JISKA (Jurnal Informatika Sunan Kalijaga)
Vol. 11, No. 3, SEPTEMBER, 2026, Pp. 427 – 437
ISSN: 2527 – 5836 (print) | 2528 – 0074 (online)

From Open Redirect to JavaScript Execution via CVE-2024-4367

Muhammad Hendra Sunarya [(1)], M. Nanda Dede Nugroho [(2)*], Reja Revaldy F [(3)], Ryan Rizky Pratama [(4)]
[1] Department of Accounting, Politeknik Negeri Banjarmasin, Banjarmasin, Indonesia
[2] PT. Vantage Point Teknologi, Banjarmasin, Indonesia
[2,3,4] Department of Electrical Engineering, Politeknik Negeri Banjarmasin, Banjarmasin, Indonesia
e-mail : hendra@poliban.ac.id, {nugroho,reja,ryan}@genbucyber.com.
* Corresponding author.
This article was submitted on 9 June 2026, revised on 28 July 2026, accepted on 29 July 2026, and published on 25 September 2026.

Abstract

This case study presents a forensically documented analysis of a multi-stage security incident involving CVE-2024-4367, an arbitrary JavaScript execution vulnerability in PDF.js. The incident occurred on one PKP Open Journal Systems (OJS) deployment. The observed chain involved an open redirect in the host platform's sign-out handler (CWE-601), a prefix-match URL validation weakness in the PDF.js viewer wrapper, and PDF.js font parsing behavior associated with CVE-2024-4367. The evidence demonstrates execution of attacker-controlled JavaScript in the viewer context and an SEO-poisoning effect; it does not establish a general vulnerability in all OJS, WordPress, Drupal, or enterprise deployments. CVSS scores are reported separately for the identified CVE and are not combined into a fabricated composite score. A defense-in-depth mitigation strategy combining reverse-proxy rules, library patches, and Content Security Policy was deployed and verified.

Keywords: CVE-2024-4367, PDF.js, Open Redirect, Vulnerability Chaining, SEO Poisoning

Abstrak

Studi kasus ini menyajikan analisis forensik insiden keamanan multi-tahap yang melibatkan CVE-2024-4367, yaitu kerentanan eksekusi JavaScript arbitrer pada PDF.js. Insiden terjadi pada satu deployment Open Journal Systems (OJS) PKP. Rantai yang diamati mencakup redirect terbuka pada handler sign-out (CWE-601), kelemahan validasi URL berbasis pencocokan prefiks pada wrapper PDF.js, dan perilaku parsing font PDF.js yang terkait dengan CVE-2024-4367. Bukti menunjukkan eksekusi JavaScript yang dikendalikan penyerang pada konteks viewer dan dampak SEO poisoning; penelitian ini tidak membuktikan bahwa seluruh deployment OJS, WordPress, Drupal, atau aplikasi enterprise rentan. Skor CVSS CVE dilaporkan secara terpisah dan tidak digabungkan menjadi skor komposit. Mitigasi defense-in-depth berupa aturan reverse-proxy, patch pustaka, dan Content Security Policy diterapkan serta diverifikasi.

Kata Kunci: CVE-2024-4367, PDF.js, Redirect Terbuka, Rantai Kerentanan, SEO Poisoning

1. INTRODUCTION

Open Journal Systems (OJS), developed by the Public Knowledge Project (PKP), is the most widely adopted open-source platform for academic journal management and publishing, with over 34,000 active installations worldwide (PKP, 2025). Serving as critical infrastructure for scholarly communication across disciplines, OJS deployments process peer-reviewed manuscripts, manage editorial workflows, and serve published content to global audiences. The security of these platforms directly affects the integrity of the academic record and the trustworthiness of scholarly dissemination channels. The incident examined in this study occurred on an OJS instance, but the vulnerability at its core— CVE-2024-4367— resides in Mozilla's PDF.js library. This case study, however, examines only its deployment within the affected OJS instance.

