WEB SEC

:3:3

What is this web sec

In network security, we cared mostly about acting, well, in a network.

For web sec, we have none of that, instead, we try and find exploits in how data is received and served on a web server.

PHP: The root of all EVIL

As far as i’m concerned, PHP errors are what cause most of the web vulnerabilities which we will be going through in this lecture.

What is PHP?

PHP is a language which tells the server how to handle certain requests. For example, you might want the server to look for a file and evaluate input with the contents, such as for a login or for a search. The problem with that is that it used wrong, PHP can make us BIG SAD.

CGI: the Common Gateway to EVIL

CGI is the Common Gateway Interface. Have you ever seen a website with a file extension .cgi? Welp, its basically the same thing as PHP. The difference is that it allows you to use any language for server side programming and provides it input through stdin and output through stdout. It’s not very common in websites anymore, but if you look at your router configuration page, especially if it’s a not-so-good-router, you might find these pages.

Why do these languages cause problems?

They dont, if used correctly.

Have any of you ever done a print(“My name is “ + name); ?

That’s what causes vulnerabilities.

If your output of a php script prints html that looks like:

“<b>found “ + $num + “ search results for “ + $term + “:</b>”

If $term isn’t sanitized, and someone puts say <img>, the html sent back is:

“<b>found “ + $num + “ search results for <img>:</b>” which is bad for obvious reasons.

Code Execution

Furthermore, on some older websites, or IoT pages (such as usually if your router has a ping function), your server is taking your input and feeding it into a “system(‘ping ‘ + ip);”

Why is this bad news?

Again, if you know basic bash, you can probably spot that you can insert a “ && reboot” to reboot the machine

Usually you find these vulnerabilities through reverse engineering firmware for IoT devices (a later lecture)

XSS

XSS stands for cross site scripting. It comes in Stored XSS, Reflected XSS and others which I won’t talk about.

Stored XSS: The server saves your malicious input such as with a username and anyone who views the page gets the payload

Reflected XSS: The server has malicious input in the URL or when you enter it in a field and if you request the website with it without knowing, you get ker’pranked.

Why do you care about XSS?

Stored XSS can be used to deface websites...

In the web, to secure information we have what's called the Same Origin Policy.

What's that you ask? The Same Origin Policy says that you can only get information from the website you're on (basically).

If I have text that is on google.com, I can't see it from twitter.com.

If you find XSS, no need to worry, anything you request will be from the origin you wanted.

Same Origin Policy part 2

Along with xss allowing you to edit and view stuff you shouldn't on a different origin, you can do what's called “cross site request forgery” which let's you call functions on other websites.

It relies upon the fact that some web servers listen for requests such as website.com/buy?item= and etc.

You can do this on embedded devices sometimes to compromise them from a website someone might visit.

Tool: Burpsuite

LETS GO FOR A FIELD TRIP!

ANOTHER FIELD TRIP: DVWA

NOTHING