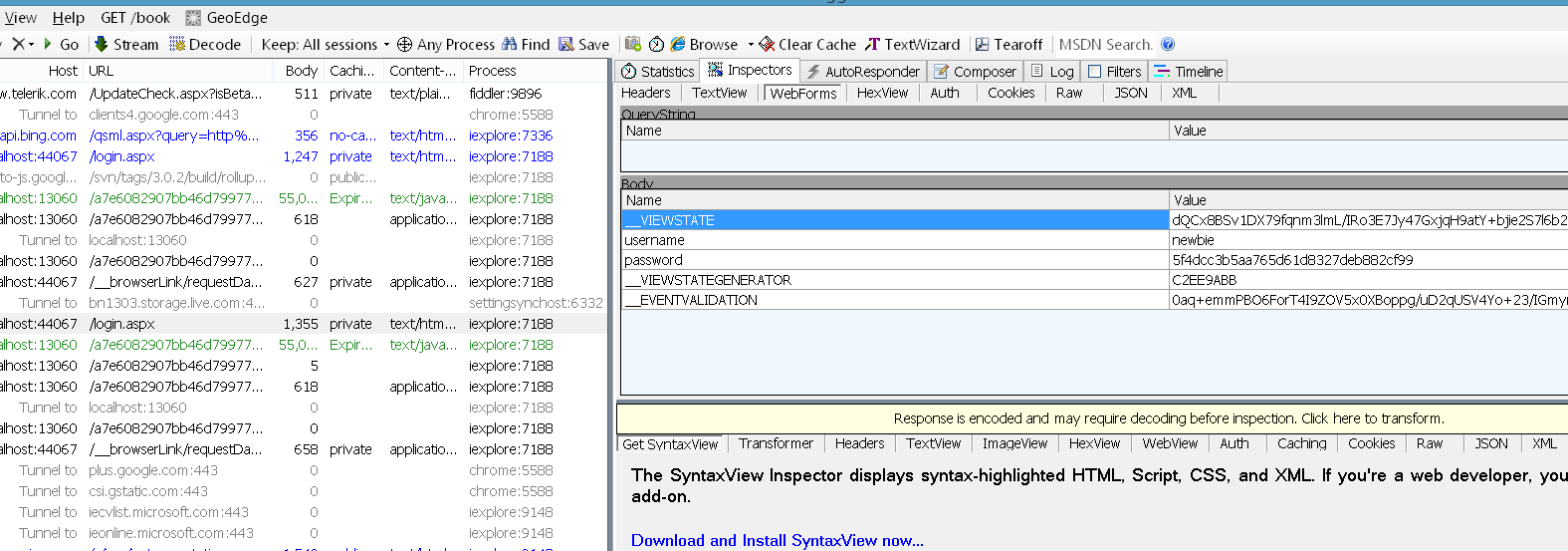
Project 5

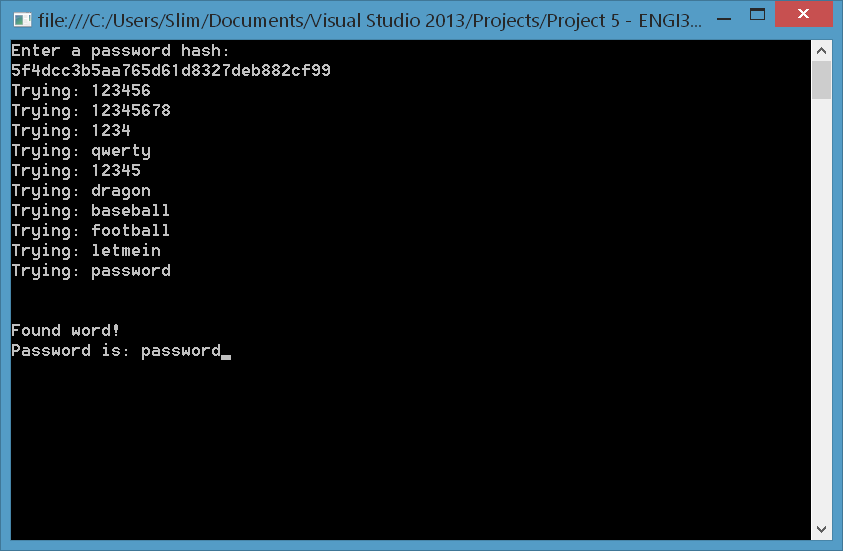
This project tested the security of an authentication database containing users and their hashed passwords. The project consists of 2 pages, one to add users by a stored procedure and another to verify their login.

Following an issue with loopback and wireshark we used fiddler to sniff the http traffic to collect the username and hash used to login.

