Lab 05 - Client-side Controls

20 points

Client-side controls make the user experience (UX) on a website much more enjoyable, but provide no real security. Security checks need to be accomplished on the server, regardless what is done on the client side. For this lab, complete the following tasks in **Lab 3. Bypassing client-side controls** on Pablo:

1. **[10 points]** Complete **all** sub-tasks **a - g**.
2. **[5 points]** Complete sub-tasks **h - i**.
3. **[5 points]** Complete item 3

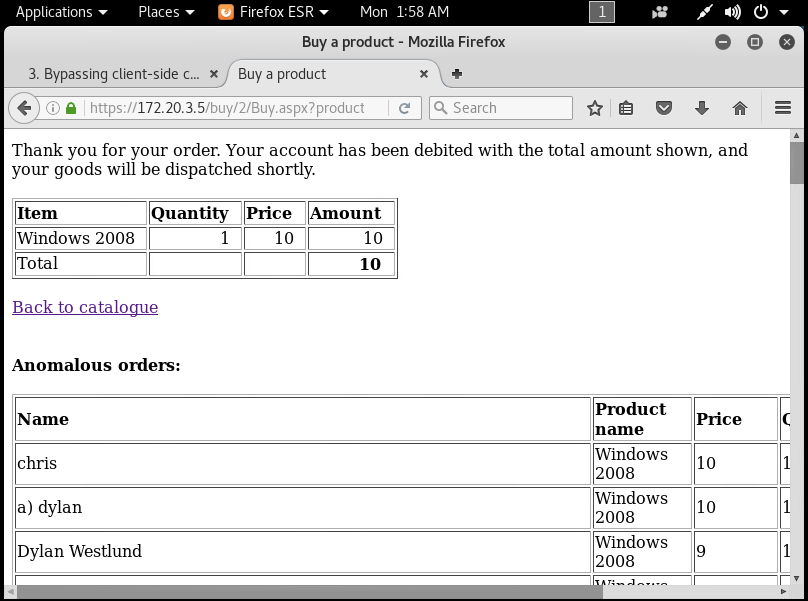
These can be challenging so do your best, if you get stuck let me know. On page two are some hints in case you get stuck!

**Deliverable:**

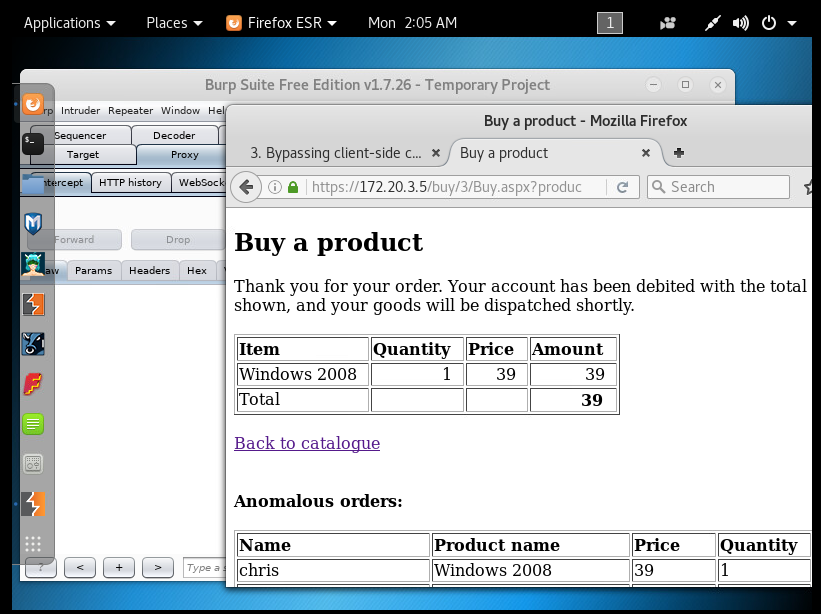
Turn in a Word Document or PDF that contains screenshots for each task as well as an explanation of what you are doing.

**Hints**

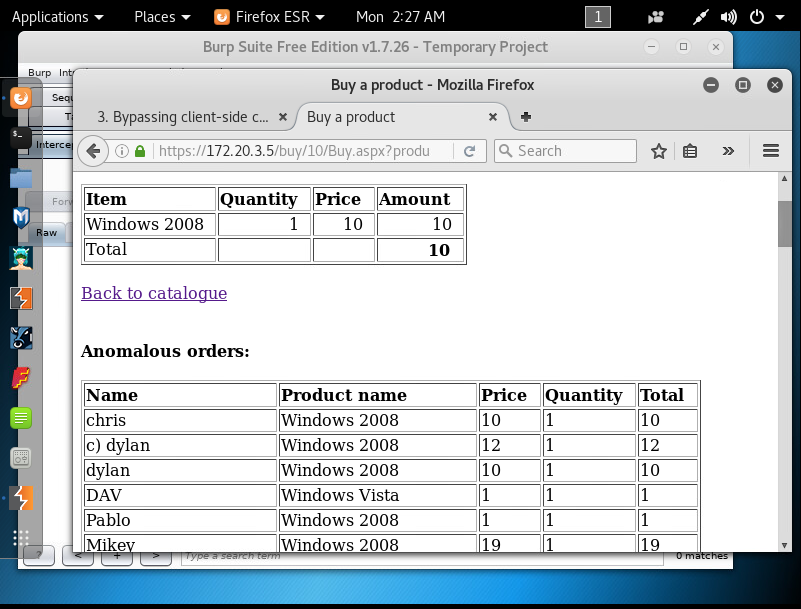
1. Purchase for a different price
   1. Easy

