

Sri Lanka Institute of Information Technology



Cyber Security Assignment (2025)
Server Side Request Forgery (SSRF)

Bug Bounty Report- 06
IT23363366

TABLE OF CONTENT

1. Scope & Objective.....	3
2. Scope & Objective.....	3
3. Enumeration and Reconnaissance.....	3
3.1 Tools Used.....	3
3.2 Steps Taken.....	3
4. Vulnerability Description.....	5
5. Affected Component.....	5
6. Impact Assessment.....	6
7. Proof of Concept (PoC)	6
7.1 Manual Testing.....	6
7.2 Automated Testing	7
8. Proposed Mitigation.....	9
9. Conclusion	9
10. References.....	9

1. Scope & Objective

Report Title: Server Side Request Forgery (SSRF)

Reported By: Raahim Mahmooth

Platform: <https://hackerone.com/>

Tested On: <https://www.mtn.com/>

2. Scope & Objective

The objective of this report is to investigate potential Server Side Request Forgery (SSRF) vulnerabilities on the target web application, **mtn.com**. SSRF vulnerabilities can allow an attacker to send malicious requests from the server to internal resources or other services, potentially leading to data leakage, unauthorized access, or other security risks.

3. Enumeration and Reconnaissance

3.1 Tools Used

- Manual check using Burp Suite
- OWASP ZAP
- [Webhook.site](https://webhook.site) for HTTP request interception
- [WaybackURL](https://waybackurl.com) for accessing historical URLs
- FFuf for fuzzing potential SSRF endpoints

3.2 Steps Taken

1. **Initial ZAP Scan:** Performed a basic scan with ZAP to detect **SSRF** vulnerabilities in the application.



Nothing found related to SSRF

- **Endpoint Discovered:** `/our-positions-certifications/?tablink=certifications-`
This endpoint included a parameter that might be prone to SSRF.

- results much overwhelming unable to **fetch SSRF prone parameters**

```
File Actions Edit View Help
(raahimhammooh@kali) ~/waybackurls
$ echo "mtn.com" | ./waybackurls | grep '.*'
https://developer.mtn.com/community/portal/site.action?s_devsite6c=userBoarding
https://hsdpportal.mtn.com/community/portal/site.action?s_devsite6c=Guide
https://hsdpportal.mtn.com/community/portal/site.action?s_devsite6c=Home
https://hsdpportal.mtn.com/community/portal/site.action?s_devsite6c=News
http://www.mtn.com/KEFRAK8financial-results/?report_cat=quarterly-results
https://www.mtn.com/bsa-MTN4Group
https://www.mtn.com/bsesheet-525329866newsitemid-2021111600649669lan-en-US&anchor=www.mtn.com#index-28md5-ea11586bb4ee8a9757071991921e0f45
https://www.mtn.com/bsesheet-541509096newsitemid-202411127496669lan-en-US&anchor=MTN+Group#index-28md5-723b29e0220a8213bdca368b883161c
https://www.mtn.com/Guserid-39662376signature-3588c622d3307e2a
http://www.mtn.com:80/?PHPSESSID=f8246dbe48e681ecd04962502a9b07c0
https://www.mtn.com/?custom-css=145b729677
https://www.mtn.com/?custom-css=4d38a82386
https://www.mtn.com/?custom-css=812a9591a1
https://www.mtn.com/?custom-css=a2f6b4fe32
https://www.mtn.com/?custom-css=e5a4508024
https://www.mtn.com/?p=37529
https://www.mtn.com/?p=3791
https://www.mtn.com/?p=3848
https://www.mtn.com/?p=3902
https://www.mtn.com/?p=3920
https://www.mtn.com/?p=3939
https://www.mtn.com/?p=3962
https://www.mtn.com/?p=39675
https://www.mtn.com/?p=39735
https://www.mtn.com/?p=39896
https://www.mtn.com/?p=40576
https://www.mtn.com/?p=4138
https://www.mtn.com/?p=42518
https://www.mtn.com/?p=42708
https://www.mtn.com/?p=4303
https://www.mtn.com/?p=4824
https://www.mtn.com/?p=4877
https://www.mtn.com/?p=4908
https://www.mtn.com/?p=4953
https://www.mtn.com/?p=4961
https://www.mtn.com/?p=4976
https://www.mtn.com/?p=4998
https://www.mtn.com/?p=5398
```

