

## **Is WordPress Vulnerable to XSS and SQLi Attacks?**

Jon Gage

GageSec.com

## TABLE OF CONTENTS

ABSTRACT .....	3
INTRODUCTION.....	4
BACKGROUND.....	5
PROBLEM STATEMENT.....	6
OBJECTIVES.....	7
LITERATURE REVIEW.....	8
METHODOLOGY ADOPTED.....	10
RESULTS – PROJECT FINDINGS.....	11
RECOMMENDATIONS.....	13
CONCLUSION.....	14
BIBLIOGRAPHY.....	15
LIST OF FIGURES.....	18

## ABSTRACT

The use of XSS (cross-site scripting) and SQL injection attacks can be used to gain access to a MySQL database. This database stores, updates, and provides all the data that allows the WordPress web application to function. Successful injections can manipulate code to display database table entries allowing an attacker to gather valuable data such as usernames, passwords, personal information, and more. The MySQL database also contains entries of most configuration settings that can be changed in the administration back-end. It is possible to inject code from the front-end and trick the SQL service to provide unintended information from the database to the front-end. In some cases, it is even possible to alter data on the database and insert malicious code to escalate access privileges to the rest of the web server to carry out more attacks. It is important to use web applications that are PCI compliant when financial information is present on the database. PCI compliance requires encryption of financial data in the databases to prevent fraud in the event of a breach.

One of the biggest countermeasures WordPress uses to help defend against XSS and SQL injection attacks is code sanitization. When code sanitization is implemented, it will omit any known characters that are used to perform XSS and SQLi attacks. An example of this would be when an attacker would type “<script>alert('XSS')</script>”, code sanitization will detect the <script> and </script> and entirely omit the input from being inserted into the MySQL database. This script is also a good way to test and see if XSS attacks are possible from the front-end.

## INTRODUCTION

XSS and SQL injections have been on the top of the “OWASP’s Top 10” list for the last decade. Attackers will attempt to exploit a web application front-end by inserting malicious code from an input field on the front-end such as a comment post or from the address bar. If successful, the malicious code tricks the SQL service into providing unintended data from the database onto the webpage. There are tools such as “WebPwn3r” and “Commix” that can be used to scan and attempt to inject code and insert, update, and retrieve data from the MySQL database.

The most common attack is an In-band SQLi, where the attacker identifies an exploit through either error-based or Union-based SQL injections, then immediately attempts to inject code that queries the MySQL database and tricks the SQL service into executing the code without proper authentication.

According to W3Tech, the majority of all known websites using a server-side programming language implement PHP and MySQL and around 42% of all websites use WordPress to drive their content.

## BACKGROUND

XSS (Cross-site Script) attacks take advantage of improperly configured web apps and server configurations that allows malicious code to be inserted into the database and is executed from the front-end without proper authentication. XSS attacks can be used to perform SQL injections by querying the MySQL database and displaying database entries onto the webpage.

Typically, these attacks are carried out by exploiting an input field that adds malicious code to a comment section or forum post to the website. It is also possible to inject php code to the address bar depending on the web server and application configurations to carry out injection attacks.

WordPress has a comment section built in and there are also numerous third-party plugins that can easily add extra functionality to WordPress ranging from “Site Reviews” to even providing a full online shopping experience such as “WooCommerce”. WordPress has an official plugin repository with a large list of “trusted” plugins. Use of these third-party plugins could potentially open vulnerabilities to the WordPress application and even the web server if the plugin is not properly maintained and constantly updated by the developers for that specific plugin.

## PROBLEM STATEMENT

Website features such as being able to post comments or participate in discussions allows users to communicate with each other and can add value to a website. An example would be that organizations can use these features to add a review section to their website where customers can post reviews of their products or services.

Improper web server and web app configurations can allow attackers to exploit these communication features and gain access the SQL databases, once accessed, they can either retrieve data or inject data, an example would be adding a user with admin privileges to access the backend of the web application or even the web server. Once the server has been breached, threat actors can steal valuable information from the databases and modify the website to either deface it. Threat actors can also implement social engineering techniques such as phishing to trick users into providing even more information from what looks like a legitimate website from a legitimate source.

