Hack Your Way Into Security Engineering

Beating the Hiring Pipeline

Who am I?

Monish Alur Gowdru

Security Engineer, Ethical Hacker & Open Source Sleuth at UltraViolet Cyber.

Agenda

- Who this is for
- Getting interviews
- Cracking the interview
- Common mistakes
- Q&A

Who this session is for

If you're in one of these three buckets, the next 43 minutes are for you.

01

The aspirant

Breaking into cybersecurity, or targeting your first Security Engineer role.

02

The switcher

Already in tech or in a security-adjacent seat, aiming to move up or across.

03

The stuck

Applying constantly. No callbacks, or interviews that stall at the technical loop.

SECTION 01

Getting Interviews

Your resume is an exploit. The ATS and the recruiter are the target.

Where applications actually die

Four gates between 'Apply' and a human conversation. Most people only optimise for the last one.

APPLY
You submit

›

ATS PARSE
Machine reads

›

RECRUITER
Seconds of scan

›

HIRING MGR
Relevance check

Does this map to my open req?

›

INTERVIEW
You're in

Now the fundamentals matter.

Optimise for the gate you're failing at.

No callbacks at all → it's the resume and the targeting. Screens but no loops → it's your story. Loops but no offer → it's depth.

Two resume mistakes that kill you

I see these on almost every resume that gets rejected.

MISTAKE 01

A duty list, not an impact list

"Responsible for performing vulnerability assessments."

That's a job description, not evidence. It tells me the role existed - not that you were good at it. Every reviewer has read that exact sentence a hundred times.

MISTAKE 02

One resume, two hundred applications

Generic resumes match nothing well. The ATS scores you low on keywords, the recruiter sees no obvious fit, and you never find out why. Volume without targeting is just faster rejection.

Also common: three pages · a skills soup of 60 tools · certifications above experience · a photo · PDF-as-image that parses to nothing

Fix the top two first - they account for most silent rejections.

Rewrite every bullet with one formula

- WEAK

Responsible for performing vulnerability assessments on internal applications and reporting findings to developers.

+ STRONG

Ran authenticated DAST plus manual testing across 40+ internal apps; found and drove remediation of 3 critical auth bypasses, cutting median fix time from 45 to 12 days.

THE FORMULA

ACTION VERB
Ran, built, automated, led

+

WHAT + SCALE
40+ apps, 3 services, 12 engineers

+

TOOL / TECHNIQUE
Burp, Semgrep, Terraform, STRIDE

+

MEASURABLE OUTCOME
Time saved, risk removed, bugs fixed

Numbers must be yours and real. If you can't defend how you measured it, don't put a number on it.

Structure: a template that parses cleanly

One column. No tables, no text boxes, no graphics. Standard section names the parser expects.

HEADER

Name · target title · city · email · LinkedIn · GitHub. No photo, no full address.

SUMMARY

Two lines, rewritten per role. Who you are, what you secure, what you're aiming at.

SKILLS

Grouped and honest: AppSec · Cloud · IaC · Languages · Tooling. Only what you can be grilled on.

EXPERIENCE

Reverse chronological. 3-5 impact bullets per role using the formula. Most recent gets the most space.

PROJECTS / LABS

Your leverage if experience is thin. Link the repo or the writeup. Two lines each.

EDUCATION + CERTS

Last. One line each. Certs are a tiebreaker, not a headline.

One page under 8 years of experience. Two pages maximum, ever.

Align to the role, mine the keywords

The job description is the answer key. Treat it like a scope document.

01

Extract the nouns

Pull every tool, framework, cloud, language and compliance term out of the JD. Those are your keywords.

02

Map to real evidence

Each keyword must connect to something you actually did. No evidence, no keyword.

03

Mirror their vocabulary

They wrote 'SAST' - don't write 'static analysis'. They wrote 'IaC security' - don't write 'Terraform scanning'. Match their words.

04

Put them where they count

Title line, summary and experience bullets. A keyword buried in a skills list carries far less weight.

Using AI to build the resume - properly

The model is a fast editor and a brutal reviewer. It is not a ghostwriter.

USE IT FOR

- Gap analysis: JD vs your resume, ranked
- Keyword extraction and mapping
- Three rewrites of a bullet you already wrote
- Tightening: same meaning, 30% fewer words
- Adversarial review: "what would you attack in this interview?"