4. focusing on parameters such as `url=`, `path=`, and `link=`, `file=` which can be targets for SSRF.
 - o Found a URL endpoint: `/wp-json/oembed/1.0/embed?url=https://www.mtn.com/2africa-now-the-longest-subsea-cable-in-the-world/&format=xml`, which is a WordPress oEmbed endpoint, potentially vulnerable to SSRF if the server fetches and embeds external URLs.

```

--(raahimamhooth@kali)~/waybackurls
$ echo "mtn.com" | ./waybackurls | grep 'file='
--(raahimamhooth@kali)~/waybackurls
$ echo "mtn.com" | ./waybackurls | grep 'path='
http://www.mtn.com:80/GenericErrorPage.htm?aspxerrorpath=/Default.aspx
https://www.mtn.com:80/GenericErrorPage.htm?aspxerrorpath=/login.aspx
http://www.mtn.com:80/GenericErrorPage.htm?aspxerrorpath=/PressOffice/Pages/ImageDownload.aspx
--(raahimamhooth@kali)~/waybackurls
$ echo "mtn.com" | ./waybackurls | grep 'link='
https://www.mtn.com/investors-shareholders/?tablink=sens
https://www.mtn.com/investors-shareholders/?tablink=ADR
https://www.mtn.com/investors-shareholders/?tablink=AGM
https://www.mtn.com/investors-shareholders/?tablink=b-bbee-certificate
https://www.mtn.com/investors-shareholders/?tablink=circulars-and-gms
https://www.mtn.com/investors-shareholders/?tablink=debt-and-
https://www.mtn.com/investors-shareholders/?tablink=debt-and-funding-updates
https://www.mtn.com/investors-shareholders/?tablink=presentations-and-transcripts
https://www.mtn.com/investors-shareholders/?tablink=sens
https://www.mtn.com/leadership/?tablink=board
https://www.mtn.com/leadership/?tablink=executive
https://www.mtn.com/legal/?tablink=terms_&_conditions
https://www.mtn.com/legal/?tablink=cookie_policy
https://www.mtn.com/legal/?tablink=mtn_privacy_notice
https://www.mtn.com/legal/?tablink=paia
https://www.mtn.com/legal/?tablink=privacy_policy
https://www.mtn.com/our-positions-certifications/?tablink=certifications
https://www.mtn.com/our-positions-certifications/?tablink=eco-responsibility
https://www.mtn.com/our-positions-certifications/?tablink=economic-value
https://www.mtn.com/our-positions-certifications/?tablink=sound-governance
https://www.mtn.com/our-positions-certifications/?tablink=sustainable
https://www.mtn.com/our-positions-certifications/?tablink=sustainable%00X000
https://www.mtn.com/reports/?tablink=additional-reports
https://www.mtn.com/reports/?tablink=cdp-reports
https://www.mtn.com/reports/?tablink=sustainable-report

```

4. Vulnerability Description

SSRF occurs when a web application is tricked into sending requests to unintended resources, potentially internal systems or third-party services, on behalf of the attacker. The vulnerability was explored in two main parts:

1. **GET /our-positions-certifications/?tablink=certifications:** This endpoint takes a URL parameter and might fetch remote resources, opening the door for SSRF.
2. **WordPress oEmbed Endpoint:** `/wp-json/oembed/1.0/embed?url=https://www.mtn.com/2africa-now-the-longest-subsea-cable-in-the-world/&format=xml` - This endpoint fetches external URLs to embed content, making it a likely candidate for SSRF attacks if the input isn't properly validated.

5. Affected Component

1. **Endpoint :** [GET/our-positions-certifications/?tablink=certifications](https://www.mtn.com/our-positions-certifications/?tablink=certifications)
2. **Endpoint :** [/wp-json/oembed/1.0/embed?url=https://www.mtn.com/2africa-now-the-longest-subsea-cable-in-the-world/&format=xml](https://www.mtn.com/wp-json/oembed/1.0/embed?url=https://www.mtn.com/2africa-now-the-longest-subsea-cable-in-the-world/&format=xml)

