ECM2426

(with Answers)

UNIVERSITY OF EXETER

COLLEGE OF ENGINEERING, MATHEMATICS AND PHYSICAL SCIENCES

COMPUTER SCIENCE

Examination, January 2020

Network and Computer Security

Module Leader: Prof Achim Brucker

Duration: TWO HOURS

Answer ALL questions.

Question 1 and Question 2 are worth 20 marks each. Question 3 and Question 4 are worth 30 marks each.

Write clearly in the sense of logic, language, and readability. The clarity of your arguments and explanations affects your grade.

The marks for this module are calculated from 70% of the percentage mark for this paper plus 30% of the percentage mark for associated coursework.

Candidates are permitted to use *non-programmable* portable electronic calculators in this examination.

This is a CLOSED BOOK examination.

Question 1

Security Fundamentals & Access Control

(a) Many modern websites allow users to log-in using external services such as Google, Facebook, or Github. This mechanism is called *single sign-on*.

Briefly explain which part of access control (identification, authentication, authorisation) is carried out by the external service.

(2 marks)

The external service provider provides, at each log-in event to the website, *authentication* to the website.

Authorisation is still done by the website itself. During the initial registration of the user at the site providing single sign-on service, the services should perform some form of identification.

Marking Scheme:

- Full marks **[2]** for *authentication*.
- Half marks *[1]* if only *identification* is mentioned.

(b) Some services (e. g., banks) use two factors for authenticating users:

(i) a log-in using username & password *and*

(ii) one-time passwords sent to the customer' s mobile phone.

Briefly explain if this is (or is not) a strong authentication mechanism for users using a mobile browser, which runs on the same phone on which they receive the one-time password.

(3 marks)

If we assume that the mobile phone of the user is compromised, an attacker can, e.g., take over the session in the mobile browser and also eavesdrop the one-time password. Hence, this is not a strong two-factor authentication for users on mobile phones.

Marking Scheme:

- **[1]** for stating that the system is not a strong authentication for users on mobile phones.

- **[2]** for a correct explanation or a correct example of an attack.

(c) Consider the following access control matrix:

	File$_1$	File$_2$	File$_3$
Alice	read, write, execute	read, write, execute	read, write, execute
Bob	read	read	read, write
Charlie	read	read	-
Dave	read, execute	read, execute	execute
Eve	read	read	-
Fiona	read, execute	read, execute	execute
Gail	read	read	read, write

Convert this access control matrix into an equivalent hierarchical role-based access control model (RBAC) by applying the following steps:

(i) Give a formal definition of the users of the equivalent hierarchical RBAC model.

(2 marks)

$$\mathit{USERS} = \{\text{alice}, \text{bob}, \text{charlie}, \text{dave}, \text{eve}, \text{fiona}, \text{gail}\}$$

Marking Scheme:

- **[2]** for the correct definition.
- at most *[1]* for an informal definition or a formal definition that does not include all users.

(ii) Give a formal definition of the set of permissions of the equivalent hierarchical RBAC model.

(2 marks)

$$\begin{aligned}\mathit{PERMISSIONS} = \{ & \\ & \text{read\_file}_1, \text{write\_file}_1, \text{execute\_file}_1, \\ & \text{read\_file}_2, \text{write\_file}_2, \text{execute\_file}_2, \\ & \text{read\_file}_3, \text{write\_file}_3, \text{execute\_file}_3, \\ \} & \end{aligned}$$

Marking Scheme:

- **[2]** for the correct definition.
- at most *[1]* for an informal definition or a formal definition that does not include all permissions.

(iii) Give a formal definition of user roles and the mapping from users to these roles.

