TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS – 2019

TRIMESTER 2

CYBR 271

SECURE PROGRAMMING

Time Allowed: TWO HOURS ******** WITH SOLUTIONS **********

CLOSED BOOK

Permitted materials: Only silent non-programmable calculators or silent programmable calculators with their memories cleared are permitted in this examination.

Printed foreign to English language dictionaries are permitted.

No other material is permitted.

Instructions: Attempt ALL questions.

There are 60 marks total.

Write answers in the spaces provided in the examination booklet.

Hand in the examination booklet.

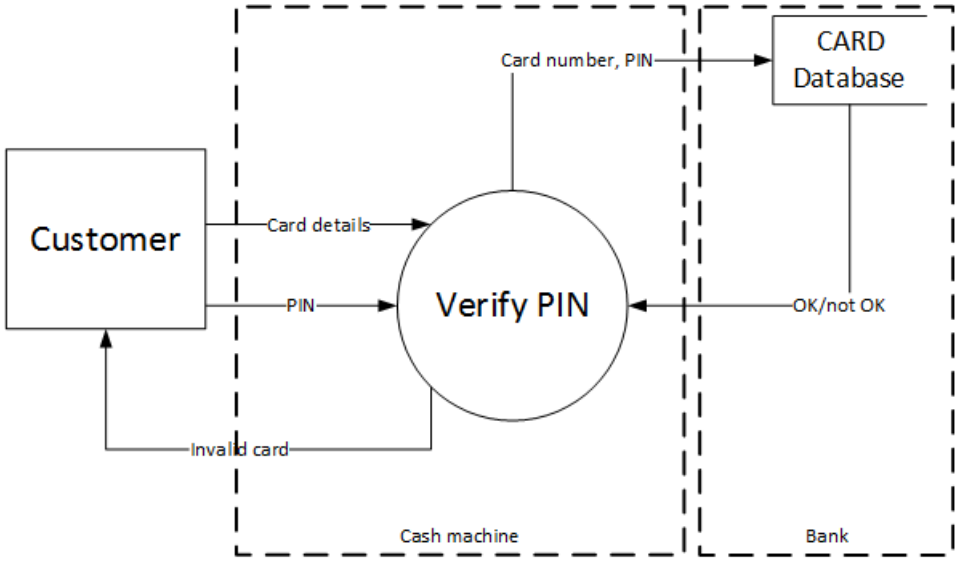
Questions	Marks
1. Threat Modelling	[8]
2. Security Principles	[8]
3. Buffer Overflow	[11]
4. Format String Vulnerability	[5]
5. SQL Injection	[4]
6. Cross-Site Scripting (XSS)	[6]
7. Cross-Site Request Forgery	[8]
8. Input Validation	[6]
9. Security Testing	[4]

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

1. Threat Modelling. **(8 marks)**

   Consider the following data flow diagram for a cash machine showing what happens when a customer presents their card to the machine before being able to withdraw cash. Apply the STRIDE methodology to develop a bottom-up threat model for the cash machine.



   i. **(2 marks)** Name an external entity, process, data store and dataflow shown in the diagram.

External entity – customer, process – verify PIN, data store – Card database, trust boundary – cash machine or bank, data flow – card details, PIN, invalid car, card number and pin, ok and not ok

   ii. **(2 marks)** Consider STRIDE. Name two threats that are applicable to a data store.

Spoofing and non-repudiation.

iii. **(2 marks)** Give an example of a threat applicable to an external entity in the diagram. Make sure that you make clear the type of threat as well as how it might be carried out.

Spoofing - attacker tries to spoof the identity of the customer by presenting the customers card. Repudiation - the attacker presents their card but later claims that they did didn't.

iv. **(2 marks)** Outline how you could *transfer* a risk associated with an information disclosure threat where an attacker observes the customer entering their PIN number?

Mitigate the risk - put the cash machine inside a cubicle only accessible using the customers card to stop people watching from behind, Eliminate it - use RFID so they customer doesn't need to enter their PIN, Transfer it - say any losses are the customer's fault, Accept it - take out insurance against potential losses.