6. Impact Assessment

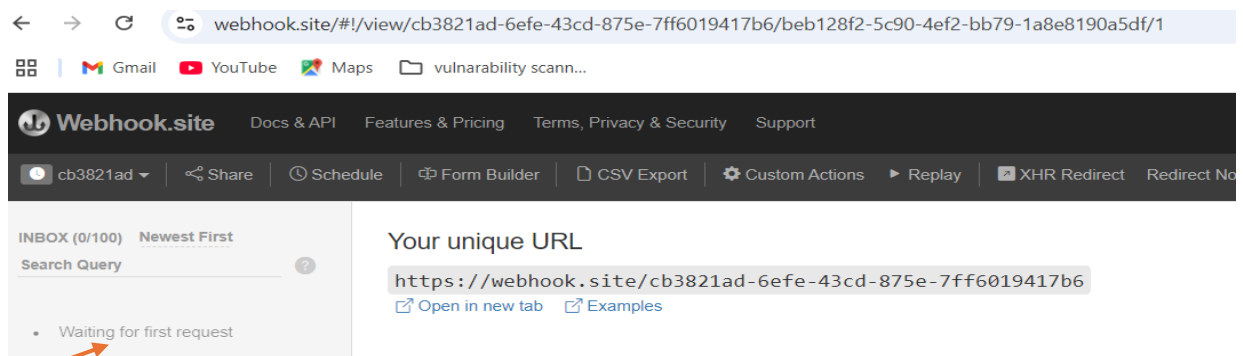
1. **SSRF Vulnerability:** If successful, an SSRF attack could allow an attacker to access internal services or resources that are not meant to be publicly accessible. This can lead to data leaks, unauthorized access to internal systems, or even the ability to interact with sensitive internal resources.
2. **WordPress oEmbed Endpoint:** This endpoint, when misused, could potentially allow attackers to fetch arbitrary resources from the internal network, which could then be exploited to gain unauthorized access or cause disruptions in service.

7. Proof of Concept (PoC)

7.1 Manual Testing

- **Testing `/our-positions-certifications/?tablink=certifications`:**
Replaced the `tablink` parameter with a webhook URL obtained from [Webhook.site](https://webhook.site). However, although the response was 200 OK, the request was sanitized by the backend, preventing SSRF.

webhook URL obtained from [Webhook.site](https://webhook.site)(act as my server)



Replaced the `tablink` parameter with a webhook URL

Target: https://www.mtn.com

Request

```
1 GET /our-positions-certifications/?tablink=https://webhook.site/cb3821ad-6efe-43cd-875e-7ff6019417b6 HTTP/2
2 Host: www.mtn.com
3 Cookie: PHPSESSID=37f12e18e9eaf629c3523e40b01b3fda; _ga_QS6MTT282Y=
  GSI.1.1745953307.1.1.1745953308.7.0.0; _ga=GAI.2.1990798675.1745953308; _gid=
  GAI.2.83107723.1745953327; cookiebar=hide
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:137.0) Gecko/20100101 Firefox/137.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://www.mtn.com/?s=heys
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Priority: u=0, i
15 Te: trailers
```

Response

```
1 HTTP/2 200 OK
2 Server: nginx
3 Date: Wed, 30 Apr 2025 01:07:04 GMT
4 Content-Type: text/html; charset=UTF-8
5 Vary: Accept-Encoding
6 Pragma: no-cache
7 X-Hacker: If you're reading this, you should visit wpvip.com/careers and apply to join the
  mention this header.
8 X-Powered-By: WordPress VIP <https://wpvip.com>
9 Host-Header: a9130478a60e5f9135f765b3f26593b
10 X-Frame-Options: SAMEORIGIN
11 X-Content-Type-Options: nosniff
12 Referrer-Policy: no-referrer-when-downgrade
13 X-Permitted-Cross-Domain-Policies: all
14 Feature-Policy: camera 'none'; microphone 'none'; geolocation 'none'
15 Permissions-Policy: camera=(), microphone=(), geolocation=()
16 Link: <https://www.mtn.com/wp-json/>; rel="https://api.w.org/"
17 Link: <https://www.mtn.com/wp-json/wp/v2/pages/49583>; rel="alternate"; title="JSON";
  type="application/json"
```

MTN About Us Sustainability Investors News People & Culture

SUSTAINABILITY

Our positions & certifications

Doing for planet Doing for people Doing it right Doing for growth Certifications

As a result no request was present in the site which tells the request was sanitized by the backend, preventing SSRF. Although the response was 200 OK

7.2 Automated Testing

- **WaybackURL & FFuf Usage:**
 - Tested <https://www.mtn.com/wp-json/oembed/1.0/embed?url=<>> :
 - Used WaybackURL to retrieve **historical URLs** and found a **potentially vulnerable endpoint**.

8. Proposed Mitigation

- **Implement Input Validation:** Ensure all parameters that accept URLs or external resources are strictly validated before making requests. Implement proper whitelisting or URL sanitization.
- **Use Firewall and Network Segmentation:** Block outgoing requests to internal resources or sensitive domains to prevent SSRF attacks.
- **Adopt Security Mechanisms:** Employ additional protections like allowing only specific external services to be fetched or using network-level controls to limit access.

9. Conclusion

After testing multiple endpoints over some time, **no SSRF vulnerabilities were found in the application**. However, proper URL sanitization should still be implemented in case new endpoints are introduced in the future. Furthermore, additional testing with Burp Suite Professional may yield more thorough results. But testing all might possibly give valuable hook.

10. References

- OWASP SSRF : https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery
- CVE-2020-15778: **SSRF vulnerability in WordPress oEmbed**