Lab 8: Hacking Web Servers

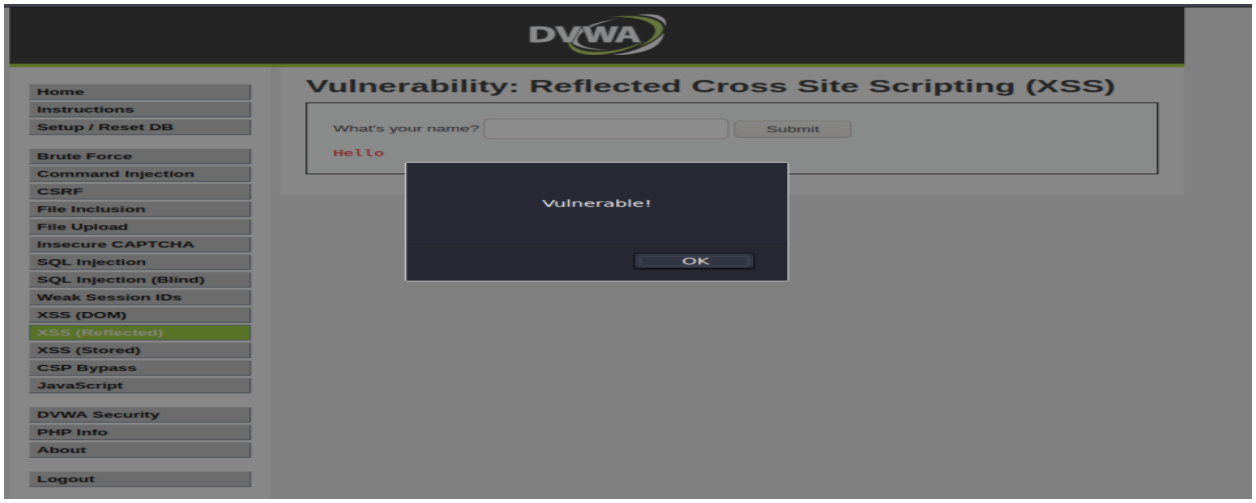
Introduction:

The objective of this lab was to study and execute some of the most common attacks against web servers, including cross site scripting, (XSS) and SQL/Command injections. To do so legally, we make use of the *Damn Vulnerable Web Application* (DVWA) available on github.

Part 1: Cross-Site Scripting (XSS)

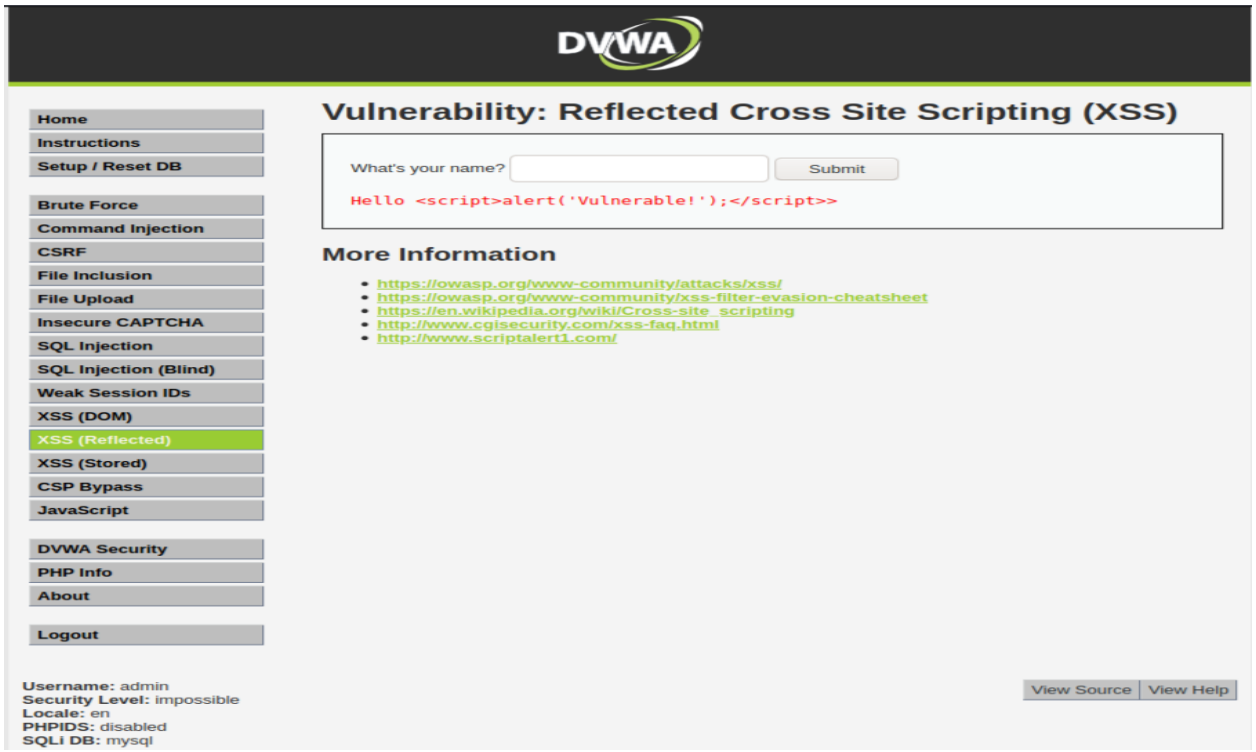
Cross-site scripting, otherwise known as XSS, is a form of code injection attack commonly employed against poorly developed web applications and users thereof. The attack works by supplying vulnerable input fields with malicious scripts, usually written in Javascript. Typically such scripts are designed to escalate an attacker's privilege over the web server itself, and or client machines. Most often for the purposes of downloading other malware or gaining unauthorized access to data such as cookies. We demonstrate proof of concept in DVWA by injecting an alert script into the text field for the user's name. The server is then sent a malformed HTTP GET request which leads to the execution of the script.

