13.4.7 Web Attack Facts

Common web application attacks include:

Attack	Description
Drive-By Download	*A drive-by download* is an attack where software or malware is downloaded and installed without explicit consent from the user. Drive-by downloads can occur in a few different ways: ▪ Through social engineering, the user is tricked into downloading the software. The user might not realize that clicking a link will install software, or the user might know that something is being installed but not fully understand what it is or what it does. ▪ By exploiting a browser or operating system bug, an attacker is able to us a site to install software without the user's knowledge or consent.
Typosquatting/URL Hijacking	*Typosquatting* or *URL hijacking* occurs when an attacker registers domain names that correlate to common typographical errors users make while trying to access legitimate websites. Common strategies include the following: ▪ Using domain names that are based on common typing errors made when users enter the legitimate site's domain name (for example, amazno.com instead of amazon.com). ▪ Using domain names that are slightly different from the legitimate site's domain name (for example, blackhorses.com instead of blackhorse.com). ▪ Using a top-level domain different from the legitimate site's domain (for example whitehouse.com instead of whitehouse.gov). The typosquatter's intentions may be benign or malicious in nature. They may be simply trying to coerce the legitimate site owner to buy the domain name from them. Alternatively, they may be attempting to compromise unsuspecting users by redirecting them to a phishing site that looks like the legitimate website. They may even use this exploit to install drive-by malware.
Watering Hole	A *watering hole attack* is a variation of a spear phishing attack, which is directed at a specific organization or person. Instead of overtly sending traditional phishing messages directly to the target, a watering hole attack is more passive than phishing; it relies on the trust the target has in specific websites. To conduct this exploit, the attacker uses reconnaissance to identify which websites the targeted person or organization frequently uses. The attacker then compromises one or more of those sites in some way, hoping that the target will access the site and be exposed to the exploit.
Buffer Overflow	A *buffer overflow* occurs when the operating system or an application does not properly enforce boundaries for data input types and amounts. Hackers submit data beyond the size reserved for the data in the memory buffer, and the extra data overwrites adjacent memory locations. The extra data sent by the attacker could include executable code that might be able to execute in privileged mode. Buffer overflow is a common attack on web servers. Countermeasures for buffer overflow attacks include: ▪ Applying all security patches to workstations. ▪ Limiting user input to less than the size of the buffer. ▪ Validating input by looking for certain symbols that may be program instructions. ▪ Implementing strict coding standards to eliminate the potential for weaknesses.
Integer Overflow	An *integer overflow* occurs when a computational operation by a running process results in a numeric value that exceeds the maximum size of the integer used to store the numeric value in memory. In other words, there simply isn't space for the large operation number. When this occurs, the value will wrap around and start again at its minimum value, in much the same way a mechanical odometer in a car rolls over to zero when it exceeds the maximum number of miles it can record. An integer overflow condition can allow an attacker to manipulate the value of variables, leading to unintended behavior from the system. For example, when purchasing items from an online store, an integer overflow attack could be used to convert the total due from a positive value to a negative value, refunding money to the customer's credit card when the transaction is executed.
Cross-Site Scripting (XSS)	*Cross-site scripting* is an attack that injects scripts into web pages. When the user views the web page, the malicious scripts run, allowing the attacker to capture information or perform other actions. ▪ XSS often relies on social engineering or phishing to entice users to click on links to web pages that contain the malicious scripts.

	▪ Some scripts redirect users to legitimate websites but run the script in the background to capture information sent to the legitimate site. ▪ Scripts can read (steal) cookies that contain identity information, such as session information. ▪ Scripts can also run under the security context of the current user. For example, scripts might execute with full privileges on the local system, or the scripts might run using the credentials used on a financial website.
Cross-Site Request Forgery (CSRF/XSRF)	Cross-site request forgery (also known as a *one-click attack* or *session riding*) is a type of malicious exploit whereby unauthorized commands are transmitted from a user to a website that currently trusts the user (perhaps because of authentication or cookies). This is almost the opposite of the XSS attack, except CSRF exploits the trust that a site has in a user's browser.
Command Injection	A *command injection* attack injects and executes unwanted commands on the application. The commands are executed with the same privileges and environment that are granted to the application. ▪ A *code injection* attack extends the default functionality of application without executing system commands. ▪ An *OS command injection* attack executes system level commands through a vulnerable application. ▪ In an *arbitrary code execution* exploit, a vulnerability in a running process allows an attacker to inject malicious code and execute it. Because it can be carried out from a remote computer, this attack is sometimes called a *remote code execution* exploit. The exploit is typically executed by manipulating the vulnerable process instruction pointer, which tells the CPU which instruction to run next. Malicious code is injected by the attacker, usually masquerading as simple input data for the running process. By redirecting the instruction pointer to the attacker's injected code, the attacker is able to take control of the process.
Zero Day	A *zero day attack* (also known as a zero hour or day zero attack) is an attack that exploits computer application vulnerabilities before they are known and patched by the application's developer.

Mitigating Web Application Attacks

Mitigation practices to protect internet-based activities from web application attacks include:

- Using the latest browser version and patch level.
- Verifying that the operating system is at the latest patch level.
- Installing antivirus, anti-spyware, pop-up blockers, and firewall software.
- Using input validation when programming services.
  - Client-side validation should be used on the local system first to identify input errors before the data is ever sent to the server. For example, if the user enters an invalid value in an email address field, the error can be detected immediately before the data is submitted.
  - Server-side validation should then be used after the data is sent to the server to detect errors. Experienced attackers can circumvent client-side validation techniques to send malicious information to the server. For example, an attacker could send data to the server from outside the application's standard user interface, bypassing any input validation measures that may have been implemented on the client.

    It's unwise to rely solely on client-side input validation techniques.

- Using add-ons to increase the security of browsing activities:
  - NoScript blocks all active content except from sites you trust.
  - Adblock Plus blocks advertisements and ad banners on the internet (which could contain malicious code).
- Training users to log out of websites when finished and not to allow applications to remember authentication.