PDF.js is Mozilla's portable Document Format (PDF) viewer, implemented in JavaScript and deployed as the default PDF renderer in Mozilla Firefox. Beyond the browser, the pdfjs-dist npm

package is one of the most widely integrated document-rendering libraries on the web, with millions of weekly downloads (npm, 2024). It is embedded in content management systems, learning management systems, document management platforms, e-signature workflows, and countless custom web applications. This study did not test the presence or exploitability of this chain in other PDF.js integrations. The security landscape for OJS itself has evolved considerably. PKP maintains a security advisory process (PKP, 2024a) and has addressed vulnerabilities including CVE-2020-8518 (PHP object injection), CVE-2021-4118 (deserialization in OJS 3.2), and CVE-2023-42457 (path traversal) (NIST, 2020)-(NIST, 2023). These incidents underscore that OJS, like other content management platforms, presents an evolving attack surface requiring ongoing vigilance (OWASP, 2024a), (MITRE, 2023a).

This study examines a real-world incident at an anonymized university electronic journal platform (7 May 2026) demonstrating an exploit chain: open redirect (CWE-601) (MITRE, 2024a), subdomain spoofing via prefix-match same-origin bypass (CWE-345) (MITRE, 2024b), and CVE-2024-4367— arbitrary JavaScript execution in PDF.js via FontMatrix objects, disclosed in the relevant Mozilla/PDF.js security materials (NIST, 2024a). While the incident occurred on one OJS deployment, the architectural prerequisites may exist elsewhere, but were not tested in this study. The chain requires (a) a web application embedding pdfjs-dist ≤ 4.1.392, (b) a PDF viewer wrapper with insufficient URL validation, and (c) an open redirect endpoint on the same origin. These conditions define the scope for future replication; their prevalence in other platforms was not evaluated here. While open redirects (MITRE, 2024a), DOM-based XSS (Dinicola & Luchaup, 2024)-(Tang et al., 2023), and PDF-based code execution (Kharraz et al., 2022) have been studied in isolation, forensic documentation of their combined exploitation in real-world infrastructure has been limited.

This paper makes the following contributions. First, forensic documentation with reproduction testing of a multi-stage exploit chain targeting CVE-2024-4367 through a real-world web platform, including the complete malicious URL structure, PDF payload dissection, and post-exploitation indicators. Second, root cause analysis demonstrating that string prefix matching on URLs fails to enforce same-origin policy as formalized in RFC 6454 (Barth, 2011)— an anti-pattern that is platform-agnostic and relevant to any PDF.js integration. Third, mapping of the attack chain to MITRE ATT&CK techniques T1189 (Drive-by Compromise) and T1203 (Exploitation for Client Execution) (MITRE, 2023b). Fourth, a defense-in-depth mitigation strategy with copy-pasteable configuration snippets applicable to Nginx, Apache, and other reverse proxies. Fifth, bounded recommendations for administrators and developers whose deployments have the same configuration; cross- platform applicability is presented as a subject for future validation. The significance of this incident is bounded by the evidence from the single OJS deployment. The observed pattern may be relevant to other applications that independently satisfy the same prerequisites, but this study does not establish prevalence or exploitability in WordPress, Drupal, enterprise applications, or the web as a whole.

2. METHODS

2.1 Case Study Design

This research employs a single-case study design following Runeson et al. (Runeson et al., 2022), selected because the incident represents what Runeson et al. terms a revelatory case: an opportunity to observe and analyze a phenomenon previously inaccessible to systematic investigation. The case is instrumentally bounded to the OJS incident as the empirical locus. The findings are not statistically generalized to other platforms. Three research propositions are defined: (P1) the observed string-based URL validation in the examined viewer wrapper created a same-origin bypass condition; (P2) the documented weaknesses formed a multi-stage attack path whose impact is described through prerequisites, evidence, and outcomes rather than a composite CVSS score; and (P3) the deployed reverse-proxy, application, and browser mitigations blocked the observed attack path in the examined deployment.

To address the validity constructs proposed by Runeson et al. (Runeson et al., 2022), the following measures are established: construct validity through multiple sources of evidence (forensic report, reproduction testing, IoC verification) and a documented chain of custody for the weaponized PDF artifact; internal validity through pattern matching between the observed attack chain and established vulnerability taxonomies (CWE, CVSS, MITRE ATT&CK); external validity through explicit scope boundaries rather than a claim of analytical generalization beyond the examined deployment; and reliability through a documented case study protocol including the reproduction testing procedure. The unit of analysis is the exploit chain as a system, bounded temporally to the 7 May 2026 incident. Architecturally, the chain is defined by the configuration: any web platform embedding pdfjs-dist ≤ 4.1.392 combined with a PDF viewer wrapper with URL-based file loading and an open redirect endpoint on the same origin. A case study database was established containing the forensic report, IoC log, reproduction test results, and reference verification records.

