Exam June 2018, questions

Computer Security & Forensics (University of Sheffield)

MODEL SOLUTIONS

SETTER: Achim D. Brucker

Data Provided:
None

DEPARTMENT OF COMPUTER SCIENCE Spring Semester 2017-2018

COMPUTER SECURITY AND FORENSICS 1.5 hours

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

All questions carry equal weight. Figures in square brackets indicate the percentage of available marks allocated to each part of a question.

1. **Security Fundamentals & Access Control**

a) Briefly explain the concepts *identification*, *authentication*, and *authorisation*. [20%]

ANSWER:

Identification: Associating an identity with a subject.

Authentication: Verifying the validity of something (usually the identity claimed by a system entity.

Authorisation: Granting (or denying) the right or permission of a system entity to access a object.

Marking Scheme:

- [5%] for each correct explanation (plus [5%] for defining all three)

b) For each of the following multi-factor authentication systems, explain briefly if it is a good or bad multi-factor authentication system:

(i) A *fingerprint* and *voice recognition*. [5%]

(ii) A *GPS signal* and a *password*. [5%]

ANSWER:

- Bad, as both factors are from the "something you are" category.
- Good, as the GPS signal is a context location (generated by "something you have") and the password is "something you know".

Marking Scheme:
[5%] for each correct explanation.

c) Instead of using a login based on username and password, many modern websites allow users to log-in using services such as Google, Facebook, or Github. This mechanism is called *single sign-on*.
Briefly explain one advantage and one disadvantage of a single sign-on solution. [10%]

ANSWER:

Advantages are:

- User do not need to learn/recall a new username/password for each website they use.
- Operators of websites do not need to maintain username/password databases that could be the target of an attack.

Disadvantages are:

- The single sign-on provider learns which other services its users are using.
- If the single sign-on provider is compromised, all websites relying on the single sign-on provider are endangered as well.

Marking Scheme:
[5%] for each advantage/disadvantage

d) Recall the Access Control Matrix Model discussed in the lecture and consider the following configuration:

- The set of subjects S is defined as follows: $S = \{\text{Alice}, \text{Bob}, \text{Charlie}, \text{Eve}\}$
- The set of objects O is defined as follows: $O = \{\text{File1}, \text{File2}, \text{File3}, \text{File4}\}$
- The set of actions A is defined as follows: $A = \{\text{read}, \text{write}, \text{execute}, \text{append}\}$
- The informal access control policy is defined as follows:
  - *Alice* is allowed to *read* all files and she is allowed to *append* to File3.
  - *Bob* can *read* and *write* File4 and *append* to File2.
  - *Charlie* is allowed to *write* to the files File1 and File2 and he is allowed to *read* File1.

- File2 can be *executed* by all subjects.
- File1 can be *read* by Eve.

(i) Specify the Access Control Matrix for this scenario. [20%]

ANSWER:

	File1	File2	File3	File4
Alice	read	read, execute	read, append	read
Bob		append, execute		read, write
Charlie	write, read	write, execute		
Eve	read	execute		

Marking Scheme:
[2%] deduction for each missing or wrong action (no negative marking applied).

(ii) Briefly explain a scenario where two subjects that collaborate can violate the integrity or the confidentially of a file that is not possible without the two subjects working together. [20%]

ANSWER:

For example, if Alice and Charlie collaborate, they can copy content from File3 into File1. This violates the confidentiality of File3 (as Charlie can not read this file) as well as the integrity of File 1 (as Alice cannot write into this file). Moreover, Eve is now also able to read the content copied from File 3.

Marking Scheme:
[10%] for naming a collaboration violating the access control matrix and [10%] for a correct explanation.

(iii) Briefly explain if Access Control Matrix Models scale well (i.e., assume a scenario with thousands of subjects and/or objects). Compare them, in this aspect, with role-based access control (RBAC) [20%]

ANSWER:

They do not scale well, as the size of the matrix grows quadratically wrt the number of subjects and objects.
RBAC scales much better, as it introduces a relation between subjects and (business) roles. Thus, at least it scales very well for large user groups.