## OBJECTIVES

- Experiment: Setup a virtual machine with a web server that is hosting the latest release of WordPress.
  - OS: CentOS Linux
    - Features required to run WordPress:
      - Apache (httpd)
      - PHP
      - MySQL (MariaDB)
      - WordPress
        - Plugins(Optional):
          - Site Reviews (Developer: Paul Riley)
          - Advanced Comment Form(Developer:Thomas Mair)
- Run various SQLi vulnerability tools through Parrot Security OS to test the vulnerabilities of WordPress and the Apache web server.
- Research historical archives.

## LITERATURE REVIEW

### **Exploiting SQL Injection: A Hand-on Example**

**(<https://www.acunetix.com/blog/articles/exploiting-sql-injection-example/>)**

Going through the process of performing an SQL injection step-by-step can help make sense of what an SQL injection is, and how it works. Agathoklis Prodromou explains how to alter a web address link to trick the SQL service into providing information from the database. The method explained changes the query from showing just the article to displaying results on the webpage to indicate that the SQL queries are being manipulated. If a query can be manipulated, then there is a very good chance, with the correct queries being altered, that gaining access to the databases backend is possible. Agathoklis then expands upon the queries to extract the administrator's password hash to be displayed onto the webpage. He then used a program called HashCat to decrypt the hash value to its original password. Once the administrator access is compromised, the attacker then has full control of the web application.

### **How to Find SQL Injection Attack Vulnerabilities?**

**(<https://geekflare.com/find-sql-injection/>)**

Chandan Kumar explains that 8% of scanned web applications are vulnerable to SQL injections. He then suggests several tools and techniques that can be used to test the vulnerability of a web application.

### **SQL Injection Attack: Real Life Attacks and Code Examples**



**(<https://www.neuralegion.com/blog/sql-injection-attack/>)**

In 2019, the popular video game “FortNite” developed by Epic Games fell victim to an SQLi attack allowing the attackers access to user accounts. Admir Dizdar also explains that in 2018, the networking technologies company, Cisco also fell victim to an SQLi attack by exploiting a vulnerability in the company’s Cisco Prime License Manager, allowing attackers to gain shell access to systems on which the license manager was deployed. These vulnerabilities have since been patched.

## METHODOLOGY ADOPTED

- Experiment:
  - Deploy a sandbox VM with an active web server and WordPress installed, then perform SQLi vulnerability scans using various tools.
    - Tools used:
      - Commix
      - SQLMap
      - Vega
      - Jsql
      - Xsser
      - Webpwn3r
    - Historical Archives:
      - Research the history of known SQLi attacks and methodologies from reliable and credible sources.

## RESULTS – PROJECT FINDINGS

Experiment: Using Jsql, Xsser, and Webpwn3r to find SQLi Vulnerabilities

After testing the web server and Web Application for SQLi vulnerabilities using Jsql, Xsser, and Webpwn3r, all attempts to perform an injection were unsuccessful. Simply having the OS up to date along with the latest releases of PHP, MySQL, and Apache, which are required to run the latest release of WordPress was enough to prevent XSS and SQLi injections from being successful using these three tools at present time.

Experiment: Commix and SQLMap to find SQLi Vulnerabilities

For good measure, it seemed necessary to expand the vulnerability testing results by utilizing two more tools to see if an SQLi vulnerability could be found. After performing a level 3(the most extensive test level) scan with Commix, it took over three hours to perform the vulnerability scan, however, the scan was not able to find an SQLi vulnerability.

When running SQLMap, the scanner attempted to exploit a comment box to inject SQL queries, however, code sanitization techniques implemented by WordPress either omitted part of the code or posted it in plain text.

```
Command: sqlmap -u 'http://192.168.56.103/?p=1#comments' -tables -  
tamper=space2comment
```

This command attempts to query the SQL database into retrieving the table entries.

(See figures 1-4 on page 17)

Historical Archive: PHPMailer 6.1.8 through 6.4.0 allows object injection through Phar Deserialization via addAttachment with a UNC pathname.

The CVE-2020-36326 vulnerability affects WordPress version 3.7 to 5.7. The vulnerability exploits PHP Object serialization to bypass sanitization techniques allowing malicious code to be passed to the SQL service, which could then be executed. WordPress uses PHPMailer to send e-mails to users notifications and activation emails for registered users. The vulnerability was published to the CVE database on April 28, 2021, and the patch was released April 29, 2021.

