This PDF Lecture is Made by
Abid Biswas
(CEH, NSE1, NSE2, NSE3, $ISC^2(CC)$, $Google\ Cyber\ Security\ Professional\ Cert$)
BSc in Information and Communication Engineering (ICE) at EWU
on the basis of the lectures of Skillsoft & Infosys collaboration.

Web application development:
really means we're talking about an application that is accessible using HTTP, the Hypertext Transfer Protocol, or HTTPS, if it's been secured.

So, what we need to do
is we need to be able to define the components that are related to developing and running a web application on a web server.

1) So, the first thing to think about is the web server itself. So, this means having an HTTP stack up and running.
   Now an HTTP server that's not protected with the PKI certificate will listen on TCP port 80 by default. That can be changed,
   but if you've got a PKI certificate and you enable an HTTPS secure binding on your server, then it will listen on TCP port 443 by default.

2) The other thing to think about is to make sure that your client devices as well as your web server is configured to only accept specific types of encryption ciphers.
   Because SSL or Secure Sockets Layer is deprecated, that should never be used, and

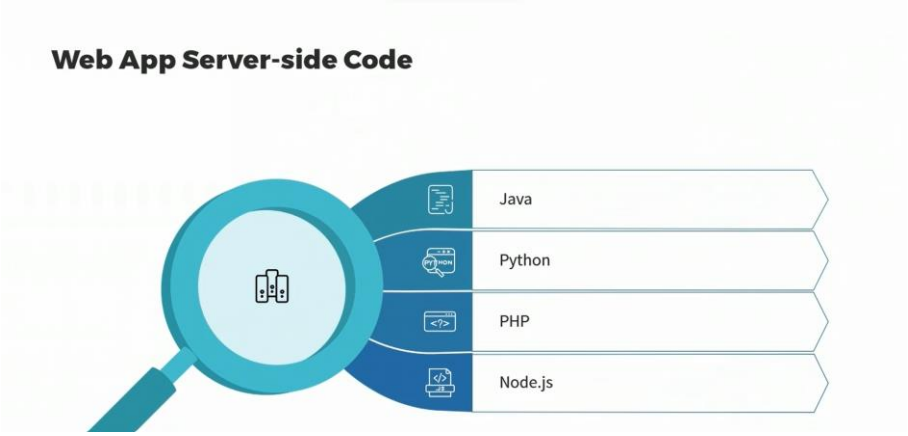
Specifically, TLS 1.2 or higher, Transport Layer Security is what should be enabled on your web server stack and ultimately on your clients as well. And once you've installed a web server, you can run a web application simply by installing a couple of files in the web server root. Even just a single file in HTML web page, if that's the nature of your app, not every web application is designed to have a user interface that the user interacts with.

3) Application Container:

But besides that, your web application might also be contained within an application container.

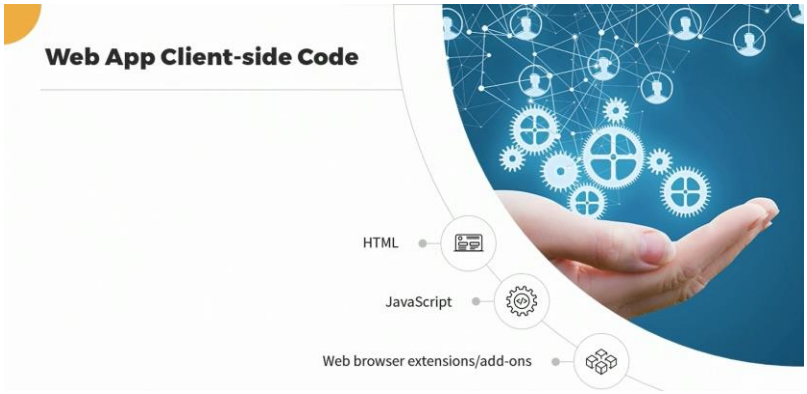
a) An application container just provides a boundary, a logical boundary around application components like all the files that comprise a web app.
b) An application container itself can also have the HTTP web server stack.
c) The great thing about application containers is they do not contain an operating system like a virtual machine does. But they're still portable, you can move application containers between different hosts that have an engine that is capable of running containerized apps like the Docker engine that you might install on Windows or Linux.

And a web app might consist of a multitude of containers working together to comprise a single overall larger application, whether dealing with the containerized web app or not.