Marking Scheme:
[5%] for the correct answer and [5%] for a correct explanation. [10%] for comparison with RBAC.

2. Cryptography & PKIs

a) (i) Briefly explain the concept of revocation lists.
[**Hint:** What information is stored in a revocation list and who maintains certification lists?] [10%]

ANSWER:

Certification Authorities (CAs) maintain certification lists. For each certificate that they issued and that is revoked, the revoked certificate (and the revocation date) is stored in the revocation list.

Marking Scheme:
[5%] for explaining that a revocation list stored revoked certificates. Additional [5%] for explaining that CA need to provide revocation lists for the certificates they issued.

(ii) Briefly explain a problem of revocation lists. [10%]

ANSWER:

Scalability is a problem: For each verification of a signature, an online check of the revocation list needs to be performed.

Marking Scheme:
[5%] for naming a problem and [5%] for a correct explanation.

b) (i) Consider a Cesar Cipher, as discussed in the lecture, with the key 5. Recall that we encode the letters by their position in the alphabet (e.g., the letter "a" is represented by the number 1, spaces are not encoded).
Encrypt the text

```
fight for your privacy
```

[15%]

ANSWER:

character	a	b	c	d	e	f	g	h	i	j	k	l	m
encoding	f	g	h	i	j	k	l	m	n	o	p	q	r

character	n	o	p	q	r	s	t	u	v	w	x	y	z
encoding	s	t	u	v	w	x	y	z	a	b	c	d	e

```
knlmy ktw dtzw uwnafhd
```

Marking Scheme:

[5%] for character table, [10%] for cipher text.

(ii) Does encrypting a text twice using Caesar Cipher improve the security? Briefly explain your answer. [15%]

ANSWER:

No, for any key k, encrypting twice is the same as encrypting once with the key $2 \cdot k$. A larger key does not increase the security of the Caesar Cipher, e.g., against frequency analysis.

Marking Scheme:
[5%] for the correct answer and [10%] a correct explanation.

c) Consider the RSA crypto scheme with the following configuration:

- Alice's public key is $(n_a, e_a) = (33, 7)$, her private key is $d_a = 3$. Alice uses this key for both signing and encryption/decryption.
- Bob's public key is $(n_b, e_b) = (65, 7)$, his private key is $d_b = 7$. Bob uses this key for both signing and encryption/decryption.
- Eve's public key is $(n_b, e_b) = (77, 13)$, her private key is $d_b = 37$. Eve uses this key for both signing and encryption/decryption.

Recall that we encode the letters by their position in the alphabet (e.g., the letter "a" is represented by the number 1, spaces are not encoded).

Alice sends to Bob the following *encrypted* message:

$$9, 1, 52, 38, 1, 45, 60$$

(i) Explain how the attacker Eve can obtain the clear text of this cipher?
[**Hint:** Recall that their keys are used for both signing and encryption/decryption.] [15%]

ANSWER:

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:

- [5%] for explaining that Eve needs to eavesdrop the message
- [5%] for explaining that Eve needs to trick Bob into signing the ciphertext.
- [5%] for explanation that the signature is the plaintext.

(ii) Demonstrate the attack by computing the plain text of the message that Alice sends to Bob. Briefly explain the necessary steps.

[**Hint:** If you did not answer 2(c)(i), show instead how Bob can obtain the plain text. Briefly explain the necessary steps.] [15%]

ANSWER:

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

$$9, 1, 52, 38, 1, 45, 60$$

- By signing the 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

$$9, 1, 13, 12, 1, 20, 5$$

- By using the same key for signing and encryption/decryption, Eve could trick Bob into decrypting the message (as signing is the same as "encrypting the message with Bob's private key"). Eve can now obtain the plain text by decoding each character of the signature using

character	a	b	c	d	e	f	g	h	i	j	k	l	m
encoding	1	2	3	4	5	6	7	8	9	10	11	12	13

character	n	o	p	q	r	s	t	u	v	w	x	y	z
encoding	14	15	16	17	18	19	20	21	22	23	24	25	26

Eve obtains the message "`iamlate`" (I am late).

