www.getastra.com

Security Assessment Report

Automated Full Scan + Manual Pentest

Demo Astra

Demo Web App

Report dated **June 28, 2024**

Assessment Performed By

Saurabh Kumar

ANALYST, REVIEWER

Abhishek Kukreti

ANALYST

About Astra

Astra is a leading Penetration Testing as a Service (PTaaS) platform which combines continuous automated scanning with on-demand manual pentests by security experts. Astra follows the highest standards for security testing, vulnerability scanning and is an active contributor to industry leading Open source security standards and tools (OWASP WSTG, OWASP ZAP).

The assessment was performed within the predefined scope of this engagement, and its findings and recommendations have been shared with the customer. A penetration test is considered a snapshot in time. The findings and recommendations solely reflect the information gathered during the assessment period and do not account for any subsequent changes or modifications.

Astra IT Inc.
help@getastra.com

2093 Philadelphia Pike 4080,
Claymont, Delaware, 19703,
United States

Table of Contents

Overview

Executive Summary

Astra was engaged by Demo Astra to perform a security assessment of **1** target during the period **27th June 2024** to **28th June 2024**. Manual pentest was performed on **1** target .

The testing was performed from a remote attacker's perspective with the following goals:

- To identify security loopholes, business logic errors and evaluate effectiveness of existing security controls in the application that pose a risk to the systems, infrastructure, or data.
- Recommend technical security best practices to improve security posture of the target applications audited.
- Explain the potential impact of the identified vulnerabilities, such as the extent of data exposure, potential financial losses, or reputational damage that could occur if they were exploited by malicious actors.
- Provide clear and actionable recommendations for addressing the identified vulnerabilities.

A total of **15 vulnerabilities/recommendations** were reported. Out of a score of 10, the highest risk score assigned to a vulnerability was **9.2**, the lowest was **2.5**, and the average score was **5.9**.

Astra verified fixes for **15** vulnerabilities, and confirmed they were fixed at the time of the rescan.

Scope of the Assessment

The assessment was performed within the predefined scope of this engagement as listed below. No assumptions about the application were made.

Type	Name	Scope	Start Grade	Closure Grade
Web App	Demo Web App	https://demowebapp.com	D	A+

Resolution Statistics

Severity	Solved	Unsolved	Help Wanted	Under Review	Accepted Risk	Grand Total
Critical	3	0	0	0	0	3
High	4	0	0	0	0	4
Medium	6	0	0	0	0	6
Low	2	0	0	0	0	2
Info	0	0	0	0	0	0
Grand Total	15	0	0	0	0	15

Overall vulnerability statistics

Pentest Details

Assessment Methodology

An **in-depth manual penetration testing** was conducted by Astra's experienced security professionals, along with an automated full vulnerability scan consisting of **9300+ tests**.

The assessment follows **industry standards** such as OWASP Web Security Testing Guide (WSTG), OWASP Top 10, OWASP Application Security Verification Standard (ASVS), NIST 800-115 etc.

Using the **same techniques as sophisticated real-world attackers**, the applications have been tested thoroughly for business logic, chained & application functionality specific vulnerabilities. This hands-on approach with manual checks goes beyond traditional scanning tools, allowing to uncover critical issues that may not be detected by automated scanners.

All false positives have been removed from the report, and once the vulnerabilities have been resolved the fixes are verified on a case to case basis.

Assessment Duration and Dates

Scan Mode	Target Name	Authentication	Started	Completed
Manual Pentest (VAPT)	Demo Web App	0 users	27th Jun 2024	28th Jun 2024

Certificates

No certificates have been issued for the scope covered as per this report. Certificates are issued when 90% or more vulnerabilities have been fixed after a manual pentest. The remaining vulnerabilities have to be of Info or Low severity. They can also be of Medium severity, if they're not immediately fixable.

Vulnerabilities

Overview Table

No.	Target	Title	Severity	Risk Score	Status	Link
1	Demo Web App	[CRITICAL] Web Shell Can Be Uploaded To Gain Complete Access To Server	Critical	9.2	Solved	Open
2	Demo Web App	[CRITICAL] Broken Access Control Leads to Unauthorized Privilege Escalation	Critical	8.9	Solved	Open
3	Demo Web App	[CRITICAL] Admin Account Takeover On Update Contact Details	Critical	8.7	Solved	Open
4	Demo Web App	Unprotected Magmi - Can Be Used As A Backdoor, Full Complete Database Access	High	6.9	Solved	Open
5	Demo Web App	Stripe API Key Disclosed	High	6.8	Solved	Open
6	Demo Web App	Possible To Bypass Work Email Only Restriction And Gain Access To Other Domains	High	6.8	Solved	Open
7	Demo Web App	Possible For Lower Privileged Users To See Details Of Admin Users	High	6.6	Solved	Open
8	Demo Web App	Outdated and Vulnerable Components In Use	Medium	5	Solved	Open
9	Demo Web App	Reverse Tabnabbing	Medium	4.8	Solved	Open
10	Demo Web App	Insecure HTTP Cookies	Medium	4.8	Solved	Open
11	Demo Web App	Missing API Security Headers	Medium	4.8	Solved	Open
12	Demo Web App	Possible To Prevent Normal Users From Booking Tickets By Performing Large Number Of False Pre-Bookings	Medium	4.7	Solved	Open
13	Demo Web App	Cross Domain Referrer Leakage	Medium	4.3	Solved	Open
14	Demo Web App	Secure SSH Access	Low	2.8	Solved	Open
15	Demo Web App	No CAPTCHA Implemented	Low	2.5	Solved	Open

Details of Vulnerabilities Found

1. [CRITICAL] Web Shell Can Be Uploaded To Gain Complete Access To Server	
Severity	Critical
Status	Solved
Risk Score	9.2/10
CWE	434: Unrestricted Upload of File with Dangerous Type
CVSS	9 (CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H)
Labels	HIPAA, PCI DSS, ISO 27001, GDPR, SOC 2, SOC 2 - Security, OWASP 2021, OWASP 2021 - A01 - Broken Access Control

Description

During the pentest, we observed that it is possible for attackers to create a new account on the website, and **upload a web-shell** through the profile photo **upload feature**.

