Project 5 — Research: Common Vulnerabilities Across Real Web Apps

Course: Secure Programming / Web Security

Student: isha bachhav

Title

File Upload Vulnerabilities in the Wild — Linking DVWA "File Upload" to real-world CVEs

Abstract

This report maps the DVWA "File Upload" lesson to three real-world vulnerabilities where insecure file upload handling led to serious compromises. Each case includes a summary of the vulnerability, how attackers exploited it, observed impact, remediation applied, and key lessons.

Selected DVWA vulnerability: File Upload (CWE-434)

DVWA's File Upload module demonstrates how accepting files from users without strict validation can allow attackers to upload and execute a malicious script on the server (CWE-434 — Unrestricted Upload of File with Dangerous Type). The cases below show real-world examples where lack of validation, poor allowlist checks, or bundled vulnerable libraries enabled remote code execution.

Case 1 — TimThumb (widely bundled in WordPress plugins) — CVE-2011-4106

Summary: TimThumb (timthumb.php), an image-resizing script commonly bundled with WordPress themes and plugins, failed to correctly validate

remote image sources and allowed attackers to place files in the cache directory and execute them.

How exploited: Attackers supplied a specially crafted URL that bypassed the source whitelist and caused TimThumb to fetch and cache an attacker-controlled file. Because the cache directory was web-accessible and TimThumb wrote files with predictable names and PHP parsing, attackers could upload PHP code and execute it by requesting the cached file.

Impact: Widespread remote code execution on many WordPress sites using vulnerable TimThumb copies; many sites were defaced or had backdoors installed.

Fix: TimThumb updated its validation logic; theme/plugin authors removed or updated vulnerable copies; server operators protected cache directories (deny PHP execution), or moved them outside the webroot.

Case 2 — ReFlex Gallery plugin (WordPress) — CVE-2015-4133

Summary: The ReFlex Gallery WordPress plugin contained an admin/scripts/FileUploader/php.php endpoint that allowed unrestricted file upload of PHP files.

How exploited: An unauthenticated attacker could upload a file with a .php extension and call it directly from the uploads directory, executing arbitrary PHP code on the server.

Impact: Remote code execution, website compromise, data theft, or pivoting to other internal systems depending on hosting privileges.

Fix: Plugin updated to restrict upload types and sanitize file names; affected sites were advised to update plugin versions and remove uploaded webshells.

Case 3 — Magento / Adobe Commerce file upload allowlist bypass — CVE-2020-24407 (and related CVEs)

Summary: Several Magento / Adobe Commerce versions had improper validation or allowlist bypasses in file upload components that allowed dangerous file types to be uploaded, potentially leading to arbitrary code execution when files were processed or stored in web-accessible locations.

How exploited: Authenticated attackers with certain administrative permissions could bypass file-type filters and upload malicious files; in some cases the uploaded files could be executed, or further flaws (deserialization, template rendering) could be chained to achieve RCE.

Impact: Arbitrary code execution on e-commerce platforms, leading to theft of customer data, site takeover, and disruption of business-critical services.

Fix: Vendor patches and security updates were released by Adobe/Magento; administrators were advised to apply updates, harden file storage locations, and restrict privileges.

Comparative analysis & mapping to DVWA

- DVWA's File Upload teaches the same core mistake: insufficient validation + placing uploaded files in web-accessible locations. The real-world cases show the same attack flow but at larger scale in third-party plugins and enterprise platforms.
- Common root causes: missing content-type checks, trusting filename extensions, writable webroot/cache directories, and outdated third-party libraries bundled into larger apps.

Reproduction & Demonstration (suggested for assignment)

1. Reproduce DVWA File Upload exploitation locally (use the vulnerable DVWA module). Take screenshots of successful webshell upload and execution.
2. For the TimThumb case, include a short writeup recreating the logic (no need to attack real sites). Show how caching to a web-accessible directory leads to execution.
3. For the plugin/CMS CVEs, include NVD references and vendor advisories (in appendix).

Mitigations & Best Practices

- Enforce strict allowlists: accept only required MIME types and enforce server-side checks (not just client-side).
- Verify file contents (magic bytes) in addition to extensions and MIME types.
- Store uploads outside the webroot or ensure they are never executed (disable PHP execution in upload directories via server config).
- Rename uploaded files to unpredictable names and avoid using user-supplied names.
- Apply least privilege to web server and application accounts; restrict administrative actions.
- Keep third-party libraries and plugins up-to-date; scan for bundled vulnerable components.

References (to include in submission)

- NVD / CVE entries for CVE-2011-4106, CVE-2015-4133, CVE-2020-24407
- CWE-434 description
- Vendor advisories and vulnerability writeups (TimThumb analyses, WordPress plugin advisories, Adobe/Magento security bulletins)

Appendix A — Full CVE summaries & timelines

(Include timeline, vendor advisory snippets, suggested remediation steps per CVE.)

Appendix B — Assessment Checklist

- ☐ DVWA vulnerability selected and mapped to CVEs
- ☐ 2–3 real-world CVEs documented with exploitation details and impact
- ☐ Fixes and recommendations enumerated
- ☐ Reproducible DVWA demo and screenshots described