  
<https://gyazo.com/84ed0a467c94a0e266878ebfa41a7912>

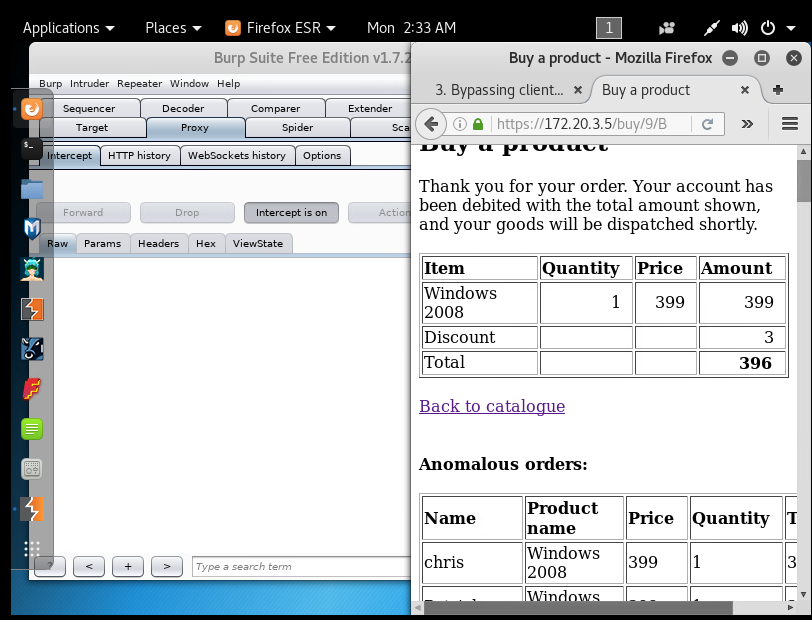
* 1. Uses encoding (ASCII Hex)