2. Security Principles. **(8 marks)**

(a) **(3 marks)** Briefly outline the main difference between a *fail-secure* and *fail-safe* computer-controlled door lock system for a car.

fail-secure - what fails it keeps the doors locked, fail-safe - when fails it opens the door automatically. Attacker triggers a failure to open the doors, this is potentially easier than hacking the system

(b) **(2 marks)** Imagine that you have to fix a security bug for a server that you created several years ago. Briefly outline the advantages and disadvantages of maintaining *backward compatibility* with older client software.

Allows older clients to still operate without requiring an upgrade but it leaves them vulnerable to the security bug.

(c) **(3 marks)** What security principle helps protect against losses due to users failing to "do the right thing"? What is a potential problem with applying it in all cases?

Employ secure defaults, tradeoff against uability of the system.

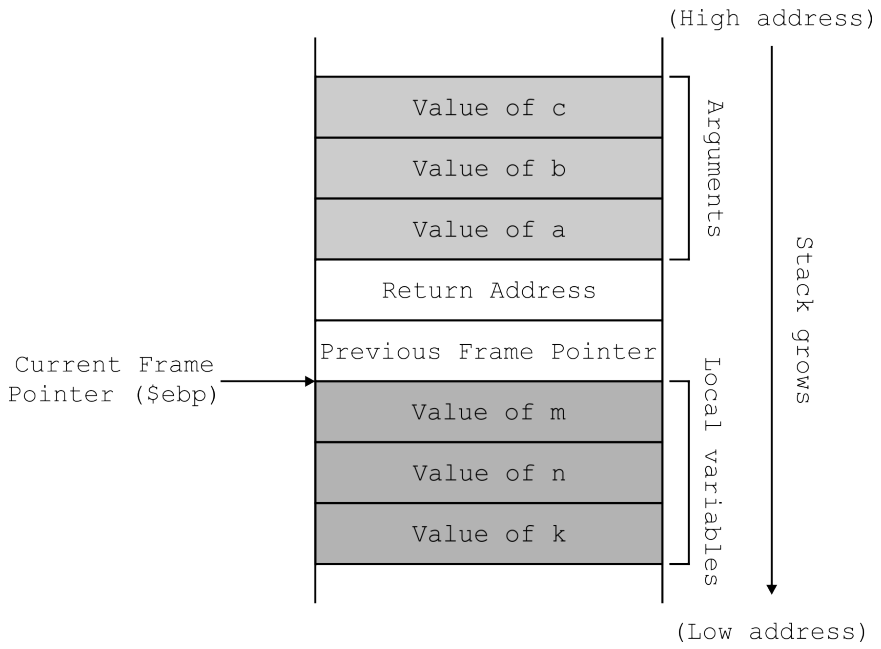
SPARE PAGE FOR EXTRA ANSWERS

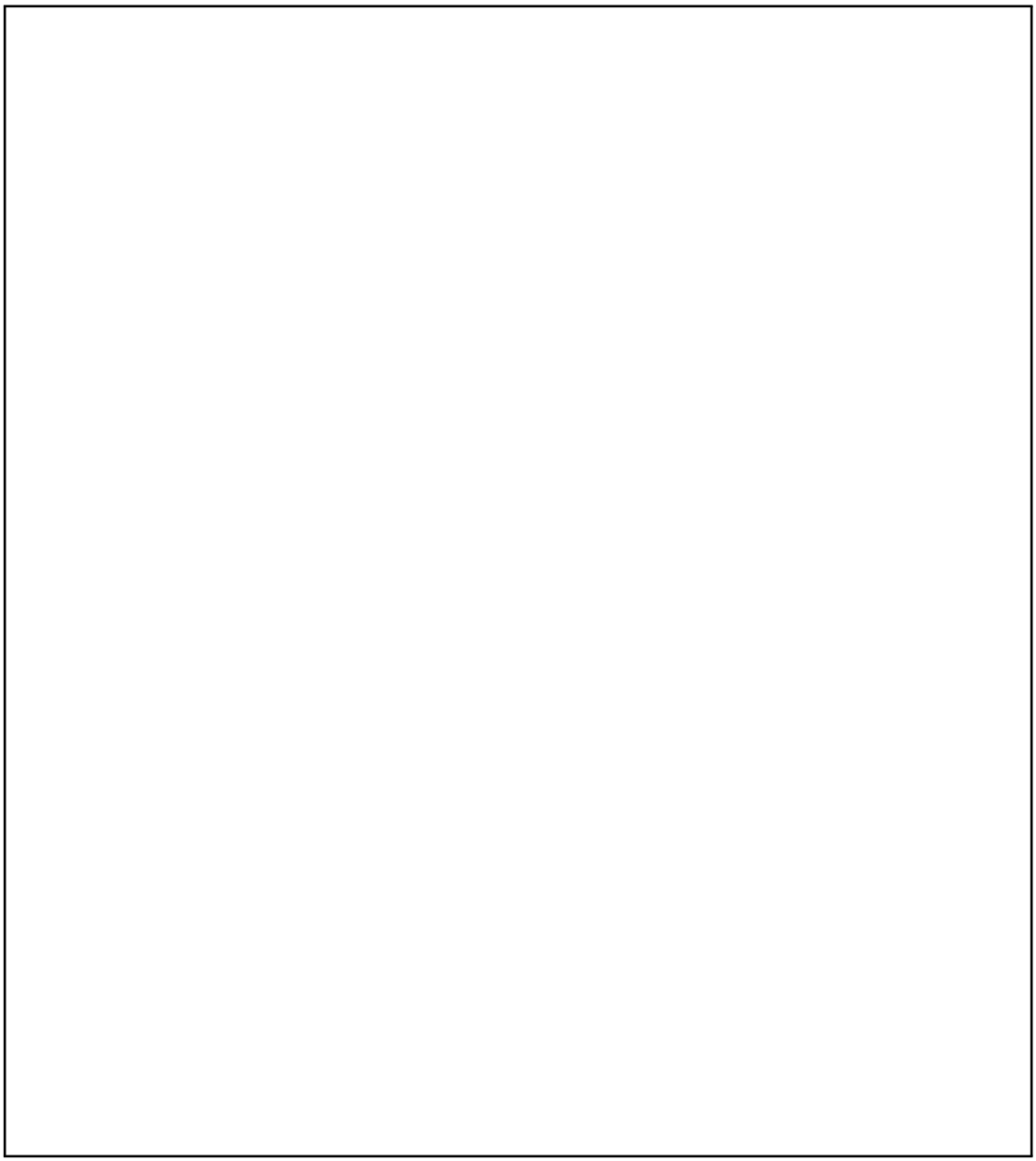
Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

3. Buffer Overflow. **(11 marks)**

(a) **(6 marks)** **Draw** the stack frame for the following function.

```
int func(int a, int b, int c)
{
   int m, n, k;
   m = a - c;
   k = m + b;
   return k;
}
```





(b) **(5 marks)** Explain what is the MAIN cause of the following buffer attack failing.

The victim program reads from a file and has created a `badfile` with the aim of getting a shell via a buffer overflow attack. The target of the attack is the `strcpy()` function that copies the user input into a buffer.

After checking the memory layout for the compiled code using the debugging mode (`gdb`), she prepared a `badfile` and injected `0xbffff200` as the new return address (the highlighted 4 bytes in the snippet below).

