SEVERITY: HIGH CONFIDENTIAL

# Paid Video Access Control on slvlog.live

Two findings, both High: unlisted YouTube embeds are sent to anonymous visitors, and a paid Vimeo recording is watchable without an account because its only protection is a Referer header the viewer can forge

One consolidated report for the SLVLOG development team - supersedes the separate security advisory and Vimeo bug report.

| TARGET PLATFORM | PREPARED BY | COMPILED | CLASSIFICATION |
|---|---|---|---|
| SLVLOG Web Application | Dinujaya Sandaruwan | November 9, 2026 | CWE-602 / CWE-200 |

---

CONTENTS

PART I - UNLISTED YOUTUBE EMBEDS EXPOSED TO UNAUTHENTICATED VISITORS

1. Executive Summary
2. Vulnerability Classification & Metrics
3. Technical Root Cause Analysis
4. Technical Walkthrough & Affected Media
5. Risk & Architectural Impact
6. Recommended Fix

PART II - VIMEO PAID VIDEO WATCHABLE WITHOUT AN ACCOUNT (SLVLOG-EMBED-403)

7. Summary
8. The Video ID in the Page Source
9. Reproduction Evidence
10. Root Causes, Ranked
11. Watching It Without an Account
12. Recommended Fix
13. Verification Checklist
14. Reproduction Environment
15. Suggested Remediation Order

HOW TO READ THIS REPORT

The two findings are independent, but they are the same mistake in two places. **Part I**: unlisted YouTube embeds are serialized into public HTML, so anyone holding one of the leaked links can play the full episode without a subscription. **Part II**: the paid Vimeo recording has its ID in the page source too, and the only control left is Vimeo's domain restriction - which checks a `Referer` header the caller writes, so copying the ID and claiming to be slvlog.live yields the full stream with no account (section 11). Part II also documents the opposite symptom: the same setting currently refuses legitimate viewers with HTTP 403. Each part proposes one fix, with one further step for Vimeo - section 15 ranks them.

PART I · SECURITY ARCHITECTURE ADVISORY · SEPTEMBER 2026

# Unlisted YouTube Embeds Are Sent to Unauthenticated Visitors

Client-side media delivery & access control analysis of the SLVLOG web application.

SEVERITY: HIGH | HERO BANNER & FEATURED CAROUSEL | WATCHABLE OUTSIDE THE SITE | CWE-602 / CWE-200

## 1. Executive Summary

A technical architecture assessment of the front-facing web platform identified an access control flaw in how protected video assets are delivered to unauthenticated clients.

While the user interface enforces subscription paywalls visually (prompting users to subscribe or sign in), the underlying streaming sources and unlisted video player embed codes are transmitted in full plaintext within the initial server-side HTML response sent to anonymous visitors. Consequently, access control is enforced only on the visual client layer, leaving restricted assets directly accessible at the data transport layer.

The practical effect is simple: **unlisted does not mean private**. Anyone who receives one of these embed URLs can watch the complete episode on YouTube - no account, no subscription, and no visit to slvlog.live.

> **CORE FINDING**
>
> Access control relies on client-side rendering suppression rather than server-side authorization gating. Full unlisted streaming embed sources are transmitted to browsers prior to authentication.

## 2. Vulnerability Classification & Metrics

| CLASSIFICATION | AFFECTED COMPONENT | TARGET SCOPE | IMPACT |
|---|---|---|---|
| **CWE-602:** Client-Side Security Enforcement | Hero Banner & Featured Carousel Initializer | Featured Episodes & Series Content | Paywall Bypass for Streaming Embeds |
| **CWE-200:** Sensitive Data Exposure | Inline Script Injection ( `window.featuredContent` ) | Unlisted Streaming Endpoints | Direct Intellectual Property Disclosure |

## 3. Technical Root Cause Analysis

In secure web architecture, authorization boundaries must reside strictly on the server. A client cannot be trusted to self-regulate access to data it has already received.

