The Project: God Father of Ai (GFA)

(GFA) Outline

- Lesson 1. Code Analysis
- Lesson 2. Look Beyond the Main App
- Lesson 3. Deep Research
- Lesson 4. Self-Learning Exploitation Tactics
- Lesson 5. Apply All Techniques via Code Analysis

First Skill: **Understanding of Programming Languages**

- You need to read, interpret, and understand code logic.
  - Key Languages:
  - C/C++ – for system-level or binary analysis
  - Python/JavaScript – for web security and automation
  - Assembly – for reverse engineering binaries

First Skill: Static and Dynamic Analysis Techniques

- Static Analysis: Analyzing source or binary code without running it.
- Tools: Ghidra, IDA Pro, Checkmarx, SonarQube
- Dynamic Analysis: Running the code and observing behavior.
- Tools: GDB, Wireshark, Burp Suite, Frida

First Skill: **Vulnerability Pattern Recognition**

- You need to read, interpret, and understand code logic.
- **Why:** To identify common security flaws like buffer overflows, injections, broken logic, etc.
- How to build this skill:
- Study CVEs and exploits.
- Practice & training with intentionally vulnerable codebases (e.g., DVWA, Juice Shop).
- Learn Secure Coding Practices

Second Skill: Look Beyond the Main App

- Third-party JS files (e.g., cdn.example.com/lib.js):
  - May leak **API keys**, **internal endpoints**, or **sensitive logic**.
  - Can expose hidden features or undocumented API routes.
  - Often contain vulnerable functions like eval() or innerHTML.

Second Skill: Look Beyond the Main App

- Internal APIs → can be discovered via frontend JS or traffic inspection.
  - Internal APIs are backend routes not directly shown in the UI but used by the app.
  - You can discover them by analyzing frontend JavaScript files for endpoint references.
  - Use browser tools like DevTools > Network tab to inspect API calls during user actions.
  - These APIs may lack proper access control, exposing functions like admin actions or data access.

Second Skill: Look Beyond the Main App

- Mobile/desktop versions of the app.Old or legacy subdomains (beta., dev., old., etc.).
  - Check mobile/desktop versions of the app—they may use different logic or APIs.
  - Explore legacy or staging subdomains like beta., dev., or old. for outdated code.
  - These versions often lack security patches or have test credentials.
  - Use tools like subfinder, assetfinder, or gau to discover hidden subdomains.

Third Skill: **Deep Research**

- **Tech Stack**.
  - Identify the backend frameworks and libraries (e.g., Node.js, Django, Laravel).
  - Each stack has its own attack surface and common misconfigurations.
  - Use tools like Wappalyzer, builtwith, or source code analysis to detect it.
  - Knowing the stack helps you focus on relevant exploits and CVEs.

Third Skill: **Deep Research**

- **CVE History**
  - Search for known vulnerabilities (CVEs) in the specific version of the tech used.
  - Older or unpatched versions often have public exploits available.
  - Use CVE sites, Exploit-DB, or Rapid7's AttackerKB for quick insights.
  - This helps you craft targeted attacks based on documented weaknesses.

Third Skill: **Deep Research**

- **GitHub / Open Source**
  - If it’s open source, inspect GitHub issues, commits, and forks.
  - You may find unresolved bugs, insecure defaults, or unpatched vulns.
  - Patch notes and PRs can reveal what was fixed or overlooked.
  - Sometimes, devs unknowingly expose sensitive info in code or comments.

Third Skill: **Deep Research**

- **Documentation / Dev Blogs**
  - Read official docs to understand hidden features or debug modes.
  - Dev blogs often reveal edge cases, misconfigurations, or future updates.
  - Some examples include verbose error logs or insecure API usage.
  - They help you exploit logic flaws or design weaknesses others miss.

Forth Skill: **Self-Learning Exploitation Tactics**

- **Payload Crafting:**
  - Learn how to build advanced payloads for XSS, SQLi, SSRF, SSTI, etc.
  - Understand encoding tricks, broken filters, and multi-layered injections.
  - Experiment with real apps or labs to see how different payloads behave.
  - Stay updated with latest bypass payloads from public writeups or GitHub repos.

Forth Skill: **Self-Learning Exploitation Tactics**

- Bypasses
  - Explore how to evade WAFs using Unicode, double encoding, and obfuscation.
  - Test HTTP verb tampering (e.g., HEAD, PUT, DELETE) to bypass logic.
  - Combine multiple techniques (e.g., CRLF + SSRF) to sneak past protections.
  - Always verify bypasses in multiple scenarios to understand effectiveness.

Forth Skill: **Self-Learning Exploitation Tactics**

- **Tools:**
  - Master Burp Suite extensions like Autorize, Turbo Intruder, and Collaborator.
  - Use ffuf, kiterunner, nuclei, and others to fuzz, enumerate, and detect vulns.
  - Write or tweak your own scripts and modules for automation.
  - Keep tools updated and customize them for advanced scanning.

Forth Skill: **Self-Learning Exploitation Tactics**

- Community:
  - Follow top hunters on Twitter/X for latest tricks and bypasses.
  - Read bug bounty writeups on HackerOne, Medium, and GitHub.
  - Join forums like Bugcrowd, Cobalt, and Discord servers to discuss tactics.
  - Learning from others shortens your trial-and-error cycle massively.

Fifth Skill: Apply All Techniques via Code Analysis:

- **Code analysis is the final weapon:**
  - **This is where all your earlier work—recon, research, and payload crafting—comes together.**
  - **You analyze backend logic, trace data flow, and find weak spots not visible from the UI.**
  - **It's the most powerful step to confirm and exploit real bugs.**

Fifth Skill: Apply All Techniques via Code Analysis:

- **Inspect the full codebase, not just the interface:**
  - **Go beyond the frontend—look at APIs, routes, internal services, and comments.**
  - **Hidden endpoints like /debug, /internal, or feature toggles can be goldmines.**
  - **This uncovers forgotten logic or weak access controls developers left behind.**

Fifth Skill: Apply All Techniques via Code Analysis:

- **Correlate CVEs with the tech stack and version:**
  - Match the exact version of libraries/frameworks with their known vulnerabilities (CVEs).
  - Check GitHub issues or changelogs to confirm unpatched bugs.
  - Reuse working exploits or PoCs tailored to that version.

Fifth Skill: **Apply All Techniques via Code Analysis:**

- **Use cutting-edge payloads + old bypasses:**
  - Modern payloads can help break filters, while old CTF tricks may bypass legacy defenses.
  - Try WAF bypasses, encodings, and alternative syntax to test edge cases.
  - Combining both increases the chance of real-world exploit success.

Thank You!