2.2 Data Collection

Primary data was collected through direct forensic examination of the affected OJS instance, including Nginx access logs showing the attacker's HTTP requests, plugin configuration files, and the weaponized PDF payload (SHA-256: aa8423846ac99b5c718f534930b7ff8484c1ed054f8bcfe1e243d70f150a2792). A timeline of events was reconstructed from log timestamps spanning the detection-to-mitigation window. Secondary data sources include the CVE-2024-4367 NVD entry (NIST, 2024a) and the correct Mozilla/PDF.js security release material (Mozilla Foundation, 2024); CWE-601 (MITRE, 2024a), CWE-79 (MITRE, 2024c), and CWE-345 (MITRE, 2024b) from MITRE; the PKP pdfJsViewer repository (PKP, 2023) as a representative PDF.js wrapper implementation; the CVE-2024-4367 weaponization proof-of-concept (Exfil0, 2024); OWASP Cheat Sheets for Open Redirect, XSS Prevention, and CSP (OWASP, 2024a)–(OWASP, 2024c); prior OJS CVEs (NIST, 2020)–(NIST, 2023); npm registry data on pdfjs-dist adoption (npm, 2024); RFC 6454 (Barth, 2011) and RFC 3986 (Berners-Lee et al., 2005) for URL and origin semantics; and the MITRE ATT&CK Framework v14 (MITRE, 2023b). All online sources were verified for accessibility as of May 2026.

2.3 Attack Chain Reconstruction

The attack chain was reconstructed following the MITRE ATT&CK framework (MITRE, 2023b). The six stages were mapped to specific ATT&CK techniques: Stages 1–2 (initial access via malicious URL) correspond to T1189 Drive-by Compromise, while Stages 3–5 (exploitation of pdfJsViewer validation flaw and CVE-2024-4367) correspond to T1203 Exploitation for Client Execution. Reproduction testing was conducted on a staging environment replicating the production deployment (OJS 3.3.0-10, PHP 7.4, Nginx 1.26, pdfjs-dist 4.1.392). Two test scenarios were executed with five trials each: Method 1 (Benign PDF) to isolate the subdomain spoofing mechanism independent of CVE-2024-4367, and Method 2 (Weaponized PDF) to confirm end-to-end exploit viability. Pass criteria required: (a) same-origin bypass confirmed via browser devtools network tab, (b) JavaScript execution verified via console output capture; CSP behavior was assessed separately during post-mitigation testing. Negative results were documented for a variant test attempting direct PDF injection without the open redirect pivot.

2.4 Vulnerability Analysis Framework

The exploit chain is modeled as a directed acyclic graph (DAG) where vertices represent individual weaknesses and directed edges represent exploit prerequisites. CVSS is not recalculated by changing one metric of an existing CVE, and CVSS scores are not added or combined to create a chain score. Therefore, this study reports the authoritative CVSS assessment for CVE-2024-4367 separately and describes the chain using prerequisite, evidence, and impact fields in Table 2. At the time of revision, the NVD record reports CVSS v3.1 CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H with score 8.8. The manuscript does not claim a CVSS v4.0 score for this CVE. The viewer URL validation weakness has no CVE score in this

study and is reported as a technical weakness with its observed evidence. The subdomain validation issue is described without asserting that CWE-345 is a definitive root classification.
Table 1 consolidates the tested deployment, component versions, observed roles, evidence sources, and remediation status. Table 3 is the primary record of the reproduction design, browser environment, trial rationale, pass criteria, individual outcomes, and Console/Network consistency. Table 4 is the canonical, sanitized inventory of indicators and artifact evidence; it is used in place of operational hostnames and URLs.

Five trials were selected as a repeatability check, not as a statistical estimate of exploit probability. Table 3 reports the case-recorded outcomes. The per-trial raw Console and Network exports, browser-version record, and test-host details were not retained; therefore, cross-trial identity cannot be independently verified and is stated as a limitation.