During homepage rendering, the backend template injects a JavaScript object named `window.featuredContent` into the DOM inside an inline `<script>` block to hydrate the featured slider. However, the database serialization query includes the sensitive content field containing raw unlisted streaming iframe embeds:

EXHIBIT A - SERVER-RENDERED DATA LEAK IN PUBLIC HTML PAYLOAD ( `WINDOW.FEATUREDCONTENT` )

```
<script>
window.featuredContent = [
   {
     "id": 471,
     "title": "ලසන්ත තාමත් අපි අතර සිනාසෙනවා (2026-09-21)",
     "slug": "-456",
     "thumbnail": "media/CIjjOjXp1TRfRQK8QceugMLpfTV1GoOQUTWrUNuS.jpg",
     "banner": "media/Ui3sSBECEIBj5SdOPc4ci9O8USDjkCMrkiUGdcRJ.jpg",
     "published": 1,
     "featured": true,
     // CRITICAL: Full unlisted embed iframe transmitted to unauthenticated client
     "content": "<iframe width="560" height="315" src="https://www.youtube.com/embed/IfBSXW15ejE?si=..." ...>
</iframe>",
     "access_level_from": 0,
     "watch_on": [5, 10, "4", "3", "7"]
   }, ...
];
</script>
```

## 4. Technical Walkthrough & Affected Media Instances

An anonymous user without an account or an active subscription initiates an unauthenticated HTTP GET request to the root URL ( `/` ). The response payload exposes multiple featured episodes that are intended to be gated behind a subscription:

| ITEM ID | CONTENT TITLE | TARGET SLUG | ASSET ID |
|---|---|---|---|
| #471 | ලසන්ත තාමත් අපි අතර සිනාසෙනවා (2026-09-21) | /-456 | IfBSXW15ejE |
| #470 | Tragedy of Saman Kumara - Episode 23 | /tragedy-of-saman-… | PZ6dwLkuXlc |
| #468 | නාඳ ගම පටන් ගත්ත තැන | /-453 | 2Mw20wSBXvY |
| #467 | වඳ මිනිසාගේ අඳෝනාව (2026-09-17) | /-452 | fkxUkKfHIjs |

EXHIBIT B - AUTOMATED EXTRACTION PATTERN (DIRECT HTTP INSPECTION)

```
$ curl -sL https://slvlog.live/ | grep -o 'https://www.youtube.com/embed/[a-zA-Z0-9_-]*'
# Output demonstrates instant leakage without session or authentication:
https://www.youtube.com/embed/IfBSXW15ejE
https://www.youtube.com/embed/PZ6dwLkuXlc
https://www.youtube.com/embed/2Mw20wSBXvY
https://www.youtube.com/embed/fkxUkKfHIjs
```

**THESE FOUR UNLISTED VIDEOS CAN BE WATCHED BY ANYONE**

The asset IDs above are the only thing needed to play these episodes. Opened directly on YouTube as `https://www.youtube.com/embed/<asset-id>`, each one plays the complete video in the browser - no account, no subscription and no visit to slvlog.live. The links can be copied, shared and bookmarked, and they keep working after the visitor leaves the site.

## 5. Risk & Architectural Impact Analysis

- **Direct monetization invalidation.** The platform's commercial model relies on paid subscriptions for exclusive series episodes. When playback URLs can be retrieved directly from the public HTML source, subscription gating is rendered ineffective.
- **Limitations of “unlisted” privacy.** Hosting videos as “unlisted” does not constitute authorization. Once an unlisted identifier is delivered to client-side code, it can be shared, bookmarked, or streamed independently without hitting the platform's authentication layer.
- **Client-side gatekeeper ineffectiveness.** UI overlays, modal dialogues, and redirect scripts provide visual barriers only. They provide no security if the underlying payload is sent across the wire before identity verification.

**ARCHITECTURAL NOTE**

This is a separation issue between frontend UI presentation and backend data authorization. Because the full embed payload is serialized into the initial page template, any client can access the underlying stream without authenticating. The fix belongs on the server: the featured slider must receive presentation data only - never raw stream identifiers - until the subscription check has passed.

## 6. Recommended Fix - Authorise on the Server, Then Send the Stream

One change closes the leak, and it leaves the interface exactly as it is. The carousel, the subscription overlay and the paywall prompts all stay; only the moment the playable URL is handed out moves.

**THE WHOLE FIX** **Take `content` out of the public payload and serve it from one authenticated endpoint**

1. **Strip the media field from the serializer.** `window.featuredContent` should carry only what a card needs to render - `id`, `title`, `slug`, `thumbnail`, `banner` and a `locked` flag. No `content`, no iframe, no stream URL.
2. **Add one endpoint, for example `GET /api/episodes/<id>/playback`.** It checks the session and the subscription on the server and returns the embed URL only when that check passes, otherwise `403`. This becomes the only place a video ID is ever sent.
3. **Point the play action at it.** A subscriber gets the URL and plays as today; an anonymous visitor gets the existing subscribe prompt and never receives a playable URL at all.