Note that address randomisation has been turned off, the victim program is compiled with executable stack protection turned off as well as made the stack executable and disabled all stack countermeasures.

```
1 00000000  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90  |................|
2 *
3 00000020  90 90 90 90 00 f2 ff bf  90 90 90 90 90 90 90 90  |................|
4 00000030  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90  |................|
5 *
6 000001e0  90 90 90 90 90 90 90 90  90 90 90 90 31 c0 50 68  |............1.Ph|
7 000001f0  2f 2f 73 68 68 2f 62 69  6e 89 e3 50 53 89 e1 99  |//shh/bin..PS...|
8 00000200  b0 0b cd 80 00                                    |.....|
```

strcp() does not stop untill a
0 is found. As the picked address has 00, this
will terminate the copying operation and all the
subsequent parts of the badfile (including the
shellcode) will not be copied. Hence, the program
will try to execute something other than the NOP
instructions (as they have not been copied into the
stack) which causes the program to crash.
Fixing would require choosing an address that is one
greater than the provided one.

4. Format String Vulnerability. **(5 marks)**

(a) **(2 marks)** Construct a string that will print out a secret value that is stored at a specific location on the stack. Assume that a value is stored 28 bytes above the `va_list` pointer on a 32-bit machine (each memory block is 4 bytes).

```
User input:  %d%d%d%d%d%d%d%d
The %x formatting token can also be used to answer
this question.
1 mark for %d, 1 mark for right number.
```

(b) **(3 marks)** Explain why an attacker implementing a code injection attack constructs the attack in terms of two separate operations. First overwriting the return address and secondly writing the malicious code to memory.

