Cyber Security Final Year Project Ideas PDF

A good cyber security final year project is defensive, runs in your own lab, and measures how well it detects or prevents attacks. Intrusion detection with machine learning, phishing URL detection, a home-lab SIEM, a vulnerability scanner for your own apps and cloud misconfiguration checks are strong choices using public datasets such as NSL-KDD and CICIDS2017.

Key takeaways

- Test only on systems you own or have written permission to test.
- Use public datasets (NSL-KDD, CICIDS2017, UCI Phishing Websites) for detection models.
- Build a small lab with virtual machines and deliberately vulnerable apps.
- Report detection rate and false positives, not just accuracy.
- The PDF gives plans and steps, not finished source code.

Who this PDF is for

This PDF is for final-year B.Tech, BCA and MCA students who want a security project that is safe, legal and measurable. Every plan is defensive or runs in an isolated lab using virtual machines and intentionally vulnerable apps such as DVWA or OWASP Juice Shop. Each plan lists data, tools, modules and viva points for a team of 3 or 4.

How to choose your project

- **Stay legal.** Scanning or attacking systems without permission is an offence under the IT Act.
- **Defence plus measurement** scores better than a tool demo.
- **Build a lab**: VirtualBox or VMware with isolated networks.
- **Mix ML and systems** if your team has both skills.

Standard level projects

1. ML-based Network Intrusion Detection System

- **Problem:** Detect attacks in network traffic.
- **Dataset:** CICIDS2017 (Canadian Institute for Cybersecurity) and NSL-KDD.
- **Tech stack:** Python, scikit-learn, XGBoost, Streamlit.
- **Main modules:** Flow feature cleaning, class balancing, model, per-attack metrics, alert dashboard.

- **Steps:** Clean flows, balance classes, train models, report detection and false positive rates per attack.
- **Show in the viva:** Why false positives matter to a security team.
- **Extensions:** Test on live flows captured in your lab with CICFlowMeter.

2. Phishing URL Detector with Browser Extension

- **Problem:** Warn users before they open a phishing link.
- **Dataset:** UCI "Phishing Websites" dataset; Kaggle "Malicious URLs dataset".
- **Tech stack:** Python, scikit-learn, FastAPI, a Chrome extension (JavaScript).
- **Main modules:** URL feature extractor, model, API, extension popup.
- **Steps:** Extract lexical and host features, train a model, serve it, call it from the extension.
- **Show in the viva:** Features that most often flag phishing.
- **Extensions:** Add page-content checks.

3. Home-lab SIEM with Wazuh

- **Problem:** Collect logs, detect suspicious events and alert.
- **Dataset:** Logs from your own lab VMs.
- **Tech stack:** Wazuh or the Elastic Stack, Windows and Linux VMs, Sysmon.
- **Main modules:** Agent setup, log collection, detection rules, dashboards, alert tests.
- **Steps:** Install the SIEM, add agents, simulate brute-force logins in the lab, tune rules.
- **Show in the viva:** One detection rule and the alert it raised.
- **Extensions:** Map detections to MITRE ATT&CK techniques.

4. Web Vulnerability Scanner for Your Own Apps

- **Problem:** Find common web flaws such as SQL injection and XSS.
- **Dataset:** DVWA and OWASP Juice Shop in your lab.
- **Tech stack:** Python, requests, BeautifulSoup, OWASP ZAP for comparison.
- **Main modules:** Crawler, payload tests, result report, comparison with ZAP.
- **Steps:** Crawl the lab app, test inputs with safe payloads, report findings, compare with ZAP.
- **Show in the viva:** Why you test only lab apps.
- **Extensions:** Add authenticated scanning.

Advanced level projects

5. Malware Classification with ML

- **Problem:** Classify Windows files as malicious or benign from static features.
- **Dataset:** EMBER dataset (feature vectors, no live malware needed).
- **Tech stack:** Python, LightGBM, scikit-learn.
- **Main modules:** Feature loading, model training, evaluation, explanation.
- **Steps:** Load EMBER features, train LightGBM, report ROC curves and false positives.
- **Show in the viva:** Why the dataset ships features instead of executable files.
- **Extensions:** Test robustness to feature changes.

6. Password Strength and Breach Checker

- **Problem:** Rate passwords and warn if they appear in known breaches.
- **Dataset:** Have I Been Pwned Pwned Passwords range API.
- **Tech stack:** Python or JavaScript, zxcvbn library, Flask.
- **Main modules:** Strength scoring, k-anonymity breach check, policy suggestions.
- **Steps:** Score strength, send only the first five hash characters to the API, compare locally.
- **Show in the viva:** How k-anonymity keeps the password private.
- **Extensions:** Add a browser extension.