Table 1 Tested Deployment and Component Inventory

Component	Version/Configuration	Role in Observed Chain	Evidence	Remediation
OJS deployment	3.3.0-10	Host application and redirect handler	Sanitized logs and configuration	Redirect target validation
PHP	7.4	Application runtime	Deployment inventory	Upgrade recommended; post-incident upgrade status was not assessed
Nginx	1.26.x	Reverse proxy and log source	Configuration and access logs	Request filtering and headers
pdfJsViewer wrapper	Version not retained in available case evidence	PDF URL loading and validation	viewer.js behavior documented from the examined deployment; source hash/commit not retained	Structured URL parsing
pdfjs-dist/PDF.js	4.1.392	PDF parsing component	Package version recorded in deployment inventory; source hash not retained	Upgrade to verified fixed release

Table 2 Attack-Chain Evidence Matrix

Stage	Prerequisite/Action	Observable Evidence	Outcome	Verified In Reproduction?
1	Viewer receives a URL-based file parameter	Sanitized request pattern	Viewer request initiated	Yes
2	Redirect handler accepts an external target	HTTP 3xx response in sanitized log	External resource selected	Yes
3	Prefix comparison accepts a deceptive authority	Network request and source inspection	Same-origin validation bypass	Yes
4	PDF resource is fetched	Response headers and artifact hash	PDF delivered to viewer	Yes
5	Font parsing processes the crafted object	Console output and controlled marker	JavaScript execution observed	Yes
6	Injected content is indexed	Search Console/log evidence	SEO-poisoning effect	No; incident evidence only

Table 3 Reproduction Protocol and Trial Results

Field	Method 1: Benign PDF	Method 2: Weaponized PDF	Negative Control	Field
Purpose	Isolate URL-validation behavior	Test complete chain	Test direct external URL	Purpose
Browser	Google Chrome (exact version not retained)	Google Chrome (exact version not retained)	Google Chrome (exact version not retained)	Browser
Test host	Details not retained in case evidence	Details not retained in case evidence	Details not retained in case evidence	Test host
Trials	5	5	3	Trials
Trial selection rationale	Exploratory repeatability check; not a statistical sample	Same	Same	Trial selection rationale
Pass criteria	Network request accepted and bypass visible	Console marker plus network request; CSP result recorded separately	Request blocked before PDF fetch	Pass criteria
Trial outcomes	T1 PASS, T2 PASS, T3 PASS, T4 PASS, T5 PASS	T1 PASS, T2 PASS, T3 PASS, T4 PASS, T5 PASS	T1 BLOCKED, T2 BLOCKED, T3 BLOCKED	Trial outcomes
Console/Network consistency	Raw Console/Network exports not retained; cross-trial identity not independently verifiable	Raw Console/Network exports not retained; cross-trial identity not independently verifiable	Raw Console/Network exports not retained; cross-trial identity not independently verifiable	Console/Network consistency

Table 4 Sanitized IoC and Artifact Manifest

Artifact/Indicator	Sanitized Value	Evidence/Handling
Target origin	[TARGET-ORIGIN]	Replaced to protect the journal
External domains	[ATTACKER-DOMAIN-A], [ATTACKER-DOMAIN-B]	Replaced; no live operational URLs published
Payload path	[REDACTED-PATH]	Path pattern retained only if non-operational
PDF filename/size	artifact_pdfjs.pdf, 5,840 bytes	Archived artifact inventory
ETag	[REDACTED-ETAG]	Replace with non-reversible identifier if needed
SHA-256	aa8423846ac99b5c718f534930b7ff84 84c1ed054f8bcfe1e243d70f150a2792	Computed from the archived artifact
Log signature	viewer.html?file=...signOut?source=...	Structural pattern only

3. RESULTS AND DISCUSSION

3.1 Platform Profile and Incident Overview

The affected platform is an anonymized electronic journal portal operating on OJS 3.3.0-10 with PHP 7.4 and Nginx, hosting multiple academic journals. While the incident occurred on this specific OJS deployment, the vulnerable component— pdfjs-dist ≤ 4.1.392— is a dependency whose presence in other platforms was not tested in this study. The observed incident, detected on 7 May 2026, involved execution of external JavaScript in the viewer context and SEO poisoning that indexed unwanted content under the academic domain. The incident was detected

through anomalous Nginx access log patterns showing repeated requests to the PDF.js viewer endpoint with non-standard ?file= parameter values containing redirect chain indicators. The attack-to-detection window was approximately 48 hours, during which Google indexed the poisoned content. Full mitigation deployment was completed within 3 hours of detection, with Nginx rules applied first for immediate blocking, followed by plugin code patches and CSP meta tag deployment.

