COMMUNITY CONTENT

Submit #785731: ProjectSend projectsend r2002 Cross-Site Request Forgery

Title	ProjectSend projectsend r2002 Cross-Site Request Forgery
Description	# Vulnerability Report: CSRF via HTTP Method Manipulation in ProjectSend ProjectSend's file upload functionality in `upload.process.php` is vulnerable to **Cross-Site Request Forgery (CSRF)** due to two critical issues: 1. **Absence of CSRF protection** — No anti-CSRF token is present. 2. **HTTP method manipulation** — The endpoint processes file upload with a POST request but that can be easily converted into a GET request and still go through. This allows attackers to craft a simple URL that triggers an upload when visited by an authenticated user. An attacker who knows (or can guess) the ProjectSend instance URL can trick a logged-in user (with roles such as "Uploader" or "Admin") into visiting a malicious page. The victim's browser automatically includes session cookies, causing the server to process the request as the authenticated user and upload an arbitrary file without consent. This attack requires **user interaction** (visiting the malicious page) but is straightforward to execute. If the endpoint enforced POST-only handling and used proper CSRF tokens, this vector would be blocked. ## Attack Flow 1. Victim is authenticated to ProjectSend → browser holds valid session cookies. 2. Victim visits attacker-controlled page (e.g., via phishing link, malicious site, or embedded content). 3. Attacker's page triggers a GET request to `upload.process.php` with upload parameters. 4. Browser sends the request (with victim's cookies) → server processes it → file is uploaded silently on victim's behalf. ## Steps to Reproduce (Authenticated Testing) You need BurpSuite for this so that you can perform browser proxy interception. 1. Log in to ProjectSend as a user with upload privileges (e.g., Uploader or Admin) and navigate to the upload page: `http://localhost:8081/upload.php`. 2. Use a proxy (e.g., Burp Suite) to capture a legitimate **POST** upload request. 3. In Burp Repeater, use the "Change Request Method" feature to convert the request to **GET**. - Parameters (e.g., `name`, `chunk`, `chunks`, `file`, etc.) move from the body to the

Community Content

Submissions are made by VulDB community users. VulDB is *not responsible* for their content nor the links to external sources.

Please use the raw information shown and the links listed *with caution*. They might contain malicious and harmful actions, code or data.

The corresponding VulDB entries contain the moderated, verified, and normalized information provided within the raw submission.

Documentation

- Submission Policy
- Data Processing
- CVE Handling

query string.
4. Send the modified GET request & observe successful file creation on the server.
5. This confirms the endpoint accepts and processes uploads via GET, enabling CSRF.

Steps to Reproduce (Attacker Perspective – Unauthenticated)

1. Identify the target ProjectSend instance URL (e.g., `http://projectsend.example.com` or internal `http://localhost:8081`).
2. Create a malicious HTML file (`payload.html`) containing an auto-submitting form:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
</head>
<body onload="document.forms[0].submit()">
  <p>Loading, please wait...</p>
  <form action="http://localhost:8081/includes/upload.process.php" method="GET" style="display:none;">
    <input type="hidden" name="storage_selection" value="local">
    <input type="hidden" name="encrypt_file" value="0">
    <input type="hidden" name="name" value="csrf_triggered.txt">
    <input type="hidden" name="chunk" value="0">
    <input type="hidden" name="chunks" value="1">
    <input type="hidden" name="file" value="pwned_by_autosubmit">
  </form>
</body>
</html>
```

3. Host this file on a server under the attacker's control or distribute via phishing (link/email).
4. Trick the victim into visiting the page while authenticated.
5. Verify in the ProjectSend upload directory: file `csrf_triggered.txt` appears automatically.

Note: For stealthier delivery, an attacker can use a hidden `<iframe src="...">` instead of an auto-submit form (works if the target lacks `X-Frame-Options` or restrictive CSP `frame-ancestors`).

Impact

- Attackers can force authenticated users to upload arbitrary files without knowledge or consent.
- Depending on server configuration and user privileges, this may lead to:
  - Storage abuse / denial of service
  - Insertion of malicious content (e.g., webshells if file-type restrictions are weak)
  - Reputation damage or compliance violations
- Exploitation requires victim interaction and an active session, but remains practical in targeted attacks (phishing, watering-hole, etc.).

CVSS v3.1 Assessment

Base Score: 5.4
Severity: Medium
Vector String:
`CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N`

Breakdown:
- **AV:N** (Attack Vector: Network) — Exploitable remotely over the network.
- **AC:L** (Attack Complexity: Low) — No special conditions; trivial to craft URL/form.
- **PR:N** (Privileges Required: None) — Attacker needs no privileges.
- **UI:R** (User Interaction: Required) — Victim must visit malicious page/link.
- **S:U** (Scope: Unchanged) — Impact stays within ProjectSend.
- **C:N** (Confidentiality: None) — No data exposure.
- **I:L** (Integrity: Low) — Unauthorized file upload (arbitrary content insertion).
- **A:N** (Availability: None) — No direct DoS.

This aligns with typical CSRF scores for state-changing actions requiring user interaction (often 4.3–6.5 range), adjusted lower here due to limited integrity impact and no confidentiality/availability effect.

Mitigation Recommendations

1. **Enforce HTTP method validation** — Reject non-POST requests in `upload.process.php` (use `$_POST` and `$_FILES` exclusively, ignore `$_REQUEST` for sensitive operations).
2. **Implement anti-CSRF tokens** — Generate and validate unique, per-session/per-request tokens for all state-changing actions (upload, delete, etc.).
3. **Add SameSite cookies** — Set session cookies to `SameSite=Lax` or `Strict` to limit cross-site usage.
4. **Set security headers** — Include `X-Frame-Options: DENY` or `SAMEORIGIN` and CSP `frame-ancestors 'self'` to block iframe-based variants.
5. **Restrict upload file types/extensions** — Prevent dangerous files even if upload occurs.

Vulnerability Tracking

- A formal CVE request has been submitted to vuldb.com.
- CVE assignment for validated CSRF vulnerabilities (especially with method manipulation) is common when reported through proper channels (e.g., MITRE, vendor CNA, or VulDB).

Credits: @Executio (Exec Hall) and @0xHamy (Hamed Kohi)

User	AquaNight (UID 88991)
Submission	03/22/2026 07:01 PM (15 days ago)
Moderation	04/05/2026 06:51 PM (14 days later)
Status	Accepted
VulDB entry	355414 [ProjectSend r2002 upload.php cross-site request forgery]
Points	17