  
<https://gyazo.com/9593e541c39a9428bccae26c3386b2ce>

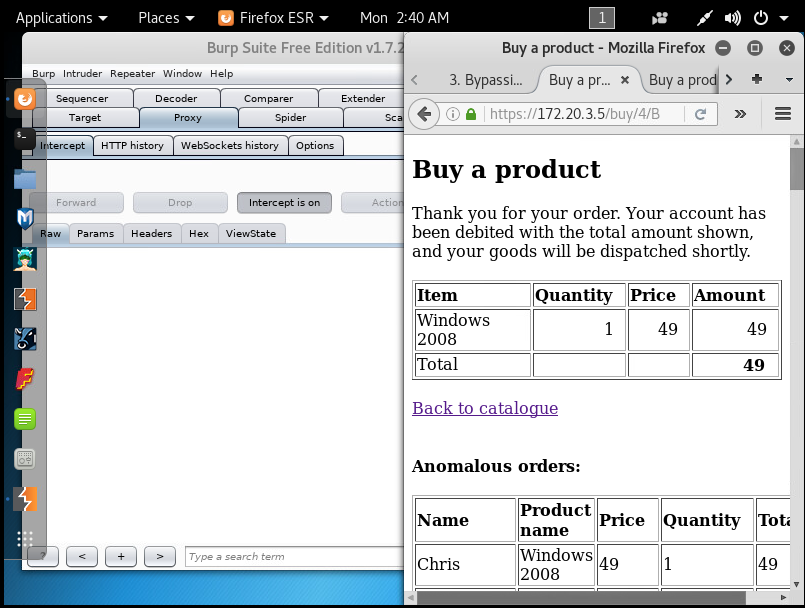
* 1. Modify HTML element attributes

  
<https://gyazo.com/6294e7ad41f3f123e96c65319df65eb9>

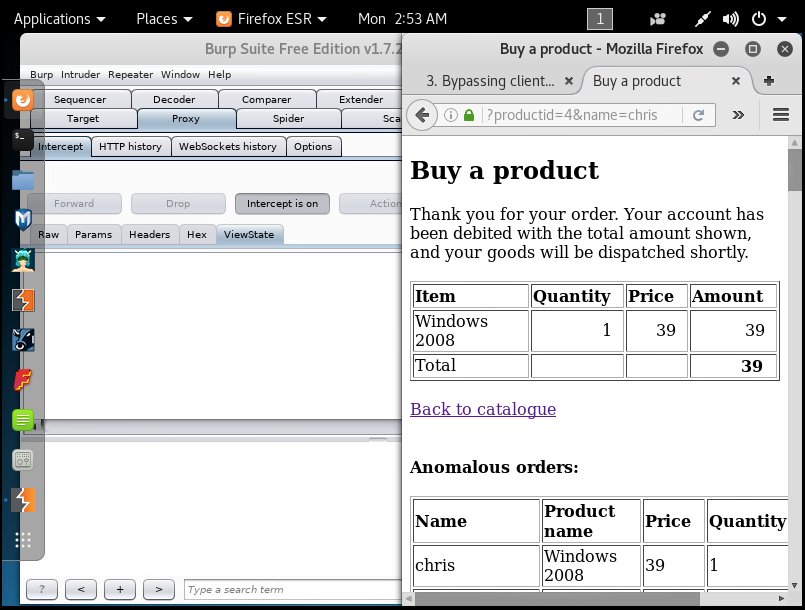
* 1. Cookies!

  
<https://gyazo.com/ad8bfcb168abd6e924884a2aa7cccfc0>

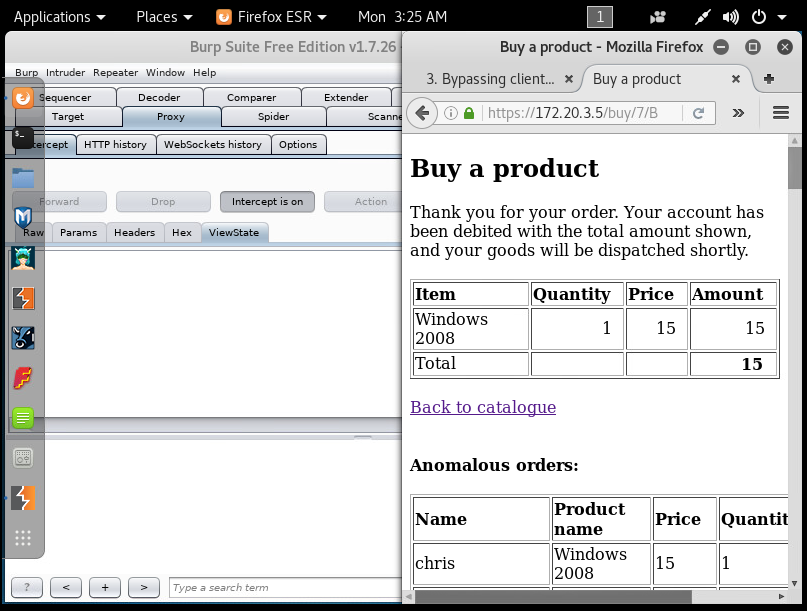
* 1. Look at chkprice, don’t try to brute-force. Where could you find another value to use?

  
<https://gyazo.com/aec2c1160e86de97abb0fc90d748bce5>

* 1. You need to attack the view state

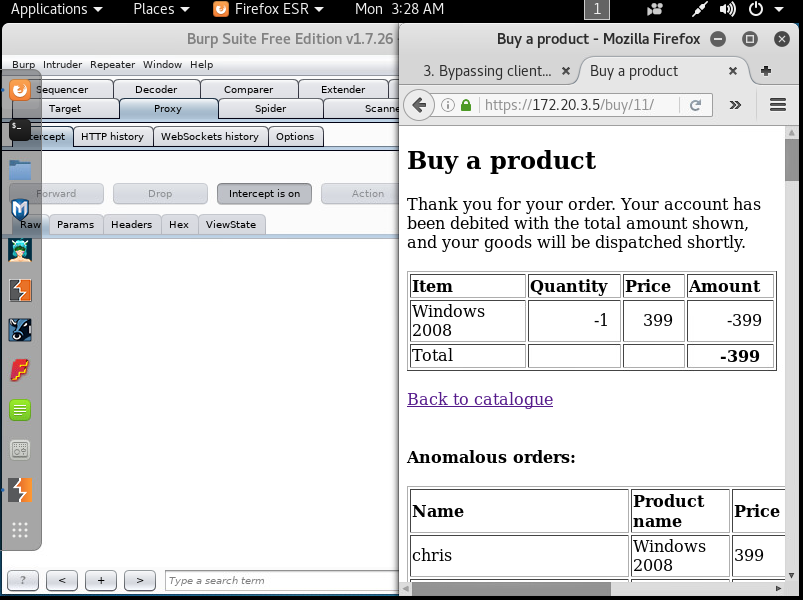
  
<https://gyazo.com/a06a2528e54a813262e7f020269235ff>

* 1. You need to attack the view state, similar to f.

