⚡Copilot vs WebGoat — Shock Prompt Pack ⚡

Read me first! WebGoat is a deliberately vulnerable app. Run it *only* on `localhost`, inside an isolated Docker container, and never against production data. These prompts are for **educational red-team demonstrations**—use them responsibly.

How to run the show

1. Fire up WebGoat in Docker (e.g. `docker run -p 8080:8080 webgoat/webgoat-8.2`).
2. Open the project folder in **VS Code** with **GitHub Copilot Chat** enabled.
3. Copy-paste each prompt (one at a time) into Copilot Chat, hit Enter, then *act* on the response live.

🌓The Prompts

Prompt #1 — Codebase Autopsy

```
You are an adversarial security analyst.
Scan the entire src/main/java tree and list every controller or servlet that:
 • accepts user input (params, headers, body) AND
 • builds SQL via string concatenation OR feeds input to Runtime.exec.
Return a table: {Class}{Method}{Line}{Why it’s explosive}.
```

Prompt #2 — Find the Weakest Link

```
From that list, pick the *smallest* method that stitches a SELECT query from
username + password.
Explain the exact taint flow from request to query execution, including line
numbers.
Highlight the single line where the bomb goes off.
```

Prompt #3 — Bypass the Gates

```
Craft an SQL-injection username payload that will log me in as admin without
knowing the password.
Output ONLY the payload string plus a one-line rationale.
```

Prompt #4 — Spray & Pray

```
Generate a UNION-based SQL-injection string that dumps every username and
password hash in the users table.
Return the ready-to-paste payload.
```

Prompt #5 — Illuminate the XSS Minefield

```
Search all JSP/Thymeleaf templates for unescaped echoes of request parameters or
DB values.
Return {File}{Line}{Variable}{Why it’s pure XSS gold}.
```

Prompt #6 — Plant a Persistent Surprise

```
For the first vulnerable template above, craft a stored-XSS payload that pops
`alert(document.cookie)` and evades any naïve filters.
Give ONLY the payload to save.
```

Prompt #7 — Shell-Factory Recon

```
Locate Java code that concatenates user input into Runtime.getRuntime().exec or
ProcessBuilder.
Summarise {File}{Line}{Command template}{Untrusted variable}.
Explain how an attacker can bolt on arbitrary commands.
```

Prompt #8 — Crack the Shell

```
For that exec hotspot, craft an input that appends "&& cat /etc/passwd" on
Linux.
Show the final command string exactly as it will execute.
```

Prompt #9 — Tunnel Through the Directory

```
Identify any endpoint that reads files based on a request parameter path.
Provide a curl example that retrieves ../../../../etc/hostname via that
endpoint.
```

Prompt #10 — Flip to Blue Team

```
Rewrite the SQL-injection method from Prompt #2 using Spring JdbcTemplate with
prepared statements.
Return the patched code ONLY.
```

🌕Mic-Drop Finale

End the demo by rerunning the attack after applying Copilot's fix—watch it *fail*—then remind the audience: **"AI magnifies *intent*. Use it to break, but above all, to build safer software."**