Using the upload areas available on the website, it is possible to upload Web Shells i.e executable PHP/ASPX/Python files. With this vulnerability, a hacker can modify/delete/steal files & access the database on the website. This means that user passwords, payment gateway keys, etc are compromised

A web shell is a script that can be uploaded to a web server to enable remote administration of the machine. Infected web servers can be either Internet-facing or internal to the network, where the web shell is used to pivot further to internal hosts

File type checks & MIME checks should be performed BOTH on the client-side (JavaScript) & on the server-side (PHP, ASPX). Note that file type checks implemented in JavaScript can easily be bypassed.

Impact

Web shells are frequently used in compromises due to the combination of remote access and functionality. Even simple web shells can have a considerable impact and often maintain a minimal presence. Web shells are utilized for the following purposes:

- To harvest and exfiltrate sensitive data and credentials
- To upload additional malware for the potential of creating, for example, a watering hole for infection and scanning of further victims
- To use as a relay point to issue commands to hosts inside the network without direct Internet access
- To use as command-and-control infrastructure, potentially in the form of a bot in a botnet or in support of compromises to additional external networks. This could occur if the adversary intends to maintain long-term persistence

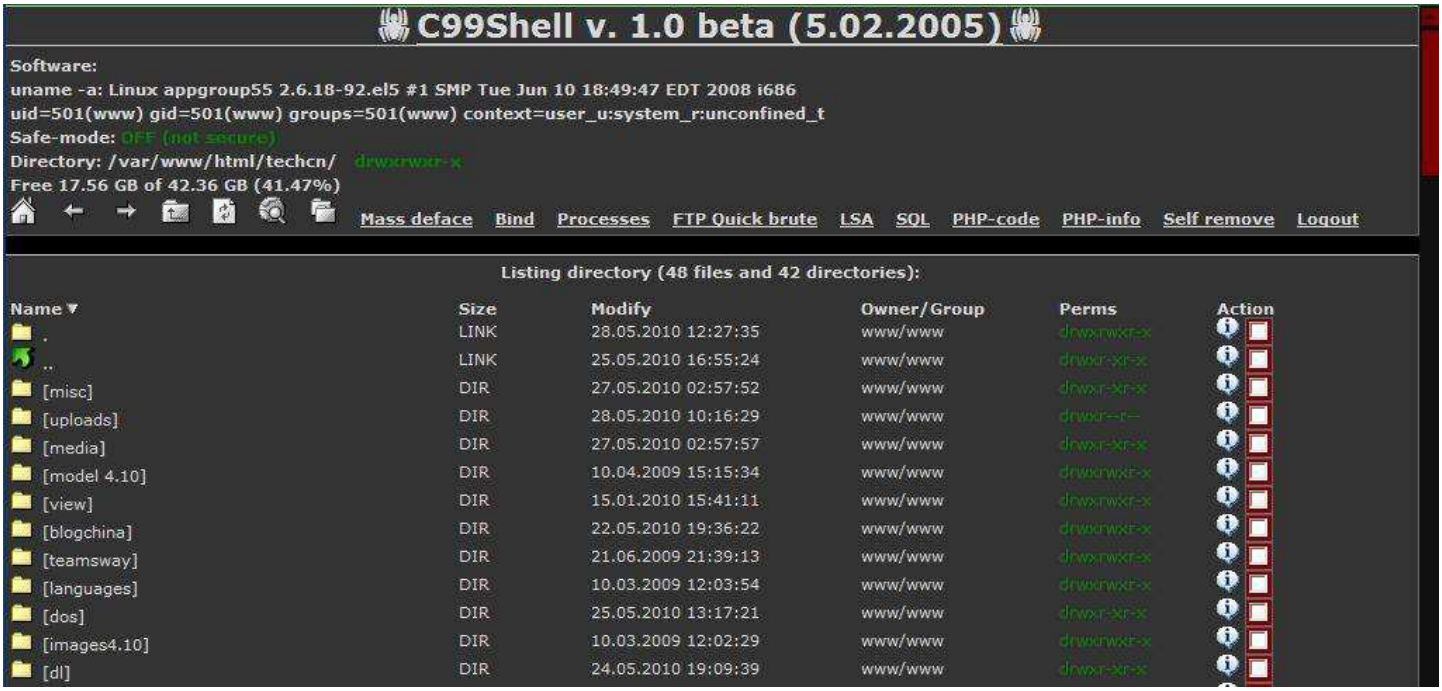
Affected Components

https://phps.example.com/2/welcome/3

POST https://phps.example.com/2/settings/account?id=47789

https://phps.example.com/2/settings/account-branding-email?id=47789

Steps to Reproduce



1. Visit https://example.com/account/create and create a free account
2. Navigate to the **Profile** page where the profile photo can be uplaoded
3. Click on **Browse for Photo** button to open the. file upload modal
4. In the window that opens, select and upload any PHP file/web-shell
5. Even though an error is displayed, it was found that the PHP file is indeed uploaded to `/public_html/2/uploads/profile_photos` folder
6. Now open the PHP file through the web browser by Direct Object Reference such as [https://example.com/uploads/profile_photos/johnDoe.jpg]()

Suggested Fix

- Once the issue has been verified, we would recommend performing an exhaustive search of .php, .aspx and other executable files in the uploads folder
- Executable file types should be rejected server-side (.php .php3 .php4 .phtml .pl .py .jsp .asp .htm .shtml .sh .cgi
- Only allow whitelisted extensions (like PDF, JPG, PNG etc.)/MIME types to be uploaded
- File type checking should be performed server-side
- Files should be checked if they contain 'double extensions
- MIME type should be checked to allow the only image
- Disable code execution in the uploads folder by adding the following code to the .htaccess file

```
<Files ~ ".(php|php3|php4|phtml|pl|py|jsp|asp|htm|shtml|sh|cgi)$">Order allow,deny Deny from all</Files>
```

- Additional steps for prevention can be found here: https://www.owasp.org/index.php/Unrestricted_File_Upload

2. [CRITICAL] Broken Access Control Leads to Unauthorized Privilege Escalation	
Severity	Critical
Status	Solved
Risk Score	8.9/10
CWE	280: Improper Handling of Insufficient Permissions or Privileges
CVSS	8.8 (CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H)
Labels	HIPAA, GDPR, ISO 27001, SOC 2, SOC 2 - Privacy, SOC 2 - Integrity, SOC 2 - Security, OWASP 2021, OWASP 2021 - A01 - Broken Access Control

Description

During our pentest, we observed that a low-privilege user can change their own privilege level to admin level. This vulnerability is a result of broken access control in the application, which fails to properly enforce authorization checks when users modify their own roles.