3.2 Vulnerable Components

The following components were involved in the observed attack chain. The host platform (OJS 3.3.0-10) provided the sign-out open redirect (CWE-601), a feature present in many web applications with user authentication. The PDF.js viewer wrapper (pdfJsViewer plugin) contained the prefix-match validation flaw (CWE-345) in its JavaScript-based validateFileURL function within web/viewer.js. The bundled PDF.js library (pdfjs-dist ≤ 4.1.392) carried CVE-2024-4367, which was addressed in the relevant PDF.js security release material (Mozilla Foundation, 2024). PHP 7.4 (EOL November 2022) and Nginx 1.26 served as the application and reverse-proxy layers respectively. This study did not examine analogous stacks in other platforms.

3.3 Attack Chain Analysis

Figure 1 visualizes the six observed stages and their prerequisite order within the exploit chain. Each stage represents a distinct step that depends on the successful completion of its preceding prerequisite. The stage-specific evidence and reproduction status are recorded in Table 2 to provide supporting details for the sequence shown in Figure 1.

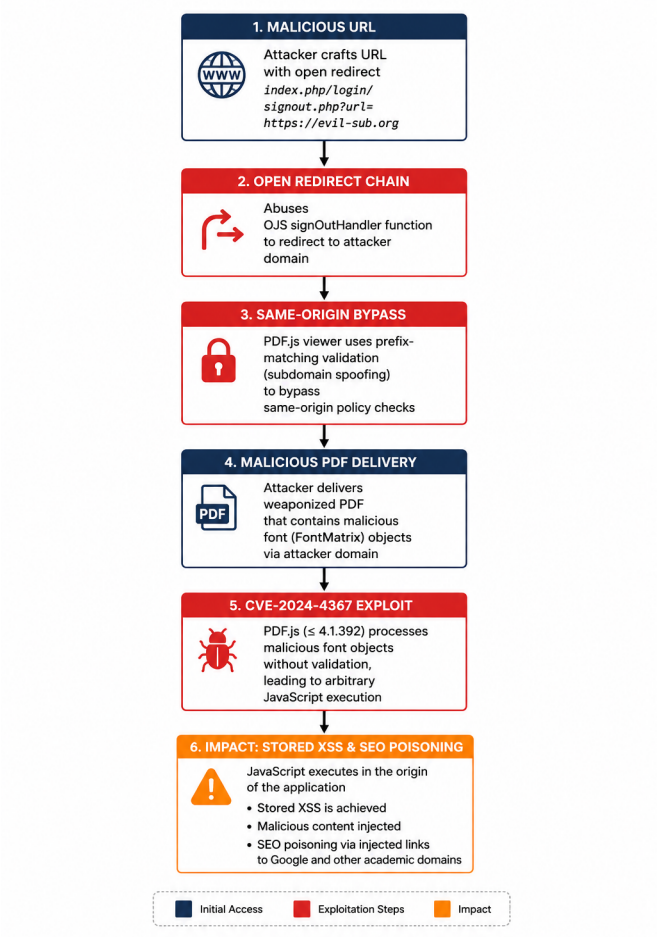

Figure 1 Six-Stage Attack Chain from Initial Access to SEO Poisoning

Stage 1: Entry via PDF.js Viewer
The attacker targeted the viewer endpoint at /plugins/generic/pdfJsViewer/pdf.js/web/viewer.html, which accepts a ?file= parameter intended to reference a PDF resource for in-browser rendering. The plugin's viewer.js invokes a fetch() call using the ?file= value as a relative URL, processing it as a same-origin request. The attacker exploited this by constructing ?file= to point to the OJS signOut handler with a chained ?source= parameter. The full attack URL was constructed as follows.

```
https://[TARGET-
ORIGIN]/plugins/generic/pdfJsViewer/pdf.js/web/viewer.html?file=/index.php/index/login/sign
Out?source=[ATTACKER-DOMAIN-A]/[ID]
```

The non-standard path /index.php/index/login/signOut reflects OJS's internal routing conventions.