Server-Side Code:

- ✓ There is the concept of server-side code that runs on the web server itself and this comes in the form of languages like Java, Python, PHP, Node.js, just to name a few.
- ✓ This means that we write code that will execute from a server perspective on the server device itself, in the server's memory. And often this is what happens to fetch dynamic information like the contents of an inbox to then present it to the client in HTML format.

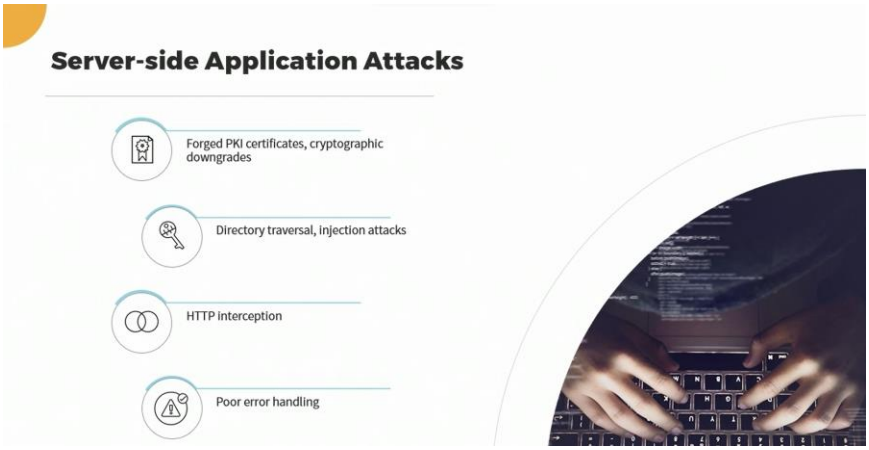


Client-Side Code:

So, then on the client-side code, we have HTML, Hypertext Markup language which is really a descriptive language that's used to format output returned from the web server.

So, you might have server-side code written in Java that generates HTML that's sent back to the client and the client renders that HTML and displays it appropriately.

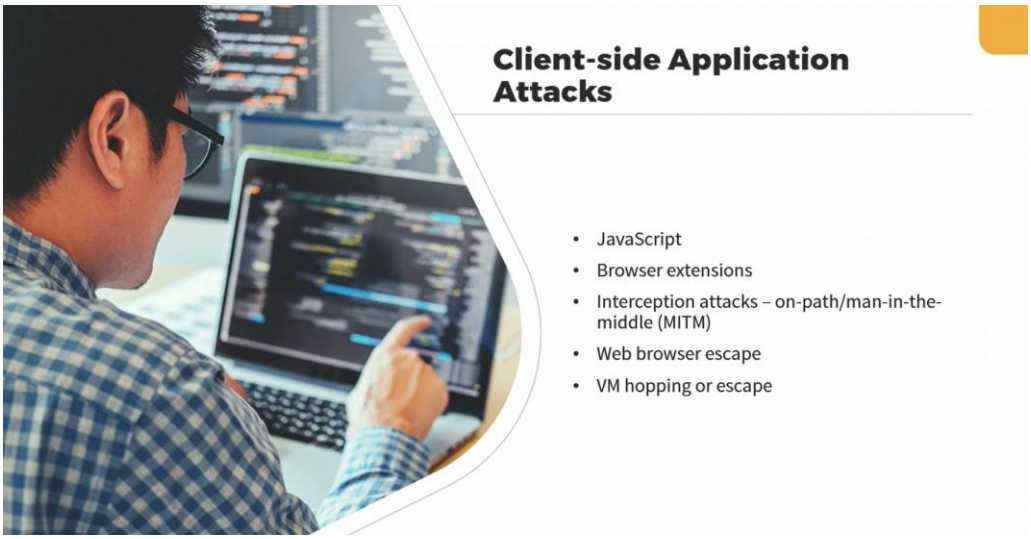
You can also have JavaScript code running on the client. JavaScript gives you many more options than you would find available with HTML. For instance, you might use JavaScript to manipulate cookies. You could also have web browser extensions and add-ons that give additional functionality like the ability to edit a PDF directly in the browser, so modern web applications might be a combination of both server-side and client-side code that's actually quite common. But as a result, we have to think about potential security issues.