NEVER LET IT

- Invent metrics, tools or employers
- Write in a voice you can't reproduce out loud
- Add a project you didn't build
- Produce the final draft, you always do the last pass

PROMPT PATTERN

```
You are a hiring manager for [role]. Here is the JD and my real experience bullets.
1) Rank the gaps.  2) List their keywords I can honestly claim.  3) Rewrite each bullet 3 ways, action + scale + tool + outcome.
Do not invent facts. Flag anything you think I cannot defend in a deep-dive interview.
```

Local LLM workflow: Ollama + open models

Same automation, none of the data leaving your laptop.

INPUT

Job descriptions

Saved JDs, feed exports, careers-page text, dumped into a local folder.

RUNTIME

Ollama, local models

A 7-14B open-weight model on your own hardware. No API key, no per-token bill.

LOGIC

Prompts + your profile

A skills matrix and real bullet bank you maintain, injected as context.

OUTPUT

Fit score + tailored draft

Ranked roles, keyword map, bullet variants, short cover note, tracker row.

PRIVATE

Resume, PII, salary and target list stay on disk.

CHEAP

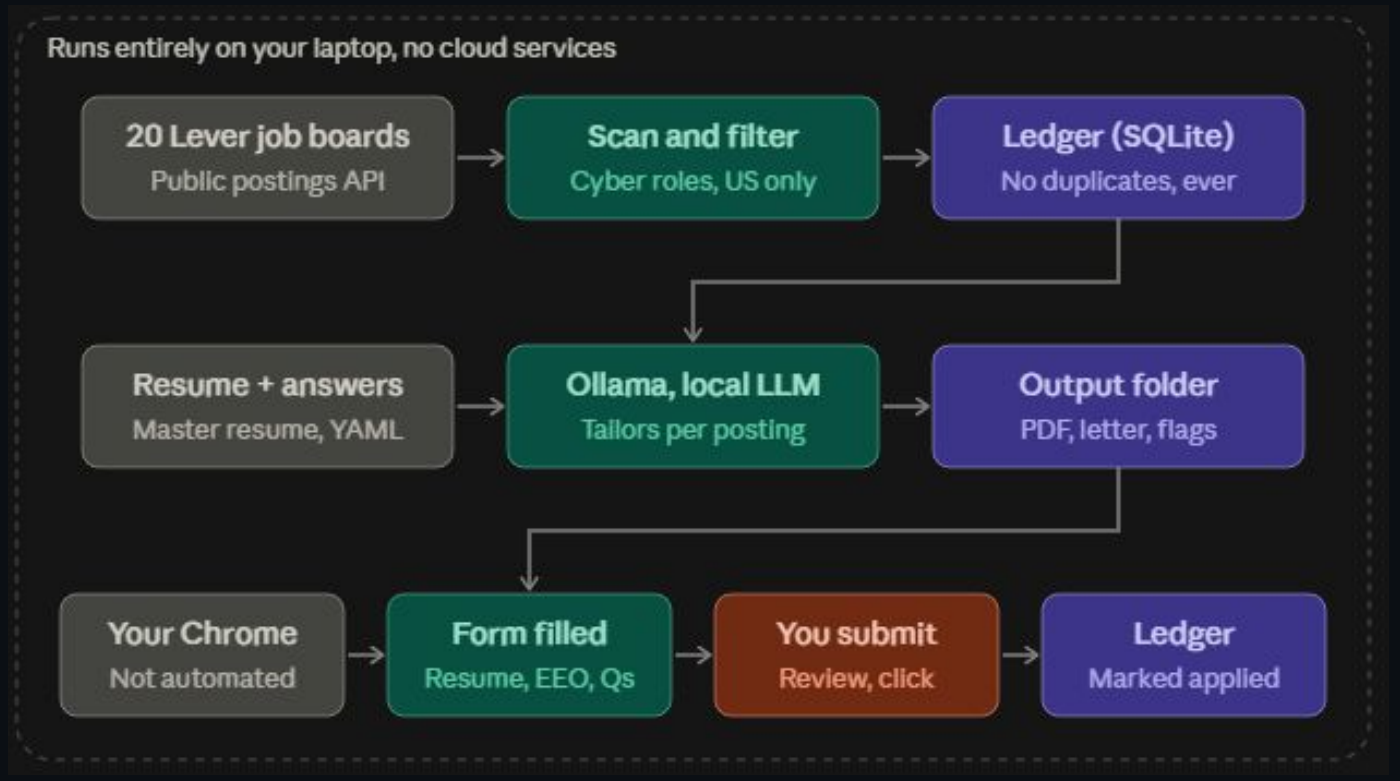
Hundreds of JDs, no metered API cost.