Marking Scheme:

- [5%] for choosing the right key (and given a reason why it is the right one)
- [10%] for the computation
- at most [10%] for the alternative problem of Bob decrypting the message, i.e. no mentioning of Eve or *clear* reference to 2(c)(i)..

d) Can symmetric cryptography be used for implementing digital signatures? Briefly explain your answer. [20%]

ANSWER:

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:
[5%] for the correct answer and [15%] for a correct explaination.

3. Security Protocols

a) Consider the following protocol:

$A \longrightarrow B: \ \{N_A, A\}_{\mathrm{pk}(B)}$
$B \longrightarrow A: \ \{N_A, N_B, A\}_{\mathrm{pk}(A)}$
$A \longrightarrow B: \ \{N_B\}_{\mathrm{pk}(B)}$

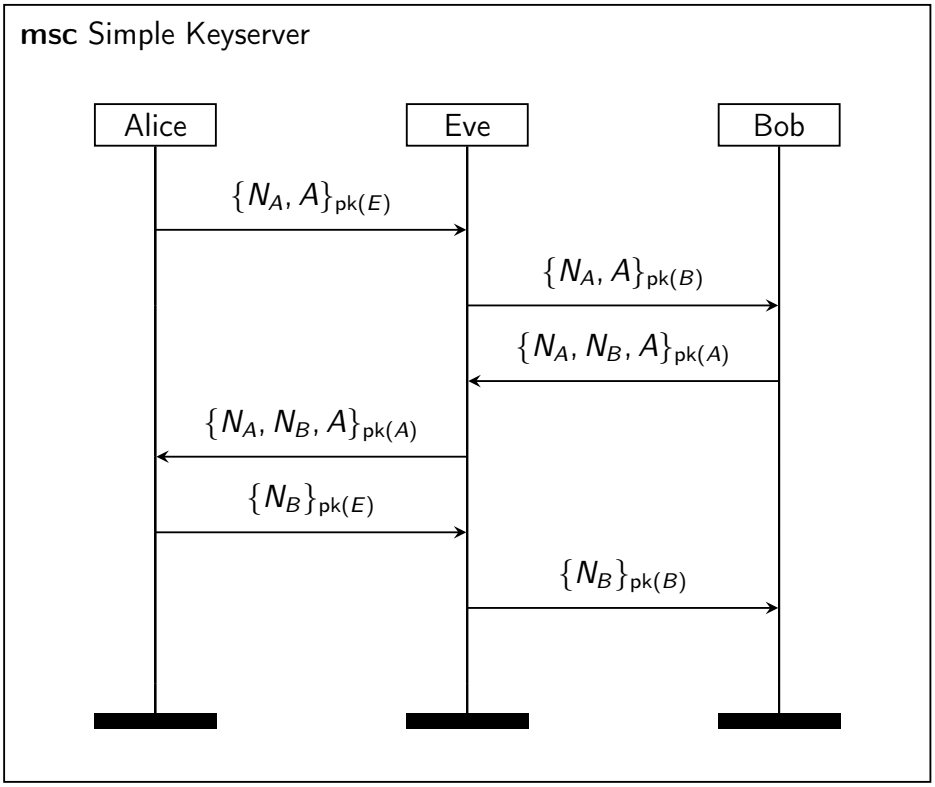
Briefly explain if this protocol is secure or not.

- If it is secure, give an informal justification why it is secure.
- If an attack is possible, explain this attack (e.g., by drawing a message sequence chart), i.e., how does it work and why it is possible.

[25%]

ANSWER:

The protocol is not secure. It can be attacked the same way as the original Needham-Schroeder protocol discussed in the lecture:



After executing the two protocol runs in parallel,

- Alice believes to speak to Eve
- Eve believes (correctly) to speak to Alice (left hand side of the chart)
- Eve believes (correctly) to speak to Bob (right hand side of the chart)
- Bob believes (wrongly) to speak to Alice (while, in fact, he speaks to Eve)