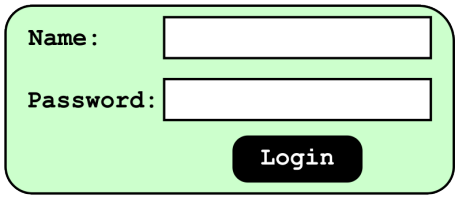
```
This is to avoid a buffer overflow countermeasure
(1 mark) where a canary value is placed below the
return address (1 mark).  Writing these separately
avoids writing over the canary value (1 mark).
```

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

5. SQL Injection. **(4 marks)**

(a) **(2 marks)** A web application uses the login form presented in the following figure to allow access only for those who have a record in the application's database. The system does not sanitize the input before forming and submitting the query to the database.




```
<?php
   $sql = "SELECT *
           FROM users
           WHERE uid='$input_name' AND password='$input_password'";
   $result = $conn->query($sql);
?>
```

Construct an input that will let you to access the system even if you are not registered.

```
Name:

Password:
ANSWER 1:
--------
Name:  anything
Password:  '  OR 1=1; --
ANSWER 2:
--------
Name:  '  OR 1=1; --
Password:  can be left blank
```

(b) **(2 marks)** Briefly explain the fundamental cause of SQL-injection attacks, and list three approaches to mitigate/prevent this type of attack.

```
Mixing data and code together.
1.  Filtering data,
2.  Encoding data, and
3.  Prepared statements.
```

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

6. Cross-Site Scripting (XSS). **(6 marks)**

(a) **(4 marks)** Briefly explain the main difference between *reflected* and *stored* XSS attacks and comment upon which can have a wider impact.

```
Reflected XSS:
--------------
1.  Non-persistent data, and
2.  only affects the targeted server/victim (spared
linearly).
Stored XSS:
--------------
1.  Persistent data, and
2.  effects everyone visited the page where the
script is stored (can spared rapidly if converted
into a worm).
```

(b) **(2 marks)** Explain why a XSS defence based upon filtering out `script` tags from user-provided data is easily evaded and give an example of how to do this.

The filtering approach is very difficult to completely mitigate all XSS attacks as there are various methods to accomplish such attacks. Some examples:
1. Filtering the <script> tags can be defeated by injecting the code into other elements such as .
2. Filtering the keyword 'javascript' can be defeated by breaking the word into two lines.
3. Filtering the word 'onreadystatechange' can be defeated by using the 'eval()' function that converts strings into commands.

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

7. Cross-Site Request Forgery (CSRF). **(8 marks)**

(a) **(6 marks)** List THREE main differences between CSRF and XSS attacks.

```
CSRF
-----
1.  Requires two websites.
2.  Less powerful.
3.  Limited to actions that can be performed on the
targeted web application.
4.  Easier to mitigate/prevent.  XSS
------
1.  Performed from the same page.
2.  More effective.
3.  Not limited by the targeted web application
(larger in scope).
4.  Harder to mitigate/prevent.
```

(b) **(2 marks)** Explain why the use of HTTPS does not protect against CSRF attacks?

Because CSRF aims at forging the request that will be sent just like other requests. Encrypting the communications can protect the privacy and integrity of the data, but not preventing forged requests from being sent from the victim's browser.

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

8. Input validation. **(6 marks)**

(a) **(2 marks)** Explain why *whitelisting* is usually preferred to *blacklisting*?

Whilelisting, if miss something the worst is that you disallow it and this is preferable to a security failure.

(b) **(4 marks)** Consider a web application that allows the uploading of images. The program performs input validation on the file name to check that it has an image file extension such as GIF, PNG or JPG.

Briefly outline TWO vulnerabilities related to canonical file name issues that might apply in this case.

First is path traversal allowing the attacker to overwrite a file outside of the intended directory on the file server and second is OS specific naming where null bytes injected to terminate the filename allowing a file with a different extension to be uploaded.

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

9. Security testing. **(4 marks)**

Briefly discuss ONE advantage and ONE disadvantage that *input fuzzing* has over manual data mutation.

Advantages: input fuzzing doesn't reflect to biases of the programmer, doesn't require access to source code. Disadvantage: random nature means that it might not find a particular hard to activate vulnerability.

****