

1. Screenshot of the www.studentXX.230.com DNS entry resolving

```
cpre230@www:~$ dig www.student15.230.com

; <>> DiG 9.16.1-Ubuntu <>> www.student15.230.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61871
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.student15.230.com.      IN      A

;; ANSWER SECTION:
www.student15.230.com.  604800 IN      A      200.35.23.202

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Tue Oct 13 06:33:00 UTC 2020
;; MSG SIZE rcvd: 66
```

2. Screenshot of the default Apache page via Lynx www.studentXX.230.com

```
Ubuntu Logo Apache2 Ubuntu Default Page
It works!

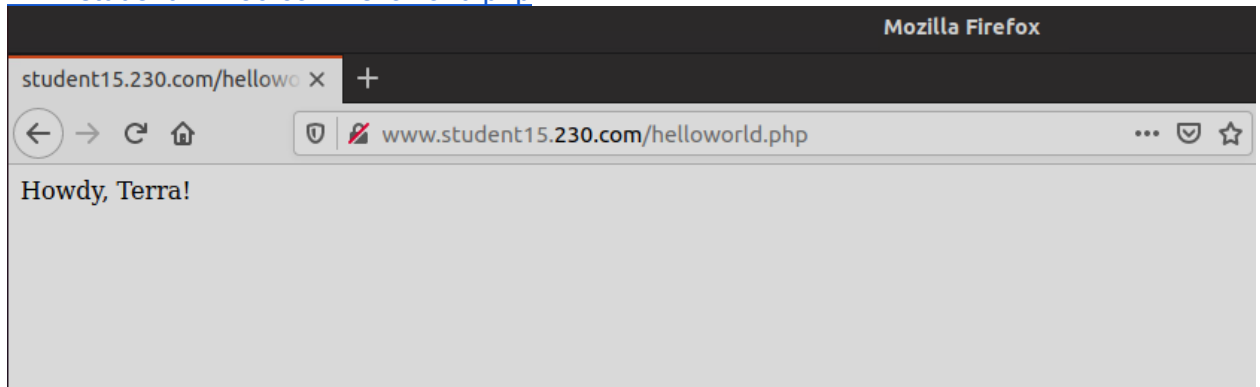
This is the default welcome page used to test the correct operation of the Apache2 server
after installation on Ubuntu systems. It is based on the equivalent page on Debian, from
which the Ubuntu Apache packaging is derived. If you can read this page, it means that the
Apache HTTP server installed at this site is working properly. You should replace this file
(located at /var/www/html/index.html) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this
probably means that the site is currently unavailable due to maintenance. If the problem
persists, please contact the site's administrator.
Configuration Overview

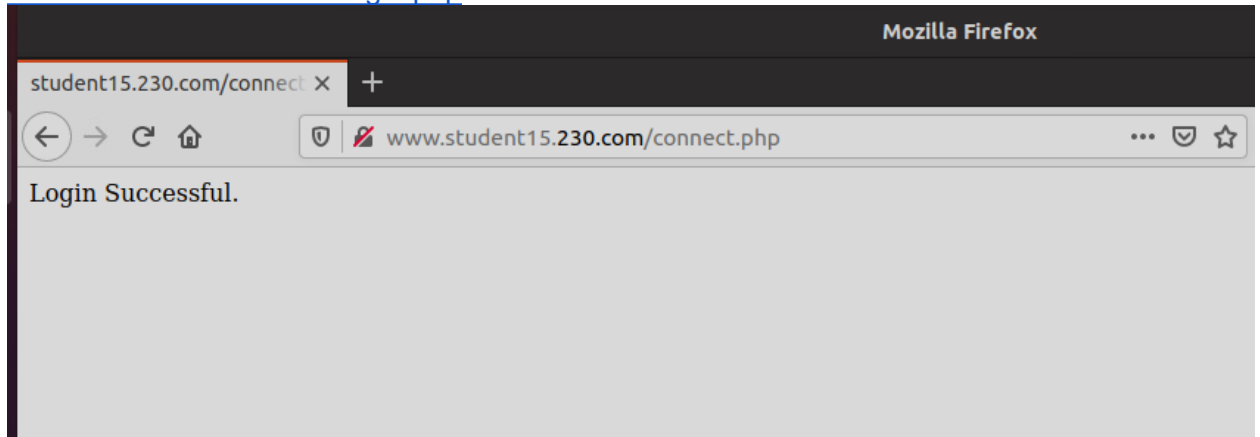
Ubuntu's Apache2 default configuration is different from the upstream default
configuration, and split into several files optimized for interaction with Ubuntu tools.
The configuration system is fully documented in /usr/share/doc/apache2/README.Debian.gz.
Refer to this for the full documentation. Documentation for the web server itself can be
found by accessing the manual if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as
follows:
etc/apache2/
-- apache2.conf
    |-- ports.conf
-- mods-enabled
    |-- *.load
    |-- *.conf
-- conf-enabled
    |-- *.conf
-- sites-enabled
    |-- *.conf
```