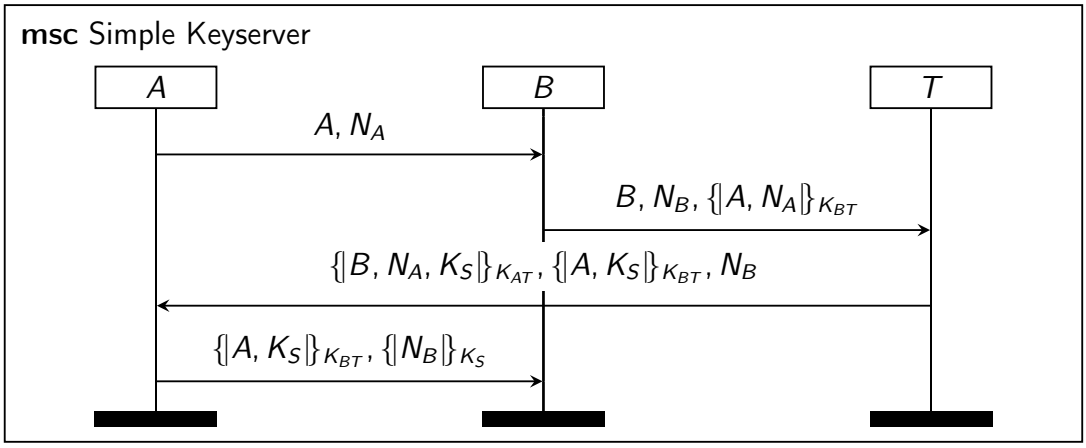
The protocol can be fixed by adding the role B into the second message (either in addition or replacing the current occurrence of role A).

Marking Scheme:

- Attack:
  - [10%] for a roughly correct message sequence chart

  - [15%] for a correct message sequence chart
- Explanation of the attack:
  - [5%] for a partial assessment that includes that Bob believes to talk to Alice.
  - [10%] for full assessment

b) Consider the following protocol in which A and B use a trusted third party T to perform mutual authentication and establish a session key K_S . Assume that initially A and T share the symmetric key K_{AT} and B and T share the symmetric key K_{BT} . A and B generate Nonces N_A and N_B, respectively. There are four steps in the protocol, as illustrated in the following message sequence chart:



Consider the message in step 4 (from A to B). Briefly explain why A encrypts the Nonce N_B with K_S. [20%]

ANSWER:

Alice proves that she knows K_S. Note that N_B is sent as plain text in step 3, hence the answer "to protect N_B" is wrong.

Marking Scheme:
[20%] for a correct explanation.

c) The Dolev-Yao closure $\mathcal{DY}(M)$ consists of *seven* deductive rules. Three of them are:

$$\frac{}{m \in \mathcal{DY}(M)}\ \text{Axiom}\ (m \in M) \qquad \frac{s \in \mathcal{DY}(M)}{t \in \mathcal{DY}(M)}\ \text{Algebra}\ (s \approx t)$$

$$\frac{\langle m_1, m_2 \rangle \in \mathcal{DY}(M)}{m_i \in \mathcal{DY}(M)}\ \text{Proj}_i$$

(i) Define the missing *four* rules of the Dolev-Yao closure *formally* and give a brief informal description of what action of the intruder they formalise.
[**Hint:** if you cannot describe the rules formally, explain informally the different actions a Dolev-Yao intruder can use for inferring new messages from an initial set of messages (this might need two or three short sentences per rule).] [25%]

ANSWER:

The missing rules are:

$$\frac{t_1 \in \mathcal{DY}(M) \cdots t_n \in \mathcal{DY}(M)}{f(t_1, \ldots, f_n) \in \mathcal{DY}(M)}\ \text{Composition}\ (f \in \Sigma_p)$$

The Composition Rule allows the attacker to apply public functions, if the attacker knows all arguments.

$$\frac{\{\!|m|\!\}_k \in \mathcal{DY}(M) \quad k \in \mathcal{DY}(M)}{m \in \mathcal{DY}(M)}\ \text{DecSym}$$

The DecSym Rule allows the attacker to decrypt messages that are encrypted with a symmetric encryption scheme, if the attacker knows the key used for encryption.

$$\frac{\{m\}_k \in \mathcal{DY}(M) \quad \text{inv}(k) \in \mathcal{DY}(M)}{m \in \mathcal{DY}(M)}\ \text{DecAsym}$$

The DecAsym Rule allows the attacker to decrypt messages that are encrypted with a asymmetric encryption scheme, if the attacker knows the private key that corresponds to the public key used for encryption.

$$\frac{\{m\}_{\text{inv}(k)} \in \mathcal{DY}(M)}{m \in \mathcal{DY}(M)}\ \text{OpenSig}$$

The OpenSig Rule allows the attacker to access the content of signed messages.