BATCHABLE

Queue it overnight, review a ranked list with coffee.

YOURS

Tune the prompts and the scoring to your own bar.


Runs entirely on your laptop, no cloud services
20 Lever job boards
Public postings API
Scan and filter
Cyber roles, US only
Ledger (SQLite)
No duplicates, ever
Resume + answers
Master resume, YAML
Ollama, local LLM
Tailors per posting
Output folder
PDF, letter, flags
Your Chrome
Not automated
Form filled
Resume, EEO, Qs
You submit
Review, click
Ledger
Marked applied

What have you actually built?

Shout out a project, a lab, a tool, a writeup. Anything hands-on.

Two or three volunteers. Sixty seconds.

Whatever you just said out loud belongs on your resume.

Side projects that earn callbacks

Thin experience is survivable. Thin evidence is not.

01

VULN APP + THREAT MODEL

Build a deliberately broken app, then write the threat model and ship the fixes. Shows offense and defense in one artifact.

02

DETECTION LAB

Small home lab, simulate a technique, write the Sigma rule that catches it. Publish the rule and the false-positive analysis.

03

LLM SECURITY HARNESS

A prompt-injection and jailbreak test suite pointed at your own chatbot. Score it, log results, publish the methodology.

04

TRIAGE AUTOMATION

A bot that pulls SAST findings, de-duplicates them and drafts a triage note locally. Solves a problem every AppSec team has.

Ship it publicly. A repo with a real README, or a writeup with your methodology and what you got wrong. Unpublished work is invisible work.

SECTION 02

Cracking the Interview

Fundamentals, hands-on proof, and the stories you tell about both.

Know the process before you're in it

Each stage screens for something different. Prepare for the stage you're actually at.

01	**RECRUITER SCREEN**	15-30 min	Role fit, comp expectations, logistics, a clean two-minute background story.
02	**TECHNICAL SCREEN**	45-60 min	Fundamentals breadth plus one deep dive. Sometimes light live coding.
03	**LOOP: DEPTH**	60 min	Vulnerability classes, exploitation, mitigations, code review.
04	**LOOP: DESIGN**	60 min	Secure design and threat modeling. "Design X." "Threat model this."
05	**HIRING MANAGER**	30-60 min	Behavioral, judgment, collaboration, your questions, closing.

Ask the recruiter for the exact loop structure and who you're meeting. They will almost always tell you.

Master the fundamentals: both Top 10s

OWASP Top 10 and OWASP Top 10 for LLM Applications. In 2026, the second one is no longer optional.

OWASP TOP 10 (WEB)

Broken access control · cryptographic failures · injection · insecure design · misconfiguration · vulnerable components · auth failures · integrity failures · logging failures · SSRF

OWASP TOP 10 (LLM APPS)

Prompt injection · sensitive information disclosure · supply chain · data and model poisoning · improper output handling · excessive agency · system prompt leakage · vector and embedding weaknesses · misinformation · unbounded consumption

FOR EVERY SINGLE ONE, HAVE FOUR ANSWERS READY

WHAT

Define it in one sentence, no jargon.

IMPACT

What does an attacker gain? Business consequence.

EXPLOIT

How, concretely. Walk the steps.

MITIGATE

Layered fixes, plus how you'd detect it.

Name the platform.

Free. Interactive labs. Genuinely the best hands-on web security training there is.

CLUE

The company behind it also makes the intercepting proxy that is open on half the laptops in this room right now.

shout it out

Secure design & threat modeling

Pick one framework. Get fluent. Then apply it to real systems until it's reflex.

STRIDE

Start here. Six categories, fast to apply, easy to narrate out loud.

PRACTICE SCENARIOS - MODEL EACH ONE END TO END

A password reset flow

Tokens, enumeration, rate limits, account recovery abuse.

File upload to object storage

Content type, path traversal, signed URLs, public buckets, malware.

An internal chatbot with tool access

Untrusted context, excessive agency, data boundaries, approval gates.