PKI Certificate & Server-Side Attacks:

- **On the server-side of things, application attacks could be related to things like forged PKI certificates.**
- Remember the **PKI certificate is required server-side if you want to enable an HTTPS binding**. So, if the server is somehow compromised, malicious actors might install a forged PKI certificate. There are many attacks that can also negotiate with the server to downgrade the cryptographic ciphers that are used, such as using **SSL version 3 instead of using TLS 1.2**.
- Directory Traversal Attack:
  Other server-side attacks would include things like directory traversal attacks, where if things aren't checked properly in the web application code, a malicious actor might modify the URL sent to the server to go back through the file system, such as to another subdirectory that isn't supposed to be exposed to the web app.

- Database Injection Attack:
  Then there are injection attacks such as database injection attacks where we have improperly validated fields that don't check for what's being supplied. If we have a field that allows searching for client IDs in the database, then we need to check that only a client ID is entered in and not executable type of code that might change the result of the query set.
- HTTP Interception:
  Other types of server-side app attacks include HTTP Interception, where malicious actor could capture and modify HTTP requests and responses, essentially change something in those transmissions before it gets to a server.
- Poor Error Handling:
  And there's also poor error handling, which is really result of not adhering to secure coding guidelines, where it might disclose sensitive information, maybe a failed authentication to a web app shows a page with the SQL query that was actually used to attempt to authenticate that user.



Client-side application attacks are many, but they can come in the form of JavaScript attacks that use JavaScript that runs on the client browser locally to do things like manipulate cookies. It could be a malicious browser extension that the user might have installed, either intentionally or maybe got installed covertly without the user's knowledge.

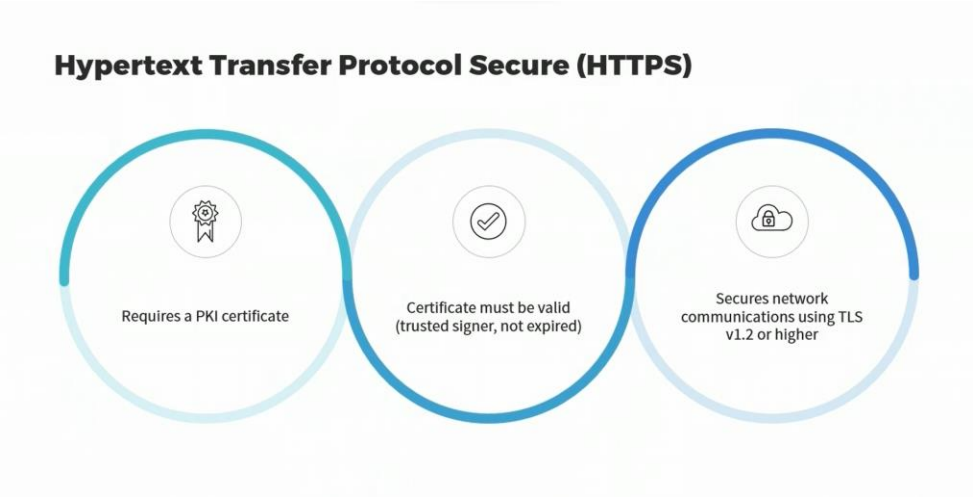
Again, interception attacks which are also called on-path and man-in-the middle or MITM attacks, where for malicious actor can get themselves on a network between the communication path of a client on the web server, depending on the nature of the traffic, they might be able to intercept it, make a change to it and send it back on its way to the client or to the server. Then there's web browser and VM hopping or escape techniques.

Modern Web Browser and Sandbox:

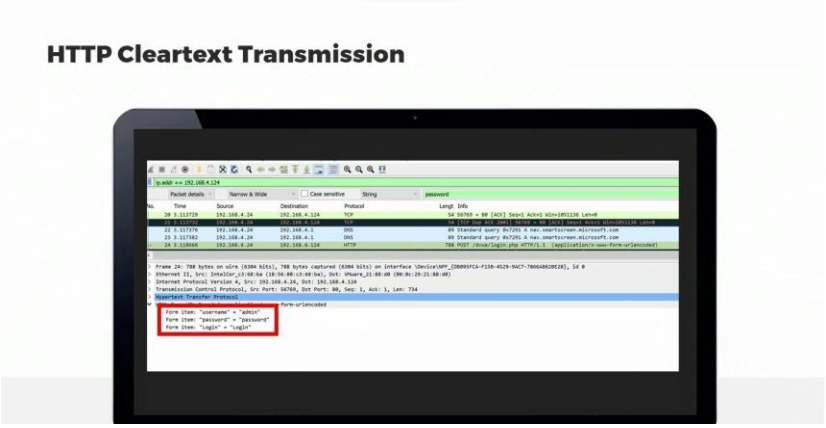
Modern web browsers are sandboxed, that means that any code running in a web page is only allowed to run in that web browser tab. In a web browser, you can open multiple tabs and usually each of those tabs is sandboxed from others. So, you don't have malicious code that can reach into another web browser tab to see what's going on, and certainly not reach into the operating system.