Impact

1. **Unauthorized Privilege Escalation:** Low privilege users can elevate their privileges to admin, gaining unrestricted access to the application.
2. **Full System Compromise:** With admin privileges, attackers can access, modify, or delete all data, change configurations, and perform any administrative actions.
3. **Data Breach:** Sensitive data can be accessed, modified, or deleted, leading to significant data breaches.
4. **Compliance Violations:** This breach can result in non-compliance with regulatory requirements, leading to potential legal consequences and fines.
5. **Reputation Damage:** Loss of trust from users and stakeholders due to compromised security and mishandling of user privileges.

Affected Components

https://getastra.com

Steps to Reproduce

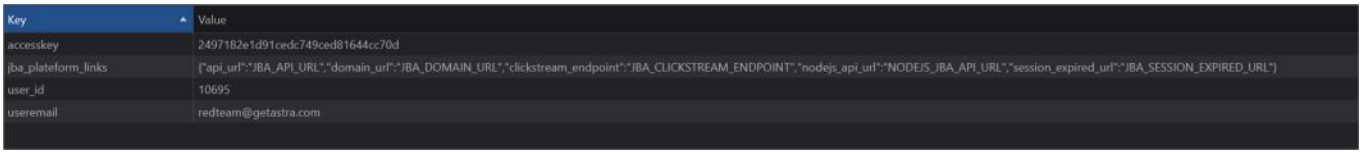
1. Log in as a low-privilege user.

2. Capture the `POST /wp-admin/admin-ajax.php` request in Burp Suite Repeater.

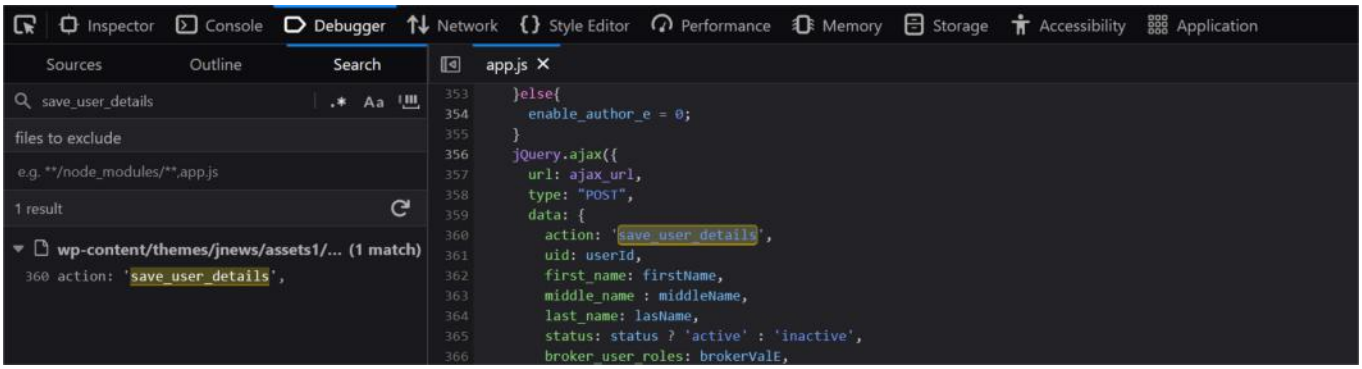
```
POST /wp-admin/admin-ajax.php HTTP/2
Host: getastra.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87
Safari/537.36 root@y9831t1wqsusceamm62s416xbohmfk88x.oastify.com
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://6t0bl1l4a0e0wmuu6em0o9q5vw1uzsygn.oastify.com/ref
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 54
Origin: https://getastra.com
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=1
Te: trailers
```

3. Delete the requested data.

4. Collect the **UID** from Local Storage.



5. Find **action** from `https://getastra.com/wp-content/themes/jnews/assets1/js/app.js`.



6. Create a request body with the collected data and add `broker_user_roles=3` to the data.

   `action=save_user_details&uid=10695&broker_user_roles=`

7. Set this body to the request and send the request.

Suggested Fix

- Make it mandatory for developers to declare 'Allowed' access for each resource, and by default, deny it.
- Unless a resource is intended to be publicly accessible, deny access by default.
- Wherever possible, use a single application-wide mechanism for enforcing access controls.
- All load/api calls in the application should check if the logged-in user has permission to access or not.

Additional References

- <https://www.hacksplaining.com/prevention/broken-access-control>;

3. [CRITICAL] Admin Account Takeover On Update Contact Details	
Severity	Critical
Status	Solved
Risk Score	8.7/10
CWE	284: Improper Access Control
CVSS	9.1 (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N)
Labels	SOC 2, GDPR, SOC 2 - Privacy, OWASP 2021 - A01 - Broken Access Control, ISO 27001, PCI DSS, OWASP 2021, HIPAA

Description

During our pentest, it was observed that the `/api/UpdateContact` endpoint facilitates the updating of profile details, utilizing parameters like `web_UserId`, `LoginName`, and `LoginEmail`. This endpoint exhibits behavior where certain fields are modified based on an authentication token, while others are altered using the `web_UserId`. Exploiting this functionality allows unauthorized modification of another user's login email by manipulating the `web_UserId` and subsequently taking control of the victim's account by resetting the password.