Historical Archive: WordPress Vulnerability Report: October 2021, Part 1 | Author: Michael Moore | <https://ithemes.com/blog/wordpress-vulnerability-report-october-2021-part-1/>

Michael Moore lists 30 plugins used on WordPress that had vulnerabilities as of October 2021. One plugin called “WP DSGVO Tools” was found to be vulnerable to XSS attacks. The vulnerability was patched but has been removed from the official repository and anyone using the plugin is advised to find a replacement ASAP.

## RECOMMENDATIONS

- Always keep Web Server, Web Applications, and all services up to date.
- Disable or remove unused services.
- Do research on any WordPress third-party plugins and themes before installing them to ensure the development of them follow proper security procedures and aren't currently vulnerable to attack.
  - Continually research CVE databases for known attacks in the past and compare how often third-plugin developers update their code in response.
- Use complex usernames and passwords for all administration accounts.

## CONCLUSION

WordPress is a powerful web application that allows users to create websites for blogging and with the use of plugins and custom themes, can add a vast amount of functionality and look really good. The latest release of WordPress and all of its dependencies required to run the web application is very secure against XSS and SQL injection attacks. The main cause for a WordPress website to have security threats usually involve the use of custom themes and adding third-party plugins that can make the core installation become vulnerable to exploits. For that reason, use as few third-party add-ons as possible and only use themes and plugins that are known to be trusted and secure, as well as researching whether the plugins being used are updated and patched on a regular basis in response to known vulnerabilities.

## BIBLIOGRAPHY

“[SECURITY] Fedora 34 Update: Php-Phpmailer6-6.4.1-1.Fc34 - Package-Announce - Fedora Mailing-Lists.” *Lists.fedoraproject.org*, 12 May 2021, [lists.fedoraproject.org/archives/list/package-announce@lists.fedoraproject.org/message/KPU66INRFY5BQ3ESVPRUXJR4DXQAFJVT/](https://lists.fedoraproject.org/archives/list/package-announce@lists.fedoraproject.org/message/KPU66INRFY5BQ3ESVPRUXJR4DXQAFJVT/). Accessed 30 Nov. 2021.

Ansari, Maria. “WordPress Market Share: How Many Websites Use WordPress in 2021?” *Barn2 Plugins*, 2 May 2021, [barn2.com/wordpress-market-share/](https://barn2.com/wordpress-market-share/).

“CVE-2020-36326 : PHPMailer 6.1.8 through 6.4.0 Allows Object Injection through Phar Deserialization via AddAttachment with a UNC Pathname.” *Www.cvedetails.com*, 28 Apr. 2021, [www.cvedetails.com/cve/CVE-2020-36326/](https://www.cvedetails.com/cve/CVE-2020-36326/). Accessed 30 Nov. 2021.

Ekşiler, Ant. “WordPress Users Urged to Update to Version 5.7.2 to Fix Critical Security Issue.” *Fuel Themes*, 15 June 2021, [fuelthemes.net/wordpress-users-urged-to-update-to-version-5-7-2-to-fix-critical-security-issue/](https://fuelthemes.net/wordpress-users-urged-to-update-to-version-5-7-2-to-fix-critical-security-issue/). Accessed 30 Nov. 2021.

Hamila, Abderrahman. “What Sanitize Mean ? And Why Sanitize in Code/Data ?” *Medium*, 30 Jan. 2018, [medium.com/@abderrahman.hamila/what-sanitize-mean-and-why-sanitize-in-code-data-5c68c9f76164](https://medium.com/@abderrahman.hamila/what-sanitize-mean-and-why-sanitize-in-code-data-5c68c9f76164).

Imperva. “What Is SQL Injection | SQLI Attack Example & Prevention Methods | Imperva.”

*Learning Center*, 2019, [www.imperva.com/learn/application-security/sql-injection-sqli/](http://www.imperva.com/learn/application-security/sql-injection-sqli/).

Moore, Michael. “WordPress Vulnerability Report: October 2021, Part 1.” *IThemes*, 6 Oct. 2021,

[ithemes.com/blog/wordpress-vulnerability-report-october-2021-part-1/](https://ithemes.com/blog/wordpress-vulnerability-report-october-2021-part-1/). Accessed 30

Nov. 2021.

