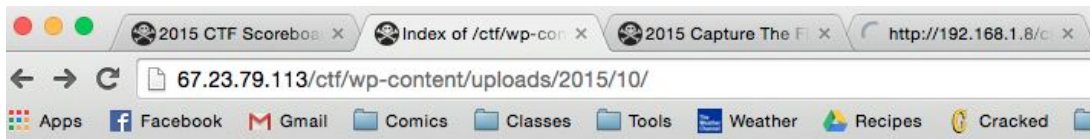Colin Hamilton, Sam Heilbron, James McCants, Nick Sempere
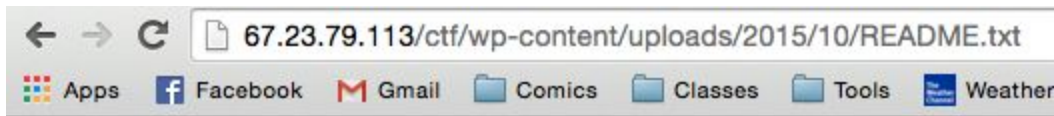Team 11

Capture the Flag 2015

Flag 1

The idea for the first exploit came from the banner at the bottom of the CTF index page: "Proudly powered by WordPress." Knowing that WordPress websites have common directories, we sought to see if we could view and traverse them. A search for WordPress directories prompted us to look for wp-admin, wp-includes, and wp-content directories. wp-content was not itself accessible (it directed to a blank page), but a further search for wp-content/uploads proved fruitful.

From there we were able to navigate the website's structure to our heart's content, and found the key living in a file helpfully named README.txt.

Flag 2

We were given the file containing this flag, but we had to figure out how to use it. It was labeled an exe file, but 1) we were wary of running an untrusted program on our machines, and 2) we had Macs and so couldn't run it anyways. Instead, we tried looking at the file as plain text to see if we could discern any meaning. It was not a text file, but we were able to see some readable text within it by searching for the string "key." It said, "Watch the video. The key is the SHA1 sum of the number, as a word in all caps, in the video."

Based on the plain text and the reference to a video, we were skeptical that this was an exe file, so we ran the file command on it -- with the response "tcpdump capture file."

From there, we realized we could examine the file in WireShark, and see that it documented an HTTP video transfer. Using WireShark's "export objects" utility, we got the video, and watched it to get the key: the SHA1 sum of the string "SEVEN".
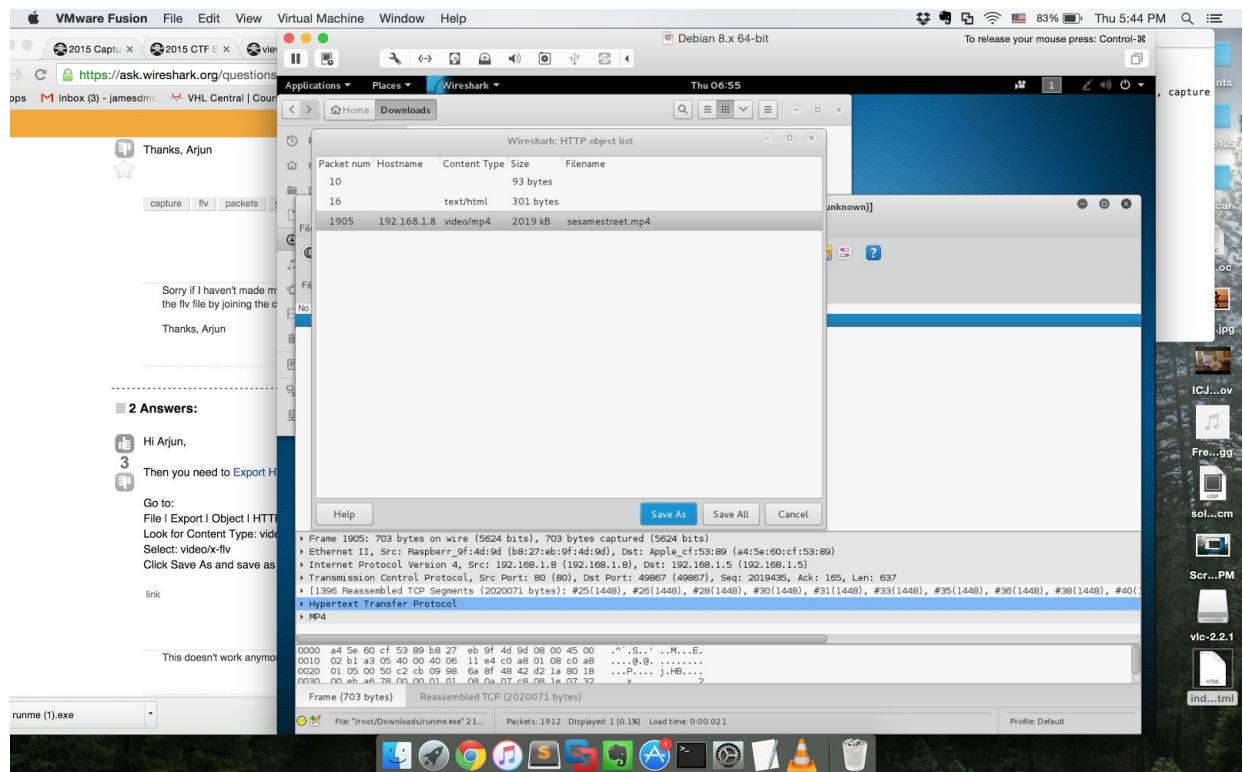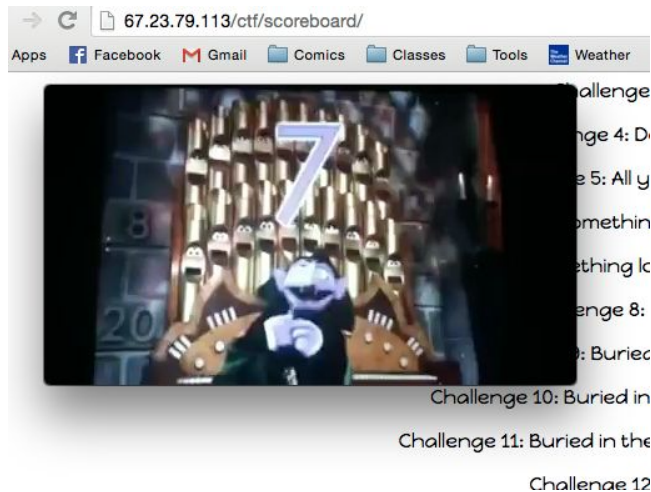
Flag 3