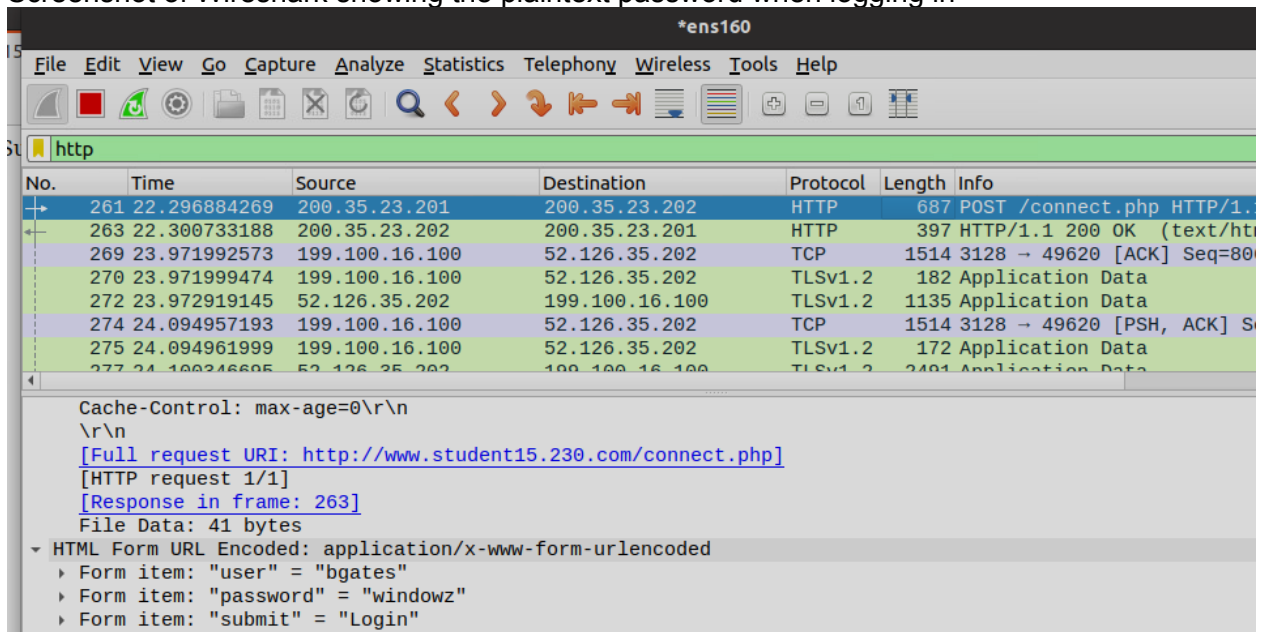
3. Screenshot of the Hello, world! PHP page via Firefox
www.studentXX.230.com/helloworld.php