Marking Scheme:

- [3%] for each correct formal rule.
- [2%] for each informal explanation of a formal rule.
- [3%] at most for each detailed but only informal description of rules.

(ii) Consider the following intruder knowledge:

$$M = \{\{\!|m|\!\}_{g(n_1)}, Axiom\{n_1\}_{\mathsf{inv}(\mathsf{pk}(a))}, \{n_2\}_{\mathsf{pk}(i)}, \mathsf{pk}(a), \mathsf{pk}(i), \mathsf{inv}(\mathsf{pk}(i)), \{\!|secret|\!\}_{\mathsf{pk}(b)}, \{\{\!|secret|\!\}_m\}_{\mathsf{inv}(\mathsf{pk}(b))}\}$$

where g is a public function (i.e., $g \in \Sigma_P$).
Prove formally that the intruder can learn the message "*secret*".
[**Hint:** if you cannot recall the formal proof notation, give a detailed informal argument.] [30%]

ANSWER:

$$\dfrac{\dfrac{\dfrac{}{\{\{\!|secret|\!\}_m\}_{\mathsf{inv}(\mathsf{pk}(b))}}\,\text{Axiom}}{\{\!|secret|\!\}_m \in \mathcal{DY}(M)}\,\text{OpenSig} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\{\{n_1\}_{\mathsf{inv}(\mathsf{pk}(a))} \in \mathcal{DY}(M)}\,\text{Axiom}}{n_1 \in \mathcal{DY}(M)}\,\text{OpenSig}}{g(n_1) \in \mathcal{DY}(M)}\,\text{Composition } (g \in \Sigma_P) \qquad \dfrac{}{\{\!|m|\!\}_{g(n_1)} \in \mathcal{DY}(M)}\,\text{Axiom}}{m \in \mathcal{DY}(M)}\,\text{DecSym}}{secret \in \mathcal{DY}(M)}\,\text{DecSym}$$

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

1	$\{\{\!\|secret\|\!\}_m\}_{\mathsf{inv}(\mathsf{pk}(b))}$	premise
2	$\{\!\|m\|\!\}_{g(n_1)} \in \mathcal{DY}(M)$	premise
3	$\{n_1\}_{\mathsf{inv}(\mathsf{pk}(a))} \in \mathcal{DY}(M)$	premise
4	$\{\!\|secret\|\!\}_m \in \mathcal{DY}(M)$	OpenSig 1
5	$n_1 \in \mathcal{DY}(M)$	OpenSig 3
6	$g(n_1) \in \mathcal{DY}(M)$	Composition $(g \in \Sigma_P)$ 5
7	$m \in \mathcal{DY}(M)$	DecSym 2,6
8	$secret \in \mathcal{DY}(M)$	DecSym 4,7

Marking Scheme:

- [10%] deduction for a correct informal proof.
- wrong but formal proof attempts that partially use the correct reasoning (i.e., correct sub-proofs) can get up to [10%].
- no deduction for small mistakes (no rule names, obvious typos, etc.)

4. Application Security

a) Consider a web application that uses the following client-side protection to prevent users from creating new "admin" accounts:

```
<select name="user[role]" id="user_role">

<option disabled="disabled" value="admin">admin</option>
<option disabled="disabled" value="lecturer">lecturer</option>
<option value="student">student</option>
</select>
```

(i) Explain briefly how an attacker can circumvent this check and create a new admin account. [15%]

ANSWER:

Using either a proxy that allows to modify requests or the developer mode of modern web browsers, an attacker can enable the two disabled options by simply removing the disabled flags using the developer mode. Since there are no server side checks implemented that enforce this policy as well, the attacker will now be able to create new accounts with any role.
Alternatively, an attacker can (either using a proxy or a tool such curl, etc.) create an hand-crafted request creating a user with admin or lecturer role.