(3 marks)

$$
\begin{aligned}
ROLES &= \{r_0, r_1, r_2, r_3\} \\
UA &= \{ \\
&\qquad (\text{charlie}, r_0), (\text{eve}, r_0) \\
&\qquad (\text{gail}, r_1), \quad (\text{bob}, r_1) \\
&\qquad (\text{dave}, r_2), (\text{fiona}, r_2), \\
&\qquad (\text{alice}, r_3) \\
&\quad \}
\end{aligned}
$$

- Charlie and Eve have the least privileges, they can only read file 1 and 2. We assign them role r_0. Note that all subjects can read file 1 and file 2.
- In addition to the privileges of r_0, Gail and Bob can read and write file 3. We assign these privileges to role r_1.
- In addition to the privileges of r_0, Dave and Fiona can execute all files. We assign these privileges to role r_2.
- Alice can read, write, and execute all files. We assign these privileges to r_3.

Marking Scheme:

- **[3]** for a formal and correct definition that defines a non-trivial role hierarchy.
- at most *[2]* for an informal definition or a partially wrong formal definition.
- at most *[1]* for the trivial solution, assigning a new role to each subject.

(iv) Give a formal definition of the role hierarchy.

(3 marks)

$$
\begin{aligned}
RH = \{ & \\
& (r_0, r_0) \\
& (r_1, r_1), (r_1, r_0) \\
& (r_2, r_2), (r_2, r_0) \\
& (r_3, r_3), (r_3, r_1), (r_3, r_2) \\
\} &
\end{aligned}
$$

We have

- r_o has the least privileges
- r_1 has all privileges that r_0 has ($r_0 < r_1$)
- r_2 has all privileges that r_0 has ($r_0 < r_2$)
- r_1 and r_2 are incomparable
- r_3 has all privileges that r_1 has ($r_1 < r_3$) and r_3 has all privileges that r_2 has ($r_1 < r_2$).

Marking Scheme:

- **[3]** for a correct (with respect to the student provided solution of the previous tasks) formal definition.
- at most *[2]* for a correct informal solution or an incorrect formal solution.

(v) Give a formal definition of the mapping from roles to permissions.

(2 marks)

$$
\begin{aligned}
UA = \{ & \\
& (r_o, \text{read\_file}_1), (r_o, \text{read\_file}_2) \\
& (r_1, \text{read\_file}_3), (r_1, \text{write\_file}_3) \\
& (r_2, \text{execute\_file}_1), (r_2, \text{execute\_file}_2), (r_2, \text{execute\_file}_3), \\
& (r_3, \text{write\_file}_1), (r_3, \text{write\_file}_2) \\
\} &
\end{aligned}
$$

Marking Scheme:

- **[2]** for a correct formal definition (with respect to the student provided solution of the previous tasks) formal definition.
- at most *[1]* for a correct informal solution or an incorrect formal solution.

(vi) Compute the permissions of Bob using your hierarchical RBAC model and check that they are equivalent to the permissions of Bob defined in the access control matrix.

(3 marks)

$$\begin{aligned}(PA \circ RH \circ UA)(\text{bob}) &= (PA \circ RH)\{r_1\}\\ &= PA\{r_0, r_1\}\\ &= \{\\ &\qquad \text{read\_file}_1, \text{read\_file}_2,\\ &\qquad \text{read\_file}_3, \text{write\_file}_3\\ &\quad \}\end{aligned}$$

The set of permissions is equivalent to the permissions given in the access control matrix.

Marking Scheme:

- **[2]** for a correct calculation of the permissions with respect to the defined access control model.
- **[1]** for stating that they are equivalent.

(Total 20 marks)

Question 2

PKIs & Cryptography

(a) In the context of Public Key Infrastructures (PKIs) for securing web sites, discuss briefly why many advocate certificates with a short (only a few weeks/months) validity.

(3 marks)

Short-lived certificates are an attempt to get rid of the need for (large) certificate revocation lists (CRLs). As a CRL needs to be checked for each access of a web sites, large CRLs slow down Web Browsers significantly.

Marking Scheme:

- full marks (**[3]**) an explanation that explains that short-lived certificates simplify the handling of CRLs.
- *[2]* for an answer that only motivates short-lived certificates by reducing the risk of losing the private key of a certificate that does not mention CRLs.