“OWASP Top 10 Security Vulnerabilities 2021.” *Sucuri*, [sucuri.net/guides/owasp-top-10-](https://sucuri.net/guides/owasp-top-10-security-vulnerabilities-2021/)

[security-vulnerabilities-2021/](https://sucuri.net/guides/owasp-top-10-security-vulnerabilities-2021/).

“Protecting against SQL Injection.” *Hacksplaining*, Hacksplaining, 2019,

[www.hacksplaining.com/prevention/sql-injection](https://www.hacksplaining.com/prevention/sql-injection).

Riley, Paul. “Gemini Labs.” *Gemini Labs* | “*Site Reviews*” *Plugin*, [geminilabs.io/](https://geminilabs.io/). Accessed 30

Nov. 2021.

“Types of SQL Injection?” *Acunetix*, 2019, [www.acunetix.com/websitesecurity/sql-injection2/](https://www.acunetix.com/websitesecurity/sql-injection2/).

“Usage Statistics and Market Share of PHP for Websites, January 2020.” *W3techs.com*, 2020,

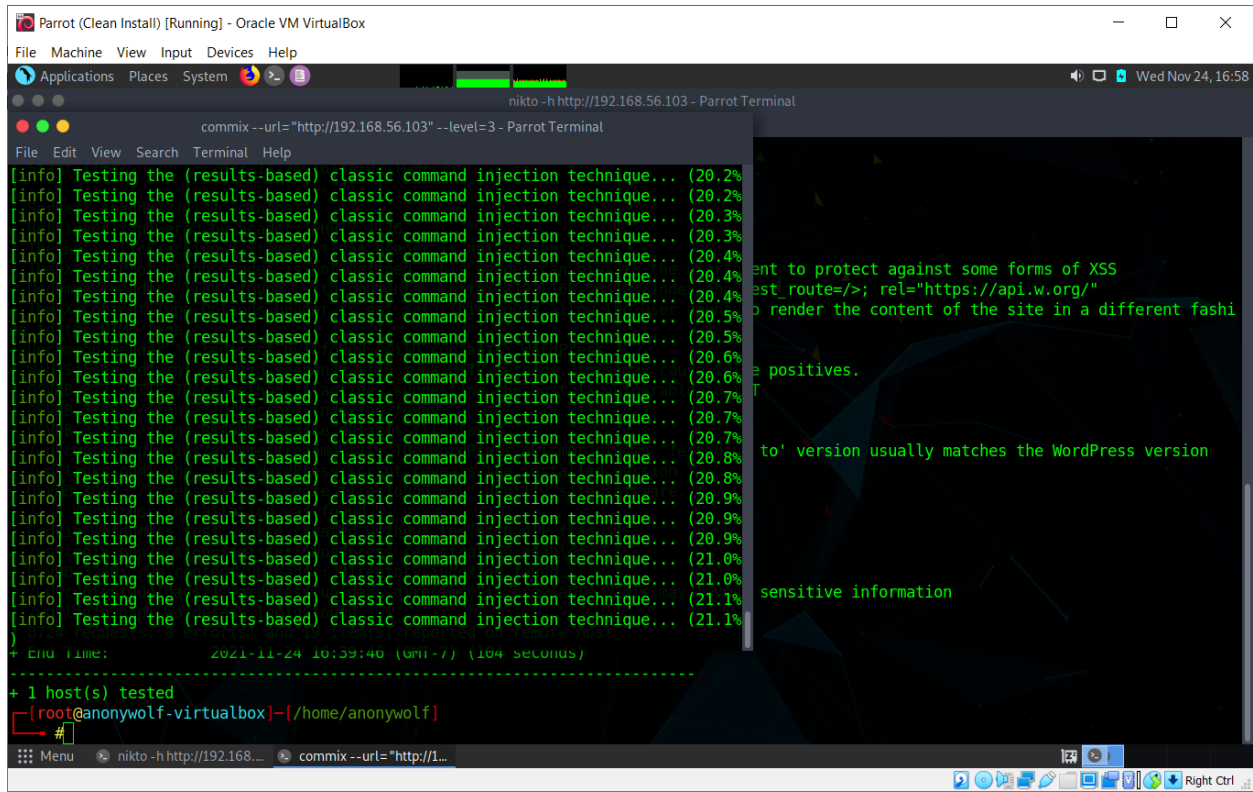
[w3techs.com/technologies/details/pl-php](https://w3techs.com/technologies/details/pl-php).