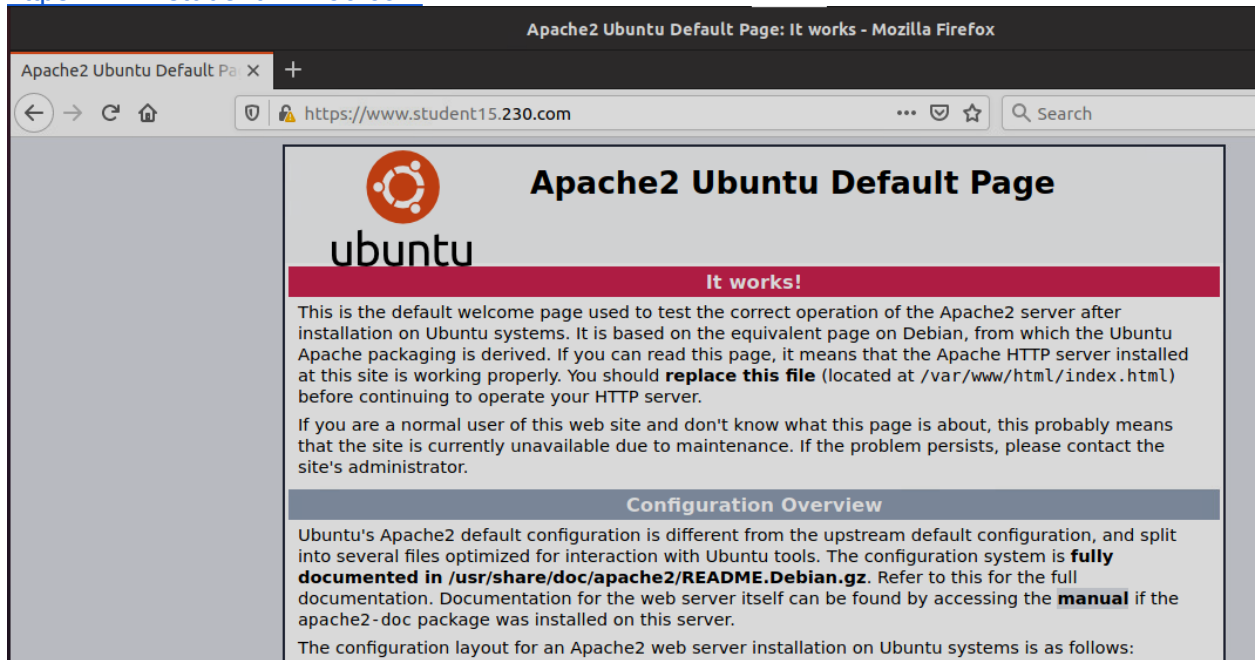
4. Screenshot of one of the users logged into PHP web page via Firefox
www.studentXX.230.com/login.php



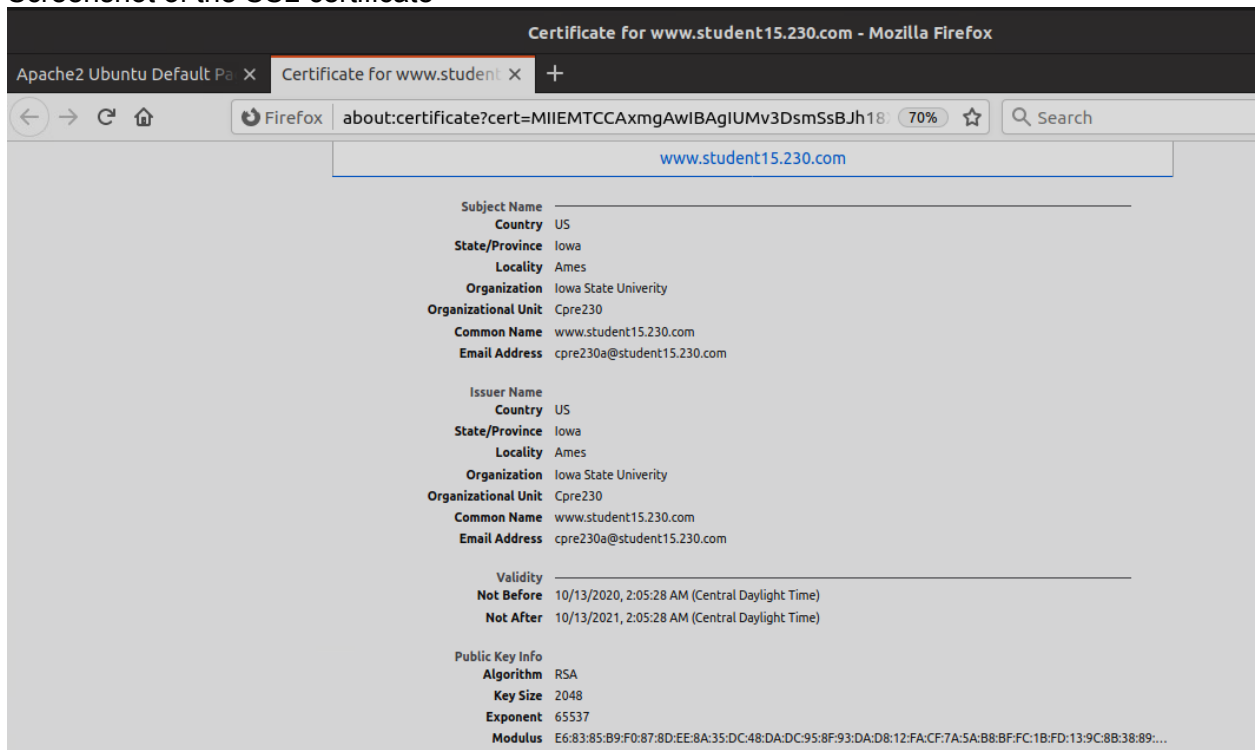
5. Screenshot of Wireshark showing the plaintext password when logging in



