Cyber Security Mini Project Ideas PDF

A cyber security mini project should be small, safe and run only on your own machines: a password strength checker, file encryption tool, port scanner for your own lab, log analyser or a caesar-to-AES encryption demo. Each teaches one security idea in one to four weeks.

Key takeaways

- Test only on your own computer or lab virtual machines.
- Python is enough for most security mini projects.
- Explain the security concept, not just the tool output.
- OWASP Juice Shop and DVWA are safe practice targets.
- Each idea fits in one to four weeks.

Who this PDF is for

This PDF is for 2nd and 3rd year students who want a first security project. Every idea is defensive or runs on your own machine or lab virtual machines. Each plan lists tools, steps and what to explain in the viva.

How to choose your project

- **Legal first**: your own devices or lab VMs only.
- **One concept per project**: hashing, encryption, scanning or logs.
- **Use Python** and its standard libraries where possible.
- **Show before and after**: weak setup, then the fix.

Beginner mini projects (1 to 2 weeks)

1. Password Strength Checker

- **Problem:** Rate how strong a password is and suggest fixes.
- **Dataset:** Common-password lists such as those in the SecLists project, used offline.
- **Tech stack:** Python, regular expressions, zxcvbn.
- **Main modules:** Rule checks, entropy estimate, common-password check, suggestions.
- **Steps:** Check length and character types, estimate entropy, compare with a common list.
- **Show in the viva:** Why length beats complexity rules.
- **Extensions:** Add a breach check with k-anonymity.

2. Classical Cipher Toolkit

- **Problem:** Encrypt and break Caesar and Vigenere ciphers.
- **Dataset:** Sample English text.
- **Tech stack:** Python.
- **Main modules:** Encrypt, decrypt, brute force, frequency analysis.
- **Steps:** Implement both ciphers, break Caesar by brute force and Vigenere with frequency analysis.
- **Show in the viva:** Why these ciphers are insecure today.
- **Extensions:** Compare with AES.

3. File Encryption Tool

- **Problem:** Encrypt and decrypt files with a password.
- **Dataset:** Your own test files.
- **Tech stack:** Python cryptography library (Fernet or AES-GCM), Tkinter.
- **Main modules:** Key derivation from password, encrypt, decrypt, integrity check.
- **Steps:** Derive a key with PBKDF2 or scrypt, encrypt with AES-GCM, detect tampering on decrypt.
- **Show in the viva:** Why you never store the password itself.
- **Extensions:** Add folder encryption.

4. Hash Generator and Verifier

- **Problem:** Generate and verify file checksums.
- **Dataset:** Your own files.
- **Tech stack:** Python hashlib.
- **Main modules:** MD5, SHA-1 and SHA-256 hashing, compare, report.
- **Steps:** Hash files, compare with published checksums, explain why MD5 is weak.
- **Show in the viva:** What a collision is.
- **Extensions:** Watch a folder for changes.

Intermediate mini projects (2 to 3 weeks)

5. Port Scanner for Your Lab

- **Problem:** Find open ports on your own lab machines.
- **Dataset:** Lab VMs on an isolated network.
- **Tech stack:** Python sockets, threading; Nmap for comparison.
- **Main modules:** TCP connect scan, threading, service banner, report.
- **Steps:** Scan your VMs, compare results with Nmap, close unneeded ports and rescan.
- **Show in the viva:** Difference between open, closed and filtered ports.
- **Extensions:** Add UDP scanning.

6. Keylogger Detector

- **Problem:** Spot suspicious processes that hook the keyboard.
- **Dataset:** Your own Windows or Linux machine.
- **Tech stack:** Python, psutil.

- **Main modules:** Process listing, suspicious name and path rules, startup entries check, report.
- **Steps:** List processes and startup items, flag unknown ones, compare with a clean baseline.
- **Show in the viva:** Limits of rule-based detection.
- **Extensions:** Add hash lookups against a known-good list.

7. SSH Log Analyser

- **Problem:** Detect brute-force login attempts from logs.
- **Dataset:** Your own Linux VM's auth logs.
- **Tech stack:** Python, regular expressions, matplotlib.
- **Main modules:** Log parsing, failed-login counts by IP, threshold alerts, charts.
- **Steps:** Parse auth.log, count failures per IP, flag bursts, chart by hour.
- **Show in the viva:** How fail2ban-style blocking works.
- **Extensions:** Auto-generate firewall block rules.

