CareerSwami · Cybersecurity & Security Audits

FREE TWO-WEEK INTERNSHIP

Project 1 — Audit a registration app you built yourself, and only that

Eight weaknesses in a fictional configuration and six lines of log. The findings are not the hard part — they are handed to you. The hard part is scoping the work legally, rating the findings honestly, and never claiming more than the evidence shows.

2 weeks	1 project	6–8 hrs
START TO SUBMIT	ONE SUBMISSION	PER WEEK

WHAT IS IN THIS BRIEF

1. The situation — the problem you are being handed
2. The four disciplines this audit is testing — the cases your work must survive
3. What you are given — your data and starting files
4. What finished looks like — the two things you hand in
5. How the two weeks run — week by week
6. Before you start — rules, tools and your starter kit
7. How the work is judged — the marking scheme
8. How to submit — one link, pasted into your dashboard

Read 1 to 4 before you do anything. They take five minutes and they are what the marking scheme is built on. Sections 5 to 8 are the how — come back to them once you know what you are solving.

1 The situation

Treat this as a real client brief, and read the scope section before anything else. In security, working outside your authorisation is not a technical mistake — it is the end of a career, and in many places a criminal offence.

You are auditing a fictional event-registration application. Your authorised scope is exactly one thing: files and applications you create yourself, locally, on your own machine. Nothing else. Not a university system, not a website that looks like a practice target, not a friend's project.

The configuration you are given has eight problems in it: debug mode on in production, a password of demo123, a cookie without the Secure flag, no limit on login attempts, roles enforced only in the interface, uploads accepting any extension, an unencrypted backup, and unversioned dependencies.

There is also a six-line log. Three failed logins for userA at nine, nine-oh-one and nine-oh-two. A success at nine-oh-three. An export_all at nine-oh-four performed by an account with viewer permissions. An upload

of invoice.exe at nine-oh-five. That sequence looks exactly like a compromise — and the whole of your judgement is tested by how carefully you describe what it does and does not establish.

> WHAT YOU ARE BEING ASKED TO DO
>
> **Write the rules of engagement, map how data moves through this application, build a threat model, and turn the eight configuration weaknesses into rated findings with evidence you can point at.**

2 The four disciplines this audit is testing

Finding weaknesses is the easy half and the one that is handed to you. These four are what separates an audit report from a list of complaints.

Scope, written down before you touch anything

Only what you built, locally, on your own machine.

Where auditors get it wrong — "I was only looking" is not a defence anywhere. Rules of engagement exist so that the boundary is agreed in writing before the work starts, not reconstructed afterwards. A report without them is not a professional document.

Clues versus proof

The log shows a pattern consistent with a compromise.

Where auditors get it wrong — Three failures then a success is consistent with a brute-force attack and equally consistent with someone who forgot their password. The export by a viewer account is the stronger signal. Write what the evidence supports and, separately, what it does not — a report that states "the account was compromised" has overclaimed.

Severity, rated by a stated method

Eight findings, not equally serious.

Where auditors get it wrong — Rating by instinct gives eight criticals and is useless for prioritisation. Pick a method — likelihood against impact, or a published scoring system — state it, and apply it consistently. A rating you cannot explain will be argued down by the first developer who reads it.

Evidence a third party could check

Each finding pointing at a specific line or observation.

Where auditors get it wrong — "Weak password policy" with nothing attached is an opinion. The same finding with the configuration line, the affected component and the reproduction note is something a developer can act on and a manager can verify. Reports get ignored for exactly this reason.

Note which of the eight findings the log actually relates to — unlimited login attempts and UI-only role enforcement — and which it says nothing about. Connecting an unrelated finding to the incident to make it sound urgent is the most common integrity failure in this work.

3 What you are given

Eight weaknesses and six log lines, all fictional, all local. The scoping, the rating method and the honesty about limits are the assignment.

The eight configuration weaknesses

`DEBUG=true` `password=demo123` `cookie Secure=false` `unlimited login attempts` `roles enforced in UI only` `uploads accept all extensions` `unencrypted backup` `unversioned dependencies`

One finding, written properly, so the standard is clear

`Finding`	Authorisation enforced in the interface only
`Evidence`	Config: roles enforced only in UI; log 09:04 export_all by a viewer account
`Affected`	Export endpoint
`Likelihood`	High — requires only a direct request, no special tooling
`Impact`	High — full registration data disclosure
`Rating`	Critical, under the method stated in section 3
`Does not establish`	That data was exfiltrated, or that the account was compromised rather than misused

Four stages of the audit

`Scope` `Model` `Test` `Report`

Authorised scope is strictly limited to files and applications you create locally yourself. No third-party systems, no real credentials, no phishing, no social engineering, no destructive testing, and no scanning of

anything you do not own. All configuration and log data here is fictional. Clues are not proof of exploitation and must never be written as though they were.

4 What finished looks like

At the end of two weeks someone should be able to read your pack, see exactly what you were allowed to do, and act on what you found.

- **Rules of engagement, a data-flow sketch and a threat model** — what is in scope, what is explicitly out, what you will not do and who to contact if something goes wrong; a sketch of how registration data moves and where it rests; and a threat model naming who might attack this, what they would want, and which weaknesses would help them.
- **A checklist, a stated rating method and an evidence register** — what you checked and what you deliberately did not; the severity method you chose, written out so a reader can apply it themselves; and a register where every finding carries the evidence, the affected component and what it does not prove.