Marking Scheme:
Up to [10%] for explanations pointing out that the client-side checks cannot be trusted, i.e., they can be circumvented by an attacker.

(ii) Explain briefly how a programmer can prevent this type of attack. [15%]

ANSWER:

The programmer needs to implement range checks in the server code of the application that needs to take the role of the current user into account.

Marking Scheme:
Up to [10%] for explanations that mention server-side checks but do not mention that the role/permissions of the currently logged-in user need to be considered.

b) Consider the following vulnerability in OpenSSL (CVE-2014-0160):

> The TLS and DTLS implementations in OpenSSL do not properly handle Heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read.
> A successful attack requires only sending a specially crafted message to a web server running OpenSSL. The attacker constructs a malformed "heartbeat request" with a large field length and small payload size. The vulnerable

server does not validate that the length of the payload against the provided field length and will return up to 64 kB of server memory to the attacker. It is likely that this memory was previously utilized by OpenSSL. Data returned may contain sensitive information such as encryption keys or user names and passwords that could be used by the attacker to launch further attacks.

What CVSS v2 Base Vector (AV:?/AC:?/Au:?/C:?/I:?/A:?) would you assign to this vulnerability? Explain your decision for each component of the CVSS v2 Base Vector. In your answer, you will need to replace the '?' in the CVSS v2 Base Vector. [30%]

ANSWER:

- Base Metrics
  - Access Vector: Network
    A *remote* user can exploit the vulnerability.
  - Access Complexity: Low
    The protocols implemented by OpenSSL are standardised and many client libraries are available. Sending a large heartbeat packet to an OpenSSL server is, thus, *easy* and sufficient to exploit the vulnerability.
  - Authentication: None No authentication is required.
- Impact Metrics
  - Confidentiality Impact: Partial
    The attacker has no control over the information that is stored in the obtained memory. Moreover, the attacker can only obtain up to 64kB memory (per exploitation).
  - Integrity Impact: None
    No data on the server can be modified.
  - Availability Impact: None
    No impact on the availability of the server.

  Thus, the CVSS v2 Base Vector is: (AV:N/AC:L/Au:N/C:P/I:N/A:N).

Marking Scheme:
In general, [5%] for each component of the CVSS Base Vector. If the assessment and the explanation is consistent for one component, but deviates from the correct solution, only minor deductions (e.g., [1%] or [2%]) are applied.

c) Consider the following Java program

```
String mname = request.getParameter("month");
String uid = session.getCurrentUserId();
PreparedStatement pstmt = conn.prepareStatement
             ("SELECT username, startdate, transaction, amount
               FROM transaction_history
               WHERE user=" + uid + " AND month=" + mname");
pstmt.execute();
pstmt.close();
```

(i) What vulnerability does this program have? Briefly explain the vulnerability in general and, in particular, why this program is vulnerable. [20%]

ANSWER:

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:

- [5%] for the correct name (SQL Injection).
- [5%] for explaining what a SQL injection is.
- [10%] for a correct explanation mentioning a wrong use of prepared statement. At most [5%] if the explanation does not mention the concept of prepared statements.

(ii) Rewrite this program to fix the vulnerability. [20%]

ANSWER:

The prepared statement needs to be rewritten to make use of placeholder parameters (?):

```
String mname = request.getParameter("month");
String uid = session.getCurrentUserId());
PreparedStatement pstmt = conn.prepareStatement
             ("SELECT username, startdate, transaction,
                      amount
               FROM transaction_history
               WHERE user=? AND month=?");
pstmt.setString(1,uuid);
pstmt.setString(2,name);
pstmt.execute();
pstmt.close();
```

Marking Scheme:

- [10%] if only one parameter is converted.
- at most [10%] for a solution using a not specified sanitisation function.
- at most [15%] for solution using a (mostly) correct specified sanitisation function.

END OF QUESTION PAPER