8. Phishing Email Awareness Quiz

- **Problem:** Train users to spot phishing emails.
- **Dataset:** Sample phishing-style emails you write (no real brands).
- **Tech stack:** HTML, CSS, JavaScript.
- **Main modules:** Email samples, clue highlighting, scoring, tips page.
- **Steps:** Write samples with clues, let users pick real or phishing, explain each answer.
- **Show in the viva:** Top phishing clues.
- **Extensions:** Track scores for a class.

Stretch mini projects (3 to 4 weeks)

9. Packet Sniffer and Analyser

- **Problem:** Capture and summarise traffic on your own network.
- **Dataset:** Your own laptop and lab network.
- **Tech stack:** Python, Scapy; Wireshark for comparison.
- **Main modules:** Capture, protocol breakdown, top talkers, unencrypted traffic alerts.
- **Steps:** Capture packets, count by protocol, flag plain-text HTTP and FTP.
- **Show in the viva:** Why HTTPS matters, shown with your capture.
- **Extensions:** Detect ARP spoofing.

10. Two-factor Authentication Demo

- **Problem:** Add time-based one-time passwords to a login page.
- **Dataset:** No external data.
- **Tech stack:** Python Flask, pyotp, a QR code library, an authenticator app.
- **Main modules:** Login, TOTP setup with QR, code verification, backup codes.
- **Steps:** Hash passwords, generate a TOTP secret, show QR, verify codes.
- **Show in the viva:** How TOTP codes are generated.
- **Extensions:** Add rate limiting.

11. SQL Injection Lab and Fix

- **Problem:** Show SQL injection in a toy app and fix it.
- **Dataset:** Your own small app on localhost.
- **Tech stack:** Python Flask or PHP, SQLite.
- **Main modules:** Vulnerable login, injection demo, parameterised query fix, retest.
- **Steps:** Build a deliberately weak login, show injection, rewrite with parameterised queries.
- **Show in the viva:** Why parameterised queries stop injection.
- **Extensions:** Add XSS demo and fix.

12. Firewall Rule Builder

- **Problem:** Create and test firewall rules on a Linux VM.
- **Dataset:** A Linux lab VM.
- **Tech stack:** iptables or nftables, ufw, Python for a simple UI.
- **Main modules:** Rule templates, apply and test, rollback, logging.
- **Steps:** Write rules to allow only needed ports, test with your port scanner, log drops.
- **Show in the viva:** Default deny in simple words.
- **Extensions:** Add geo-blocking rules.

Mini project report structure

A mini project report is usually 10 to 20 pages. Keep it short and clear:

1. **Title page and certificate** in your department's format.
2. **Introduction:** the problem in one paragraph and your objectives as a list.
3. **Tools used:** operating system, Python version, libraries and lab setup (VMs and network).
4. **Design:** one block diagram or flowchart, plus a diagram of your lab or tool flow.
5. **Implementation:** what each module does, with 3 to 6 screenshots.
6. **Results:** screenshots, before and after results, and a note that all tests were on your own systems.
7. **Conclusion and future scope,** then references.

Ask your guide for the exact format before you start writing.

Common mistakes to avoid

- Testing on college or public systems without permission.
- Writing your own encryption algorithm instead of using tested libraries.
- Storing passwords in plain text, even in demos.
- Showing a tool's output with no explanation.

Frequently asked questions

What is a good cyber security mini project for beginners?

A password strength checker, file encryption tool or hash verifier. They need only Python and teach core ideas.

Is a port scanner legal as a mini project?

Yes, if you scan only your own machines or lab VMs. Scanning others' systems without permission can be an offence.

Which language is best for security projects?

Python, because of libraries such as Scapy, cryptography and hashlib, and many learning resources.

Do I need Kali Linux for mini projects?

No. Most ideas here run on Windows or any Linux with Python. Kali is useful later for lab testing.

Does this PDF include full source code?

No. It gives the problem, data, stack, modules and steps for each project. You write the code yourself, which is what examiners test in the viva.

Is this PDF free?

Yes. Fill in the short form on this page and the PDF downloads right away.

Sources and further reading

- OWASP Juice Shop
- Python cryptography library
- Scapy documentation
- OWASP Cheat Sheet Series

About this PDF

Prepared by the Uncodemy training team as practice project ideas. Datasets and tools named are public; check each dataset's licence before use. We update it every January. Corrections: info@uncodemy.com.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550