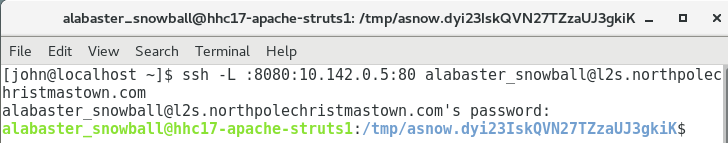
# Elf Web Access--Encryption Gone Wrong Part 2, Reconnaissance

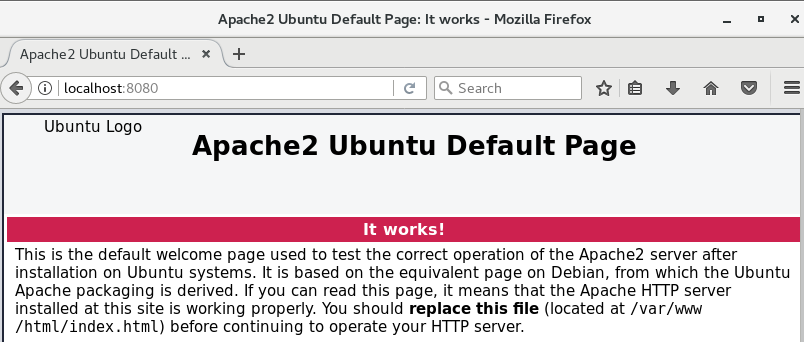
In Part 1, you should have connected to the Elf Web Access server. As always, there are several ways to do it.

## Local SSH Tunnel--Linux

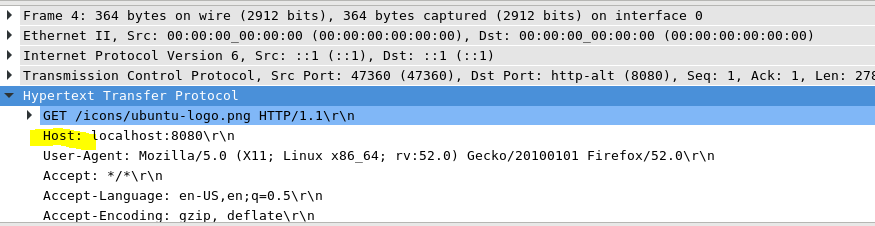
This is almost identical to the way you connected to the SMB server. Nmap scans from the l2s server during the SMB challenge told us that there is a server at 10.142.0.5 named mail or ewa.northpolechristmastown.com, and it listens on ports 22 (SSH), 25 (SMTP), 80 (HTTP), 143 (IMAP mail client), and 2525 (?). Any of the ports could be useful. If Elf Web Access is like Outlook Web Access it works on web ports so let’s try port 80 first. We can make our end of the SSH tunnel listen on port 8080 instead of 80, just so that there aren’t conflicts with Internet sites.  
ssh -L :8080:10.142.0.5:80 [alabaster\_snowball@l2s.northpolechristmastown.com](mailto:alabaster_snowball@l2s.northpolechristmastown.com)



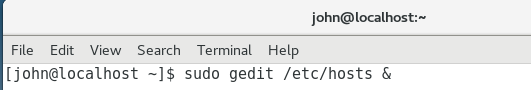
Now that the tunnel is established, connect to the mail server. We will connect to 127.0.0.1 or localhost because that is where our end of the tunnel is. We put :8080 after the site name to tell the browser to use port 8080.

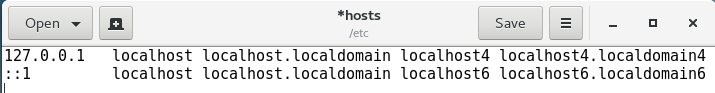


Hmm, that doesn’t look like an EWA server. If we ran a packet capture, we would find that the browser put localhost in the HOST field of the HTTP Header.

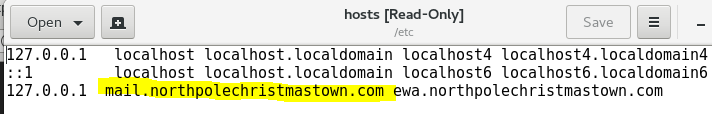


We need to change our hosts file, so we can use the DNS name.