“What Is SQL Injection & How to Protect against It | Barracuda Networks.”

*Www.barracuda.com*, [www.barracuda.com/glossary/sql-injection](http://www.barracuda.com/glossary/sql-injection).

## LIST OF FIGURES



```
Parrot (Clean Install) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System Wed Nov 24, 16:58
nikto -h http://192.168.56.103 - Parrot Terminal
commix --url="http://192.168.56.103" --level=3 - Parrot Terminal
File Edit View Search Terminal Help
[info] Testing the (results-based) classic command injection technique... (20.2%)
[info] Testing the (results-based) classic command injection technique... (20.2%)
[info] Testing the (results-based) classic command injection technique... (20.3%)
[info] Testing the (results-based) classic command injection technique... (20.3%)
[info] Testing the (results-based) classic command injection technique... (20.4%)
[info] Testing the (results-based) classic command injection technique... (20.4%)
[info] Testing the (results-based) classic command injection technique... (20.4%)
[info] Testing the (results-based) classic command injection technique... (20.5%)
[info] Testing the (results-based) classic command injection technique... (20.5%)
[info] Testing the (results-based) classic command injection technique... (20.6%)
[info] Testing the (results-based) classic command injection technique... (20.6%)
[info] Testing the (results-based) classic command injection technique... (20.7%)
[info] Testing the (results-based) classic command injection technique... (20.7%)
[info] Testing the (results-based) classic command injection technique... (20.7%)
[info] Testing the (results-based) classic command injection technique... (20.8%)
[info] Testing the (results-based) classic command injection technique... (20.8%)
[info] Testing the (results-based) classic command injection technique... (20.9%)
[info] Testing the (results-based) classic command injection technique... (20.9%)
[info] Testing the (results-based) classic command injection technique... (20.9%)
[info] Testing the (results-based) classic command injection technique... (21.0%)
[info] Testing the (results-based) classic command injection technique... (21.0%)
[info] Testing the (results-based) classic command injection technique... (21.1%)
[info] Testing the (results-based) classic command injection technique... (21.1%)
)
+ End Time: 2021-11-24 10:39:40 (GMT-7) (104 seconds)
-----
+ 1 host(s) tested
[root@anonywolf-virtualbox]-[~/home/anonywolf]
#
```