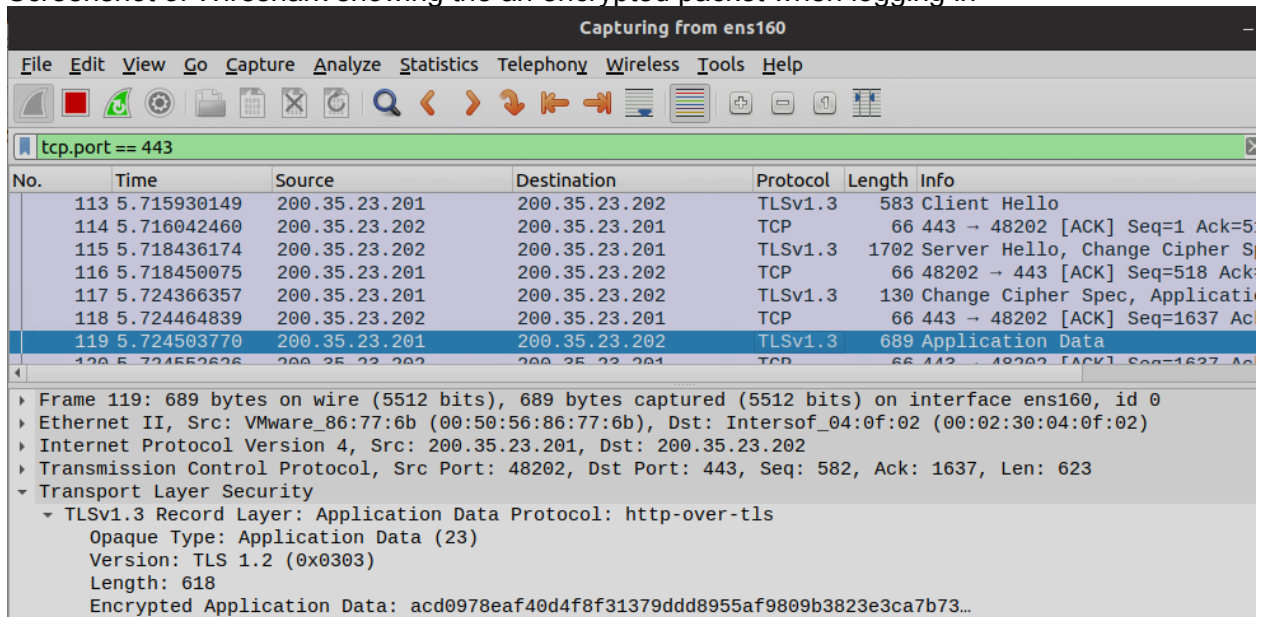
6. Screenshot of the Apache default page on HTTPS via Firefox:
<https://www.studentXX.230.com>