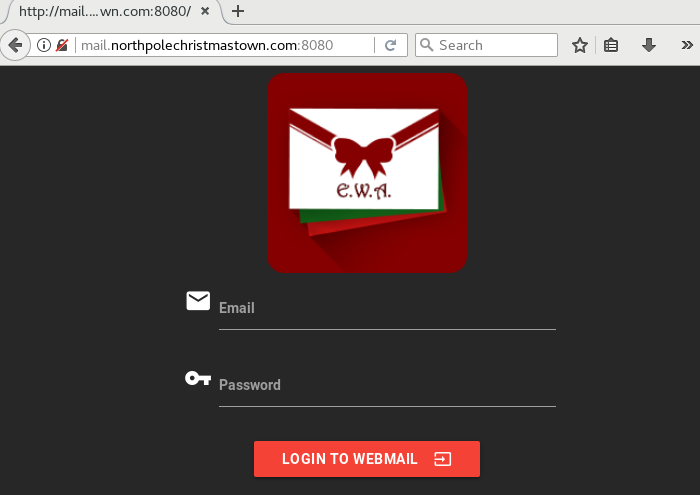




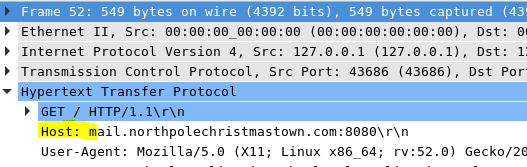
Adding mail.northpolechristmastown.com…



When we connect to the mail server, this happens:

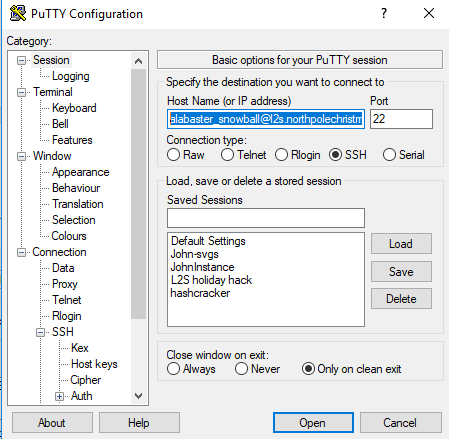


WooHoo! Notice that the Host header has changed, and the web site likes it.

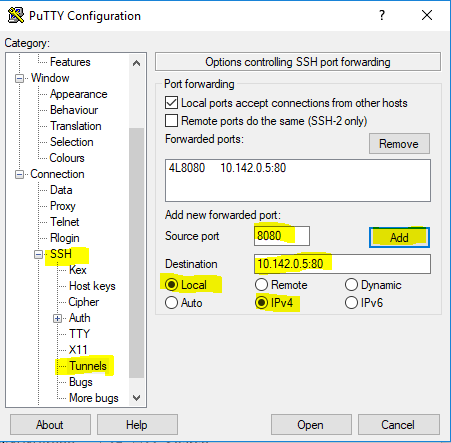


## Local SSH Tunnel--Windows and PuTTY

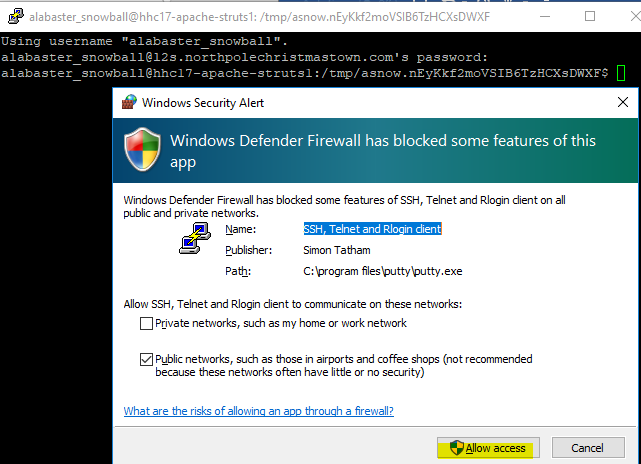
Open PuTTY, and with Session selected enter [alabaster\_snowball@l2s.northpolechristmastown.com](mailto:alabaster_snowball@l2s.northpolechristmastown.com) into the host field.