The third exploit came from an attempt to fool the server into letting us log in. For this we utilized Burp Suite. Burp was able to generate a sitemap, from which we saw the page "admin.php." We were greeted by a simple login tool, which sent a post request that Burp's proxy was able to intercept. Under the assumption that the request would be sent to a SQL database of users, we injected SQL code into the parameter of the username: user' or '1' = '1.

As this always evaluates to true, the server let us through -- we knew it succeeded because, while failed login attempts would normally redirect back to the login page, we were sent instead to a new page called "main.php." The page appeared to be a 404 page, but an inspection of the HTML revealed the flag in a comment.
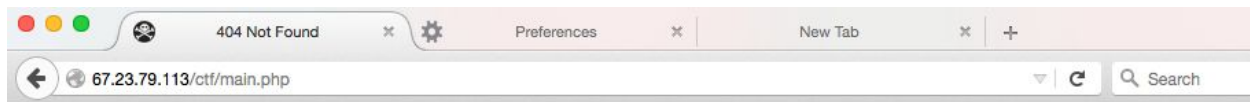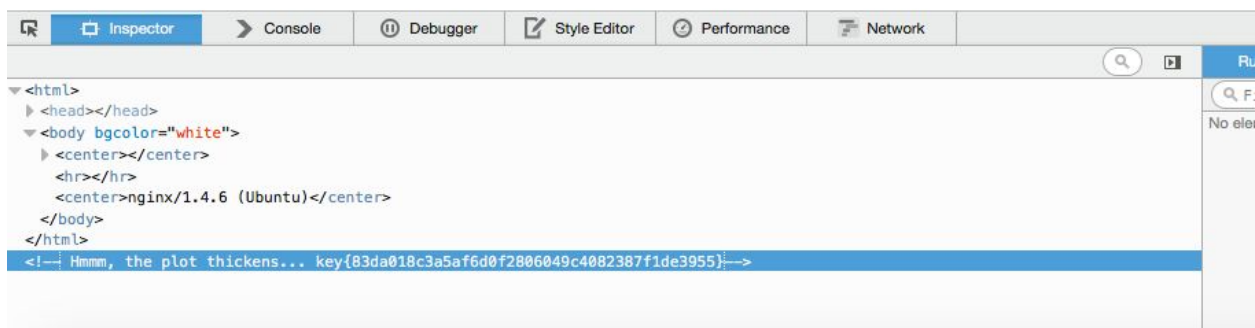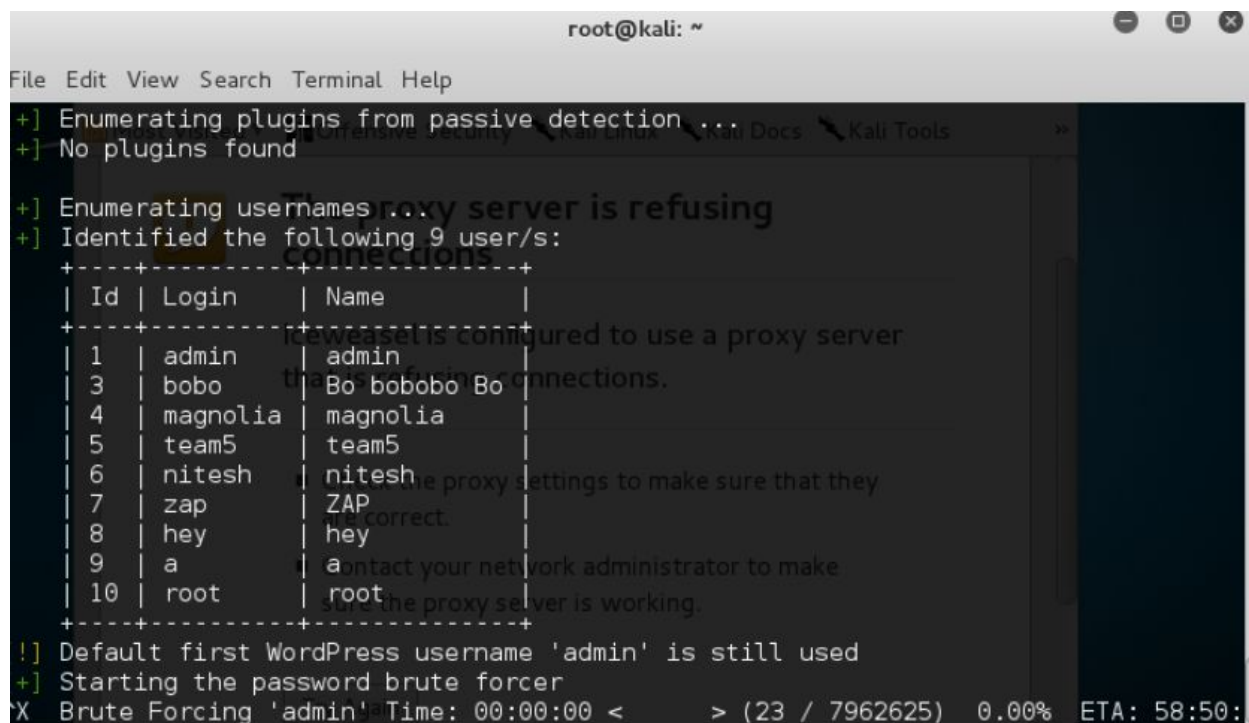
Failed Attempts

Failed Attempt 1

Once we had established that the website was built from WordPress, we attempted to gain administrator access (wp-admin). For all WordPress sites, "admin" is the default username, so first we confirmed that the root user hadn't removed this login. By submitting a random password with the username "admin", and receiving the error message that the password for "admin" was incorrect, we were able to confirm that "admin" was still a valid user. However, it made sense that the root user would have created other profiles with various administrator rights. Therefore a simple wpscan on the login url returned all valid usernames for the website. By appending a wordlist to the command, a brute force attack on each username was initiated. The command was:

The result was:

Since there was a flag hint about "bobo" we attempted to gain access to the user "bobo". Unfortunately, the wordlist was large enough that we ran out of time to iterate through the entire list on that user. However, we were confident that gaining access to that user would have resulted in finding a flag.

Failed Attempt 2

Of course, the WordPress login was not the only potential exploit that required authentication bypass; we also attempted to crack the target IP's SSH password and gain entry as root. We chose a tool called Hydra, a brute-force cracker designed for use against remote authentication services. By specifying the port number (22), a wordlist (the Metasploit framework includes several brilliant ones), and the target's IP address, we commenced the attack. Unfortunately, it too was a very time-consuming process and did not produce any successes before the deadline.

Failed Attempt 3

A third attempt came on the message board. Along with many other groups, we attempted to inject code into the server through post submission, without success. We also tried to modify the URL query parameter to inject SQL code -- using id=1 or 1=1. This actually

somewhat worked, in that it revealed all posts made on the message board. However, it did not reveal any keys.