Figure 1-commix

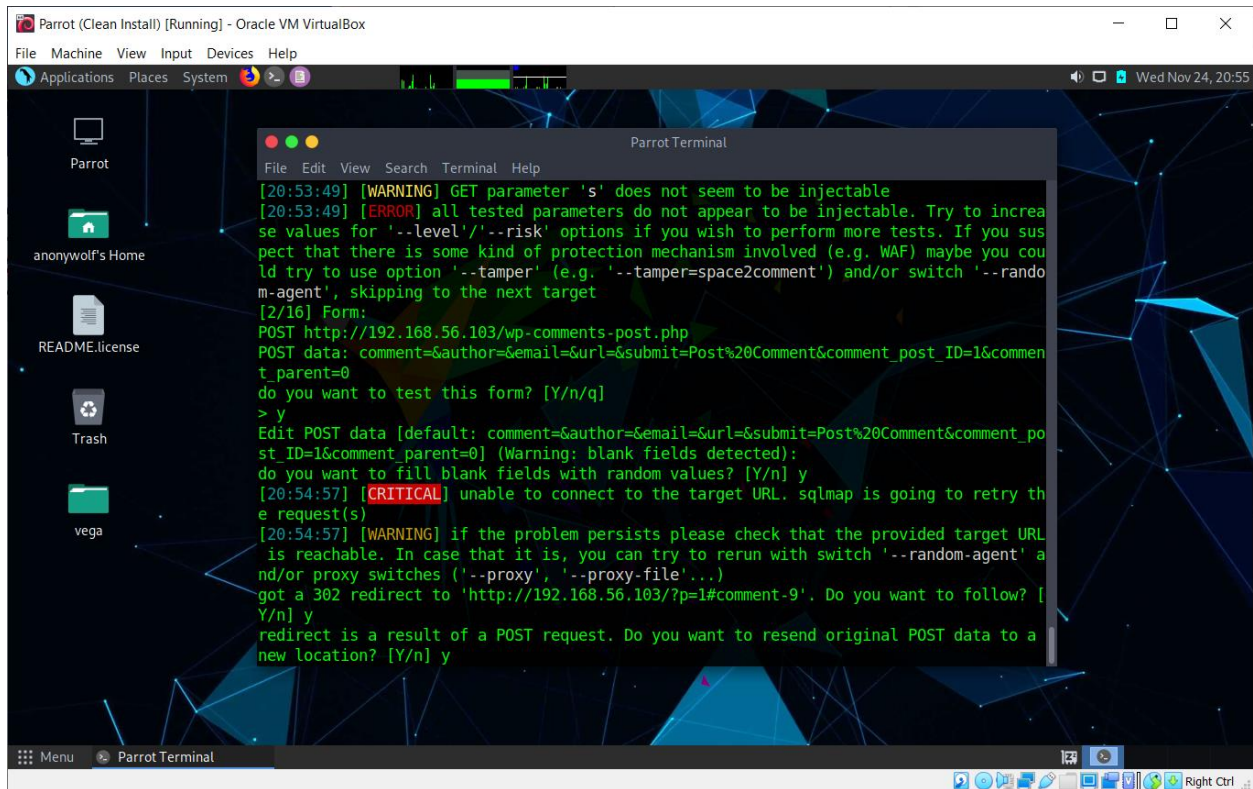


Figure 2-sqlmap

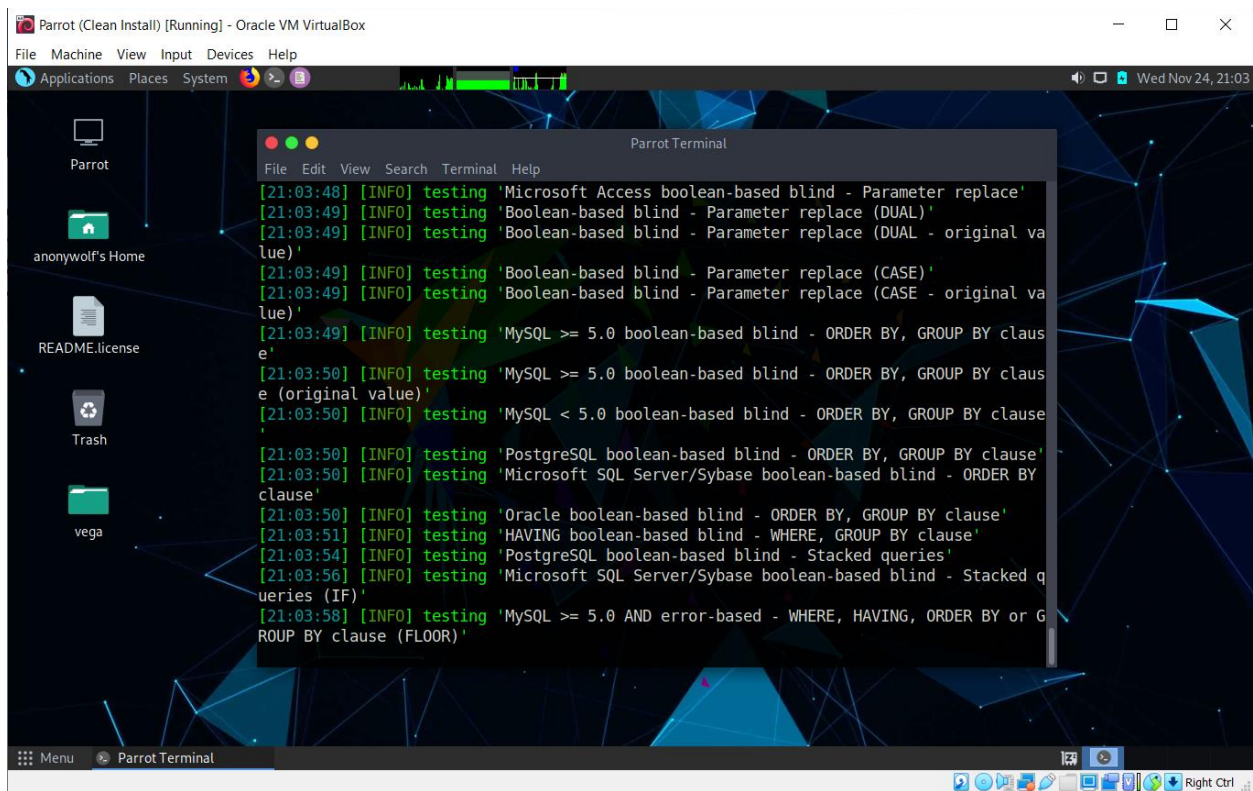


