Assignment 2

CS 458 Computer Security and Privacy – Shun Da Suo | sdsuo |20509411

Written Question 1: Security Policies – BIBA Integrity Model/Bell-La Padula Confidentiality Model (15 marks)

Part a) Bell-La Padula Confidentiality Model

i. Read Only
ii. None
iii. None
iv. Both
v. Write Only

Part b) Dynamic BIBA Model

i. File A has new integrity: (TA, {assignments})
ii. Carol has same integrity: (Student, {assignments, marks})
iii. File C has new integrity: (Professor, {assignments, marks})
iv. File D has new integrity: (TA, {assignments})
Bob has new integrity: (TA, {assignments})
File E has new integrity: (TA, {assignments})
v. File F has new integrity: (Student, {marks})
Alice has new integrity: (TA, {assignments})

Written Question 2: Password Cycling and Similarities (9 marks)

Part a) Two Problems

1. Since Alice's password consists of only 7 lower case letters, it is at risk of a brute force attack. From lecture, we know that a cluster of high-end graphics cards is capable of brute forcing every possible eight-character password containing upper- and lower-case letters, digits, and symbols in hours.
2. Alice's password is a valid word in the English dictionary, which makes it even more vulnerable. Since valid words are more likely to be used in passwords than random combination of characters, attackers often use them as roots of password combinations to optimize brute force attacks.

Part b) Worst Case Possibilities

Since Eve know Alice's strategy, there are only 10 possible digits Alice may append to her original password.

Part c) Potential Password

Kittens0$000

Part d) Worst Case Possibilities

1. If Eve knows the password after ii), she needs to test out all choices of 1 symbol and 3 digits:

$$6 * 10 ^ 3 = 6000 \text{ possibilities}$$

2. If Eve does not know the password after ii), she needs to test out all choices of 1 symbol and 4 digits:

   10 * 6 * 10 ^ 3 = 60000 possibilities

Written Question 3: Password Hashing (9 marks)

Problem 1

SHA-1 is a standard cryptographic hash that is relatively cheap to compute. This implies that it would be feasible and inexpensive to mount an offline guessing attack.

An iterated hash function that is expensive to compute (such as bcrypt) or uses a lot of memory (such as scrypt) should be used. This would slow down a guessing attack significantly, but is barely noticeable for users.

Problem 2

18-bit salts may be too short for the application. The purpose of including salts is to modify the function used to hash each user's password so that each stored password hash must be attacked individually. If there is a large number of users in Alice's user account database, probability of having repeated salts may be high with only 18 bits.

To remedy this, Alice should simply use longer salts. Research shows that random 64-bit salts are very unlikely to ever repeat even with a billion registered users.

Problem 3

This password scheme is vulnerable to interception attacks since passwords or hashes are often transmitted in plaintext on the network.

To prevent such an attack, Alice can make ensure the network is encrypted (i.e. with TLS). She can further enhance security by employing one-time passwords (i.e. with a challenge-response protocol.

Programming Question 1: SQL Injection (5 marks)

b) SQL Injection Prevention

To prevent SQL injection attacks, one can use prepared statement with variable binding (as known as parameterized queries) instead of dynamic queries. This system forces the developer to first define all the SQL queries, and then pass in each parameter to the query later. Thus, the database can see a clear distinction between query data and query syntax, regardless of what the user inputs.

Programming Question 2: Cross-site Scripting (9 marks)

b) Same-Origin Policy and XSS

No, same origin policy does not prevent XSS attacks, since the attacks originate from malicious code injected into the original website. The victim's browser will not prevent execution of the malicious code since it does come from the same origin as the website that the victim is visiting.

c) XSS Mitigation

In this case, user input is not checked and sanitized before being inserted into HTML element content. To mitigate this without modifying the database, we can perform HTML escaping for the user inputs (e.g. encode '&' with '&amp'). An example may look like this:

```
String safe = ESAPI.encoder().encodeForHTML( request.getParameter( "input" ) );
```

As a result, the user's browser will interpret the malicious code as plaintext instead of valid HTML elements.

Programming Question 3: Cross-site Request Forgeries (16 marks)

c) Defences Against CSRF Attacks

i. One can employ a synchronizer (CSRF) token system, where any state changing operation requires a secure random token. The attacker, without control of the victim's browser, cannot access this random token, and thus will fail to supply this in maliciously forged requests.

ii. The server can be set up to inspect header information of the received requests and verify the source and target origin of the request matches as expected. One can identify the source origin via either the origin header or the referrer header, and the target origin via X-Forwarded-Host. After identifying the origins, one can simply compare the two values and verify if it is a cross-origin request.

d) HTTPS and CSRF

No, configuring the server to use HTTPS instead of HTTP does not prevent CSRF attacks. In a CSRF attack, the victim is tricked into performing an request with his or her browser, which sends previously established identifying state information. Thus, the addition of authentication and secure communication has no bearing on the vulnerability.

e) CSRF and CORS

If Alice is using a modern browser which enforces the same origin policy, then it will prevent Eve from reading data from our website, since it does not use a CORS header. There is, however, nothing preventing state changing requests from being executed on our website. Furthermore, this is an entirely client side restriction, and it is only enforced through Alice's browser.