Server-Side Request Forgery (SSRF) (<=1.11.36)

High **ywarnier** published **GHSA-q74c-mx8x-489h** 2 weeks ago

Package

Chamilo LMS (PHP)

Affected versions

<=1.11.36

Patched versions

1.11.38, 2.0

Description

Summary

Field	Detail
Vulnerability	Server-Side Request Forgery (SSRF)
Affected Software	Chamilo LMS
Affected Version	≤ 1.11.36
Component	`main/inc/ajax/social.ajax.php` → `SocialManager::verifyUrl()` + `SocialManager::readContentWithOpenGraph()`
CVSS v3.1 Score	**7.7 (High)** — AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N
CWE	CWE-918: Server-Side Request Forgery (SSRF)
Authentication	Required (any authenticated user)

Vulnerability Description

Chamilo LMS contains a **Server-Side Request Forgery (SSRF)** vulnerability in the Social Wall feature. The endpoint `read_url_with_open_graph` accepts a URL from the user via the `social_wall_new_msg_main` POST parameter and performs **two server-side HTTP requests** to that URL without validating whether the target is an internal or external resource:

1. **Guzzle HTTP client** (`SocialManager::verifyUrl()`) — sends a GET request to check if the URL returns HTTP 200
2. **cURL** (`OpenGraph::fetch()` via `SocialManager::readContentWithOpenGraph()`) — fetches the page content and parses OpenGraph metadata

There is **no validation** on:

- IP address (private ranges: `127.0.0.0/8` , `10.0.0.0/8` , `172.16.0.0/12` , `192.168.0.0/16` , `169.254.0.0/16`)
- URL scheme (allows `http://` , `https://` , potentially `file://` , `gopher://` , `dict://`)
- Hostname (no blacklist for `localhost` , internal Docker DNS names, cloud metadata endpoints)

This allows an authenticated attacker to force the server to make arbitrary HTTP requests to internal services, scan internal ports, and access cloud instance metadata.

Impact

An authenticated attacker (any role including student) can:

1. **Scan internal network** — Enumerate live hosts and open ports behind the firewall by observing response times and status codes.
2. **Access internal services** — Read content from services that are only reachable from the server (e.g., admin panels, databases, monitoring tools, Docker internal DNS).
3. **Steal cloud credentials** — On cloud-hosted instances (AWS, GCP, Azure), access the metadata endpoint (`169.254.169.254`) to obtain IAM roles, access keys, and security tokens.
4. **Bypass network segmentation** — The server acts as a proxy, allowing attackers to reach resources behind firewalls and network ACLs.
5. **Chain with other vulnerabilities** — Use SSRF as a pivot to exploit internal services (e.g., unauthenticated admin panels, Redis, Elasticsearch).

Affected Code Locations

File	Line(s)	Issue
`main/inc/ajax/social.ajax.php`	354–366	`$_POST['social_wall_new_msg_main']` passed to `verifyUrl()` and `readContentWithOpenGraph()` without IP/scheme validation
`main/inc/lib/social.lib.php`	2117–2138	`verifyUrl()` — Guzzle GET request to arbitrary URL, no internal IP blocking
`main/inc/lib/social.lib.php`	2086–2112	`readContentWithOpenGraph()` — passes URL to `OpenGraph::fetch()`, returns parsed content to user
`main/inc/lib/opengraph/OpenGraph.php`	50–63	`fetch()` — cURL request with `FOLLOWLOCATION`, no scheme or IP restriction

---

Remediation

Fix 1 — Validate URL scheme and block internal IPs

File: `main/inc/lib/social.lib.php`, before line 2122 in `verifyUrl()` and line 2088 in `readContentWithOpenGraph()`

```
private static function isUrlSafe(string $url): bool
{
    $parsed = parse_url($url);

    // Allow only http and https schemes
    if (!isset($parsed['scheme']) || !in_array($parsed['scheme'], ['http', 'https'])) {
        return false;
    }

    $host = $parsed['host'] ?? '';
    if (empty($host)) {
        return false;
    }

    // Resolve hostname to IP
    $ip = gethostbyname($host);
    if ($ip === $host) {
        return false; // DNS resolution failed
    }

    // Block private/reserved IP ranges
```

```
    $blockedRanges = [
        '127.0.0.0/8',      // Loopback
        '10.0.0.0/8',       // Private Class A
        '172.16.0.0/12',    // Private Class B
        '192.168.0.0/16',   // Private Class C
        '169.254.0.0/16',   // Link-local (cloud metadata)
        '0.0.0.0/8',        // Non-routable
        '100.64.0.0/10',    // Shared address space
        'fc00::/7',         // IPv6 private
        '::1/128',          // IPv6 loopback
    ];

    $ipLong = ip2long($ip);
    foreach ($blockedRanges as $cidr) {
        [$range, $bits] = explode('/', $cidr);
        $rangeLong = ip2long($range);
        if ($rangeLong === false) continue;
        $mask = -1 << (32 - (int)$bits);
        if (($ipLong & $mask) === ($rangeLong & $mask)) {
            return false;
        }
    }

    return true;
}
```

Apply the check:

```
public static function verifyUrl(string $uri): bool
{
    if (!self::isUrlSafe($uri)) {
        return false;
    }
    // ... existing Guzzle code ...
}
```

Fix 2 — Disable dangerous cURL options

File: `main/inc/lib/opengraph/OpenGraph.php`, lines 53–54

```
// Before (dangerous)
curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);

// After (safer — limit redirects and block internal IPs)
curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
curl_setopt($curl, CURLOPT_MAXREDIRS, 3);
curl_setopt($curl, CURLOPT_PROTOCOLS, CURLPROTO_HTTP | CURLPROTO_HTTPS);
curl_setopt($curl, CURLOPT_REDIR_PROTOCOLS, CURLPROTO_HTTP | CURLPROTO_HTTPS);
```

Fix 3 — Re-validate IP after DNS resolution (anti-DNS rebinding)

```php
public static function verifyUrl(string $uri): bool
{
    if (!self::isUrlSafe($uri)) {
        return false;
    }

    $client = new Client();
    try {
        $response = $client->request('GET', $uri, [
            'timeout' => 10,
            'verify' => false,
            'allow_redirects' => ['max' => 3],
            'on_stats' => function ($stats) {
                // Re-check resolved IP after redirect
                $effectiveUri = $stats->getEffectiveUri();
                if (!SocialManager::isUrlSafe((string)$effectiveUri)) {
                    throw new \RuntimeException('Redirect to internal IP blocked');
                }
            },
        ]);
        return 200 === $response->getStatusCode();
    } catch (Exception $e) {
        return false;
    }
}
```

References

- CWE-918: Server-Side Request Forgery
- OWASP SSRF Prevention Cheat Sheet
- PortSwigger: SSRF
- Chamilo LMS GitHub

Severity

High 7.7 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	Low
User interaction	None
Scope	Changed
Confidentiality	High

Integrity	None
Availability	None

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N

CVE ID

CVE-2026-31941

Weaknesses

No CWEs

Credits

 **elliSzAt** Reporter