Impact

- **Account Takeover**: Exploiting the vulnerability enables attackers to change another user's login email by manipulating the `web_UserId`, subsequently taking control of the victim's account.
- **Password Reset Exploitation**: By updating the victim's `LoginEmail`, attackers can then initiate a password reset process using the newly set email address.
- **Complete Account Compromise**: With control over the victim's email and the ability to reset their password, attackers can fully take over the victim's account, accessing sensitive information and performing actions on their behalf.
- **Data Exposure**: Unauthorized access to and modification of personal details such as email addresses may lead to exposure of sensitive user information.
- **Reputation Damage**: Exploitation of this vulnerability can severely damage the organization's reputation and erode user trust.

Affected Components

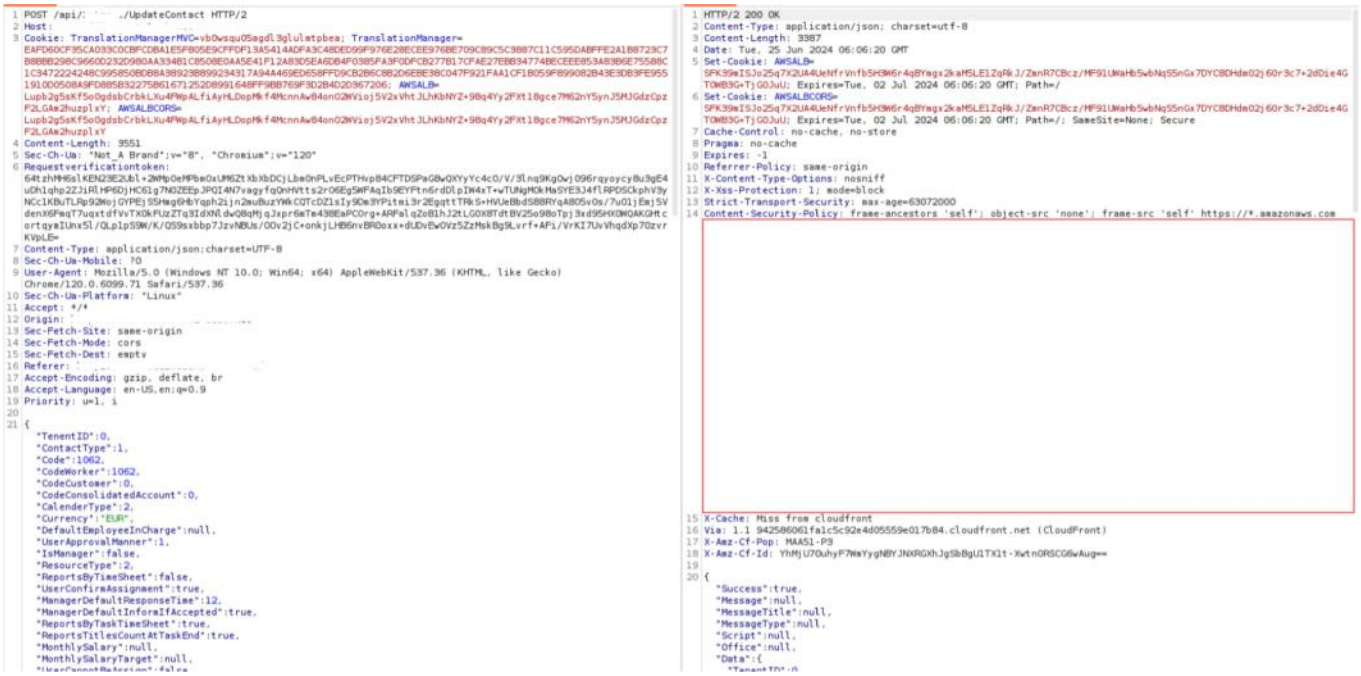
https://getastra.com

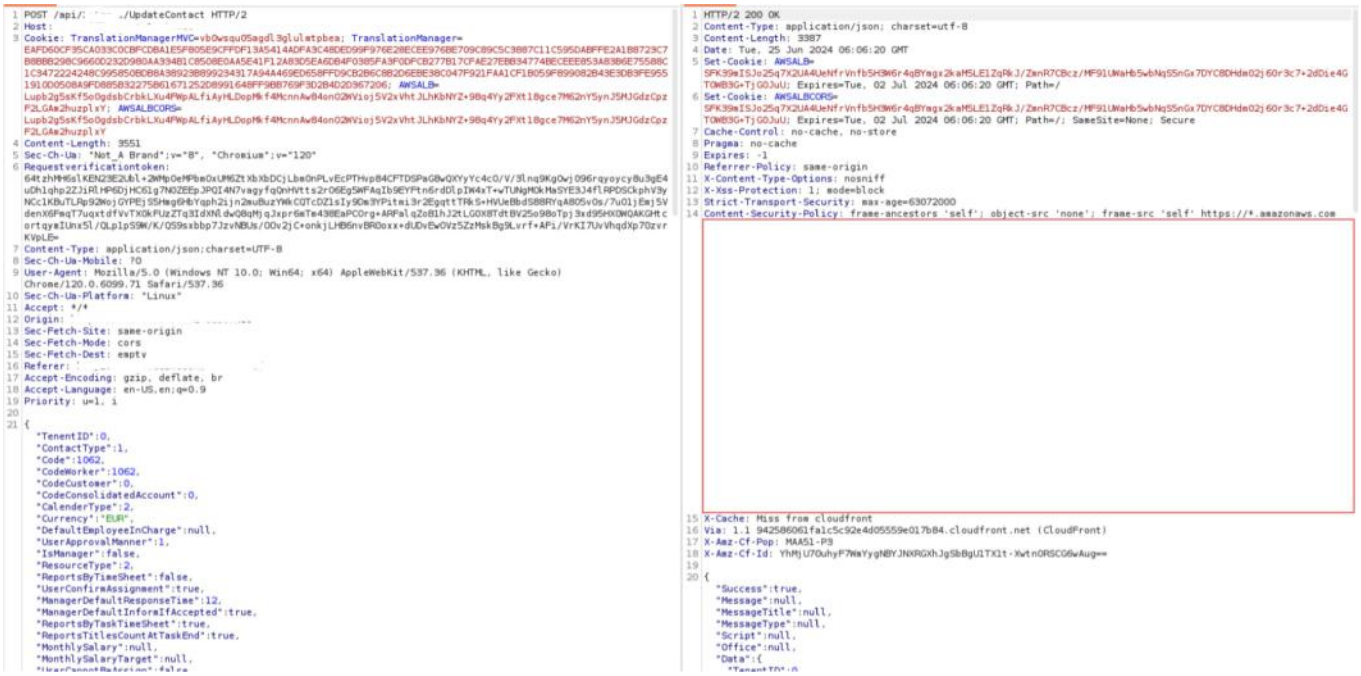
Steps to Reproduce

1. **Authentication**: Obtain a valid authentication token or session for your own account.
2. **Identify Target User**: Identify the `web_UserId` of the target user whose account you intend to compromise.
3. **Modify Request**: Send a request to `/api/UpdateContact`, altering the `web_UserId` parameter to match the target user's `web_UserId`. Update the `LoginEmail` parameter to a new email address controlled by the attacker.
4. **Verify Changes**: Confirm that the request successfully updates the target user's `LoginEmail` to the specified address.
5. **Password Reset**: Utilize the newly set `LoginEmail` to initiate a password reset process for the target user's account.
6. **Account Takeover**: Complete the password reset process and gain unauthorized access to the target user's account using the new credentials obtained.
7. And we would be able to takeover any users account.

