NTNU

Department of (Computer and Information Science)

Examination paper for (TDT4237) (Software Security)

Academic contact during examination: Jingyue Li

Phone: 9189 7446

Examination date: 16-December-2016

Examination time (from-to): 9.00-13.00

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 – (45 points)

1) (5 points) Explain what SQL injection attack is and its vulnerability exploited by this attack. List at least three countermeasures of this attack.
   - (20%) A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS) and so on (reference: OWASP guide).
   - (20%) The main vulnerability exploited by SQL injection attack is that the input data from the client is not sanitized or filtered.
   - (60%) Lists of countermeasures (three out of five):
     - Blacklisting
     - Whitelisting
     - Escaping
     - Prepared statement & bind variables
     - Mitigating impact

2) (5 points) Explain what buffer overflow is, and list at least three methods/strategies to defend against buffer overflow.

- (40%) Buffer overflow attack is that hackers use malformed inputs to write its malicious code to overrun a buffer's boundary and to overwrite adjacent memory locations. The malicious code in the overwritten location can later be jumped into and be executed.
- (60%) Buffer overflow countermeasures (three out of five):
  - 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

3) (5 points) List the three general ways of authentication and discuss their advantages and disadvantages.
   - (40%) Lists
   - Something you know (e.g. password)
   - Something you have (e.g. mobile phone)
   - Something you are (e.g. fingerprint)
   - (60%)
   - Something you know
     - Advantage: Simple to implement and Simple to understand and use, recovery, distribution, remember.
     - Disadvantage: Easy to crack
   - Something you have
     - Advantage: Hard to crack
     - Disadvantage: Can be stolen and forged. Strength of authentication depends on difficulties of forging
   - Something you are
     - Advantage: Hard to crack and Hard to be stolen
     - Disadvantage: Accuracy: False negative/False positive. Social acceptance and privacy issues. Hard key management (not replaceable)

4) (5 points) Explain encryption and decryption algorithm of One Time Pad (OTP), and explain why it is insecure to use the same key to encrypt two or several messages using OTP.
   - (40%) Encryption and decryption of OTP

     Key has the same length as the plain text
     Encryption $c = E(k,m) = m \oplus k$
     Decryption $D(k,c) = c \oplus k = (m \oplus k) \oplus k = m$

     K is encryption key, M is plain text, C is Ciphertext.

   - (60%) Why it is insecure to use the same key to encrypt two or several messages using OTP

     $C_1 = E(k,m_1) = m_1 \oplus k$
     $C_2 = E(k,m_2) = m_2 \oplus k$
     Eavesdropper does:
     $C_1 \oplus C_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) \quad m_1 \oplus m_2$
     $m_1 \oplus m_2$ reveals m_1, m_2 information

     Any example to explain this is acceptable.

5) (5 points) Explain what digital signature is and how digital signature is used in the SSL/TLS handshake process to exchange the secret key.
   - (40%) What is digital signature?
     - Digital signature is to make signature depend on document. It is acceptable if students show that they understand that Digital signature is made by encrypting the document using private key and decrypting/verifying the signature using public key.

     - More complete answers of Digital Signature
       Bob first sends own public key (PK_{Bob}) to Alice.
       Bob sign the document using F^{-1}. Sign (SK_{Bob}, m) = F^{-1} (SK_{Bob}, Hash (m)) generates sig.
       Alice wants to know if the document is the one signed by Bob
       Alice receives PK_{Bob}, sig, and m.
       Check if F(PK_{Bob}, sig) = Hash (m) ?

- Yes: the document is signed by Bob
- No: the document is not signed by Bob

- (60%) How digital signature is used in SSL/TLS handshake process?

  First of all, Public Keys(PK) of Certification Authority (CA), i.e. PK_{CA} is embed in all browsers and web servers when browser and servers are shipped.

  In one step of the handshaking process, the document contains PK of the web server is Digitally signed by CA using CA's private key (SK_{CA}). Since every browser has PK_{CA}, the browser can use PK_{CA} to decrypt the document contains the PK of the web server. Then the browser will get PK of the web server. The PK of the web server will later be used to exchange secrete key between browser and server.

