Anonymous

https://hackerdna.com/labs/anonymous

Author: Dorothy Spencer
Date: 02/16/2026
Platform: HackerDNA
Difficulty: Easy Level
Flag Points: 10 (1 flag worth +10 pts)
**Additional points can be earned by doing a community Writeup*
Objective: This version of the Anonymous lab demonstrates how insecure configurations in legacy services like FTP can expose sensitive information. Even when a website appears empty or locked down, underlying services may still be accessible — and misconfigured.

Your mission in this lab is to:

Identify the exposed service
Authenticate using anonymous access
Retrieve the flag from the server
Understand why this misconfiguration is dangerous

Vulnerability Summary:
The target machine exposes an FTP service that allows login using the classic:

- **Username:** anonymous
- **Password:** *(blank)*

This is a common misconfiguration in older systems. Once authenticated, the server allows access to sensitive files — including the flag needed to complete the lab course.

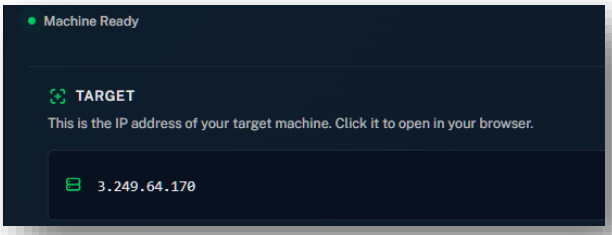
The vulnerability is a combination of:

- Anonymous FTP access enabled
- No directory restrictions
- Flag stored in a publicly accessible location
- Active Mode FTP causing client-side failures

Target Example: https://hackerdna.com/labs/anonymous

Start the Machine: Once the machine loads, you're given a target IP.

Example Below.



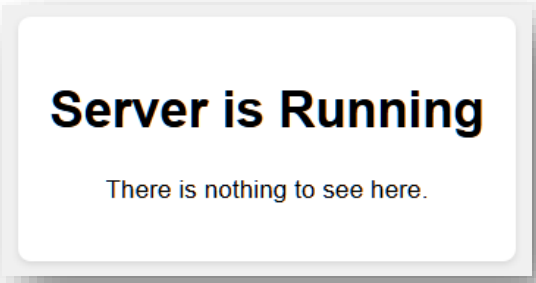
When the machine loads, navigating to the target IP shows a simple page:

Server is Running

There is nothing to see here.

This is a decoy.

The real entry point is not the web server — it's the FTP service running on the same machine.



Initial Enumeration Attempts

Like many beginners (and honestly, many pros), I first tried:

- Directory enumeration
- File enumeration
- Viewing page source
- Searching for hidden content in DevTools

None of these revealed anything useful.

This is because the web server is intentionally empty. The real vulnerability is on a different port.

Exploitation - Connecting to the FTP Service (The "Anonymous" Login)

Using PowerShell, I navigated to my Downloads folder and attempted to connect:

**Different IP shown here as to that above since I restarted my Lab a few times.*

```
ftp 52.215.93.96
```

**Different IP shown here as to that above since I restarted my Lab a few times.*

The Technique: We used the standard "***Anonymous FTP***" credentials:
Username: anonymous
Password: (Empty/Enter)
The Result: 230 Login successful. This confirmed that the “anonymous” FTP login was enabled.

Pivoting to PowerShell WebClient (The Fix)

To bypass the Active Mode problem, I switched to PowerShell’s WebClient, which uses Passive Mode by default.

Passive Mode works because:

- *I can* initiate the connection
- The server does not need to connect back
- Firewalls allow outbound connections

***Crucial detail:** Unlike the basic FTP client, WebClient defaults to **Passive Mode**. It tells the server: "Don't call me, I'll call you." This allows the file to pass through your firewall because your computer initiated the request.*

Here’s the exact sequence:

Step-by-Step Script Breakdown:

```
# Step A: Define the source
$url = "ftp://52.215.93.96/flag.txt"

# Step B: Initialize the web engine
$wc = New-Object System.Net.WebClient

# Step C: Execute the download
$wc.DownloadFile($url, "$home\Downloads\flag.txt")
```

Technical Explanation for each line:

Line A: I create a variable called $url. By using the ftp:// prefix, to tell PowerShell exactly which protocol to use.

Line B: I create an instance of the WebClient class. This is a powerful .NET object that handles complex networking tasks (like handshakes and error checking) automatically.

Line C: This is the execution phase. DownloadFile takes two arguments: ***(Where it's coming from, Where it's going).***

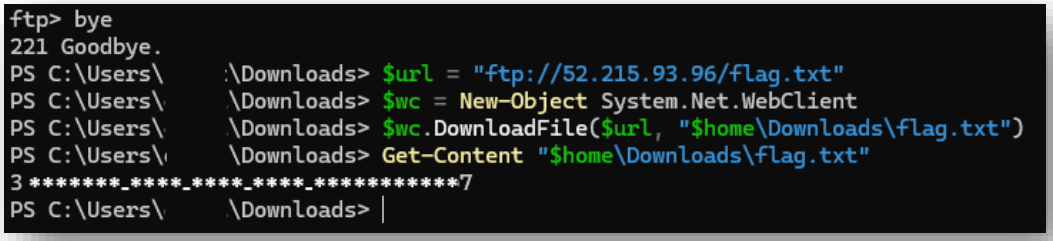
The "Verification" Phase

Once the file was on my local disk, I used a native PowerShell command to read it without opening Notepad.

The Command: `Get-Content "$home\Downloads\flag.txt"`
The Function: `Get-Content` reads a file line-by-line and outputs it directly into your terminal.
The Benefit: In a security context, this is safer and faster than opening files in a GUI editor, as it prevents accidental metadata changes.

Retrieve the Flag

Copy the flag exactly as shown in screen shot below.



Common Pitfalls:

I wasted time enumerating the web server. The web server is intentionally empty. The vulnerability is on FTP, not HTTP.

Common Pitfalls I Hit (and How I Solved Them)

I wasted time enumerating the web server — it's intentionally empty.
I initially tried to retrieve the file using the built-in FTP client, but Active Mode caused a 425 error.
I tried running PowerShell commands inside the FTP shell — which doesn't work.
The real vulnerability was anonymous FTP access, not anything on the web server.

What I learned here:

- Recognize when a web server is a decoy
- Identify exposed services beyond HTTP
- Understand Active vs. Passive FTP modes
- Use PowerShell to retrieve files from FTP
- Troubleshoot failed transfers
- Pivot quickly when a tool or protocol fails