Figure 1: XSS Vulnerability



This kind of vulnerability is easily patched, as demonstrated by changing DVWA's security level to 'impossible.' Doing so changes the source code such that the input from the text field is sanitized and treated as a string. The exploit therefore fails when we attempt to run it again.

Figure 2: XSS Vulnerability Patched



<u>Part 2: SQL Injection</u>

As the name implies, SQL injections are another form of code injection attacks affecting mySQL databases. The principles of this type of attack are quite similar to those of XSS, relying on the injection of carefully crafted strings into vulnerable input fields of a web application. The goal of which being the generation and execution of malicious SQL statements on the target

server. Such attacks can easily be conducted with simple trial and error, iteratively trying different inputs until the structure of the SQL statement can be inferred.

Figure 3: Displaying All Users with `x'='x`

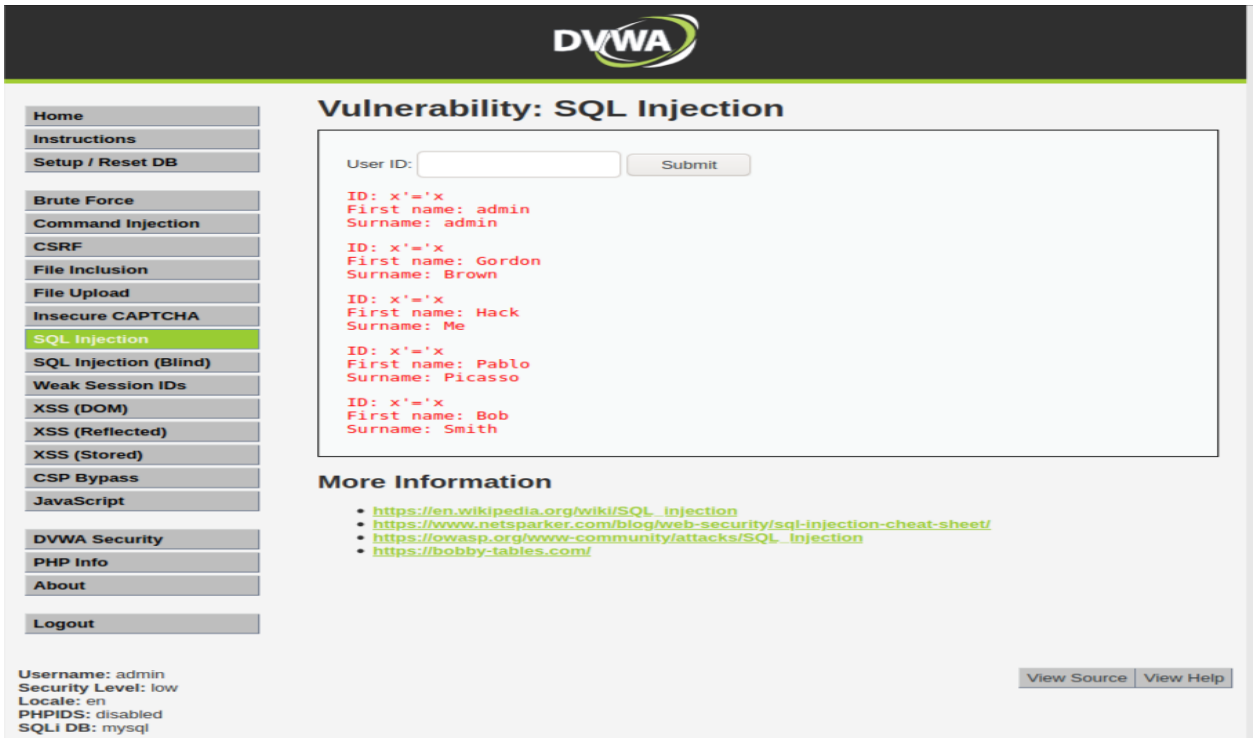
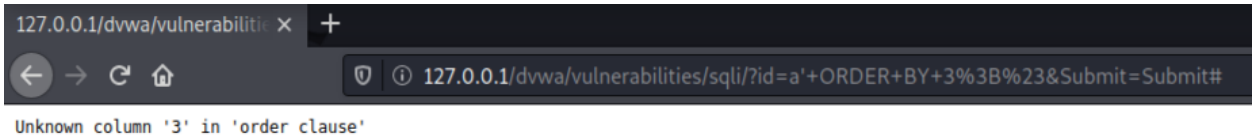


