Security I Quiz

TOTAL POINTS 10

1. Which of the following is true of the economy of mechanism design principle (select all that apply)? 1 point

- [x] Using less complicated code to implement the same functionality simplifies security audits.
- [x] Using less code to implement the same functionality simplifies security audits.
- [ ] Using less code to implement the same functionality will always mean that the code is less extensible and modular.

2. Assuming that "runner" is the only implementation of Runner and never needs to be changed, which of the following refactorings of the code shown below would best demonstrate the economy of mechanism design principle (select all that apply)? 1 point

```
public class DataHandler {

  public interface Runner {
    public void run(Runnable r);
  }

  private final ExecutorService executor = Executors.newFixedThreadPool(1);

  private final Runner runner = new Runner() {
    public void run(Runnable q){
      executor.submit(q);
    }
  }

  public void runIt(Runner r, Runnable q){
    r.run(q);
  }

  public void fetchData() {
    runIt(runner, new FetchDataTask());
  }

  public void writeData() {
    runIt(runner, new WriteDataTask());
  }

}
```

- [ ]
```
public class DataHandler {

  public interface Runner {
    public void run(Runnable r);
  }

  private class MyRunner {
    public void run(Runnable q){
      executor.submit(q);
    }
  }

  private ExecutorService executor;

  private Runner runner;

  public DataHandler() {
    executor = Executors.newFixedThreadPool(1);
    runner = new MyRunner();
  }

  public void runIt(Runner r, Runnable q){
    r.run(q);
  }

  public void fetchData() {
    runIt(runner, new FetchDataTask());
  }

  public void writeData() {
    runIt(runner, new WriteDataTask());
  }

}
```

- [ ]
```
public class DataHandler {

  private ExecutorService executor;

  public DataHandler() {
    executor = Executors.newFixedThreadPool(1);
    runner = new MyRunner();
  }

  public void runIt(ExecutorService r, Runnable q){
    r.submit(q);
  }

  public void fetchData() {
    runIt(executor, new FetchDataTask());
  }

  public void writeData() {
    runIt(executor WriteDataTask());
  }

}
```

- [x]
```
public class DataHandler {

  private final ExecutorService executor = Executors.newFixedThreadPool(1);

  public void fetchData() {
    executor.submit(new FetchDataTask());
  }

  public void writeData() {
    executor.submit(new WriteDataTask());
  }

}
```

- [ ]
```
public class DataHandler {

  private ExecutorService executor;

  public DataHandler(ExecutorService ex){
     executor = ex;
  }

  public void fetchData() {
     executor.submit(new FetchDataTask());
  }

  public void writeData() {
     executor.submit(new WriteDataTask());
  }
}
```

3. Which of the following are reasons why the principle of least privilege is important (select all that apply)? 1 point

- [x] Each privilege that your code possesses creates an additional security responsibility to protect that privilege.
- [ ] Each privilege that your code possesses increases the capabilities of your app and makes development faster.
- [ ] Each privilege that your code possesses increases the memory consumption of your application.

4. In Android, which of the following are true of apps that do not apply the principle of least privilege (select all that apply)? 1 point

- [ ] They increase build times.
- [ ] They consume more space on disk.
- [x] They increase the chance that an app will gain access to a capability that requires a uses-permission without declaring the associated uses-permission.
- [ ] They make development easier by allowing the developer to use any device capability they want without having to change the manifest.

5. Which of the following could be examples of secure defaults (select all that apply)? 1 point

- [ ] encrypting data that the user has explicitly flagged as sensitive
- [x] forcing the user to set a password on first use
- [ ] using a default username and password to protect data until the user sets their own password

6. Why are secure defaults important (select all that apply)? 1 point

- [x] A user may not know there are any default settings to change.
- [x] A user may not have the expertise to change default settings in a secure manner.
- [x] A user may not bother changing default settings.
- [x] A user may assume that the default settings are secure.

7. Which of the following is an example of secure defaults (select all that apply)? 1 point

- ( )
```
Map<String,StorageEngine> storageEngines = new HashMap<>();

public void init() {
  storageEngines.put("secure", new EncryptedStorageEngine());
  storageEngines.put("plain", new PlainTextStorageEngine());
}

public void store(String type, String data){
  storageEngines.get(type).store(data);
}
```

- ( )
```
Map<String,StorageEngine> storageEngines = new HashMap<>();

public void init() {
  storageEngines.put("secure", new EncryptedStorageEngine());
  storageEngines.put("plain", new PlainTextStorageEngine());
}

public void store(String type, String data){
  if(storageEngines.get(type)!=null){
    storageEngines.get(type).store(data);
  }
  else {
    System.out.println("Unable to store "+data);
  }
}
```

- (•)
```
Map<String,StorageEngine> storageEngines = new HashMap<>();

private encryptedEngine = new EncryptedStorageEngine();

public void init() {
  storageEngines.put("secure", encryptedEngine);
  storageEngines.put("plain", new PlainTextStorageEngine());
}

public void store(String type, String data){
  if(storageEngines.get(type)!=null){
    storageEngines.get(type).store(data);
  }
  else {
    encryptedEngine.store(data);
  }
}
```

8. A number of recent news articles have highlighted apps that asked for a large number of uses-permissions. Which design principle are these apps most likely to violate (select all that apply)? 1 point

- [ ] Economy of Mechanism
- [x] Least Privilege
- [ ] Complete Mediation
- [ ] Secure Defaults

9. What is wrong with the code below (select all that apply)? 1 point

```
public static int SECURE = 1;
public static int MOST_SECURE = 2;
public static int INSECURE = 0;

public void sendData(int securityLevel, String data){
  if(securityLevel > 0){
     sendEncrypted(data);
  }
  else if(securityLevel == 0){
     sendPlainText(data);
  }
}
```

- [x] In Java, new integer variables are initialized to zero, which will cause the default behavior to be insecure.
- [x] The flags for the security levels are not final, which could lead to runtime changes of their values and unexpected behavior.

10. When is it appropriate to ignore these security principles in order to develop new functionality (select all that apply)? 1 point

- [ ] if ignoring them helps meet a deadline
- [ ] if ignoring them leads to improved performance
- [x] never

- [x] I, **Shubham Kumar**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Saving... Submit