forms
- TR3: The attacker can inject malicious code when filling forms
- TR4: CSRF attack. The attacker lures therapist to click on links or run scripts that do malicious operations, e.g., changing medical record.

- TR5: Attackers can steal information transmitted between web browser, mobile and server through eavesdropping.
- TR6: SD-card information is stolen and the information is disclosed to attackers.
- TR7: Session fixation. The attacker gives user attacker's own session token, user logs in, and then the attacker's token is elevated.
- TR8: Session theft. The attacker steals session information of the patient or doctor and impersonates them.
- TR9: DDOS. The attacker sends a huge number of packets to the server and makes the server unavailable to legitimate user.
- TR10: Credentials on the server are stolen and disclosed to attackers.
- TR11: Secrete keys are not managed properly and are disclosed to attackers.
- TR12: The attacker modifies the data stored in the system and misleads the therapist.

- Derive security requirements from each technical risk identified, and design and describe black-box penetration test cases (including test steps and expected results of each step) to verify each derived security requirement (10 points)

(10/30 points) The following are examples of security requirements and tests that are valid. There are other security requirements and tests that are also regarded as valid, but not are listed here. If you list 10 valid requirements and tests, you will get full points. Each requirement and test will count one point. Each test should have test steps and expected test results.

- Req1: The system should have strong password policy.
- Test1: Perform dictionary brute force password attack.
- Expected test results: The password should not be cracked with limited effort.

- Req2: Prepared statement should be applied or inputs should be sanitized everywhere the user can give inputs.

- Test2: Input malicious SQL injection command in every place where free text can be typed in.
- Expected test results: No SQL injection attack is succeeded.

- Req3: Input sanitization of code injection should be performed at every place where free text can be typed in and some parts of text will be echoed back to the browser.
- Test3: Input malicious JavaScript.
- Expect test results: The JavaScript is sanitized and will not execute.

- Req4: The system should have mechanism to check that all commands are issued from legitimate users.
- Test4: Log in and get a valid session token. Use the token to call script of the system from another session.
- Expected results: The script will not execute.

- Req5: All communications between clients and servers should be encrypted properly.
- Test5: Use eavesdropping tools to collect information transmitted between clients and servers and read the information.
- Expected test results: The information is encrypted and is not human-readable.

- Req6: All information in the SD card should be encrypted properly.
- Test6: Download the data from the SD card and open it and read it.
- Expected results: The information is encrypted and is not human-readable.

- Req7: The system should issue new session token when the user logs in.
- Test7: Access the website and read the session token received. Log in the system and read the session token received after log in.
- Expected results: The session tokens should be different.

- Req8: All session should automatically expire after 10 minutes.
- Test8: Log in the system and get the session token. After 11 minutes, use the same token to log in.

- Expected. The log in is rejected.

- Req9: The system should have backup servers to defect against DDOS.
- Test9: Simulate 10,000 packets and send them to the server.
- Expected results: The system can still respond a request within 3 seconds.

- Req10: All credential files should be encrypted.
- Test10: Download the credential files from the server, open and read them.
- Expected test results: The information is encrypted and is not human-readable.

- Req11: All secret keys should be stored in a safe place.
- Test11: Perform various attacks to find the secret key of the systems.
- Expected results: The secrete key cannot be found in the system.

- Req12: All sensitive data in the system should be hashed.
- Test12: Open the sensitive data in the system and read them.
- Expected results: The sensitive data should appear as results after hashing, e.g., 256 bits. Another method is to read the software code to see if a proper hash function has been used to hash the information before storing them.

- This application must be compliant with General Data Protection Regulation(GDPR). List data of this application that can directly or indirectly identify a natural person, and discuss how company A can address the privacy issue of this application from transparency, fair use, and minimalization perspectives (5 points)

(2/30 points) These are examples of data that can directly or indirectly identify a natural person. If you list minimum four out of the following list or other valid ones, you will get the full points.

- Direct data (patients): email, credit card information
- Direct data (therapists), name, office address, CV, Bank account

- Indirect data (patient): username, medical record, age, gender, sensor data

(3/30 points) How to address privacy issue from transparency, fair use, and minimization perspectives. These are just reference answers. We will also give you full points if you provide other valid answers.

- Transparency: Company A should inform the user what the user data is used for and who will get access to the data. The data usage should be logged.
- Fair use: Nobody who are not eligible or who do not need to get access to certain data should get access to them.
- Minimization: Company A should only collect and store data that is needed.