(b) Explain briefly why using the *one-time pad* (also called Vernam cipher) to encrypt several messages with the same key is either:

- secure, giving an explanation as to why it is so;
- insecure, giving an explanation as to why it is so.

(3 marks)

Using the same key twice is *not secure*: the simplest attack possible is a known plain text attack. Assume an attacker knows both the message m and the cipher text $c = m \oplus k$ (encrypted with the key k). Now she can easily compute $m \oplus c = m \oplus (m \oplus k) = (m \oplus m) \oplus k = k$. Thus, if the key is re-used, an attacker can now easily decrypt further messages. A more complex attack is a crib dragging (but students are not expected to be able to explain crib dragging) attack.

Marking Scheme:

- **[1]** for the answer "not secure"

- **[2]** for either an example demonstrating an attack or a correct algebraic argument. At most *[1]* for an informal explanation.

(c) Can symmetric cryptography be used for implementing digital signatures? Briefly explain your answer.

(4 marks)

No, signatures require a secret used for creating a signature and a public part for validating it. This is not possible with a symmetric encryption scheme, as the key for creating signatures and the key for validating signatures are the same. Hence, anybody that can validate a signature could also generate signatures. Thus, a signature cannot be linked to one subject.

Marking Scheme:

- **[1]** for the correct answer and
- **[3]** for a correct explanation.

(d) Consider the RSA crypto scheme with the following configuration:

- Alice's public key is $(n_a, e_a) = (33, 7)$, her private key is $d_a = 3$.
- Bob's public key is $(n_b, e_b) = (65, 7)$, his private key is $d_b = 7$.
- Eve's public key is $(n_e, e_e) = (77, 13)$, her private key is $d_e = 37$.

All key pairs are used for both signing and encryption/decryption.

Alice sends to Bob the *cipher text*:

$$9, 59$$

(i) Briefly explain why there is an attack that allows Eve to obtain the plain text of this cipher text.

(5 marks)

(ii) Demonstrate the attack by showing how Eve can obtain the plain text of the message that Alice sends to Bob.

(5 marks)

- Eve needs to convince (e.g., by social engineering) to convince Bob to sign the encrypted message that Alice did send to Bob. We assume here that Eve was able to eavesdrop this message. The signature of this message is identical to the plain text if Bob uses the same keypair for en/decrypting and signing.

Marking Scheme:

- **[1]** for explaining that Eve needs to eavesdrop the message
- **[2]** for explaining that Eve needs to trick Bob into signing the cipher text.
- **[2]** for explanation that the signature is the plain text.

- We assume that Bob is willing to sign all messages that we show him. First, Eve asks Bob to sign the message

$$9, 59$$

- By signing this message, Bob computes the signature by computing $m = c^7 \mod 65$ for each character c of the cipher text. Bob obtains the signature

$$9, 19$$

Marking Scheme:

- **[3]** for choosing the right key (and giving a reason why it is the right one)
- **[2]** for the computation

(Total 20 marks)

Question 3

Security Protocols

(a) How useful would the Dolev-Yao attacker model be for finding *denial of service* attacks?

(4 marks)

Denial of service attacks can, fundamentally, not be discussed within the Dolev-Yao attacker model. The Dolev-Yao model has no notion of service quality, processing time of messages, etc.

Marking Scheme:

- **[1]** for the correct answer that denial of service cannot be analysed
- **[3]** for a correct justification

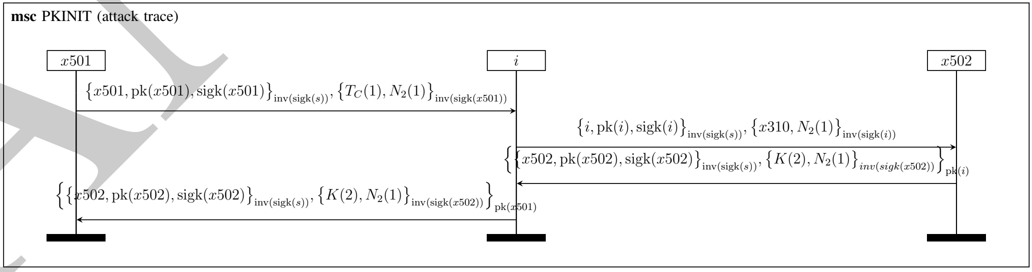
(b) Consider the following protocol with the two roles C and KAS and the two public functions $\mathrm{pk}(\_)$ and $\mathrm{sigk}(\_)$:

Step 1: $C \longrightarrow KAS$:
$\{C, \mathrm{pk}(C), \mathrm{sigk}(C)\}_{\mathrm{inv}(\mathrm{sigk}(s))}, \{T_C, N_2\}_{\mathrm{inv}(\mathrm{sigk}(C))}$
Step 2: $KAS \longrightarrow C$:
$\big\{\{KAS, \mathrm{pk}(KAS), \mathrm{sigk}(KAS)\}_{\mathrm{inv}(\mathrm{sigk}(s))}, \{K, N_2\}_{\mathrm{inv}(\mathrm{sigk}(KAS))}\big\}_{\mathrm{pk}(C)}$

State whether the protocol transmits the key K securely (i.e., secretly and authentically) from KAS to C. If it is secure, explain why. If it is not secure, explain the possible attack and briefly explain a possible fix.

(6 marks)

The protocol is not secure. The attack similar to the attack against NSPK:


msc PKINIT (attack trace)
$x501$
i
$x502$
$\{x501, \mathrm{pk}(x501), \mathrm{sigk}(x501)\}_{\mathrm{inv}(\mathrm{sigk}(s))}, \{T_C(1), N_2(1)\}_{\mathrm{inv}(\mathrm{sigk}(x501))}$
$\{i, \mathrm{pk}(i), \mathrm{sigk}(i)\}_{\mathrm{inv}(\mathrm{sigk}(s))}, \{x310, N_2(1)\}_{\mathrm{inv}(\mathrm{sigk}(i))}$
$\big\{\{x502, \mathrm{pk}(x502), \mathrm{sigk}(x502)\}_{\mathrm{inv}(\mathrm{sigk}(s))}, \{K(2), N_2(1)\}_{inv(sigk(x502))}\big\}_{\mathrm{pk}(i)}$
$\big\{\{x502, \mathrm{pk}(x502), \mathrm{sigk}(x502)\}_{\mathrm{inv}(\mathrm{sigk}(s))}, \{K(2), N_2(1)\}_{\mathrm{inv}(\mathrm{sigk}(x502))}\big\}_{\mathrm{pk}(x501)}$

The fix is very similar to the one proposed by Lowe for the Needham–Schroeder protocol. KAS must simply include the agent

identity of C in his response:
$KAS \longrightarrow C$:
$\big\{\{KAS, \mathrm{pk}(KAS), \mathrm{sigk}(KAS)\}_{\mathrm{inv}(\mathrm{sigk}(s))}, \{K, N_2, \mathbf{C}\}_{\mathrm{inv}(\mathrm{sigk}(KAS))}\big\}_{\mathrm{pk}(C)}$
Now, an attacker cannot simple forward the encrypted message.

Marking Scheme:

- **[1]** for stating that the protocol is insecure
- **[3]** for describing the attack
- **[2]** for fix and justification why the attack is prevented

(c) In the Dolev-Yao attacker model, consider the following intruder knowledge:

$$M = \Big\{ \{\!|\, \mathrm{inv}(\mathrm{pk}(b)) \,|\!\}_{g(n_2)},\ \ \{\{n_1\}_{\mathrm{pk}(i)}, \{n_2\}_{\mathrm{inv}(\mathrm{pk}(a))}\}_{\mathrm{inv}(\mathrm{pk}(b))},$$
$$\mathrm{pk}(b),\ \ \mathrm{pk}(a),\ \ \mathrm{pk}(i),\ \ \mathrm{inv}(\mathrm{pk}(i)),\ \ \{\mathit{secret}\}_{\mathrm{pk}(a)},\ \ \{\mathit{secret}\}_{\mathrm{pk}(b)} \Big\}$$

where g is a public function (i.e., $g \in \Sigma_P$).
Using natural deduction, prove formally that the intruder can learn the message "*secret*".