A CI/CD pipeline with third-party actions

Secrets, supply chain, build integrity, branch protections.

Source code review

The round most candidates skip preparing for - and the one that separates people fastest.

01 **Learn the sinks**
Per language: command exec, deserialization, SQL string building, path joins, crypto misuse.

02 **Trace source to sink**
Where does user input enter, what touches it, and what validation actually applies on that path?

03 **Write it up as a finding**
Title, severity, affected path, proof of concept, concrete fix. Practice the format, not just the spot.

04 **Study real diffs**
Pull a CVE fix commit and read it before the advisory. Guess the bug from the patch.

Coding preparation (if the role requires it)

Some Security Engineer loops include a coding round. Ask the recruiter.

PATTERNS THAT ACTUALLY SHOW UP

- Arrays and strings: two pointers, sliding window
- Hashing: sets, counters, dedup, frequency maps
- Stacks and queues: parsing, matching, BFS
- Basic recursion and simple graph traversal
- Linked lists: traversal, reversal, cycle detection (optional)

HOW TO SEARCH EFFICIENTLY

```
leetcode > problem set
  filter: Company = <your target>
  filter: Difficulty = Easy + Medium
  sort:   Frequency, descending
  then:   "Top Interview Questions" list
→ work the top 40, not all 3000
```

PYTHON

Pick one language and stop switching. Know its idioms cold.

OUT LOUD

Narrate while you code. Silence reads as being stuck.

EDGE CASES

State them before you type. Empty, null, huge, duplicate.

2-3 A DAY

Three weeks of consistency beats a panicked weekend.

Behavioral: build a story bank

Loops assess collaboration, communication and judgment under pressure. Prepare 8-10 STAR stories.

S **Situation**
Context in two sentences

T **Task**
What you owned

A **Action**
What you specifically did

R **Result**
Outcome, ideally measurable

COVER THESE TEN THEMES - ONE STORY EACH, REUSABLE ACROSS QUESTIONS

01	Conflict with an engineer or team	02	A failure you owned
03	Influence without authority	04	An incident under time pressure
05	Disagreeing with a decision	06	Mentoring or levelling someone up
07	Security vs shipping tradeoff	08	An ambiguous problem you scoped
09	Your highest-impact work	10	Feedback that changed how you work

Two - three minutes per story. Lead with the result, then back into how you got there.

Mock interviews and real feedback

You cannot self-assess delivery. Rehearsing in your head is not rehearsing.

PEER NETWORKS

Security Discord and Slack communities, local meetups, conference study groups. Trade mocks - you interview them, they interview you.

LINKEDIN OUTREACH

Message engineers who already hold the role you want. Short, specific ask: 30 minutes, one mock, one piece of feedback.

MOCK PLATFORMS

Peer and professional mock interview services for coding and systems rounds. Anonymous practice removes the fear of looking bad.

RECORD YOURSELF

Answer a question to a camera and watch it back. Painful, and the fastest fix for filler words and rambling.

Ask for one specific improvement, not "how did I do?"

"What was the weakest part of that answer?" gets you something actionable. "Good job" gets you nothing.

Common mistakes

The ones I see cost people offers, in rough order of frequency.

01

Two hundred applications, one resume

Volume without targeting is just faster rejection.

02

Definitions without depth

You can name the vuln. You freeze at "how would you exploit it?"

03

No clarifying questions in design rounds

Jumping straight to solutions is the most common design-round failure.

04

Rambling behavioral answers

Five minutes, no result, no "I". Two minutes, lead with the outcome.

05

Trashing a previous employer

Nothing ends a loop faster. Describe the problem, not the villain.

06

Claiming work you can't defend

AI-written bullets and borrowed projects collapse at the first follow-up.

Bonus: having no questions for your interviewer. Bring three, and make one of them about how the team measures success.

If you remember five things

01 **Diagnose your failing gate.**

02 **Every bullet: action, scale, tool, outcome.**

03 **Run the workflow locally.**

Ollama plus an open model gives you volume and tailoring without leaking your own data.

04 **Four questions, every vulnerability.**

What, impact, exploit, mitigate — for both OWASP Top 10s.

05 **Only claim what you built.**

Then go three questions deep on it without breaking a sweat.

Questions?

Let’s Connect

Monish Alur Gowdru
Security Consultant