Figure 3-sqlmap

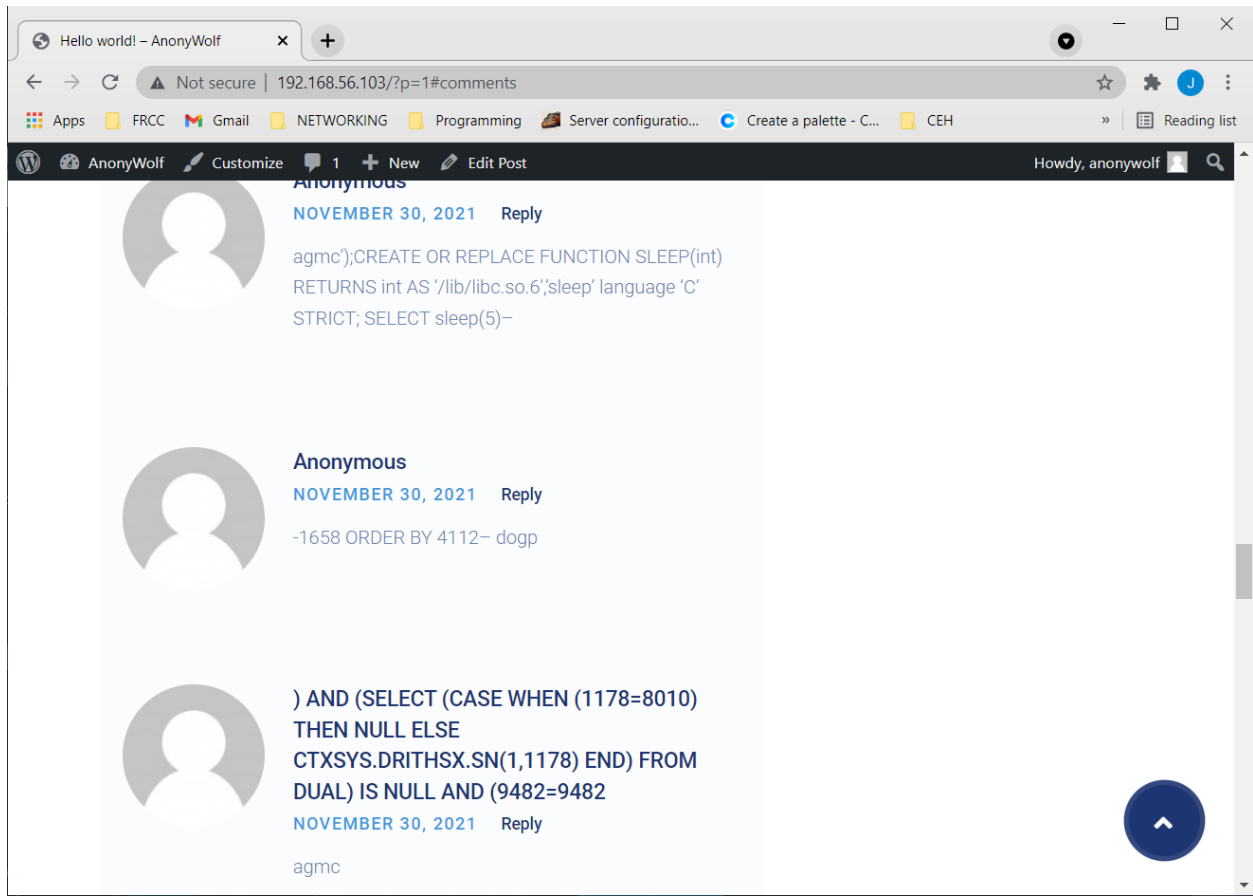


Figure 4-sqlmap\_results