Figure 4: Deducing the Number of Columns with `a' ORDER BY 3;#`



Unknown column '3' in 'order clause'

By iteratively increasing the number before `;#`, we can deduce that the database does not contain a column by the name of 3. Further shedding insight as to what SQL query is working behind the scenes.

Figure 5: Display mySQL Version with `a' UNION ALL SELECT 1, @@version;#`

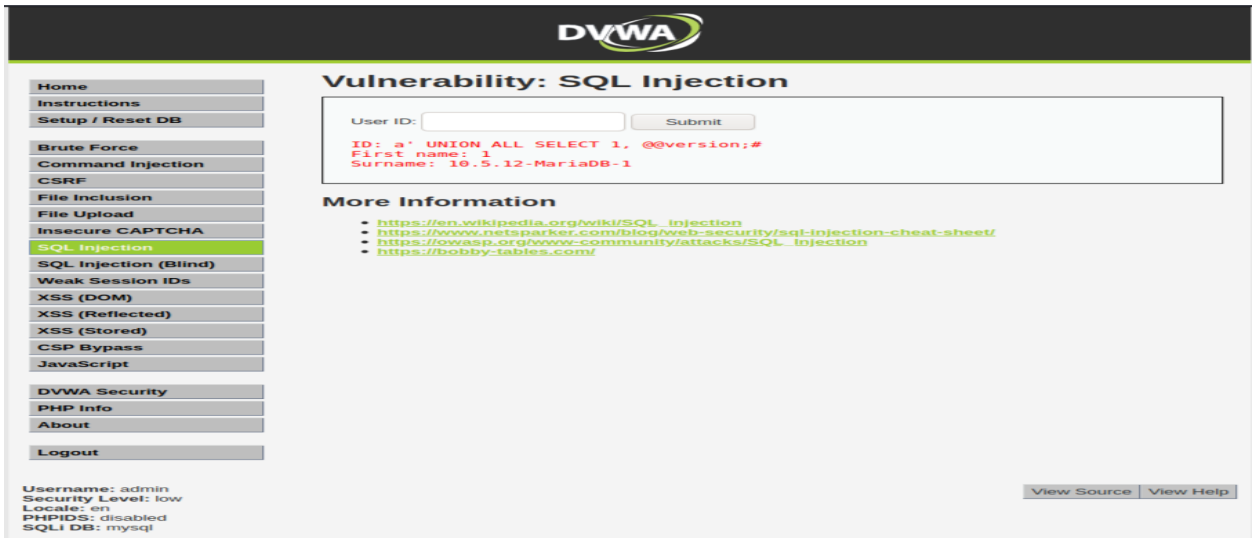
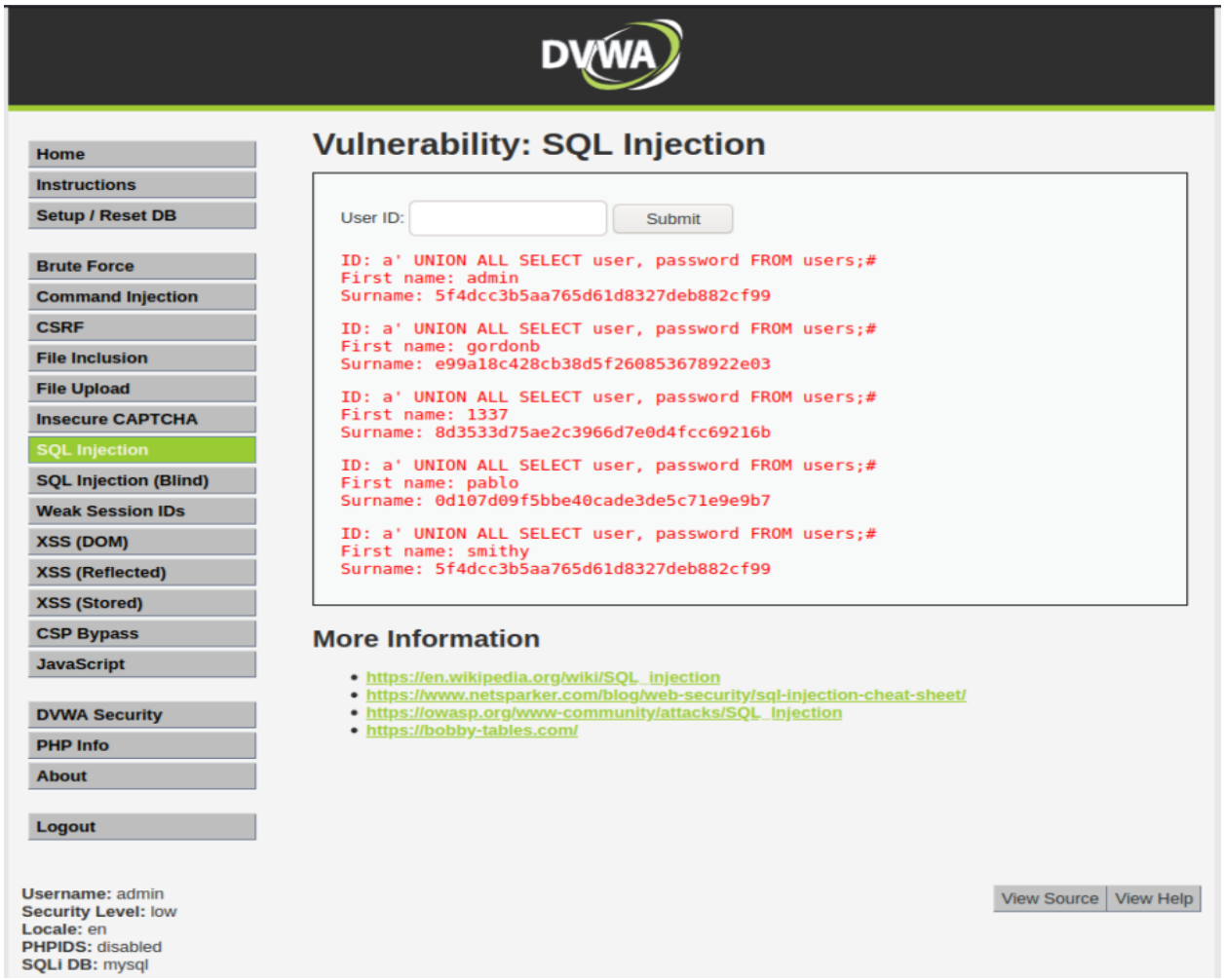


Figure 6: Display Server Owners with `a' UNION ALL SELECT system_user(), user ;#`

Figure 7: Display User Credentials with `a' UNION ALL SELECT user, password FROM users;#`



It is common practice among web developers to only store the hash values of sensitive information such as passwords or banking information. This approach however, is only as good as the hash function it relies upon, and is vulnerable to collision and or pass the hash attacks.