7. Cloud Misconfiguration Scanner

- **Problem:** Find risky settings in a cloud account you own.
- **Dataset:** Your own AWS free-tier account with test resources.
- **Tech stack:** Python, boto3, Prowler for comparison.
- **Main modules:** Checks for public S3 buckets, open security groups, missing MFA, report.
- **Steps:** Create deliberately risky test resources, write checks, compare with Prowler, fix and re-scan.
- **Show in the viva:** The shared responsibility model.
- **Extensions:** Add auto-fix suggestions.

8. Honeypot with Attack Analytics

- **Problem:** Attract and record attack attempts to learn attacker behaviour.
- **Dataset:** A low-interaction honeypot on an isolated cloud VM you own.
- **Tech stack:** Cowrie SSH honeypot, Elastic Stack or Python analysis.
- **Main modules:** Honeypot deployment, log shipping, analysis of commands and source countries, report.
- **Steps:** Deploy Cowrie with strict isolation, collect logs for two weeks, analyse patterns.
- **Show in the viva:** Isolation steps that stop the honeypot being misused.
- **Extensions:** Compare SSH and Telnet honeypot traffic.

Research-oriented projects

9. Adversarial Attacks on Intrusion Detection

- **Problem:** Test whether small feature changes let attacks evade an ML detector.
- **Dataset:** NSL-KDD or CICIDS2017.
- **Tech stack:** Python, scikit-learn, an adversarial ML library.
- **Main modules:** Baseline detector, evasion attacks, adversarial training, results.
- **Steps:** Train a detector, craft evasion samples within valid feature limits, retrain, compare.
- **Show in the viva:** Detection rate before and after defence.
- **Extensions:** Write it up for a student conference.

10. Email Phishing Detection with NLP

- **Problem:** Detect phishing emails from their text.
- **Dataset:** Enron email dataset for legitimate mail plus public phishing email corpora.

- **Tech stack:** Python, scikit-learn, Hugging Face Transformers.
- **Main modules:** Text cleaning, TF-IDF baseline, transformer model, header features.
- **Steps:** Combine text and header features, compare models, check false positives on normal mail.
- **Show in the viva:** Which words and headers signal phishing.
- **Extensions:** Add an Outlook or Gmail add-on prototype.

11. DevSecOps Pipeline

- **Problem:** Add security checks to a CI/CD pipeline.
- **Dataset:** One of your own apps on GitHub.
- **Tech stack:** GitHub Actions, Semgrep, Trivy, OWASP ZAP baseline scan.
- **Main modules:** Static analysis, dependency scan, container scan, dynamic scan, gate rules.
- **Steps:** Add each scan as a pipeline stage, fail builds on high-severity issues, track fixes.
- **Show in the viva:** Issues found at each stage.
- **Extensions:** Add secrets scanning.

12. Steganography Detection

- **Problem:** Detect hidden data in images.
- **Dataset:** Images you create with and without hidden data using open tools.
- **Tech stack:** Python, OpenCV, scikit-learn or a CNN.
- **Main modules:** Stego image generation, feature extraction, classifier, results.
- **Steps:** Hide data with LSB methods, extract statistical features, train a detector.
- **Show in the viva:** Why LSB changes are hard to see but detectable statistically.
- **Extensions:** Test other embedding methods.

Report structure

A final-year report is marked on clarity as much as on code. A structure most universities accept:

1. **Abstract** (150 to 250 words): problem, method, main result.
2. **Introduction:** background, problem statement, objectives and scope.
3. **Literature review:** 8 to 15 papers or existing systems, and the gap your project fills.
4. **Requirements and design:** functional and non-functional requirements, architecture diagram, the lab network diagram, threat model, and data or log flow.
5. **Implementation:** modules, tools, key algorithms and screenshots.
6. **Testing and results:** detection rate, false positive rate, findings tables and screenshots, with a note that all testing was in your own lab.
7. **Conclusion and future scope.**
8. **References** in the format your department asks for (IEEE style is common).

Follow your own university's template if it gives one; it overrides this list.

Common mistakes to avoid

- Scanning college or public sites without written permission.

- Running real malware outside an isolated lab.
- Reporting only accuracy on imbalanced attack data.
- A tool demo with no measurement or analysis.

Frequently asked questions

Which cyber security project is best for a final year?

An ML-based intrusion detection system, a phishing URL detector, a home-lab SIEM or a cloud misconfiguration scanner. Each is defensive and measurable.

Is it legal to do hacking projects?

Only on systems you own or have written permission to test. Use lab apps such as DVWA and OWASP Juice Shop inside virtual machines.

Which datasets are used for intrusion detection?

NSL-KDD and CICIDS2017 are the most common public datasets for student intrusion detection projects.

Do I need Kali Linux?

It helps for lab testing tools, but many projects here use Python, a SIEM and public datasets instead.

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

- Canadian Institute for Cybersecurity datasets (CICIDS2017, NSL-KDD)
- OWASP Top 10
- MITRE ATT&CK
- Wazuh documentation
- EMBER dataset on GitHub

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