Web Application Recon Using Browser DevTools (Chrome / Edge / Brave / Firefox)

Playbook for Bug Hunters (Beginner Intermediate)

November 6, 2025

Abstract

This playbook expands the DevTools walkthrough into a detailed, copy-paste friendly LaTeX document you can print or save as a PDF. It explains exactly what to click, what to type, how to extract endpoints, how to intercept and modify requests client-side, and includes short exercises and a compact checklist.

Contents

1 Getting started opening DevTools

1. Open the target site in Chrome / Edge / Brave. (Firefox UI is similar but labels differ slightly.)
2. Open DevTools:
   - Windows/Linux: `F12` or `Ctrl+Shift+I`
   - macOS: `Cmd+Option+I`
3. Switch to the **Network** panel. If it's empty, press `F5` (reload).

2 Network tab UI anatomy and quick recipes

2.1 Columns you must know

Name	Path or resource requested; click to inspect.
Status	HTTP status codes (200, 401, 403, 404, 500). Non-200s often indicate interesting behavior.
Type	Resource type: `fetch`, `xhr`, `document`, `script`, `websocket`. Use this to filter.
Initiator	Shows which script or resource triggered the request. Click to jump to that source.
Size/Time	Useful for performance analysis and noticing large payloads or slow endpoints.

2.2 Essential checkboxes and filters

- `Disable cache` forces fresh requests, important while testing.
- `Preserve log` keep the log across navigations (useful for redirects auth flows).
- Filters: click `XHR` or `Fetch` for APIs, `WS` for WebSockets, `Doc` for navigation, `JS` for scripts.
- Text filter bar: match hostnames, `graphql`, `api`, or parameter names (e.g., `token`).

3 Inspecting a single request (step-by-step)

Click any row to reveal subpanels. Inspect the following in order:

3.1 Headers

- **General:** full Request URL and method (GET/POST/etc.).
- **Request Headers:** Authorization, Cookies, Content-Type, custom headers (`X-...`).
- **Response Headers:** `Set-Cookie`, `Server`, caching headers. Server strings may leak technology.

3.2 Payload / Request

If method is POST/PUT/PATCH view Form Data or raw JSON. This reveals parameter names and shapes.

3.3 Response / Preview

Review JSON trees, HTML, or images. Look for tokens, URLs, error messages or internal hints.

3.4 Cookies

Which cookies were sent and set? Session cookies and CSRF tokens live here.

4 Copying requests and replaying

4.1 Copy as cURL

Right-click `Copy` `Copy as cURL`. Paste into a terminal to reproduce the request exactly. Example:

```
curl 'https://example.com/api/user' \
  -H 'Authorization: Bearer eyJ...' \
  -H 'Content-Type: application/json' \
  --data-raw '{"id":123}' --compressed
```

Modify URL, headers, or payload to test parameter tampering.

4.2 Copy as fetch

Copy into the Console to replay from the browser context (useful because cookies and CORS behave like the real client):

```
fetch("https://example.com/api/user", {
  "headers": { "accept": "application/json", "content-type": "application/json"
  "body": "{\"id\":123}",
  "method": "POST",
  "credentials": "include"
});
```

5 Finding endpoints in JavaScript (Sources panel)

5.1 Open files and global search

1. Open **Sources** or press `Ctrl+P` to open files by name.
2. Use global search (`Ctrl+Shift+F`). Run searches for: `fetch(`, `XMLHttpRequest`, `/api/`, `graphql`, `new WebSocket`, `Authorization`, `API`$_{U}RL$, `baseUrl`.

`2. For minified scripts click the {}` *`Pretty print`* `button to reformat and search again.`

5.2 Common patterns to look for

- Initialization objects: `window.`$_{CONFIG}$,`process.env`,`API`$_{U}RL$.
- `Template strings building endpoints: e.g., `` ` ``$API_{B}ASE/v1/users/$`id`` ` ``. Search for backticks.`

6 Breakpoints to intercept modify requests

6.1 XHR / fetch breakpoints

In **Sources** panel right sidebar **XHR/fetch Breakpoints**:

- Add a breakpoint with a string such as `api/user` or `graphql`.
- Trigger the app action. DevTools pauses on the JS line creating the request.
- Inspect local variables, change them in the Console, alter payloads and headers, then press resume.

Use cases: remove client-side CSRF header; change body before send; see how UI builds a request.