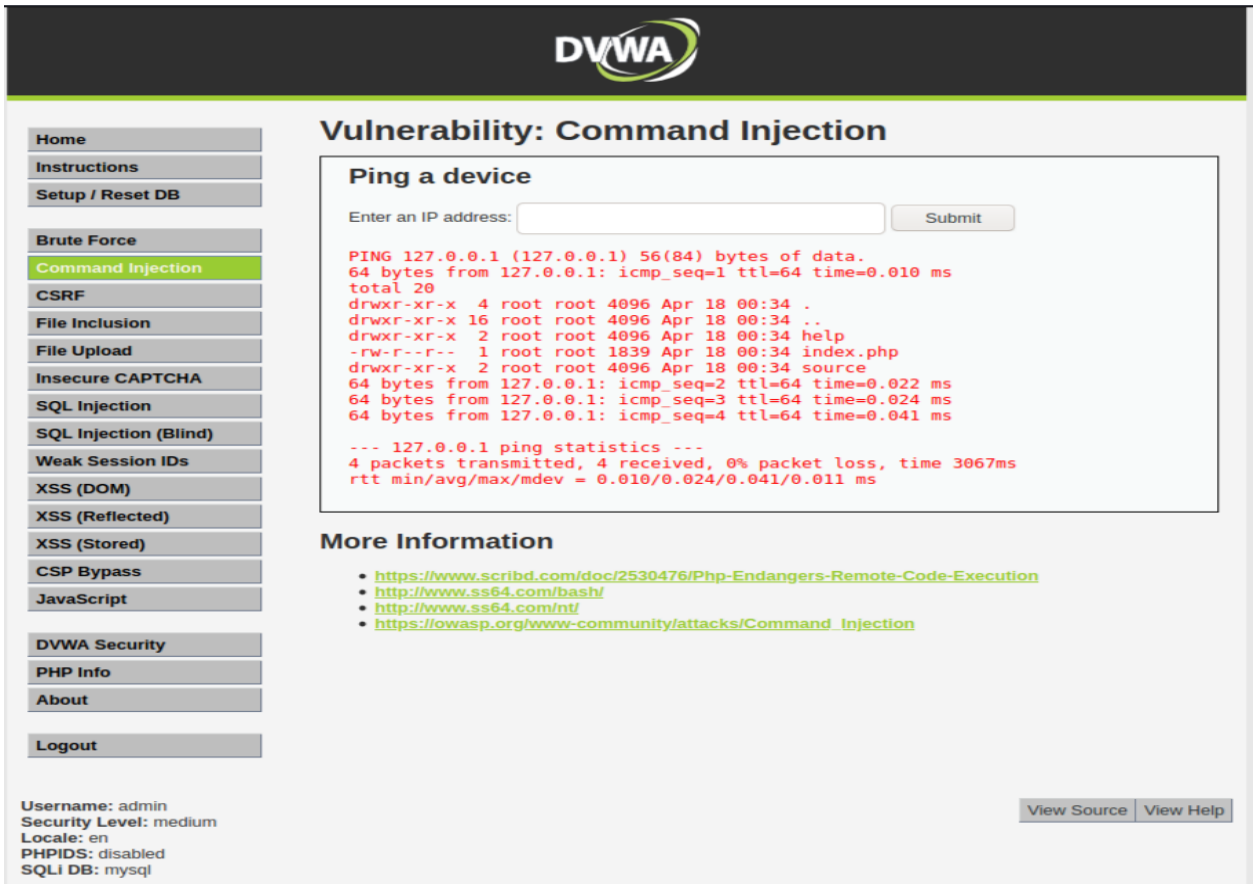
Part 3: Command Injection

The exploitation of poorly programmed web forms for the sake of running malicious code is a common denominator among most attacks targeting web applications. The last attack we will examine is that of command injection, which if done correctly, affords attackers the ability to remotely execute shell commands on the target server. Mechanically, the process is very similar to an XSS or SQL injection attack, save for the fact that shell commands are injected in lieu of scripts of mySQL statements. Web applications that support UNIX style command utilities are particularly vulnerable. Namely, the DVWA ping web application, which supports UNIX style ping commands.

