1 What is “OS-command injections”? Explain how it happens.

OS-command injections are a type of attacks attackers convert queries (SQL queries, but not necessarily limited to SQL) to commands to operating systems (to let the operating systems at an attacked host computer execute some commands for remote attackers, such as creating a new user account for attackers, installing backdoors, and etc.).

A well-known technique to launch OS-command injection attacks is to attach “$” at the top of the text message destined to a SQL server. As soon as a SQL server at an attacked host computer detects “$” at the beginning of each SQL query, the SQL server forwards it the operating system, by replacing “$” by “exec”, which is a system call to let the operating system execute the rest as a command.

OS-command injection attacks exploit the servers that do not properly perform sanitization in SQL server.

2 Explain how “Cross-Site Scripting (XSS) attacks” are performed by attackers.

Cross-Site Scripting attacks are another type of attacks attackers exploit lack of sanitization (at web servers). As follows:

① Attackers prepare a web page that hosts a web page, which contains a malicious script and wait for victim web users to visit their web sites. The attackers’ web pages contain a link to another web server who does not perform sanitization, especially to the malicious scripts prepared by the attackers.

② When a victim web user visits the attacker’s web site, the victim web user’s web browser downloads the attacker’s web page that contains malicious script (the web user will not see the malicious script in the web page).

③ When the victim web user clicks a link to another web server that does not perform sanitization, the malicious scripts prepared by the attacker are transferred to the web server (the one that does not perform sanitization).
The web server (the one that does not perform sanitization) returns the malicious scripts prepared by the attacker to the web user’s browser.

When the web user’s browser receives the attacker’s malicious through another web server, the web user’s browser executes the attacker’s malicious scripts as those from the (innocent) web server (instead of from the attackers – there is no way for the web user to see that the scripts are actually from the attacker, not from the innocent web server).

(3) What is “onload” attribute for “<body>” html tag? Why can it cause a security risk?

- The “onload” attribute lets web browsers to automatically start executing some (but specific) activities or even programs as soon as a web page (such as “index.html”), which contains “onload” attribute, is (down)loaded to a browser.
- The “onload” attribute is known to cause a security risk, primarily because it can let attackers to perform many activities, such as executing some processes, installing a malicious software tools, and even redirecting our web browsers to harmful web sites, such as XSS web sites by kicking web users (i.e., “us”) from the control loop in web navigations.

(4) What risks exist if error messages in web applications are not well designed?

If error messages are not carefully designed, such “careless error messages” can provide attackers the information about a server host computer or a server software process. For example, a careless error message as shown below provide the structure of an underlying database tables, which let attackers devise malicious messages for effective SQL injections.

(5) What are “back doors”?

Back doors are processes (those that do not show their presence in a host computer) that waits for connections using a connecting point other than the standard connecting points (e.g., TCP ports #20, #21, #22, and #23, and etc.) to establish a connection with only users who know the presence of such (network) processes. Back doors are not necessarily a malware (although many attackers use back doors to control a hijacked network host computers).