Information Disclosure Vulnerability Exploitation in Web Applications – Exposed Backup File Reveals Source Code and Credentials

Vulnerability Overview/Executive Summary

I Recovered Database Credentials From a Forgotten Backup File Left Publicly Accessible on the Server.

During a penetration test for a software firm, I encountered an overlooked but highly impactful vulnerability — the presence of a raw backup file containing Java source code on a production server. These .bak or .old files are often created during testing or migration and accidentally deployed to public-facing environments. In this case, the file contained sensitive backend logic and hardcoded PostgreSQL credentials.

This type of information disclosure vulnerability is dangerous not because of technical misconfiguration, but because of operational oversights. When a backup file contains authentication secrets and application logic, it equips attackers with everything they need for privilege escalation and lateral movement.

Discovery Process

Using Burp Suite's **Content Discovery** tool under Engagement Tools, I initiated a scan for common hidden paths and file extensions. One of the findings included a .bak file under a /backup/files/ directory — a likely development artifact. Retrieving and reviewing the file confirmed it was a copy of a key Java source file (ProductTemplate.java.bak) containing hardcoded credentials to the application's production database.

Exploitation Method

1. Logged in as **john doe** and launched content discovery from Burp Suite.
2. Located /backup/files/ProductTemplate.java.bak among discovered resources.
3. Sent a GET request to the file using Repeater and reviewed the response body.
4. Identified a plaintext hardcoded credential block, including the PostgreSQL password.

5. Extracted the password and confirmed database access was possible using this data.

Proof of Concept

- **Leaked File:** /backup/files/ProductTemplate.java.bak
- **Leak Detail:** String dbPassword = "pgSecureAdmin@123";
- **Response:** HTTP/2 200 OK
- **Result:** Java source code exposed with database credentials in plaintext

Potential Impact

- Full database access via hardcoded credentials
- Exposure of backend application logic
- Accelerated exploitation of other vulnerabilities (SQLi, auth bypass)
- Regulatory compliance failures and breach notifications

Mitigation/Recommendations

- Never deploy backup files or code artifacts to production
- Enforce file extension allowlists and denylists in the web server
- Restrict access to folders like /backup/, /admin/, or /dev/ using server config
- Conduct routine audits of deployed assets
- Automate deployment scripts to exclude temporary and sensitive files

Reflections/Learning Points

This engagement reinforced the idea that secure software isn't just about code — it's about discipline. Backup files, forgotten dev resources, or logs can reveal more than vulnerabilities ever could. Content discovery should never be skipped in assessments. It often leads straight to the attacker's jackpot.