6) (5 points) List software security touchpoints in order of effectiveness. (List 40%, Order 60%)
   1. Code review (bugs)
   2. Architectural risk analysis (flaws)
   3. Penetration testing
   4. Risk-based security tests
   5. Abuse cases
   6. Security requirements
   7. Security operations

7) (5 points) Explain why Biba model can help protect data integrity.
   - (50%) No improper modification of high integrity object from low classified subject (No write up), e.g. Software downloaded from web not write to OS
   - (50%) High integrity object is not contaminated due to read and use unreliable data (No read down), e.g. Signaling sys. does not use data from passenger info. sys.

8) (5 points) Explain what “protection level” is in Android platform. What are the Normal, Dangerous, and Signature protection levels?
   - (50%) Protection level defines how risky a permission is and controls how to grant the permission.
   - (50%) Explain different protection levels
     - Normal (no user approval needed)
     - Dangerous (require user approval)
     - Signature (only for apps with same certificate)

9) (5 points) Explain what BSIMM is, what OpenSAMM is, and the main differences between them.
   - (50%) BSIMM is a descriptive models describe what is actually happening. The idea of BSIMM is to build a maturity model from actual data gathered from well-known large-scale software security initiatives.
   - (50%) OpenSAMM is a prescriptive models describe what you should do. OpenSAMM has defined several maturity levels to define and measure how well the software security initiatives of a company are. OpenSAMM consists of 4 core business functions (governance, construction, verification, deployment), which has three security practices with 3 levels of maturity.

Problem 2 – (20 points)

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

- Identify what you think is the main security vulnerability
- 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 (5 points)

Source: https://www.htbridge.com/vulnerability/

```
<?php
    $SessionID = SHA256($UserName);
    if (empty($_COOKIE["SESSION_ID"])){
            setcookie("SESSION_ID",$SessionID);
5.     }
6.  ?>
```

- (40%) Main vulnerability: line 4. Possible to predicate the session id by attackers, because the session ID is generated by using the UserName.
- (60%) Fix: To add randomness in the session generation.

Code snippet 2 (5 points)

Source: http://shiflett.org/articles

HTML form

```
1.  <form action="buy.php" method="POST">
2.  <p>Symbol: <input type="text" name="symbol" /></p>
3.  <p>Shares: <input type="text" name="shares" /></p>
4.  <p><input type="submit" value="Buy" /></p>
5.  </form>
```

buy.php

```
1.  <?php
2.      session_start();
3.      if (isset($_REQUEST['symbol'] && isset($_REQUEST['shares'])) {
4.         buy_stocks($_REQUEST['symbol'],$_REQUEST['shares']);
5.     }
6.   ?>
```

Note: buy_stocks () is a user defined function to trade stocks based on value of variables “symbol” and “shares”.

- (40%) Main vulnerability: line 4 in buy.php. Possible to run CSRF attack because the code does not check if the request is sent from authorized users.
- (60%) Fix: To use **hidden** token as countermeasure.

There are some other minor vulnerabilities in this code. Each minor vulnerability and its fix counts only 20%.

Code snippet 3 (5 points)

Source: https://www.wordfence.com/

```
1.  <html><body>
2.  <form action="signup.php" method="POST">
3.  <p><input type = "text" size = "20" value = "" name = "user"></p>
4.  <p><input type="submit" value="Sign Up" /></p>
5.  </form>
6.  <?php
7.       If($_POST[‘user’]{
8.           file_put_contents(‘userlist.txt’, $_POST[‘user’]. "\n", FILE_APPEND|LOCK_EX);
9.      }
10.  ?>
11. <h2> All users signed up:</h2>
12. <?php>
13.    $ list = file_read_contents (‘userlist.txt’);
14.    foreach (split ("\n", $list) as $re){
15.    echo $re. "<br />";
16.    }
17. ?>
18. </body></html>
```

Note: file_put_contents () is a PHP function to write a string to a file.

File_get_content () is a PHP function to read entire file into a string.

- (40%) Main vulnerability: line 8 and line 13. Possible to run stored XSS attack, because the user input is stored in a file without being sanitized or filtered. The content of the file is later echoed back to the browser. If attacker types in malicious script, the malicious script can also run. If the student cannot identify it as "**stored**" XSS, the student can only get 20% credit here.
- (60%) Fix: Sanitize and filter the inputs.