Figure 8: Low Security Level Command Injection With `127.0.0.1 && ls -la`

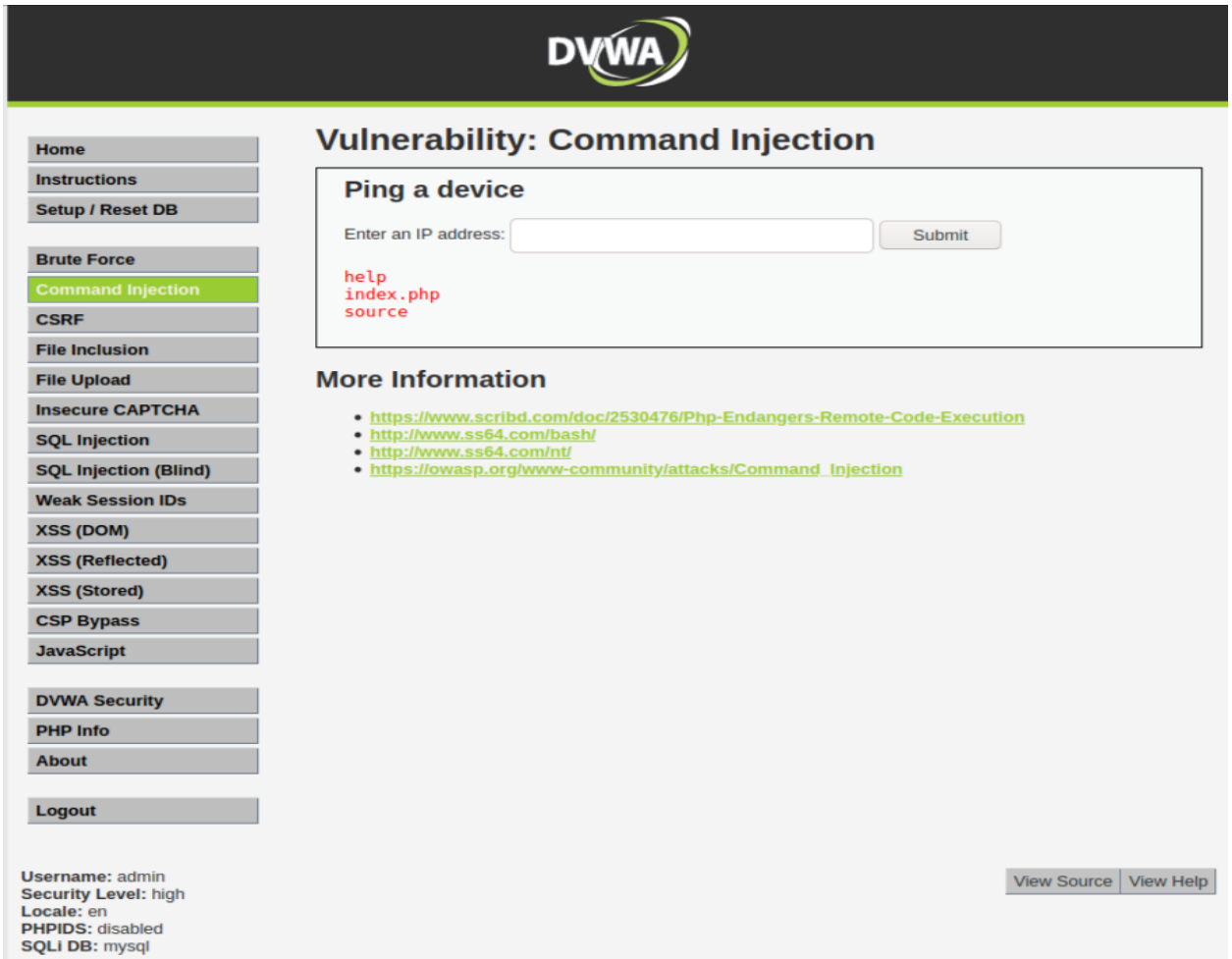
Low security level allows us to view the directories the web application is running on along with their permissions and owners. Potentially invaluable information to an attacker.

Figure 9: Medium Security Level Command Injection With `127.0.0.1 & ls -la`



Medium security level still works as intended with some minor adjustments to the injection statement.

Figure 10: High Security Level Command Injection With `127.0.0.1|ls`



With high security, we can still view the directories the server is running on, but not their permissions or owners.