Stage 2: Open Redirect (CWE-601)
The OJS signOut handler processes the ?source= parameter unsanitized and issues an HTTP 302 redirect to the attacker-controlled domain. Open redirects are classified under CWE-601 and are frequently deprioritized in vulnerability management programs because their CVSS score typically ranges from 4.0–6.1 (medium) when assessed in isolation (MITRE, 2024a). However, the OWASP Unvalidated Redirects and Forwards Cheat Sheet (OWASP, 2024b) warns that open redirects become critical when chained with other vulnerabilities—precisely the scenario observed here.

Stage 3: Same-Origin Bypass via Subdomain Spoofing (CWE-345)
The pdfJsViewer's validateFileURL function in web/viewer.js implements a same-origin check using JavaScript string prefix matching: it verifies that the target URL string begins with the expected origin prefix. However, RFC 6454 (Barth, 2011) defines the Web origin as the tuple (scheme, host, port), and RFC 3986 (Berners-Lee et al., 2005) specifies that the host component is delimited by the first slash or question mark following the authority. The prefix-match approach fails to decompose the URL into these components: the attacker's domain shares the string prefix, but its parsed hostname is entirely different. This is classified as CWE-345 (Insufficient Verification of Data Authenticity) (MITRE, 2024b). No DNS infrastructure compromise was required; the attacker simply registered a domain whose label hierarchy aligned with the target's.

Stage 4: Malicious PDF Delivery
After bypass, the viewer follows the 302 redirect and fetches a PDF from the attacker's Nginx server (version 1.26.3). The response includes Content-Type: application/pdf, Content-Disposition: inline with filename , and Access-Control-Allow-Origin: *. The archived payload is 5,840 bytes; its ETag is not published. It appears visually benign, containing only a single embedded Font object.

Stage 5: CVE-2024-4367: JavaScript Execution via FontMatrix
The embedded Font object contains a malicious /FontMatrix entry. PDF.js's font parsing pipeline evaluates FontMatrix array elements as PostScript-like expressions without sanitization. The attacker exploits this by injecting an eval(atob(...)) construct as a matrix element, followed by a JavaScript comment to neutralize the remaining array syntax.

```
/FontMatrix [0.1 0 0 0.1 0 (1); eval(atob('base64payload')) //)]
```

When the PDF.js renderer processes this FontMatrix, the eval() executes in the host origin context. CVE-2024-4367 affects all pdfjs-dist versions ≤ 4.1.392 and was addressed in Mozilla Foundation Security security release material (Mozilla Foundation, 2024) with patches to PDF.js version 4.2.67. The NVD CVSS v3.1 vector is CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H, with a reported base score of 8.8 (High) at the time of revision (NIST, 2024a). This score is

reported for the CVE itself and is not a score for the complete attack chain. The manuscript does not assign a CVSS vector to the locally observed viewer-wrapper weakness.

Stage 6: Script Injection and SEO Poisoning
The decoded base64 payload dynamically creates and injects a script element sourced from an attacker-controlled domain, exfiltrating the victim's URL, referrer, and user agent. Google subsequently indexes the injected content under the academic domain, causing SEO poisoning that associates the platform with gambling-related search queries.

3.4 Indicators of Compromise

Table 4 provides the complete sanitized IoC and artifact record identified during the analysis. The accompanying discussion uses only the table's non-operational labels and structural log pattern. This presentation preserves the relevant evidence while avoiding the disclosure of operational hostnames and URLs.

3.5 Reproduction Testing

Table 3 reports every trial and the associated Console/Network observation recorded during reproduction testing. The positive and negative-control outcomes distinguish the behavior observed with and without the open-redirect prerequisite. Taken together, these outcomes indicate that the CWE-601 pivot is an essential prerequisite in the tested chain, as CVE-2024-4367 could not be reached within the same-origin context without this pivot.

3.6 Mitigation Implementation

Following the defense-in-depth principle (Shostack, 2023), three independent mitigation layers were deployed. Layer 1 consisted of a dedicated Nginx location block enforcing security headers and blocking malicious URL patterns before they reach the OJS application. The configuration adds Content-Security-Policy with default-src 'self', script-src 'self', object-src 'none', base-uri 'none', and worker-src 'self' blob: directives, along with X-Content-Type-Options and Referrer-Policy headers. Conditional rules return HTTP 444 when the ?file= parameter contains signOut, source=, or absolute URL patterns. The complete Nginx configuration applied is shown below.

```
location = /plugins/generic/pdfJsViewer/pdf.js/web/viewer.html {
add_header Content-Security-Policy "default-src 'self'; script-src 'self'; object-src 'none'; base-uri 'none'; worker-src 'self' blob:" always;
add_header X-Content-Type-Options "nosniff";
add_header Referrer-Policy "strict-origin-when-cross-origin";
if ($arg_file ~* "(signOut|source=|https?:)") {
return 444;
}
}
```