Code snippet 4 (5 points)

Source: https://www.htbridge.com/vulnerability/

HTML form

```
1. <form action="upload_picture.php" method="post" enctype="multipart/form-data">
2. <input type="file" name="filename"/>
3. <input type="submit" name="submit" value="Upload"/>
4. </form>
```

upload_picture.php

```
1. <?php
2. $picture = "pictures/". basename($_FILES["uploadedfile"]["name"]);
3.  if(move_uploaded_file($_FILES["uploadedfile"]["tmp_name"], $picture)){
4.     echo "You have successfully uploaded you picture.";
5. }
6. else{
7.     echo "Error!";
8. }
9. ?>
```

Note: move_uploaded_file() is a PHP function to move file to a new location.

- (40%) Main vulnerability: line 3. The file uploaded is not check to see if it is a valid picture file. Attacker can upload malicious files to the server.
- (60%) Fix: Sanitize the file type before uploading it.

Problem 3 – (35 points)

Case description:

Company A is developing a web application to help people rent out their private cars in a period when they do not need to use their cars.

The owner of the car needs to open an account on the website and register their personal information (including social security number, name, home address, email, phone number), car information, car availability, the account for receiving the deposit, and the account for receiving the rent. The possible renter of the car also needs to open an account and register their personal information.

When the renter wants to rent a car, he or she can search for availability of cars, compare prices, make a reservation, and pay the deposit using a credit card to the deposit account. After the car is returned, the owner and renter need to log into the system and agree on the amount of deposit to be returned. After the agreement, the rent will be transferred to the car owner and the rest of the deposit will be returned to the renter. Company A will receive 0.1% of the rent from the car owner as expense of using the web application. Company A also gets paid by other companies, such as auto repair shops, insurance companies, and road rescue companies, who advertise their services in the web site.

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

- Identify business goals and business assets (5 points).
  - (50%) Business goals should at least include:
  - Attract many users to use the system to rent the car
  - Attract many companies to advise their services on the web site
  - (50%) Business assets should include:
  - **Credentials of the users and admin**
  - Personal information of the car owner and car renter;
  - Car reservation log
  - Bank transactions between users
  - Contact information of companies advertising their services

- Identify and prioritize business risks (5 points).

  You should use the risk matrix below to prioritize business risks.

		Probability		
	Risk matrix	**Low**	**Medium**	**High**
Consequence	**Low**	**L**	**L**	**M**
	Medium	**L**	**M**	**H**
	High	**M**	**H**	**H**

- (20%) Students know how to use the risk matrix to prioritize business risks
- (80%) Students can make several meaningful business risks as examples shown in the following table

Student should briefly explain/argue the rationale of the ranking.

Business risks (not a complete list, just examples)	**Probability**	**Consequence**	**Risk**
BR1: System unavailable	M	M	M
BR2: User identity or credential is stolen	M	H	H
BR3: Partners contact information is tampered	L	M	L
BR4: Bank transaction is tampered	H	H	H

- Make an overall misuse case diagram (10 points)
  (20%) The diagram should include typical actors, such as car owner, car renter, **internal attacker**, external hacker, and companies put advertisement.
  (20%) The diagram should include vulnerability, use case, misuse cases, security case, and corresponding links (e.g. threaten, mitigation, and exploit)
  (60%) The content of the diagram is complete and meaningful.

- Draw an attack tree for at least one of the threats in the misuse case diagram (10 points)
  (20%) Students show they understand what attack tree is.
  (80%) The content of the attack tree is complete and meaningful.

- Discuss possible legal issues to be considered in the contract if you are hired as penetration tester to test this application (5 points)
  - (20%) for each of the one listed item below. Sum up to 100%.
    – Defined scope of test in details, Range of IP address, computers, servers
    – Customer has legal authority to authorize the test
      - E.g. cloud customer authorize a pen test does not mean cloud provider has authorized the test

- Ensure copyright permits reverse engineering or code review
- Damage control, notify customer in writing about potential harm
- You want your customer to indemnify and hold you harmless for damages resulting from you doing what you say you are going to do
- Promise to use professionalism and skills commonly found in industry, not promise find all or even substantially all vulnerabilities
- Privacy issues, E.g. is access to sensitive information to be reported?
- Data ownership, Customer owns findings, Pen tester owns methods for conducting tests
- Duty to warn, E.g. What if you discover a zero-day vulnerability that may have system wide or industry wide impact?