  
<https://gyazo.com/72032ff3e5bbce3b6ab17b875ed6cb39>

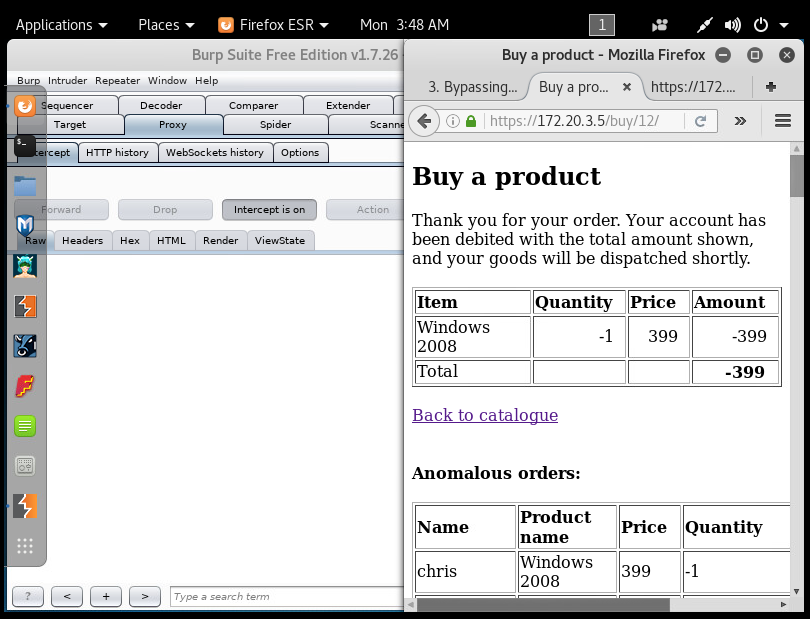
1. Price

h. Bypass Javascript

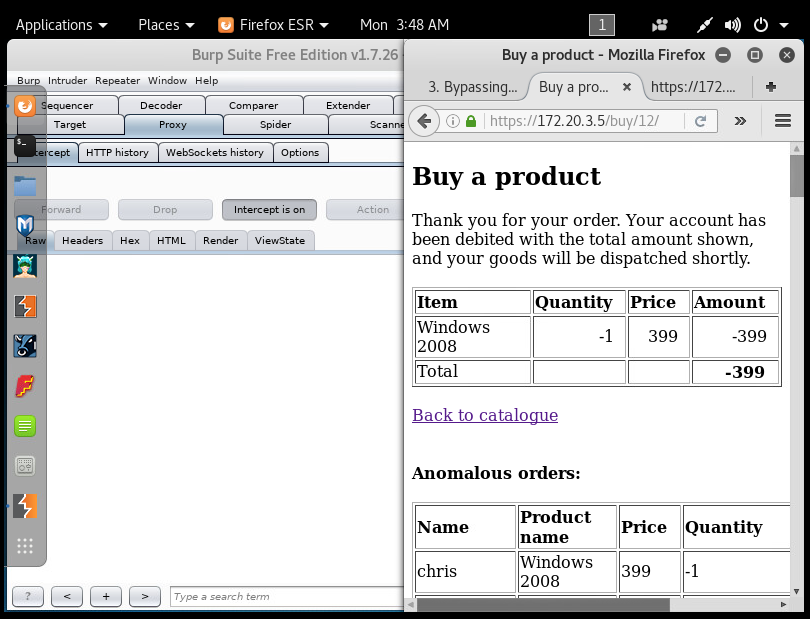
  
<https://gyazo.com/009d58e692319ad77a1ca1d53358cdf9>

i. This uses obfuscation, look at the function (in javascript) that is called before the

form is posted, how could you use this to generate arbitrary values?

  
<https://gyazo.com/e4e0ead30114ebca04055bff30029a5d>

For getting payed to pirate:



On both of the JS validatioins (h & i) I get a default off 0 when a would-be-overflow gets submitted. I am getting the same behavior even when editing the header itself on H (meaning that this validation is performed on the server side?)

Java had similar results.

my device doesn’t have Silverlight installed (it’s a linux right, and Silverlight is for Microsoft, anyway I couldn’t pull up the page to look for an install, I think I isolated my machine’s network capabilities at some point so I wasn’t accidentally messing with networks out of scope.)

and it appears that flash isn’t installed either.

The book refers to serialized data, which I can’t say I recognize, and mentions potentially walking through a debugger or decompiling the bytecode, I’m not sure how to go about identifying the bytecode, or locating it (beyond what we’ve done so far with burp, maybe I just don’t recognize it?)

Anyway. I’m unsure what I’m looking for.

From the video I got the impression it would be doable with the JS validations (h & i), but I just don’t see it.