### **Question 1**

Prompt: What is the coolest animal no longer on planet earth? To solve this puzzle, you'll need to start your journey at the following URL: <https://web.cs.wpi.edu/~rjwalls/start-here/>. Important: Do not brute force the server! Brute-forcing is not required and it might get us all in trouble. Your video should describe the process required to solve the puzzle.

Whats the coolest animal that is no longer on planet earth? It’s none other than the infamous Sloth Astronaut. To solve this puzzle, I used a combination of a user's password reuse, and eavesdropping with Wireshark to exploit the HTTP Basic Authentication flaw of transmitting packet information in cleartext.

For step one, I was pretty set on the password being passw0rd because it’s one I found during an earlier assignment. Since we talked about the dangers of password reuse in an earlier class, I thought this would be a clever way to demonstrate that flaw to us. I tried a number of different usernames but eventually found the correct one which was robert with the password as passw0rd.

After succesfully making it past step 1, I followed the directions and downloaded the packet capture file. I then opened it wireshark and analyzed it for where to go next. In the server reponses, it’s was clear that the next subdomain to visit was not-so-secret.

By expanding the HTML Tab in Wireshark, it allows us to see the full HTML header sent by the server, and because of the flaws in the HTTP Basic Encoding, the account information is sent over plaintext using just base64 encoding. With wireshark able to automitcally decode it, I was able to find the next pair of credentials to log in, which was username bro, password let me in.

Now knowing the subdomain to visit, I just needed to find the credientials. This is were the HTTP Basic Authentication flaw came into play. Inside wireshark we’re able to analyze the packets HTTP header. Knowing the credentials in the response are transmitted over cleartext with base64 encodin. Wireshark is easily able to decode the credentials and display them. Which were username bro, and password let me in.

Upon entering the decoded credentials, I was able to successfully log in and complete the challenge. .

### **Question 2**

Prompt: What is cross-site scripting, i.e., XSS attacks? If possible, demonstrate an example of how such an attack can be used to steal authentication cookies. I do not have any good reference materials for this one, so you'll have to find those materials yourself. Please make sure to cite your references.

SOURCE: <https://medium.com/@laur.telliskivi/pentesting-basics-cookie-grabber-xss-8b672e4738b2>

https://portswigger.net/web-security/cross-site-scripting

*Cross-Site Scripting attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side javascript, to a different end user*

Criss Site Scripting is a hacking technique where an attacker attempts to inject malicious javascript code into an otherwise trusted website that is commonly visited by other users.

Once the code is injected into the website, and a new user visits the web page, the injected code executes silently on their browser. Using this injection technique, an attacker can gain access to sensitive information stored in a user's browser, such as their cookies.

For the purpose of this video, I’m going to demonstrate how an attacker could steal authentication cookies using a Stored XSS attack.

Cookies are important target to attackers because if succesfully captured, it contains information that allows the attacker to impersonate their vitcim at the website they visited, even without knowing the vitcims passwords.

Which is why a Stored XSS attack is a common way hackers try to steal cookies. They do so by first finding a vulnerable website. They creating a web server to catch and store stolen cookies, which are forwarded to them by the malicious javascript code to inject into the vulnerable website.