12.2.2 Web Application Facts

Web applications are programs that run on a web server and are accessed by a web browser. With the advancement of cloud-based storage and services, the use and functionality of web applications has increased dramatically. Web applications process, store, and distribute information on demand. The difference in a local application and a web application is that the majority of the processing is done remotely on a web server. A request is sent from the client device to the web server. The server processes the request and returns the results to the client.

This lesson covers the following topics:

- Web application server
- Web application vulnerabilities

Web Application Server

There are three types of web applications:

- Browser-based web applications are run in a web browser. These applications can have access to data on the web server or other local systems, depending on need.
- Mobile apps are probably the most common type of application used today. Mobile apps are similar to other applications, but are specifically designed to work on mobile operating systems such as Apple iOS or Android.
- Client-based apps are run in a separate client-side application that needs to be installed before the web app can be used.

There are many benefits to using web apps over local apps. There is no need to purchase the app, install it on a host system, or provide processor and storage resources. Processing and storage happen remotely. Administration is easier because there are fewer installations and updates on client computers. If the app can be run over a browser, there is no need to develop the application for every operating system.

Web Application Vulnerabilities

The following table lists several web application vulnerabilities:

Vulnerability	Description
Cookies	Cookies are a necessary part of web applications; they are also a notable vulnerability. Because HTTP was not designed to save state information, cookies were created to store information about user preferences and web activities. Cookies contain several parameters, including the cookie's name, the cookie's value, the expiration date, the related URL, and the related domain. In addition to accessing valid cookies to gather information, an attacker could also use malicious cookies to track and store online activity. Web developers must take care to safely and securely handle cookies. Extreme methods such as restricting all cookies would make web browsing very difficult for the user. It is possible, however, to restrict specific types and sources of cookies.
Flawed web design	Because web developers assume that no one will read their code, they'll sometimes leave notes in the code that serve as reminders. Because it's easy to access a web page's source code, developers should be cautious not to include information that could be valuable to an attacker.
Buffer overflow	Ideally, a buffer should have a limit and stop accepting data when it reaches that limit. Without the necessary restrictions, the buffer will overflow when an application or process tries to send more data than the buffer is able to hold. The result can be corrupt or lost data, memory access errors, or system crashes. An attacker can exploit a buffer overflow to save malicious code in overflow areas with executable code.
Input validation	As data is entered into an application, a feature known as input validation verifies the data. For example, if a form requests a phone number, alphabetical characters would not be acceptable. When there is no restriction on the data type, it's easy to make mistakes. This may seem like a small concern, but in addition to making the data useless, extreme errors could result in a system crash.
Scripting errors	Similar to a flawed web design, scripting errors can create vulnerabilities to a system. A script that has been written incorrectly could result in sensitive information, such as passwords or usernames, available to a potential attacker. Additionally, if a default script or sample script is uploaded to a server, an attacker may be able to gather valuable information and access restricted portions of a system. Upload bombing and poison null byte attacks are designed to target possible errors in scripting. Upload bombing is the act of loading many files onto a server, hoping to fill up the server's drives and crash the system. A poison null byte attack sends special characters to the script. If the script is unable to handle the characters, the script may provide access that it otherwise would not.
Session management	When a client effectively connects to a server, a session is created. This session assigns session IDs, encryption, and permissions to a specific client for a period of time. After the session is closed, the session information should be removed. Failure to create a specified end to a session can result in unattended sessions that an attacker could take over. Weak session IDs can be easily guessed or predicted. As a result, an attacker can easily exploit a session ID for use in session hijacking. Additionally, lack of password reset verification such as an old password, security questions, or an email verification

	can make it easy for an attacker to change users' passwords.