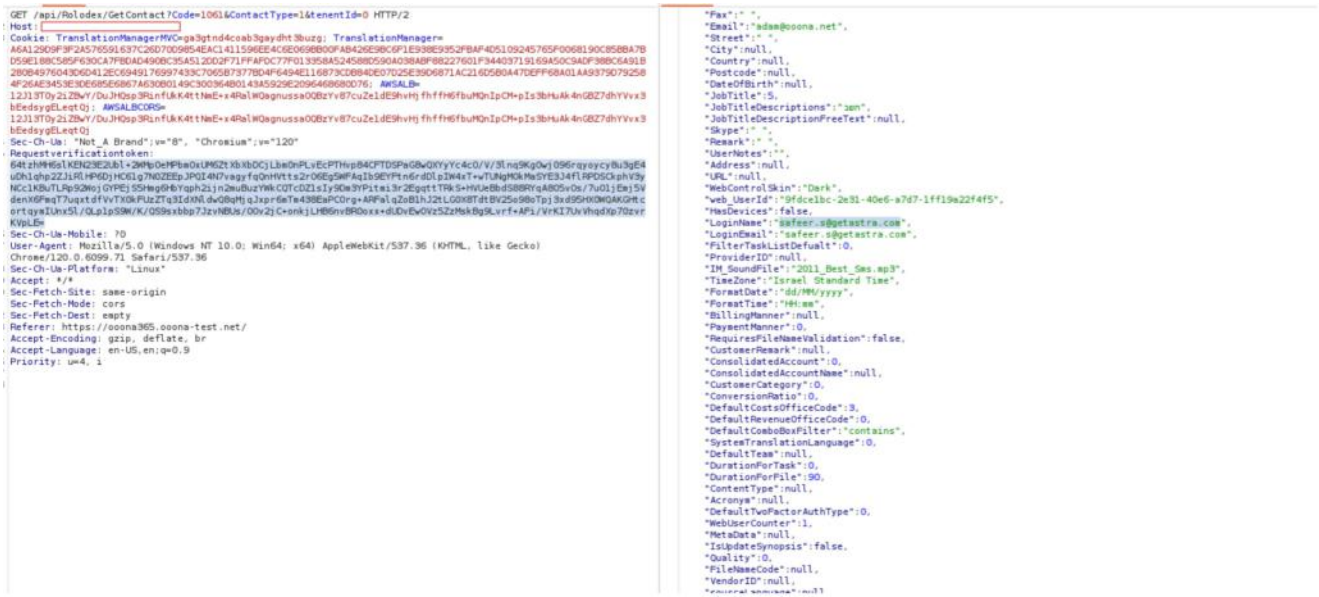
POC

- Profile Update with another user id

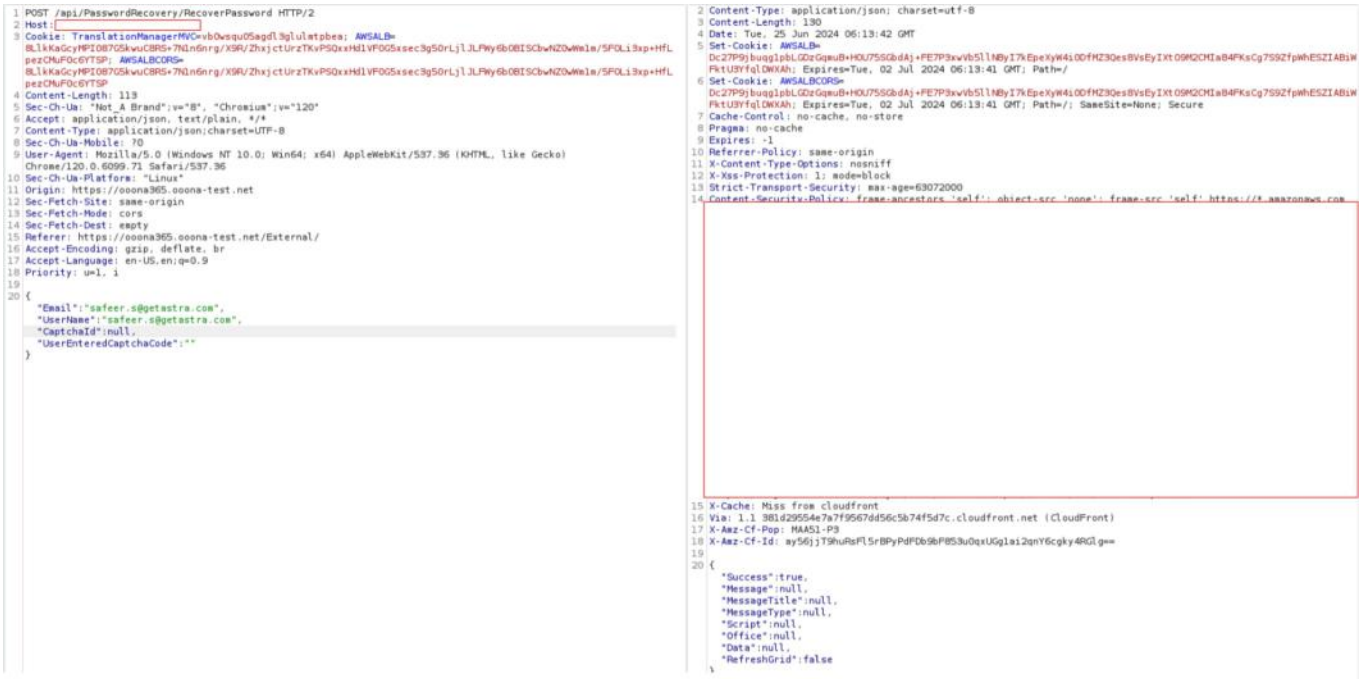




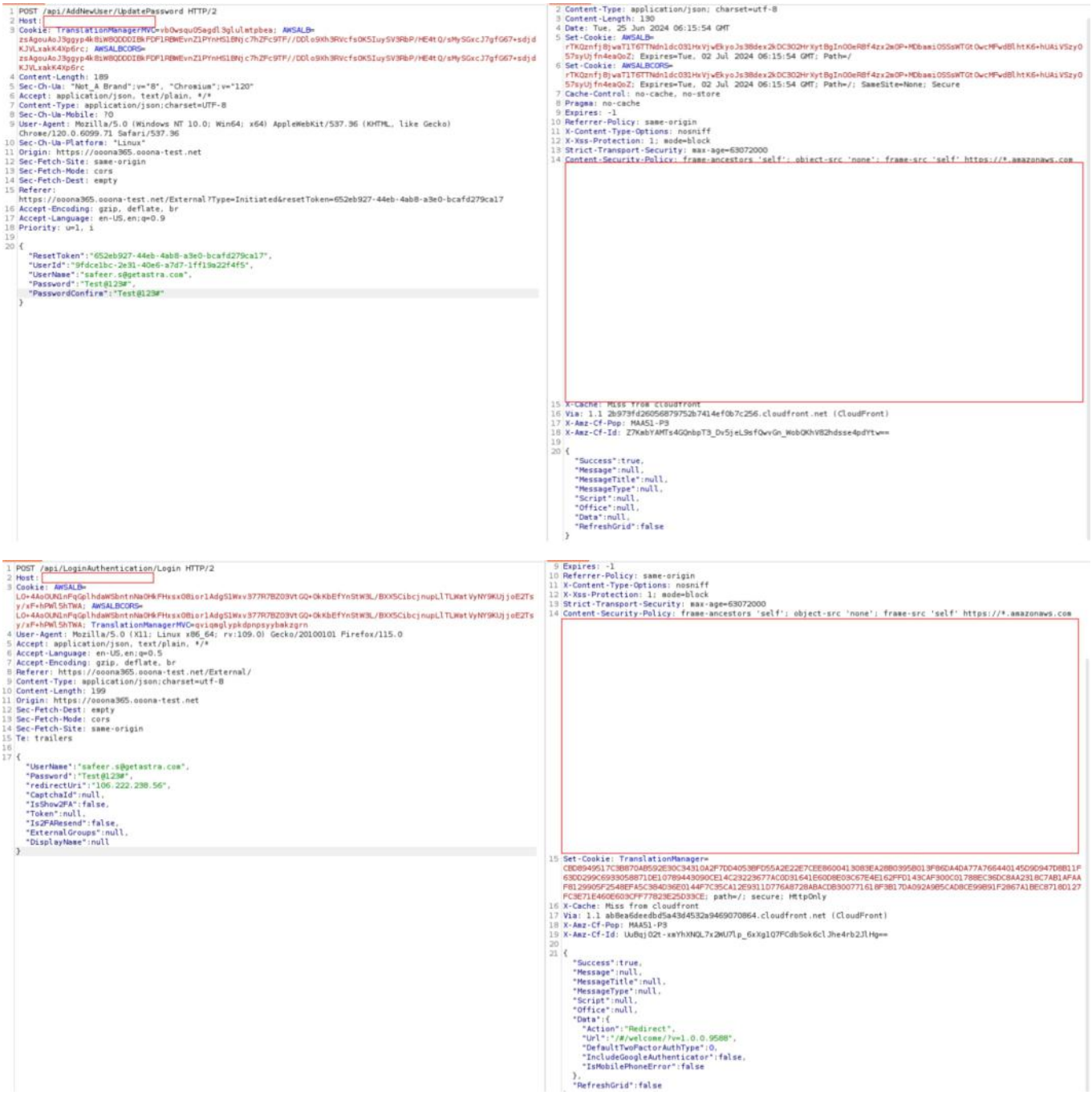
- And updated details can be seen on the fetch user request. And this has successfully updated the email of another user.



- And Initiated a password request



- And successfully updated the password and takeover the account.



Suggested Fix

- Make it mandatory for developers to declare 'Allowed' access for each resource, and by default, deny it.
- Unless a resource is intended to be publicly accessible, deny access by default.
- Wherever possible, use a single application-wide mechanism for enforcing access controls.
- All load/api calls in the application should check if the logged-in user has permission to access or not.

Additional References

<https://www.hacksplaining.com/prevention/broken-access-control>;

4. Unprotected Magmi - Can Be Used As A Backdoor, Full Complete Database Access	
Severity	High
Status	Solved
Risk Score	6.9/10
CWE	0: Vulnerability
CVSS	7.5 (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N)

Description

During our pentest, we discovered that the MAGMI endpoint isn't configured properly and can be used maliciously.

MAGMI (Magento Mass Importer), is a popular Magento Data Import Tool, that often is used without any protection in its default location (`/magmi/web/magmi.php`).