6.2 DOM and Event Listener breakpoints

Right-click an element in **Elements** `Break on subtree modifications / attribute modifications`. Use Event Listener breakpoints to pause on `click`, `submit`, or timers.

7 WebSocket and SSE analysis

7.1 WebSocket (WS) frames

Filter to `WS` in Network. Select a WebSocket request and open the *Frames* tab to inspect sent/received JSON.

7.2 Server-Sent Events

SSEs show as `event-stream`. Click the request to see streamed data in real time.

8 Application panel storage and source maps

8.1 Where secrets and endpoints hide

- **Local / Session Storage**: look for keys like `token`, `API`$_URL,$`user`.
- **IndexedDB:** `contains structured caches and sometimes responses.`
- **Service Workers:** `may intercept or rewrite requests.`
- **Manifest and source maps:** `manifest.json and source maps (*.map) can reveal original source and endpoints.`

9 GraphQL-specific tips

- GraphQL requests are usually POSTs to `/graphql` with JSON body: `"query":"..."`. Inspect the query string in **Payload**.
- Copy queries to reproduce and extend (use authorized testbeds only). Look for names of types and fields to craft new queries.

10 Quick extracts and automation

10.1 Extract endpoints via Console

Paste this into Console to list unique resource URLs:

```
(() => {
  const urls = performance.getEntries().map(e => e.name);
  return [...new Set(urls)].join('\n');
})();
```

10.2 Export HAR

Right-click network table `Save all as HAR with content`. HAR files can be parsed by tools to build endpoint lists.

11 Hands-on walkthrough (example flow)

1. Open target site; enable `Disable cache` and `Preserve log`; reload.
2. Filter `XHR`. Trigger a search box; note `GET /api/search?q=...`. Copy as cURL, replay and modify query param `q` to test input handling.
3. Find profile update flow: inspect the `POST /api/user/update`. Set XHR breakpoint for `/api/user/update`. Pause, inspect `X-CSRF` header value (maybe from localStorage), remove it in Console and resume observe server response for CSRF enforcement.
4. Search Sources for `fetch('/api/user/update')`, examine client-side validation logic and input sanitization.

12 Beginner exercises

1. On an authorized lab, capture the login request, copy-as-cURL, and replay while changing the username parameter. Observe error messages.
2. Find a page using GraphQL. Copy the query and try to request additional fields (read-only).
3. Find a minified JS file, prettify it, and search for `fetch(`. Identify the base URL. Document it.
4. Set an XHR breakpoint, change the request body, and resume. Document server responses.

13 Ethics and safety

Always have written permission. Prefer read-only probes. Avoid destructive API calls (`DELETE`, `PUT`) unless explicitly authorized.

14 Compact checklist

- Open DevTools Network `Disable cache`, `Preserve log`, filter `XHR`. Reload.
- Search Sources for `fetch`, `XMLHttpRequest`, `graphql`, `api`, `token`.
- Inspect **Headers**, **Payload**, **Response**. Copy as cURL/fetch.

- Check Application tab: LocalStorage, SessionStorage, Cookies, IndexedDB, Service Workers.
- Set XHR/fetch breakpoints; test WebSockets; export HAR.

15 Appendix: Useful copy-paste snippets

15.1 Console: list unique URLs

```
let urls = performance.getEntries().map(e=>e.name).filter((v,i,a)=>a.indexOf(v)=
console.log(urls.join('\n'));
```

15.2 cURL template (modify)

```
curl 'https://TARGET/api/endpoint' \
  -H 'Authorization: Bearer REPLACE_TOKEN' \
  -H 'Content-Type: application/json' \
  --data-raw '{"param":"value"}' --compressed
```

15.3 fetch template (browser console)

```
fetch('https://TARGET/api/endpoint',{
  method:'POST',
  headers:{'Content-Type':'application/json','Authorization':'Bearer REPLACE'},
  credentials:'include',
  body:JSON.stringify({param:'value'})
}).then(r=>r.json()).then(console.log)
```

16 Want more?

If you want I can also:

- Provide a LaTeX version compiled to PDF (I can generate the PDF for you).
- Produce a small Python/Bash script that parses a HAR file and extracts unique API endpoints.
- Walk you step-by-step through a specific PortSwigger or OWASP Juice Shop lab.

Prepared for safe, ethical testing only.