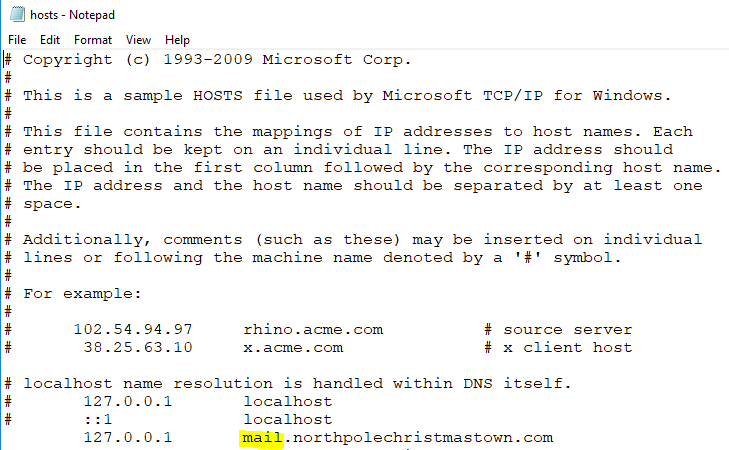
Then, with Connection -> SSH -> Tunnels selected, enter 8080 in the Source Port. Enter 10.142.0.5:80 in the Destination. Select Local and IPv4. Open the connection.



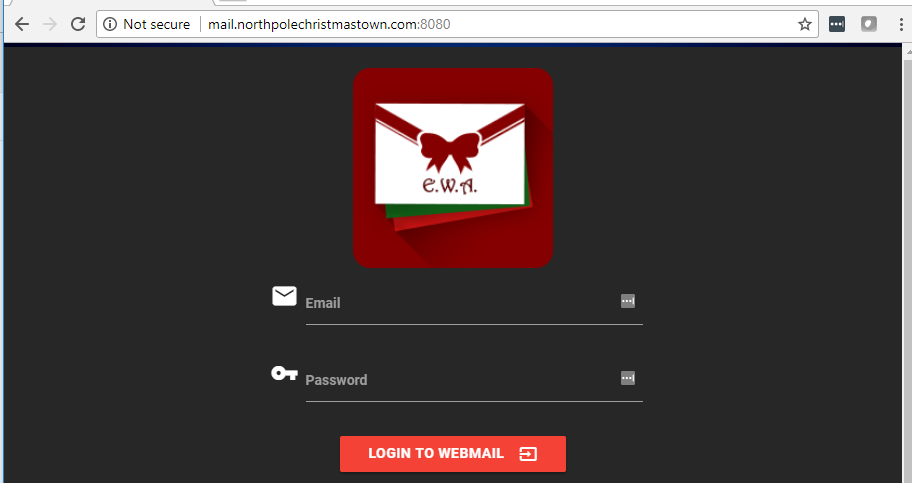
When you log in, you may have to allow PuTTY to open a hole in the firewall for 127.0.0.1:8080.



Pointing the browser to 127.0.0.1 will get the same generic server page as in the Linux example, so we’ll need to make an entry in C:\Windows\System32\drivers\etc\hosts.

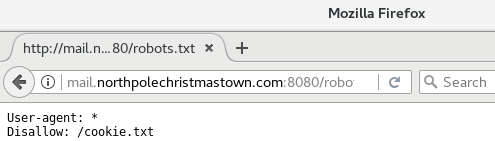


Then the browser should be able to connect.