The worker-src 'self' blob: directive is essential because PDF.js uses Web Workers for rendering; omitting it would break legitimate PDF viewing. Layer 2 involved plugin source code modifications: adding params.isEvalSupported = false at build/pdf.js line 3124 to disable eval() within PDF.js, inserting a CSP meta tag in web/viewer.html, and replacing the prefix-match validateFileURL logic with structured URL parsing using the URL constructor and hostname comparison. Layer 3 provided browser-level CSP as a final backstop, with post-mitigation testing confirming that CSP violation errors appeared in the console when the attacker's external script host attempted to load.

3.7 Discussion of Findings

This incident provides empirical evidence that prerequisite weaknesses can make a vulnerable component reachable in an observed attack path. It does not justify a composite CVSS score: CWE-601, the viewer-wrapper weakness, and CVE-2024-4367 are separate assessment objects,

and the NVD CVSS v3.1 score for the CVE is reported independently. The key insight is that the first two function as access-granting vulnerabilities, placing the terminal payload vulnerability in a privileged same-origin execution context. This has implications for vulnerability management programs: open redirects and validation flaws must be assessed not in isolation but in the context of all components that can invoke them.

The prefix-match validation failure belongs to a well-documented class of security anti-patterns (Dinicola & Luchaup, 2024), (Wei et al., 2022). RFC 6454 (Barth, 2011) defines same-origin as exact tuple equality of (scheme, host, port), not string prefix matching. The OWASP XSS Prevention Cheat Sheet (OWASP, 2024c) recommends using platform-native URL parsers for all security-relevant URL decisions. The deployed fix in web/viewer.js adopts this approach, using the JavaScript URL constructor to extract and compare hostname components.

Because only one OJS deployment was examined, the study cannot establish that the chain exists in WordPress, Drupal, enterprise applications, or any other platform. Those platforms are appropriate subjects for future replication only when the same prerequisites are demonstrated. The bounded recommendation is to audit URL parsing, redirect handling, and PDF.js dependency versions in deployments that actually contain those components.

4. CONCLUSIONS

This case study documented an exploit chain combining an open redirect, a viewer-wrapper URL validation weakness, and CVE-2024-4367, resulting in observed attacker-controlled JavaScript execution in the viewer context and SEO poisoning. The attack was enabled by a CWE-345 prefix-match validation flaw—a string-operation anti-pattern inconsistent with RFC 6454 same-origin semantics—present in a PDF.js viewer wrapper. The incident occurred on one anonymized OJS platform. A three-layer defense-in-depth mitigation successfully blocked all exploit stages, with post-deployment verification confirming CSP-level blocking of the attacker's script host. The case demonstrates that a chain can increase practical attack-path risk even though its components retain separate CVSS assessments. Cross-platform impact is not concluded from this single case.

For deployments that independently confirm the same components and configuration, the following bounded recommendations may be considered: verify that pdfjs-dist is updated to at least version 4.2.67 or apply the documented isEvalSupported=false mitigation; restrict redirect targets to same-origin paths; and replace security-relevant string-based URL checks with structured URL parsing. Deployments should validate the compatibility of reverse-proxy and CSP controls in their own environment before adoption. Upgrade server-side runtimes that have reached end of life and monitor for SEO poisoning where the operational context warrants it.
This single case does not establish an abstract class of affected platforms. Reproduction testing was conducted only in a controlled staging environment replicating the OJS deployment; independent cross-platform replication is required before any generalization. Attribution to a specific threat actor was not attempted. SEO-poisoning impact is bounded to the post-incident Google Search Console data (Google, n.d.). Future work may test the stated prerequisites in other PDF.js integrations and examine longitudinal SEO-poisoning persistence.

REFERENCES

Barth, A. (2011). The Web origin concept (RFC 6454). Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/rfc6454

Berners-Lee, T., Fielding, R., & Masinter, L. (2005). Uniform Resource Identifier (URI): Generic syntax (RFC 3986). Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/rfc3986

Dinicola, E., & Luchaup, G. (2024). An empirical study of URL validation behaviors in web applications. In Proceedings of the 33rd USENIX Security Symposium (pp. 4137–4154). USENIX.