Write the "what this does not prove" column before you write the recommendations. It is the discipline the rest of the report depends on.

5 How the two weeks run

Work through it in two parts — but you submit **once**, at the end, not once per week.

PART 1 · WEEK 1

Agree the boundary and model the threat

Write the rules of engagement first — before opening a single file. Then map how data moves through the application and work out who would want it and how they would try.

Produce

- Rules of engagement: in scope, out of scope, prohibited actions, contact
- A data-flow sketch: where registration data enters, moves and rests
- A threat model: likely actors, what they want, which weaknesses help them

> Your rules of engagement must say explicitly that no third-party system is in scope and no destructive testing will take place. A threat model that lists only external attackers has ignored the one the log points at — a legitimate account doing something it should not.

PART 2 · WEEK 2

Rate the findings, and record the evidence

Turn the eight weaknesses into findings with a consistent rating method. For each, record what you observed, what it means, and — the column that matters most — what it does not establish.

Produce

- A checklist of what was examined and what was deliberately not
- A rating method, written out so a reader could apply it
- An evidence register: finding, evidence, affected component, limits

> Not every finding can be critical. If your ratings do not separate, the method is not working. The log-related findings must distinguish what the log is consistent with from what it proves.

6 Before you start

- Write the rules of engagement before you examine anything. In a real engagement, the order is the same and it is not negotiable.
- State your severity method and apply it consistently. Eight criticals is the same as no ratings at all.
- Separate what the evidence supports from what it suggests. Every finding gets a "does not establish" line.
- The log is consistent with several explanations. Say which, rather than choosing the most dramatic one.
- Recommendations must be specific to a component. "Improve security" is not a recommendation; "enforce role checks server-side on the export endpoint" is.

Your tools

A text editor, a document editor and the local mock application in the starter kit. No scanner, no exploitation framework and no paid tool is required — and none is to be pointed at anything you did not build. No ID document is ever required.

CS_cybersecurity-security-audits_Starter_Kit.zip — dummy-config.txt, local_mock.py, test_observed_weaknesses.py, a rules-of-engagement template, an evidence register, a threat model template, and a file index.
https://careerswami.com/wp-content/uploads/careerswami-projects/CS_cybersecurity-security-audits_Starter_Kit.zip

7 How the work is judged

WHAT IS ASSESSED	WEIGHT
Problem and requirements — did you understand what you were solving	20%
The core artefact — the thing you actually made	40%
Validation and evidence — how you know it works	25%
Clarity and reproducibility — can someone else open and follow it	15%

Strong	Complete, correct, evidence-backed and easy to inspect.
Developing	A useful partial result, with the gaps explained.
Needs revision	Core deliverable missing, claims unverified, or the work cannot be opened.

Credit reflects demonstrated work, not visual polish. A small, honest result beats a large, unsupported one.

What counts as evidence

expected result → actual result → the configuration line, log entry or observation → conclusion. Then say what the evidence does not establish.

Why this one is worth doing. Security hiring filters hardest on judgement, not on tool knowledge. Anyone can run a scanner; the question is whether you will scope the work in writing, rate findings by a method you can defend, and refuse to say the account was compromised when the evidence only says the pattern is consistent with it. An audit report that distinguishes clue from proof is rarer than a certification, and worth more in the room.

8 How to submit

One link, pasted into your dashboard

1. Put your work somewhere with a link — Google Drive, Figma, GitHub, Notion, Loom. Whatever fits the work.
2. **Set it so anyone with the link can view.** A locked link is the single most common reason a submission comes back.
3. Include a short README or cover page saying what to open first, and where each deliverable lives.
4. Open your dashboard, paste the link into the submit box, and add a line or two on anything you want the reviewer to know.

Name files `CS_cybersecurity-security-audits_YourName_v1`. Keep your own copy. A screenshot on its own is not enough for anything interactive. A video is optional — annotated screenshots or narrated slides are fine.

Submitted and heard nothing, or the link came back? Email **support@careerswami.com** with the email address you enrolled with.

Work honestly. Use fictional data and your own work. Credit anything you reused, and disclose AI assistance — including what you checked yourself. Never invent interviews, users, tests or results. Nothing here requires you to spend money, contact real organisations, or deploy anything publicly. Never upload identity documents, real customer records or credentials.

Stuck, confused, or something is broken? Write to us.

`support@careerswami.com`

Or WhatsApp **+91 7383138418**. We answer on working days. Use it for anything at all — a part of this brief that does not make sense, a starter file that will not open, a dashboard link that has stopped working, a deadline you need moved, or a submission you are not sure about.

To get a useful answer fast, spend twenty minutes narrowing the problem down first, then tell us four things: what you expected, what actually happened, what you already tried, and your one specific question. Include your track name and the email you enrolled with.

Asking is not penalised. Submitting partial work with a clearly stated blocker always scores better than a finished answer you copied, and better than a silent no-show.

Optional stretch

Optional, and only once the main work is done — cap it at an hour. Write the remediation plan in priority order with effort estimates, or build a small local check that detects one of the findings automatically. Label it clearly as a stretch.