PUBLIC PAYLOAD BEFORE AND AFTER

```
// before — every anonymous visitor can read the playable URL
{ "id": 471, "title": "…", "content": "<iframe src=\"https://www.youtube.com/embed/IfBSXW15ejE…\">" }

// after — the public payload carries no media, only a state flag
{ "id": 471, "title": "…", "locked": true }

// returned only to an authorised viewer by /api/episodes/471/playback
{ "embed": "https://www.youtube.com/embed/IfBSXW15ejE" }
```

**ONE THING THIS FIX DOES NOT DO**

It stops the site from handing out the links. It does not make an unlisted YouTube URL private: once a paying viewer has it, the link still plays anywhere and YouTube does not tell you who watched. If the recordings themselves must stay controlled, they need a host that authorises each viewer rather than unlisted visibility.

**PART II · SECURITY FINDING · NOVEMBER 4, 2026**

## Vimeo Paid Video Is Watchable Without an Account

Vimeo video 1242676130 - "NADHAGAMA MELBORNE SL VLOG" · the recording slvlog.live embeds for subscribers.

SLVLOG-EMBED-403 | SEVERITY: HIGH - PAID CONTENT WATCHABLE | REFERER-ONLY GATE | NO ACCOUNT REQUIRED

## 7. Summary

**Bug ID:** SLVLOG-EMBED-403 · **Severity:** High (paid video watchable without an account) · **Component:** Vimeo embed published on slvlog.live · **Reporter:** Dinujaya Sandaruwan

The paid Vimeo recording is published to anonymous visitors. Its numeric ID sits in the page source that slvlog.live returns to anyone - no login, no subscription - and that same ID addresses the video directly on Vimeo.

The only thing between that ID and the video is Vimeo's embed setting, which allows playback for requests claiming to come from slvlog.live. The claim is an HTTP `Referer` header, written by whoever makes the request. Send the ID with `Referer: https://slvlog.live/` and Vimeo answers **HTTP 200** with the full signed HLS configuration; the recording then plays anywhere, with no account and no subscription (section 11).

The same setting produces the opposite failure for real viewers: because the live hosts are not correctly allow-listed, their player request is refused with **HTTP 403** and the page shows a blank box. The net result is the worst of both worlds - **legitimate visitors cannot watch, and anyone who copies the ID can.**

## 8. The Video ID in the Public Page Source

This is Part I's delivery pattern, repeated on Vimeo: the embed below is part of the page HTML sent to every visitor, subscribed or not. Nothing on the server checks who is asking before the video identifier is handed out.

```
<div style="padding:56.25% 0 0 0;position:relative;">
  <iframe
    src="https://player.vimeo.com/video/1242676130?badge=0&amp;autopause=0&amp;player_id=0&amp;app_id=58479"
    frameborder="0"
    allow="autoplay; fullscreen; picture-in-picture; clipboard-write; encrypted-media; web-share"
    referrerpolicy="strict-origin-when-cross-origin"
    style="position:absolute;top:0;left:0;width:100%;height:100%;"
    title="NADHAGAMA MELBORNE SL VLOG">
  </iframe>
</div>
<script src="https://player.vimeo.com/api/player.js"></script>
```

**THE EMBED IS CORRECT CODE PUBLISHED TO THE WRONG AUDIENCE**

`allow`, `referrerpolicy`, `app_id`, the responsive wrapper and the Player SDK tag are exactly what Vimeo's embed generator produces. What is wrong is who receives it: `1242676130` is readable from View Source, a network log or a saved page, with no account and no subscription - which is the starting point of the bypass in section 11.

## 9. Reproduction Evidence - From Public Page to Playback

```
# --- evidence 1: the video ID is in the page served to anonymous visitors
$ curl -s https://slvlog.live/ | grep -o "player.vimeo.com/video/[0-9]*"
player.vimeo.com/video/1242676130    # no login, no subscription needed to read it

# --- evidence 2: Vimeo refuses the embed when no approved referrer is sent
$ curl -i "https://player.vimeo.com/video/1242676130?badge=0&autopause=0&player_id=0&app_id=58479"
HTTP/2 403                      # "Sorry" page - what a real viewer sees

# --- evidence 3: oEmbed confirms the video is domain-restricted, not open
$ curl -s "https://vimeo.com/api/oembed.json?url=https://vimeo.com/1242676130"
{"type": "video", "video_id": 1242676130,
 "domain_status_code": 403, ...}      # only guard = domain restriction

# --- evidence 4: same request with the site's Referer -> signed stream
$ curl -i -H "Referer: https://slvlog.live/" \
       "https://player.vimeo.com/video/1242676130/config"
HTTP/2 200
{"request": {"files": {"hls": {"cdns": {
      "fastly_skyfire":            {"avc_url": "https://vod-adaptive-*.vimeocdn.com/exp=<ts>~hmac=
<sig>/.../playlist.m3u8"},
      "akfire_interconnect_quic": { ... } } } } },
 "video":   {"title": "NADHAGAMA MELBORNE SL VLOG", "duration": 14944, ...}}
```