(10 marks)

$$
\dfrac{
  \dfrac{}{\{\mathit{secret}\}_{\mathrm{pk}(b)} \in \mathcal{DY}(M)}\,\mathrm{Axiom}
  \qquad
  \dfrac{
    \dfrac{}{\{\!|\, \mathrm{inv}(\mathrm{pk}(b)) |\!\}_{g(n_2)} \in \mathcal{DY}(M)}\,\mathrm{Axiom}
    \qquad
    \dfrac{
      \dfrac{
        \dfrac{
          \dfrac{
            \dfrac{}{\{\{n_1\}_{\mathrm{pk}(i)}, \{n_2\}_{\mathrm{inv}(\mathrm{pk}(a))}\}_{\mathrm{inv}(\mathrm{pk}(b))}}\,\mathit{Axiom}
          }{\langle \{n_1\}_{\mathrm{pk}(i)}, \{n_2\}_{\mathrm{inv}(\mathrm{pk}(a))} \rangle}\,\mathit{OpenSig}
        }{\{n_2\}_{\mathrm{inv}(\mathrm{pk}(a))} \in \mathcal{DY}(M)}\,\mathrm{Proj2}
      }{n_2 \in \mathcal{DY}(M)}\,\mathrm{OpenSig}
    }{g(n_2) \in \mathcal{DY}(M)}\,\mathrm{Comp.}\ (g \in \Sigma_P)
  }{\mathrm{inv}(\mathrm{pk}(b)) \in \mathcal{DY}(M)}\,\mathrm{DecSym}
}{\mathit{secret} \in \mathcal{DY}(M)}\,\mathrm{DecAsym}
$$

The proof can also by given using the notation used in the book by Huth and Ryan:

1	$\{secret\}_{\mathrm{pk}(b)}$	premise
2	$\{\!\mid \mathrm{inv}(\mathrm{pk}(b)) \mid\!\}_{g(n_2)}$	premise
3	$\{\{n_1\}_{\mathrm{pk}(i)}, \{n_2\}_{\mathrm{inv}(\mathrm{pk}(a))}\}_{\mathrm{inv}(\mathrm{pk}(b))}$	premise
4	$\langle \{n_1\}_{\mathrm{pk}(i)}, \{n_2\}_{\mathrm{inv}(\mathrm{pk}(a))} \rangle$	OpenSig 3
5	$\{n_2\}_{\mathrm{inv}(\mathrm{pk}(a))}$	Proj2 4
6	$n_2 \in \mathcal{DY}(M)$	OpenSig 5
7	$g(n_2) \in \mathcal{DY}(M)$	Composition ($g \in \Sigma_P$) 6
8	$\mathrm{inv}(\mathrm{pk}(b)) \in \mathcal{DY}(M)$	DecSym 2,7
9	$secret \in \mathcal{DY}(M)$	DecAsym 1,8

Marking Scheme:

- **[10]** for a correct formal proof
- *[4]* deduction for an informal proof.
- wrong but formal proof attempts that partially use the correct reasoning (i.e., correct sub-proofs) can get up to *[8]*.
- no deduction for small mistakes (not all rules named, obvious typos, etc.)

(d) Consider the following skeleton of a key exchange protocol using a trusted server S.

Step 1: $B \longrightarrow A: \; B, A, \{\!\mid B, A, \exp(g, Y) \mid\!\}_{\mathrm{sk}(B,S)}$
Step 2: $A \longrightarrow S: \; \{\!\mid A, B, \exp(g, X) \mid\!\}_{\mathrm{sk}(A,S)}, \{\!\mid B, A, \exp(g, Y) \mid\!\}_{\mathrm{sk}(B,S)}$
Step 3: $S \longrightarrow A: \; \ldots$
Step 4: $A \longrightarrow B: \; \ldots$

(i) Complete the protocol, i.e., what messages need to be sent in *Step 3* and *Step 4* to ensure that A and B can use the Diffie-Hellman half-keys to compute a shared full-key.