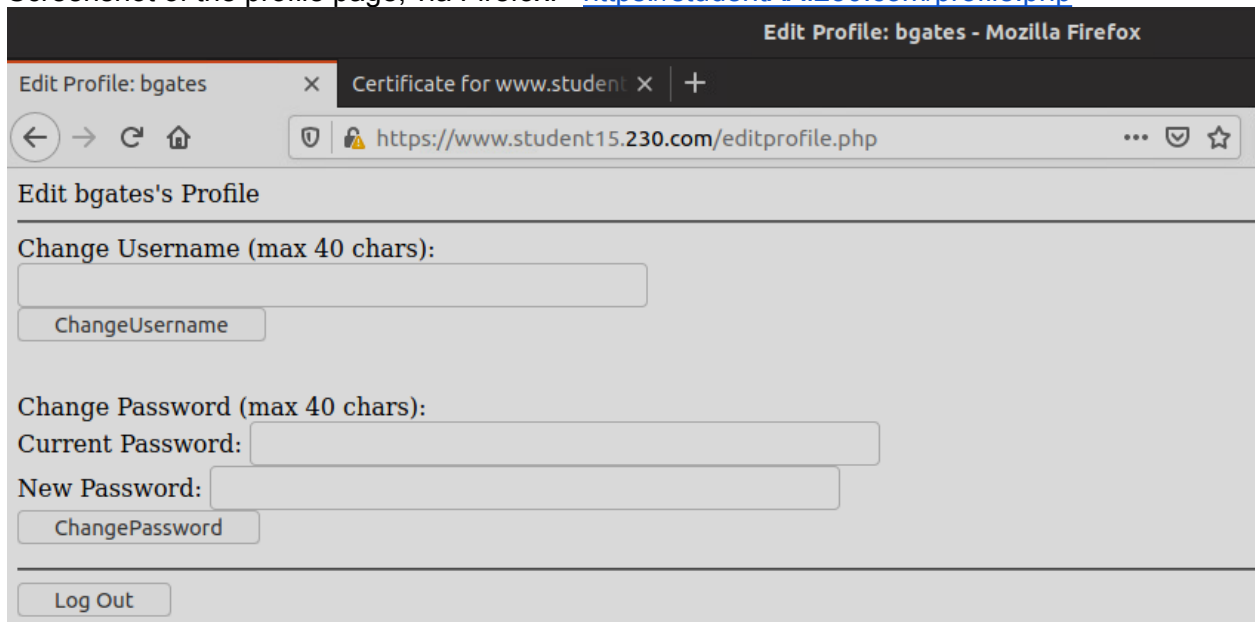
7. Screenshot of the SSL certificate



8. Screenshot of Wireshark showing the an encrypted packet when logging in



9. Screenshot of the profile page, via Firefox: <https://studentXX.230.com/profile.php>



10. Screenshot of the database, with changed username and password
(I entered the ' OR '1' = '1 in the current password field, and it set all the passwords to "password". It makes sense though. It's probably just checking which passwords match and changing those passwords. So when, they all equal true/1. It just sets them all.)

```
mysql> select * from login_info.UsernamePassword;
+-----+-----+-----+
| usernameID | username | password |
+-----+-----+-----+
|          1 | Stevie   | password |
|          2 | bgates   | password |
|          3 | mzuckerberg | password |
|          4 | emusk    | password |
|          5 | jbezoz   | password |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

11. Description of what SQL injection is, and how you might go about preventing it

It seems like SQL injection is basically passing SQL code into a web interface like a text box or password box. This text gets passed to the server and handled by some set of sql code, so the passed code may be executed to some effect. This then can set values in SQL or retrieve passwords or ...

To prevent it, it would seem like you could check for SQL commands / statements within your text before it is handled by SQL and deny the use of that text. This, if done perfectly / without flaws, would completely prevent the use. Another idea may be preventing the use of certain characters that are commonly used/needed in SQL injections. As well, you could minimize its potential threat with a principle of least access, so, if I did what I did in the picture above, I wouldn't be able to change let's Stevie and bgates. Since those two are admin users and can't be changed through the web interface at all.