### Step-by-step interpretation

| # | CHAIN STEP | EXPECTED | OBSERVED / WHERE IT BREAKS |
|---|---|---|---|
| 1 | Anonymous visitor loads the slvlog.live page and reads its HTML | No video identifier in public HTML | `player.vimeo.com/video/1242676130` present in the response - the ID is readable with no account and no subscription. |
| 2 | Player request without an approved `Referer` | Player HTML (HTTP 200) | HTTP 403 “Sorry” - precisely what a real viewer experiences as a dead player box. |
| 3 | oEmbed probe: `domain_status_code` | 200 if “embed anywhere”, 403 if specific-domains | **403** - the video's only guard is a domain restriction, so the next row is the whole test. |
| 4 | Config request for the same ID, adding `Referer: https://slvlog.live/` | Refused, if the gate were an authorisation check | **HTTP 200** with the full signed HLS configuration - the gate judged only a header, and the stream is now playable anywhere. |

## 10. Root Causes, Ranked

PRIMARY **The paid video is delivered to anonymous visitors**

The embed and its ID are serialized into the public HTML, exactly as in Part I. Nothing on the server checks whether the requester is a subscriber before the identifier is sent; the subscription only decides what the interface draws. That exposure (CWE-200) is what makes section 11 possible.

SECONDARY **The only remaining guard is a header the caller writes**

Vimeo's "specific domains" setting evaluates the `Referer` header. It refuses requests with no approved referrer - the 403 real viewers see - and returns the signed configuration whenever the header reads `https://slvlog.live/`. It is a hotlink control, not an access control, so it cannot protect paid content on its own.

TERTIARY **CDN stalls on the demuxed audio playlist (network robustness)**

| REQUEST (SAME URL) | DIRECT FETCH (CURL) | THROUGH A 4×15S RETRY PROXY |
|---|---|---|
| master playlist, `fastly_skyfire` | 0.3 s | 0.3 s |
| master playlist, `akfire_interconnect_quic` | 1.5 s | 22.4 s (1 stall + 1 retry) |
| audio playlist (545 KB) | 0.6 / 0.9 / 5 / 7.6 / 17.6 s … occasionally over 120 s | 0.7 s (retries hide stalls) |
| video segment (~4 MB) | n/a | 0.6 - 2.3 s, every fetch OK |

The video ships audio as a separate HLS track; its playlist is large (545 KB / ~2,494 entries - the recording is 4 h 9 min) and the CDN intermittently stalls far past any player's sensible timeout. Symptom: `audioTrackLoadTimeOut` - video frames render, audio never starts, or the player dies. This affects slvlog.live viewers on constrained networks too, wherever the player is embedded.

ADDITIONAL **Silent failure when the Player SDK cannot load**

`player.vimeo.com/api/player.js` is loaded from an external host at page render time. During testing that request hung on a constrained network, leaving a blank player with no message - which also masks primary cause (10.1) from viewers and support staff.

## 11. Watching It Without an Account

This is the counterpart to Part I. The Vimeo setting does not protect the recording; it only checks whether a request *claims* to come from an approved domain.

Vimeo's embed gate reads the `Referer` header, which is supplied by the requester. Send the same player request with `Referer: https://slvlog.live/` and the gate answers **HTTP 200** with the full signed HLS configuration (evidence 3 in section 9). With that one header in place, the stream plays in a player that is not on slvlog.live at all - no login, no password, no membership.

**WATCHING OUTSIDE THE OFFICIAL SITE WAS REPRODUCED**

Playback of video 1242676130 away from slvlog.live, with no account and no subscription, was reproduced using nothing but the ID read from the public page: a small local player requests the stream configuration while presenting the site as the referrer, then plays the manifests Vimeo returns. The player itself is not part of this report, and nothing specialised is needed - the mechanism is one standard HTTP header, so anyone who knows the video ID can do the same.