That doesn't mean it's impossible, there are some known attacks that have occurred when that can be done. Same with virtual machines, if we have a compromised virtual machine, is it possible for an attacker

to back out of that VM and get into the underlying hypervisor running the VM? Anything is possible, we just have to make sure we keep things as hardened and up to date as possible when it comes to patches. So, securing the communication channel between a client and a web server can be done with HTTPS, we've talked about this. So, it means that we have a PKI, a public key infrastructure certificate on the server.



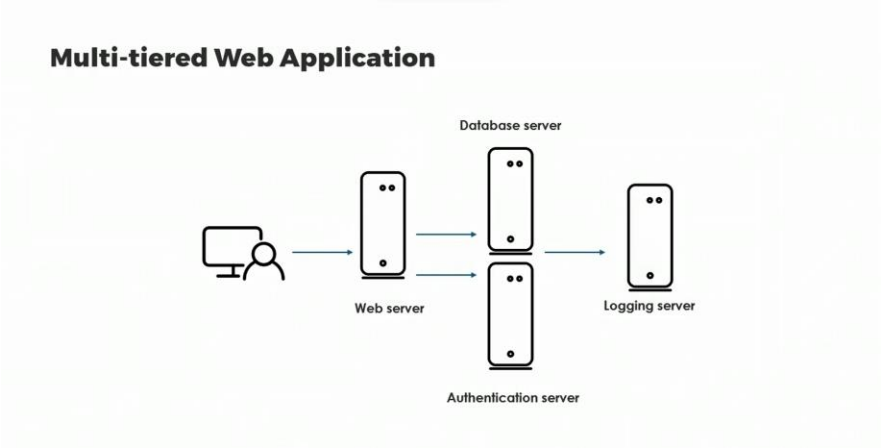
Now that certificate needs to be valid. In other words, it needs to have a trusted signer or a trusted issuer in order for clients to trust it, and the certificate can't be expired. Certificates have a validation period, certificates have an expiration date, kind of like a passport or a driver's license, where after that, they are no longer trusted. Until they get renewed ahead of time before they expire, or if a new certificate is issued after they expire. So, HTTPS then is used to secure network communications, ideally using TLS v1.2 or better. Now, if you don't use HTTPS, that means you can end up with HTTP clear text transmissions.



In the screenshot, we have the Wireshark free tool that is captured network traffic and what's been highlighted here at the bottom is the username, password, and Login information supplied from a web form. If nothing is encrypted, then it's possible to capture that traffic and retrieve it and see it in plain text.

[Video description begins] A slide appears with the heading HTTP Cleartext Transmission. It contains a screenshot of HTTP screen. It contains a toolbar , with various icons on the top. It further contains the

following tabs: Packet details, Narrow & Wide, Case sensitive, String, and password. It further contains a table with the following column headers: S.No, Time, Source, Destination, Protocol, Length, and Info. Below, it contains the output pane with various code. The following lines are highlighted. Line 1 reads: Form item: "username"="admin". Line 2 reads: Form item:"password"="password". Line 3 reads: Form item:"Login"="Login" [Video description ends]



Multi-Tired Web Application:

Another notion to bear in mind about web application development is having a multi-tiered web app, instead of everything running on one server, you might split it up. For example, we might have a front-end web server that has the files required for the web server front-end for the UI, the user interface that the GUI interacts with and depending on what actions are taken in the web app, it might talk to a back-end database server or an authentication server and ultimately to a logging server to record web application activity. These could be physical servers, virtual machines, or any combination thereof. All of this might also be behind the load balancer, which takes the client requests and routes it to a pool of servers, so you would have multiple servers running all of these things to increase availability and performance.