Incorrect implementation of this tool can be abused to gain full access to a Magento installation, especially taking into account `CVE-2014-8770` vulnerability and public exploits available

Impact

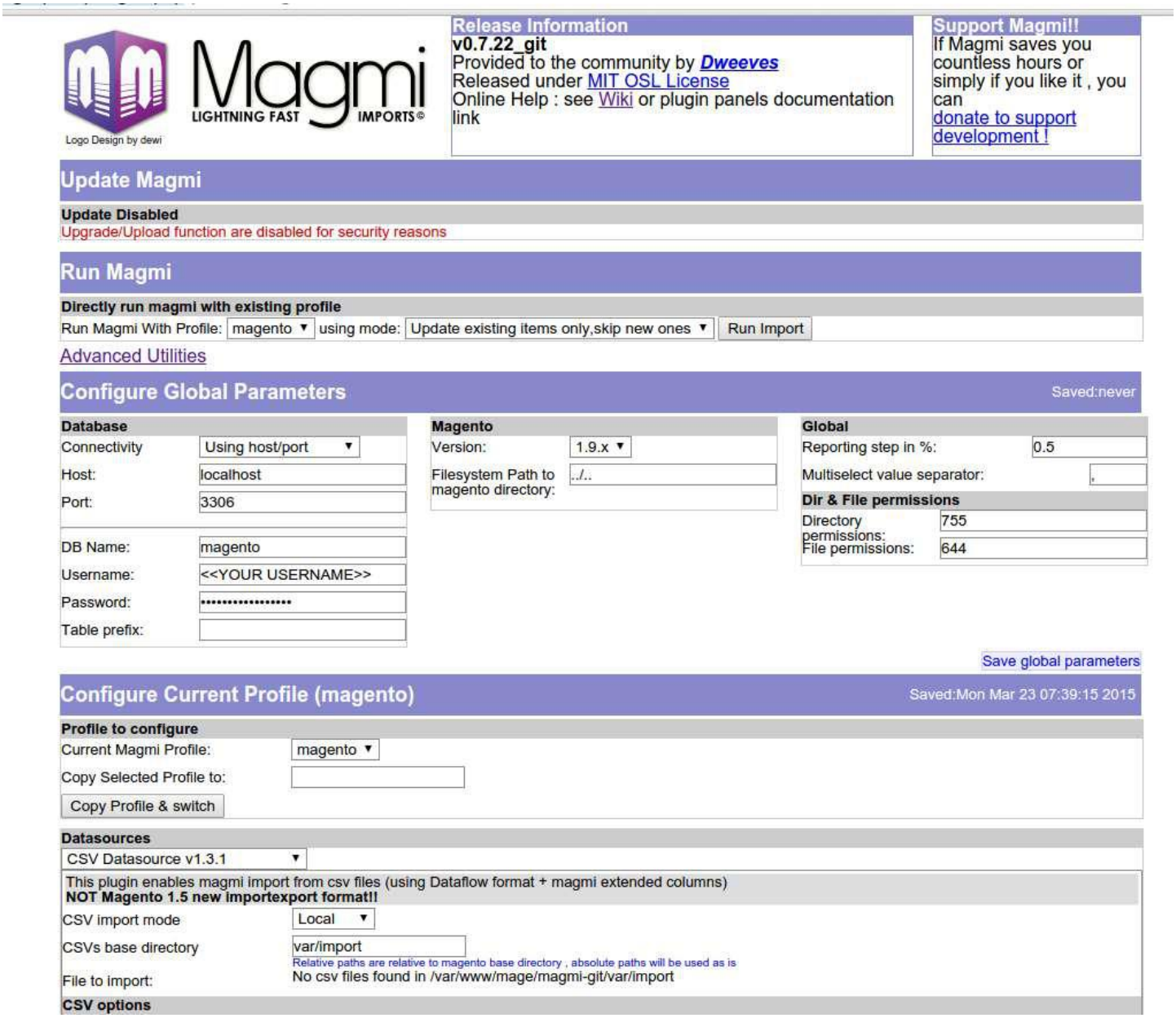
Full compromise of your Magento Store

Affected Components

https://store.example.com/magmi/web/magmi.php

https://ecomm.example.com/magmi/web/magmi.php

Steps to Reproduce



1. Visit the affected URL

Suggested Fix

There are several ways of restricting access to /magmi/ possible. You can select any way that suit your needs and qualification

- **Move /magmi/ out when don't need it** The most simple way that requires absolutely no knowledge of webserver magic. Just navigate to your Magento root directory in your web-filemanager (FTP or SSH are also just fine) and move `/magmi/` folder or into another folder that is already protected, preferably renaming it.

- **Restrict access by IP address**

- **Apache2 with .htaccess enabled**

- Add the following lines on top of `/magmi/.htaccess and /magmi/web/.htaccess` files

```
Order deny,allow
Deny from all
Allow from 100.111.100.108
```

5. Stripe API Key Disclosed	
Severity	High
Status	Solved
Risk Score	6.8/10
CWE	0: Vulnerability
CVSS	8.2 (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:H/A:N)

Description

During our pentest, we discovered a possible Leak of **Stripe API Key** in the response body. Disclosure of valid private keys may lead to unauthorized access to any systems that use them for authentication. Please verify whether any keys disclosed are actually valid, and whether their disclosure within the application is appropriate.