Two consequences for SLVLOG:

- **The embed restriction is a hotlink control, not an access control.** It is still worth configuring correctly so genuine visitors can watch - that is the outage described in section 7 - but it cannot decide who may watch.
- **If the vlog must be restricted, the control has to live with the host or the server:** a Vimeo password or private link (a shared secret, but far stronger than a header check), or delivery through SLVLOG's own authorised endpoint. The same reasoning applies to the YouTube embeds in Part I.

## 12. Recommended Fix - Gate Delivery, Then Authorise the Viewer

Two changes, in this order. The first stops the ID reaching anonymous visitors; the second makes a copied ID worthless.

**REQUIRED** **Serve the Vimeo embed only to authenticated subscribers**

The same pattern as section 6, applied to this page: the public payload carries the card and a `locked` flag, never the video ID.

1. **Remove the iframe from the page HTML.** Nothing an anonymous visitor receives should contain `player.vimeo.com/video/1242676130`.
2. **Return it from one authenticated endpoint**, for example `GET /api/episodes/<id>/playback`, which checks the session and the subscription and answers `403` otherwise.
3. **Play from that response.** A subscriber sees the same player as today; everyone else gets the subscribe prompt and never learns the ID.

**WHILE YOU ARE IN VIMEO STUDIO**

The 403 that blocks genuine viewers is this same setting seen from the other side. Allow-list both `slvlog.live` and `www.slvlog.live` - or choose “Anywhere”, which costs nothing once the ID is no longer public - so that closing the exposure does not leave real visitors on Vimeo's “Sorry” page.

REQUIRED · CLOSES THE BYPASS **Authorise the viewer, not the domain**

Gating delivery removes the free scrape. It does not stop a subscriber - or anyone they forward the ID to - from replaying the `Referer` trick, and Vimeo's domain restriction cannot prevent that: it is a hotlink control. Per-viewer protection has to come from the host:

- **Signed, expiring playback URLs** issued after the subscription check - a host that supports per-session tokens, or SLVLOG's own signed HLS delivery. A copied URL stops working, and a copied ID returns nothing.
- **If Vimeo must stay:** its per-viewer privacy options (password or private link) are the available control. Treat the password as a shared secret - a viewer can pass it on - but it is far stronger than a header the caller writes.

## 13. Verification Checklist

- [ ] **No ID in public HTML**
  An anonymous request to the page returns no `player.vimeo.com/video/…` and no `1242676130`.
- [ ] **Endpoint gated**
  Unauthenticated `GET /api/episodes/<id>/playback` → `401` / `403`; a subscribed session → `200` with the embed.
- [ ] **Referer trick dead**
  Take the ID and request the stream with `Referer: https://slvlog.live/` but no session - the response must not yield a playable signed URL.
- [ ] **Real viewers**
  The video plays on slvlog.live and on `www.`; logged out, the page shows the subscribe prompt rather than a blank box.
- [ ] **Slow connection**
  Long-play on a constrained network; if audio gaps persist, offer a download or a lower default rendition.

## 14. Reproduction Environment

macOS · Google Chrome · residential/constrained network. Every probe above is a plain curl command. The streaming measurements come from a small local HLS player built for that purpose; neither the player nor its source is attached to this report, and no specialised tooling is needed to reproduce any finding.

## 15. Suggested Remediation Order

Both findings are rated High. Part I exposes episodes; Part II exposes the paid recording and, at the same time, breaks the player for real viewers. Four steps, in this order:

| STEP | ACTION | WHERE | EFFORT |
|---|---|---|---|
| 1 | Allow-list both host variants in Vimeo Studio (section 12) - restores playback for legitimate viewers today | Vimeo Studio | Minutes, no deploy |
| 2 | Take the Vimeo ID out of the public payload and serve it from an authenticated endpoint (sections 6 and 12) | Backend serializer + one endpoint | Small change |
| 3 | Move paid recordings to per-viewer authorisation - signed, expiring URLs - so a copied ID is worthless (section 12) | Hosting / architecture | Decision + migration |
| 4 | Apply the same rule to the YouTube episodes, which play for anyone once an ID leaks (section 6) | Product / architecture | Decision |

---

This consolidated report was compiled by **Dinujaya Sandaruwan** to assist the SLVLOG engineering team in strengthening application security and media protection, and in restoring reliable playback of the embedded vlog. It combines the September 2026 security architecture advisory and the November 4, 2026 Vimeo bug report without changing either set of findings.

I would welcome the opportunity to discuss these findings in detail with your development team and collaborate on implementing robust, server-validated media delivery architectures. The player used for the streaming measurements can be shared separately if the team wants to reproduce the results.

Prepared for the SLVLOG Development Team · Confidential · Compiled November 9, 2026