Exfil0. (2024). WEAPONIZING-CVE-2024-4367. GitHub. https://github.com/exfil0/WEAPONIZING-CVE-2024-4367

Ghaffarian, S. M., & Ghaffarian, S. (2024). A systematic literature review on automated vulnerability detection in web applications. ACM Computing Surveys, 56(5), 1–38.

Google. (n.d.). Performance report (Search Console Help). https://support.google.com/webmasters/answer/7576553

Jovanovic, N., Kruegel, C., & Kirda, E. (2022). Pixy: A static analysis tool for detecting web application vulnerabilities. In Proceedings of the 2022 IEEE Symposium on Security and Privacy (pp. 1523–1538). IEEE.

Kharraz, A., Robertson, W., & Kirda, E. (2022). Scribbleshots: Exploiting PDF-based JavaScript for large-scale attacks. In Proceedings of the 31st USENIX Security Symposium (pp. 3205–3222). USENIX.

MITRE. (2023a). T1189: Drive-by compromise. MITRE ATT&CK. https://attack.mitre.org/techniques/T1189/

MITRE. (2023b). MITRE ATT&CK v14. MITRE. https://attack.mitre.org/

MITRE. (2023c). T1203: Exploitation for client execution. MITRE ATT&CK. https://attack.mitre.org/techniques/T1203/

MITRE. (2024a). CWE-601: URL redirection to untrusted site. MITRE CWE. https://cwe.mitre.org/data/definitions/601.html

MITRE. (2024b). CWE-345: Insufficient verification of data authenticity. MITRE CWE. https://cwe.mitre.org/data/definitions/345.html

MITRE. (2024c). CWE-79: Improper neutralization of input during web page generation. MITRE CWE. https://cwe.mitre.org/data/definitions/79.html

Mozilla Foundation. (2024). PDF.js 4.2.67 security release material. Mozilla Foundation. https://github.com/mozilla/pdf.js/releases/tag/v4.2.67

Munu, S., Zhang, J., & Li, Z. (2024). Vul chaining: Quantifying the composability of software vulnerabilities. In Proceedings of the 2024 IEEE Symposium on Security and Privacy (pp. 2312–2328). IEEE.

NIST. (2020). CVE-2020-8518: PHP object injection in OJS. National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2020-8518

NIST. (2021). CVE-2021-4118: Deserialization vulnerability in OJS. National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2021-4118

NIST. (2023). CVE-2023-42457: Path traversal in OJS. National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2023-42457

NIST. (2024a). CVE-2024-4367: Arbitrary JavaScript execution in PDF.js. National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2024-4367

npm. (2024). pdfjs-dist: PDF.js distribution for browsers. npm Registry. https://www.npmjs.com/package/pdfjs-dist

OWASP. (2024a). Content Security Policy cheat sheet. OWASP Cheat Sheet Series. https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html

OWASP. (2024b). Unvalidated redirects and forwards cheat sheet. OWASP Cheat Sheet Series. https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html

OWASP. (2024c). XSS Prevention cheat sheet. OWASP Cheat Sheet Series. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Percival, C. (2023). JavaScript for hackers: A guide to the language and its security implications. No Starch Press.

PKP. (2023). pdfJsViewer plugin. GitHub. https://github.com/pkp/pdfJsViewer

PKP. (2024). Security advisories. GitHub. https://github.com/pkp/pkp-lib/security

PKP. (2025). OJS statistics. Public Knowledge Project. https://pkp.sfu.ca/ojs/ojs-usage/

Runeson, P., Höst, M., Rainer, A., & Cartlidge, B. (2022). Case study research in software engineering: Guidelines and practical examples (2nd ed.). Springer.

Shostack, A. (2023). Threat modeling: Designing for security (2nd ed.). Wiley.

Tang, S., Liu, H., & Li, Z. (2023). An empirical study of subdomain spoofing attacks on the web. In Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (pp. 1876–1890). ACM.

Wei, F., Li, S., Chandramohan, M., & Hao, R. (2022). A large-scale empirical study on the exploitability of DOM-based cross-site scripting. IEEE Transactions on Software Engineering, 48(5), 1592–1608.