(5 marks)

Step 3: $S \longrightarrow A: \; \{\!\mid A, B, \exp(g, X), \exp(g, Y) \mid\!\}_{\mathrm{sk}(A,S)},$
$\{\!\mid B, A, \exp(g, Y), \exp(g, X) \mid\!\}_{\mathrm{sk}(B,S)}$
Step 4: $A \longrightarrow B: \; \{\!\mid B, A, \exp(g, Y), \exp(g, X) \mid\!\}_{\mathrm{sk}(B,S)}$

Marking Scheme:

- **[3]** for Step 3 and **[2]** for Step 4.
- At most *[2]* in total for a solution that is "in principle" correct.

(ii) In this protocol, the protocol designers decided to use Diffie-Hellman half-keys. Name the property established by this and explain briefly against which attack the Diffie-Hellman half-key exchange protects the users of this protocol.

(5 marks)

In this protocol, the server only learns the Diffie-Hellman half-keys generated by the agents, i.e., $\exp(g, X)$ and $\exp(g, Y)$. From this information, the server *cannot* compute the full-key $\exp(\exp(g, X), Y)$ (respectively, $\exp(\exp(g, Y), X)$). Hence, the attacker *cannot* decrypt messages send with these session keys. This is called *perfect forward secrecy*.

Marking Scheme:

- **[1]** for naming perfect forward secrecy
- **[4]** for the correct explanation

(Total 30 marks)

Question 4

Application Security

(a) An application is using the following approach for storing passwords:

- For each user, a unique and long random salt is generated.
- The salt is appended to the plaintext password, the resulting string is hashed using a secure hashing algorithm.
- Both the salt and the hash are stored in a database.

Given this scenario, answer the following questions:

(i) Explain briefly why the salt should be unique for each user.

(2 marks)

If the same salt is used for all users, two users choosing the same password will have the same hash. Moreover, the attacker can still use a (rather) efficient dictionary attack, using a application specific dictionary that she generated using a word list and the fixed hash.

Marking Scheme:

- **[i]**f dictionary attack or rainbow tables are mentioned and a hint is given, why this can be done efficiently.

(ii) Assume that the currently used hashing algorithm is no longer considered to be (highly) secure. Therefore, you want to switch to a different hashing algorithm. Briefly explain an approach for securely migrating the users of your application to the new hashing algorithm.

(4 marks)

One secure approach is to extend the database with an additional column storing the hashing algorithm (or the information if a user has already been migrated). Whenever a user logs in, we check if the user has already been migrated. If yes, check the hash using the new hashing algorithm. If not, we check the hash using a the old hashing algorithm and ask the user to change the password. Now we can generate a new salt and hash (including the new salt) the password using the new hashing algorithm. Finally, we store salt, hash, and the information that the user has been migrated in the database.
After a grace period, we might want to disable user that have not logged in (i.e., accounts still using the old hashing algorithm).

Marking Scheme:

- **[4]** for a correct explanation.
- At most *[3]* if the explanation is correct in principle but contains some errors (e.g., no new salt is generated).

(b) A web application is using the following function for protecting itself against Cross-site Scripting (XSS):

JavaScript

```
function sanitiseXSS(str){
   var re = new RegExp("^([a-z0-9]*)$");
   var retval = ""
   if (re.test(str)) {
     retval = str;
   }
   return retval;
 }
```

(i) Explain briefly if this sanitisation function will successfully protect an application against client-side XSS or not.

(3 marks)

The sanitisation function is secure, as only a sequence of the lowercase letter a–z and the digits 0–9 is accepted, all other input is rejected.

Marking Scheme:

- **[1]** for stating that it is secure.
- **[2]** for the correct explanation why it is secure.

(ii) Does this sanitisation function implement black listing or white listing? Why?

(3 marks)

White listing, as it specifies the allowed characters.