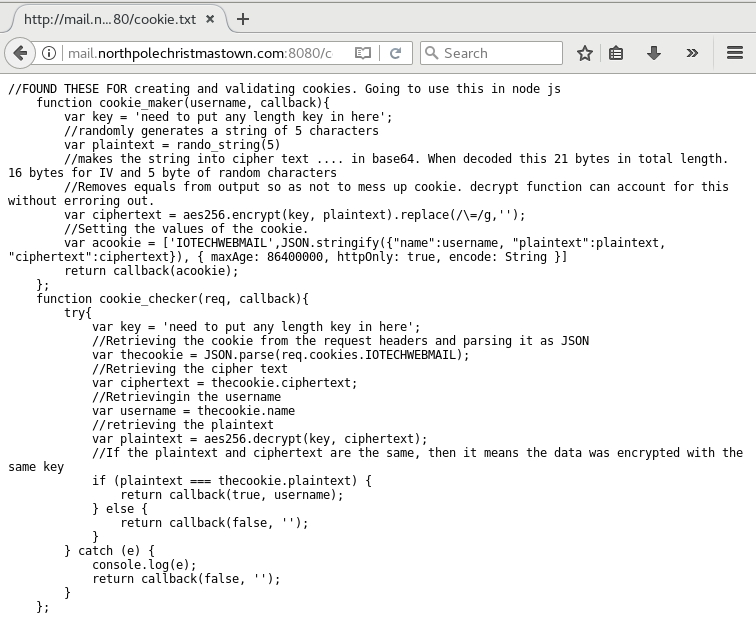


## Question 1

The robots.txt file tells search engines to not to scan the files you specify. It does prevent honorable search engines (Google, Bing, Yahoo, etc.) from showing those files as answers to queries. It does not prevent dishonorable people (us, for example) from viewing the robots.txt file and then viewing the prohibited files. The robots.txt file provides no secrecy at all. In fact, it highlights the files to adversaries. I don’t know why people insist on believing that entering file names or directories in a robots.txt file somehow makes the files invisible. Anyway, putting <http://mail.northpolechristmastown.com:8080/robots.txt> in the navigation bar of our browser shows the following.



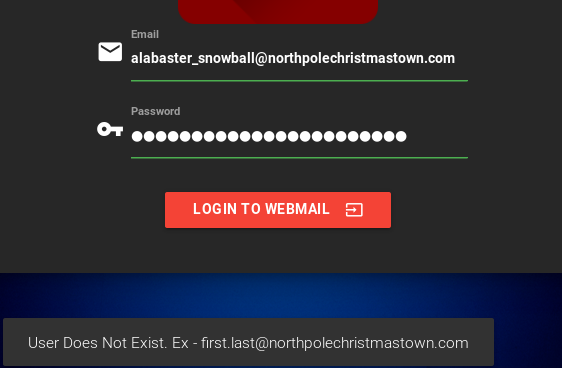
It appears that Alabaster does not want us to look at the cookie.txt file. Of course, we have no choice but to browse to <http://mail.northpolechristmastown.com:8080/cookie.txt>



Well, well. What have we here?

## Question 2

When we try to log in with Alabaster’s user name and @northpolechristmastown.com we see this.



Note the kind warning to tell us that we should be using an email address of [alabaster.snowball@northpolechristmastown.com](mailto:alabaster.snowball@northpolechristmastown.com). Many people consider this to be a security problem, but Alabaster has given priority to helping his users. Also, a good login system gives the same error message for user name errors as it does for password errors so that attackers cannot discover valid user names. Anyway, logging in with the correct address and the password we have for Alabaster does not work. At least Alabaster pays attention to his password reuse policy sometimes!

## Questions

The nice cookie.txt file appears to give us server-side code for Alabaster’s cookie system. Another source of information about the cookie may be found in the Developer’s Tools section of our browser. Between the two, we can learn a lot. Note: To view cookies using [Chrome Developer Tools](https://developers.google.com/web/tools/chrome-devtools/), select More Tools -> Developer Tools. Look in Application, Storage, Cookies. To view cookies using [Firefox Developer Tools](https://developer.mozilla.org/en-US/docs/Tools/), you must first enable the [Firefox Storage Inspector](https://developer.mozilla.org/en-US/docs/Tools/Storage_Inspector), as shown below.

1. Based on the content of the cookie.txt file, what kind of web server is Alabaster running? Hint: if you go past the top two or three lines, you’ve gone too far. You may also gain some insight by looking at the Nmap scan results from the SMB challenge. If you used the -A flag you’ll see something interesting on TCP 3000.
2. What language is the server-side code written in?
3. While you are looking at the EWA site, use your browser’s developer tools to check for cookies. What is the format of the cookie Alabaster uses?
4. Study the server-side code in the function cookie\_checker from cookie.txt. How does the server determine whether the cookie we submit is valid?
5. Use your favorite search engine to look for the encryption function name used in cookie.txt (aes256.encrypt) along with the answer to question 1. Find anything?
6. Based on Pepper’s Hints (especially 3 and 4), and the comments in cookie.txt, how could we create a cookie that the server might accept even if we don’t know the key?

