**Homework 1**

**Question 1:**

Why attacks on HTML5 based apps are possible? Why native apps are immune to such attacks

HTML5 based web applications have to run in a web container that supports HTML5 and JavaScript. The containers are meant to sandbox the code to protect the operating system, the middleware allows JavaScript to make calls to system in native language. Web frameworks are susceptible to code injection as they accept both code as well as data and execute the code. So if in the data segment of the application, code is given, it will be executed without checking further.

In case of native apps such as those written in C++ or Java, this kind of code injection attack is not possible because, the bytecode has already been generated as a result of code getting compiled by the Java compiler. SO there is a clear segregation of data and code here, any input that is received will be treated as a data, as the JVM or the system knows that the incoming input has to be treated as data and not code.

**Question 2:**

What do the attacks have in common?

SQL Injection

Cross-Site Scripting attack

Attack on system(command)

The 3 attacks have the following in common,

1. No segregation between code and data.
2. String has code and data for input. The code and data are supplied to all of these scenarios by the user. The program is supposed to combine the code and data and generate the input to the system. If the segregation between code and data is not done. The system will not see any boundary between code and data and this will lead to execution of code embedded in data.
3. The code and data are supplied to all of these scenarios by the user.
4. SQL, has a database, Javascript has a javascript engine and system has a shell.
5. All the attacks take advantage of the fact that the coder knows the boundary of data and code but the engine does not know this distinction. And is therefore susceptible to the code injection attacks.

String supplied to respective engine

String constructed based on user input

: Database  
Engine : JS Engine  
 : Shell

User input   
(Data)

**Question 3:**

Why do cross-site Ajax requests need to be restricted but the normal cross site requests do not?

AJAX requests are submitted with user credentials and allow the caller to read the returned data.

XMLHTTPRequest allows you to post data with credentials such as cookie data to the destination site and read the data that is returned from the site, which includes the secure data of that site. So, if a cross-site Ajax request is allowed to facebook, then we will get the cookie data along with all the credentials needed to establish a secure connection to facebook. This will allow an attacker to create the same session that the victim is accessing.

In case of normal cross-site requests, the authentication data is verified and the sensitive data is not posted to the attacker.

**Question 4:**

Explain clickjacking attack.

Clickjacking attack can be performed by the use of iframes. It is also knows as UI redress attack. It is a malicious technique of tricking a Web user into clicking something different from what the user perceives they are clicking on, thus potentially revealing confidential information or taking control of their computer while clicking on seemingly harmless web pages. It is a browser security issue. Clickjacking can be understood as an instance of the confused deputy problem. Most websites implement a server side prevention known as frame busting, where conditional checks in javascript ensure that the frame with the website is the frame that is on top. Another protection is the use of X-Frame-Options and set the property to DENY, this can be done in PHP. X-Frame-Options detect and prevent frame-based UI redressing attacks Other methods include installation on client side, such as plugins and extensions to the browser, this type of prevention just warns the user of the clickjacking attack.

**Question 5:**

Stateless nature of the web has led to the data integrity and workflow problems.

In case of data integrity, stateless nature of the web results in the possibility of manipulation of the data by attacker as the server and client interact with other and the server makes use of the critical data that is returned by the client. This data can be manipulated by a malicious third-party, which can lead to vulnerabilities such as incorrect transactions, incorrect commits etc.

Stateful nature of the web will not allow this to happen as the server is aware of the client that is sending the data and thus fetches the data securely and performs the necessary computations.

Workflow is a sequence of events that need to be completed for a transaction to be successful.In case of workflow problems, the stateless nature of the web allows for workflow based attacks, as in each call to a server we may or may not be talking to the same server. This attack can be done by allowing the skipping of the different steps that are involved in for example a purchase of products. We can directly load the final process instead of providing the data for all the processes, which is possible as the server doesn’t remember if the previous steps have been followed or not. If the order is not enforced then, we can bypass security checks to reach the final process. So the order must be enforced to avoid this kind of a workflow attack.

In stateful nature of the web since the client server system always knows which step they are in, it is always aware of the sequence of transactions being followed and so workflow attack will not be successful.

**Question 6:**

Design disable and enable functionalities for the file-descriptor’s capability mechanism.

Design suggestion:

The file descriptor has an fd flag bit that is being used. So we can check this value and add another value that this flag can accept, by doing this we can modify this bit value and whenever it is checked we can call a fd\_enable or fd\_disable function on the file descriptor, we can disable the capability by calling the fd\_disable and changing the fd flag to say 2 for disable and 3 for enable. So when we don’t want this privilege we can relinquish this privilege by using fd\_disable and calling fd\_enable when we want certain privileges in the application, this can be done by checking the flag when access control is being enforced.

The kernel can also use this function, if the kernel wants to remove the capability it can call fd\_disable on a particular process and similarly the fd\_enable can be called on the process by the kernel itself.

However, this enable and disable functionality is not effective as by simply using buffer overflow attacks the capability can be re enabled by an attacker. The attacker, will run a malicious code and since the enable and disable functions are accessible to the user program through an interface to the kernel, even though the kernel will disable the capability, the malicious program can re enable the capability. Thus rendering this enable and disable functionality useless.