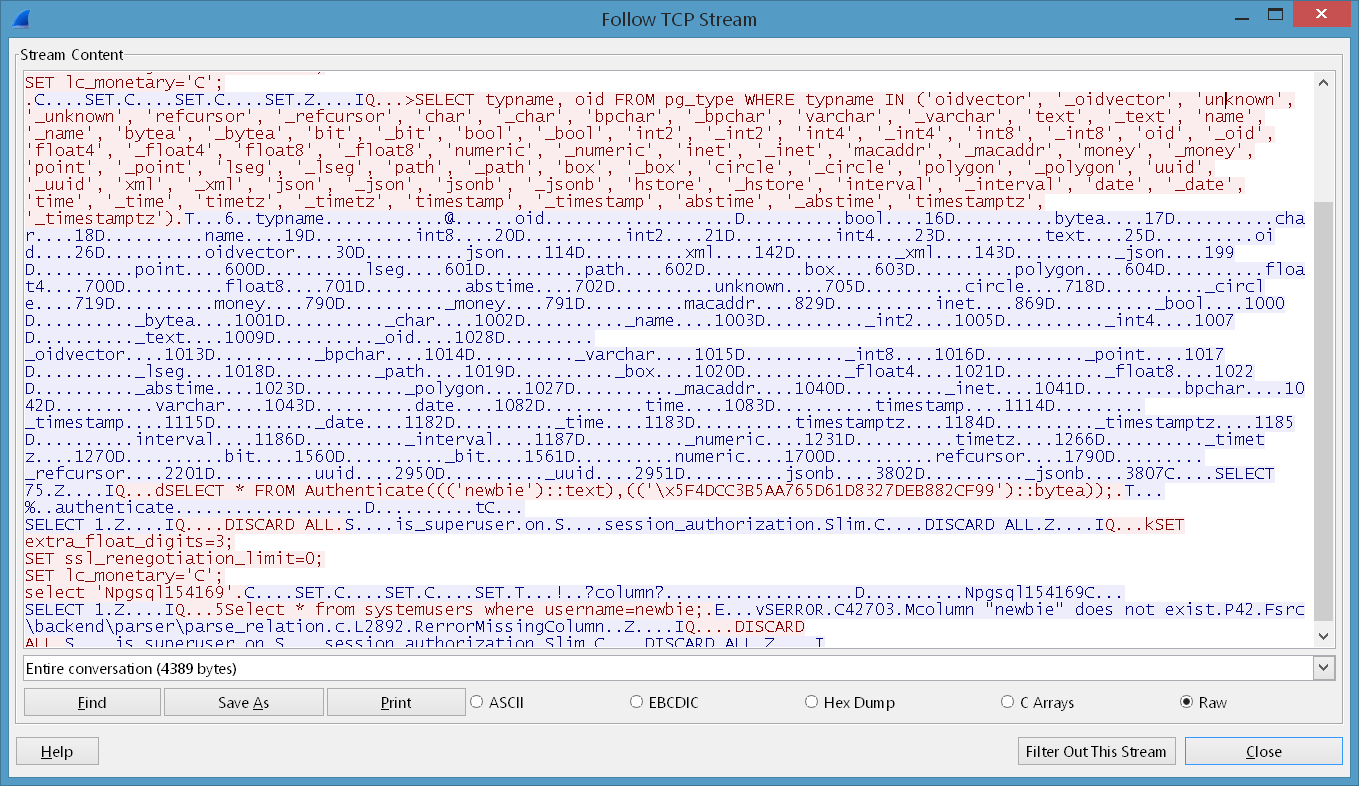


\_\_VIEWSTATE=dQCx8BSv1DX79fqnm3lmL%2FIRo3E7Jy47GxjqH9atY%2Bbjie2S7l6b2lmmMP55asYDyL%2BGBUCScriD5Dhn0K8qM184WUIP43pOIU4%2B7oy2nr8%3D&username=newbie&password=5f4dcc3b5aa765d61d8327deb882cf99&\_\_VIEWSTATEGENERATOR=C2EE9ABB&\_\_EVENTVALIDATION=0aq%2BemmPBO6ForT4I9ZOV5x0XBoppg%2FuD2qUSV4Yo%2B23%2FIGmynKmIvTaVPd5WK3fohpqu7GsQPpt18mLrVoT76x%2B0crW34nnYVy4NdQiO%2B8eRUO%2BSEtr9pSbqtezKu9gW4TZRNwPGx9ZuGWQpErz1A%3D%3D

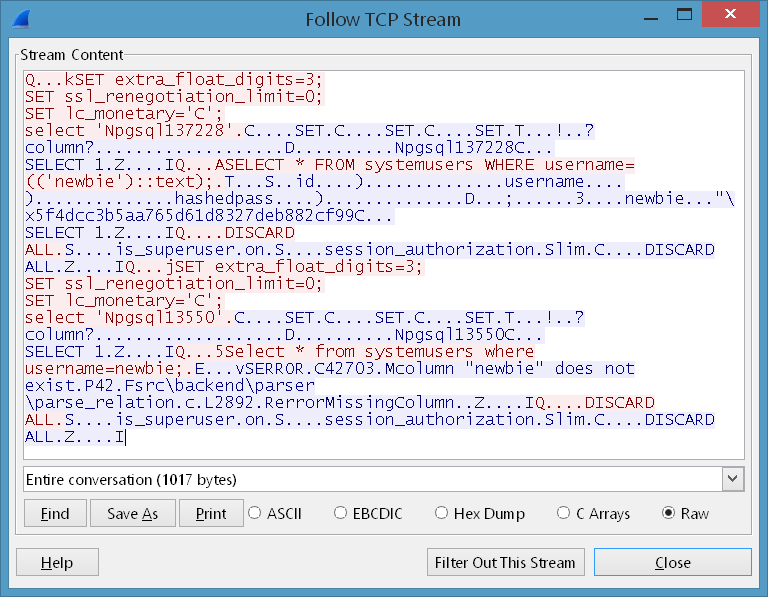
The authentication page would hash the plain text password before sending it to the server using javascript. A list of plain text English words were hashed and compared to the intercepted hash in order to reverse engineer the password by a Rainbow attack. Storing passwords as salted hashes would make it much more difficult in finding the plaintext password for this type of attack.



We were also able to find the communication to the database. Below is the sql communication using a stored procedure in sql: Sending plaintext to the stored procedure is a bad idea and should be hashed beforehand.



Below was using the ‘parameters’ and stored procedure option in c#:



This is bad if a webserver and database server are on different machines or are susceptible to man in the middle attack as this plain text communication can be intercepted. In both cases, it’s better to use some kind of challenge authentication so that sensitive information isn’t comprised if the communication happens to be intercepted. Preferably all communication can implement SSL (HTTPS) or some type of encryption for communication.