Marking Scheme:

- **[1]** for stating that it applies white listing.
- **[2]** for the explanation why it is white listing and not black listing.

(c) Consider the following Python program

Python

```
# dbuser and dbpwd are obtained from a
# configuration file
con = pymysql.connect(’localhost’, dbuser,
                  dbpwd, ’database’)
# get parameters from web request
uid = getRequestParameter("uid")
with con:
  cur = con.cursor()
  cur.execute("SELECT * FROM users WHERE user_id="
          + uid)
  firstname, lastname = cur.fetchone()
  print("Welcome " + firstname + " " + lastname)
```

(i) What vulnerability does this program have? Briefly explain the vulnerability in general and, in particular, why this program is vulnerable.

(4 marks)

The program is vulnerable to *SQL Injection*. While the program uses a prepared statement (which is usually recommended), the prepared statement is used wrongly, i.e., the parameters are still passed to the SQL statement by unchecked string concatenation. Thus, an attacker can pass SQL commands as parameters (e.g., `month`) that are then executed by the database (and, e.g., return additional data or modify data).

Marking Scheme:

- **[1]** for the correct name (SQL Injection).
- **[1]** for explaining what a SQL injection is.
- **[2]** for a correct explanation mentioning a wrong use of prepared statement.
- At most *[2]* if the explanation does not mention the concept of prepared statements.

(ii) Modify this program to fix the vulnerability.

(3 marks)

The prepared statement needs to be rewritten to make use of placeholder parameters (`%s`):

```
    cur.execute("SELECT * FROM users WHERE
              user_id= %s", uid)
```

Marking Scheme:

- **[3]** if the syntax is not correct but the fix attempted shows the right idea, i.e., a mostly correct specified prepared statement.
- at most *[2]* if a correct white-listing is used.
- at most *[1]* if a correct black-listing is used.

(iii) Briefly explain why your modified version is secure.

(3 marks)

The *correct* use of a prepared statement provides a strict separation of code (SQL) and data (parameters of the SQL query). This prevents any attack that tries to inject SQL code into a part of a SQL query where data is expected.

Marking Scheme:

- **[3]** for a correct explanation (with respect to the solution provided by the students, i.e., if a secure white listing approach has been used, a correct explanation gives full marks).
- At most *[2]* for mostly correct explanation.

(d) Consider the following vulnerability in SearchBlox, an enterprise search and data analytics service (CVE-2015-0970):

> A cross-site request forgery (CSRF) vulnerability in SearchBlox Server before version 8.2 allows remote attackers to perform actions with the permissions of a victim user, provided the victim user has an active session and is induced to trigger the malicious request.

What CVSS v3 Base Vector would you assign to this vulnerability? Provide a brief justification of your assessment.
Hint: Recall the CVSS v3 Base Vector components:

(AV:[N,A,L,P]/AC:[L,H]/PR:[N,L,H]/UI:[N,R]/
S:[U,C]/C:[H,L,N]/I:[H,L,N]/A:[H,L,N])

(8 marks)

The CVSS v3 Base Vector is

Attack Vector	Network	A victim must access a vulnerable system via the network.
Attack Complexity	Low	A phishing email does not absolutely require victim reconnaissance.
Privileges Required	None	The attacker does not need any permissions to perform this attack, the attacker lets the victim perform the action on the attacker's behalf.
User Interaction	Required	The victim must click a specially crafted link provided by the attacker.
Scope	Unchanged	The vulnerable component is SearchBlox. The impacted component is also SearchBlox as the actions only affect the SearchBlox configuration.
Confidentiality Impact	High	The attacker can obtain permissions to view all confidential data contained in SearchBlox.
Integrity Impact	High	User accounts can be modified at will as well as SearchBlox configuration.
Availability Impact	High	SearchBlox configuration may be modified such as to disable services.

Marking Scheme:

- In general, *[1]* for each component (**[8]** in total) of the CVSS Base Vector. If the assessment and the explanation is consistent for one component, but deviates from the correct solution, still full marks are awarded.

(Total 30 marks)