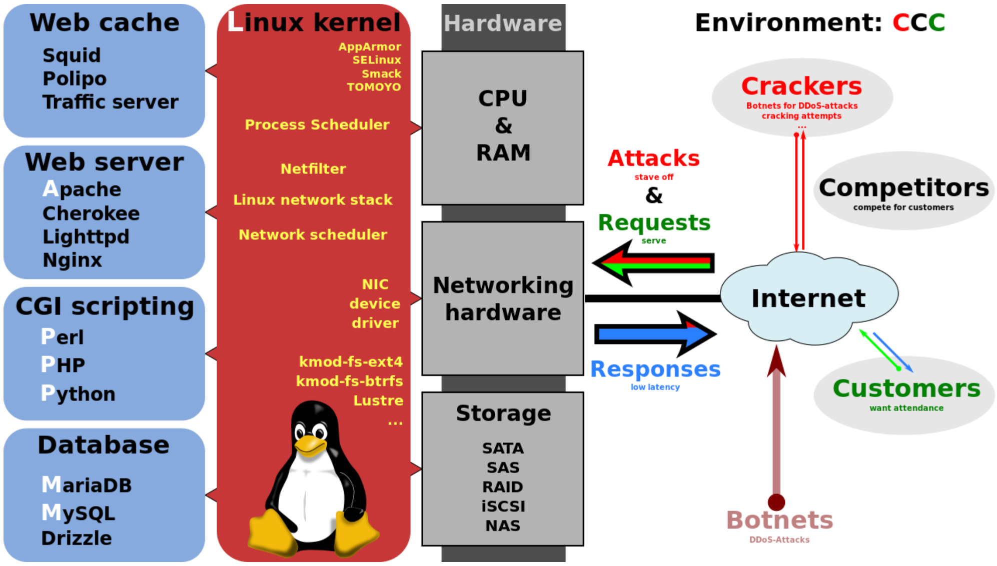
13.1

1. Discuss the plugins. This course has us use Cookie Manager and Tamper Data. The idea here is that these are not the only web tools available. Talk about firebug, burp suite, google developers console, and even scripting interaction (python via pwntools, scapy or bs2). Try to give some context on when you would use what tool and that it is also user preference. Ask the students what they use when and why

13.2

2. XAMMP stack. What is this stack? In more general, what is the common LAMP stack?



https://en.wikipedia.org/wiki/LAMP_(software_bundle)#/media/File:LAMP_software_bundle.svg

Quickly make sure everyone understand the different pieces that generally make up a web server (that it is more than just apache HTML/CSS) and, because it is more, has additional attack vectors.

3. Discuss the javascript language. Ask who has coded in Javascript before? Make sure everyone understands java/javascript not the same. Quickly discuss that javascript (and loading of a website in general), is single threaded but asynchronous. This causes issues when doing more complicated things because you have to be careful of order (i.e. in normal code you call a function, save the output to a variable and then use that variable. In javascript, it might still be working on the function you called and move on so it tries to use your variable before its the what you expect it would be because the first function has not yet finished. (http://blog.thefirehoseproject.com/posts/why-is-javascript-so-hard-to-learn/ #4 does a good job explaining this issue). You will not write complicated javascript in here, but you need to be able to debug it. Talk about alert() verse console.log() and and how to view the console in your browser (i.e. Chrome developers console).

4. Introduction into XSS. Ask to example what a XSS attack is? What is the vulnerability? How do you fix this issue (input sanitization, how you use user supplied fields). With a XSS, who are we attacking? The server or the client (person going to the server). This is an important distinction.

5. Talk about cookies. How do cookies work? What are the required fields? Is a cookie has domain="example.com" can you read that cookie from www.example.com ? What about domain=".example.com"? (both cookies can be read from www.example.com). How do you set a cookie to a specific domain (do not specify a domain and it will be only valid from the domain that issued it exactly. What does the Expire="" field do? What about max-age=? (difference is one is a fix time, one is seconds from when received cookie). What do the Secure and HttpOnly attributes do?

The `Secure` *attribute is meant to keep cookie communication limited to encrypted transmission, directing browsers to use cookies only via secure/encrypted connections. However, if a web server sets a cookie with a secure attribute from a non-secure connection, the cookie can still be intercepted when it is sent to the user by man-in-the-middle attacks. Therefore, for maximum security, cookies with the Secure attribute should only be set over a secure connection.*

The `HttpOnly` *attribute directs browsers not to expose cookies through channels other than HTTP (and HTTPS) requests. This means that the cookie cannot be accessed via client-side scripting languages (notably JavaScript), and therefore cannot be stolen easily via cross-site scripting (a pervasive attack technique).*

5. More about cookies. Other than the required name/value and the field discussed above, what other fields can a cookie have? Is there a limit to these fields (no, can have as many key=value pairs as you would like)? What are authentication cookies? Is this authentication the same for all sites? Does a site use only one authentication cookie? Two? Can these be encoded?

6. Lastly, talk about cookie context. If I go to example.com, will I only get cookies from example.com (no). How do I get cookies from other sites? This discussion should lead to a discussion on iframes and document domains. Iframes are essentially separate, individual webpages that are loaded and embedded within the page you're currently looking at. So you will get many cookies from a single page, but you are really loading a bunch of minipages. Your facebook widget inside an iframe will have access to your facebook cookie, but it will not be able to access the example.com cookie. All subpages do get some hierarchical information so facebook.com is aware that it was called in a iframe from example.com which is part of what allows cookies to track you as you move around the internet.

7. Talk about sessions. How long is a cookie valid? When you go to amazon after not being there a while, does it prompt you right away to log in (no - you can still add stuff to your cart / wish list). How about when you finally want to check-out (yes). What if you open two windows and specifically logout on one? If you log out with Admin from the example from PWK after it sends the cookie, will that session cookie still be valid? The main idea here is the timing also matters.

13.3

8. Need to have a quick discussion on GET vs POST. Talk about www.example.com?name=Test&comment=Where .What is the ? what does the & do? How do you send characters (i.e. can you just put a %?)

9. Reinforce the difference between an RFI and LFI vulnerability (LFI subclass of RFI, RFI means can include remote whereas LFI mean can include only additional local code so you need to work with what is on the box).

10. Talk about PHP. Can you see the source for PHP by going to via source in a browser? What is displayed? How about with html? How about with javascript?How do you output what is seen by client when visiting your page with php?

11. What does the linux command tail do? head? The -f option?

12. Instead of appending the null byte %00, what else could you have done in the example (simply renamed to evil.txt.php).

13. What do all the ../../../ do? What happens if you use too many ../? What are some techniques you can use to figure out where you are in the directory structure to determine the path to the correct file?