Impact

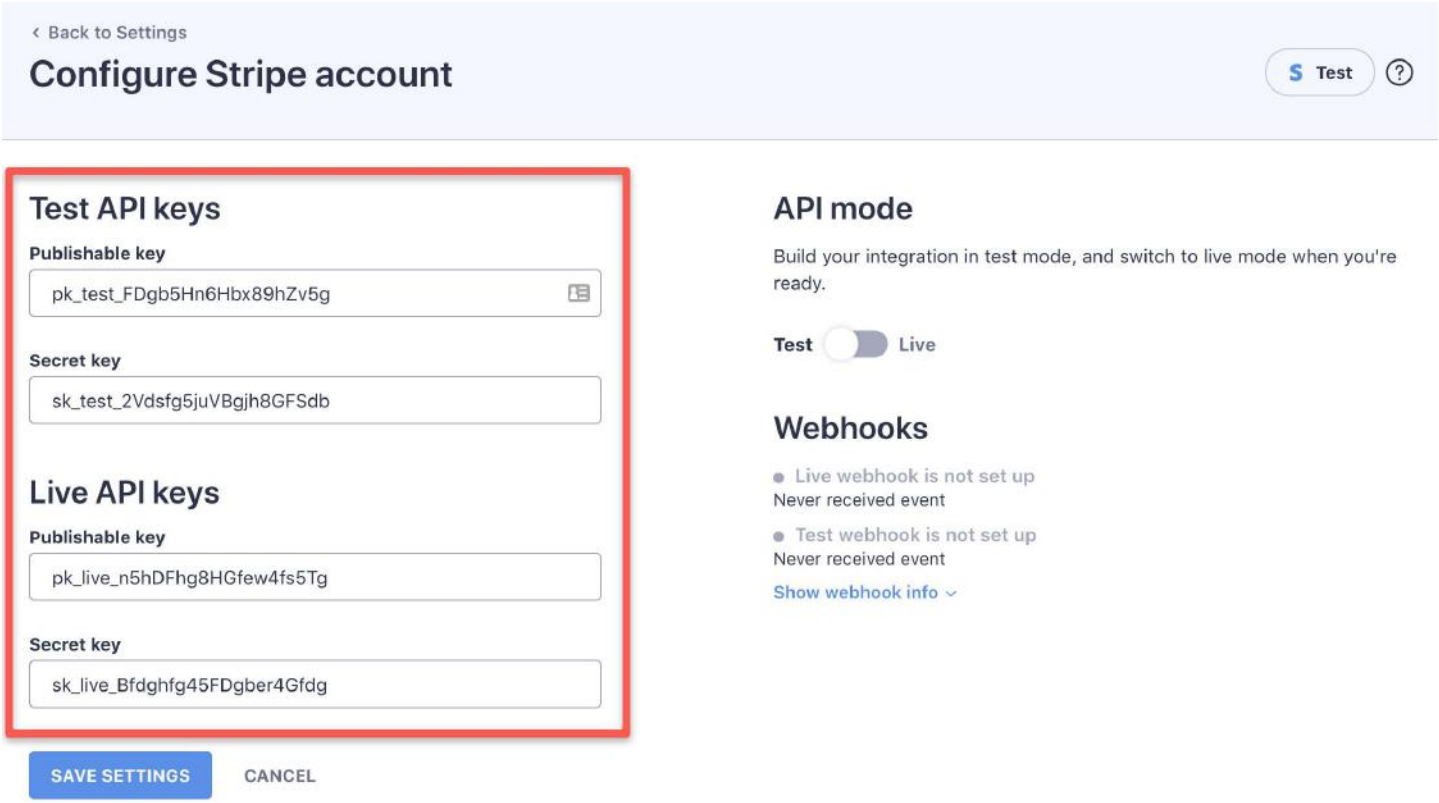
Stripe API keys are used to very webhook calls, encrypt data and make API calls to Stripe. An attacker can impersonate you and perform unintended actions such as:

- Downloading customer PII
- Modifying gateway settings
- Stealing payments

Affected Components

https://example.com/main-es2018.js

Steps to Reproduce



1. Open the affected URL in your browser
2. Right click and select **View Source**
3. Search for the API key mentioned in the Payload section

Suggested Fix

1. Make sure that the disclosed key is removed or has sufficient permissions to prevent exploitation
2. Avoid embedding sensitive API keys in JavaScript files since they can be accessed by anyone
3. Use secrets management for storing sensitive API keys

Additional References

- Web Security Academy: Information disclosure

6. Possible To Bypass Work Email Only Restriction And Gain Access To Other Domains	
Severity	High
Status	Solved
Risk Score	6.8/10
CWE	0: Vulnerability
CVSS	8.1 (CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N)

Description

During our pentest, it was noted during the testing that the sign up process has restrictions set up in place that requires only work emails to be used. However, we were able to bypass this restriction and gain access to other domains registered on the website.

Impact

It was possible for unregistered attackers to sign up using any email, bypassing the work only email restriction, and fetch data of registered organizations. This information can later be used for nefarious purposes.

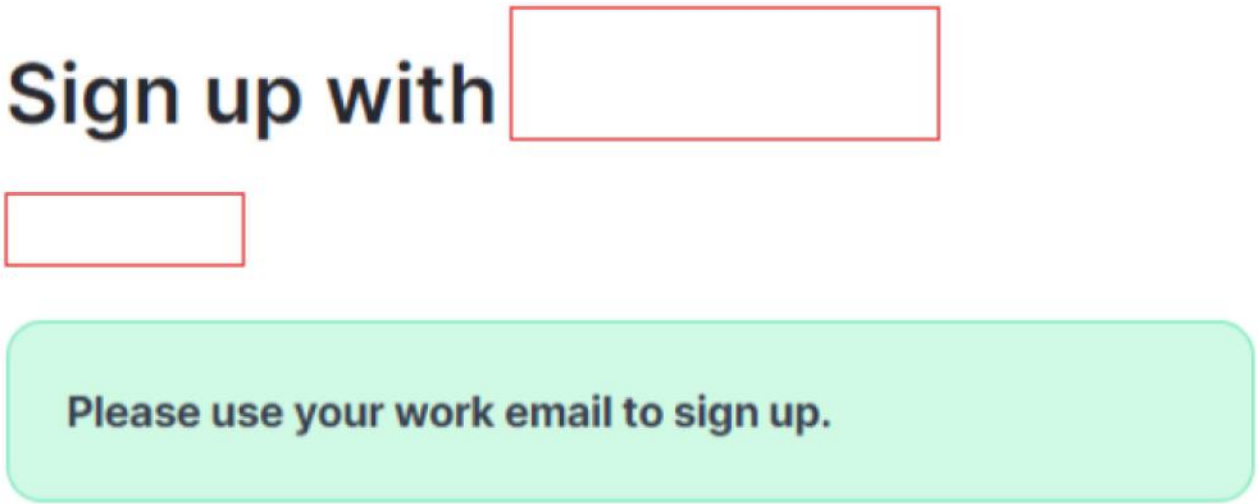
Affected Components

https://getastra.com

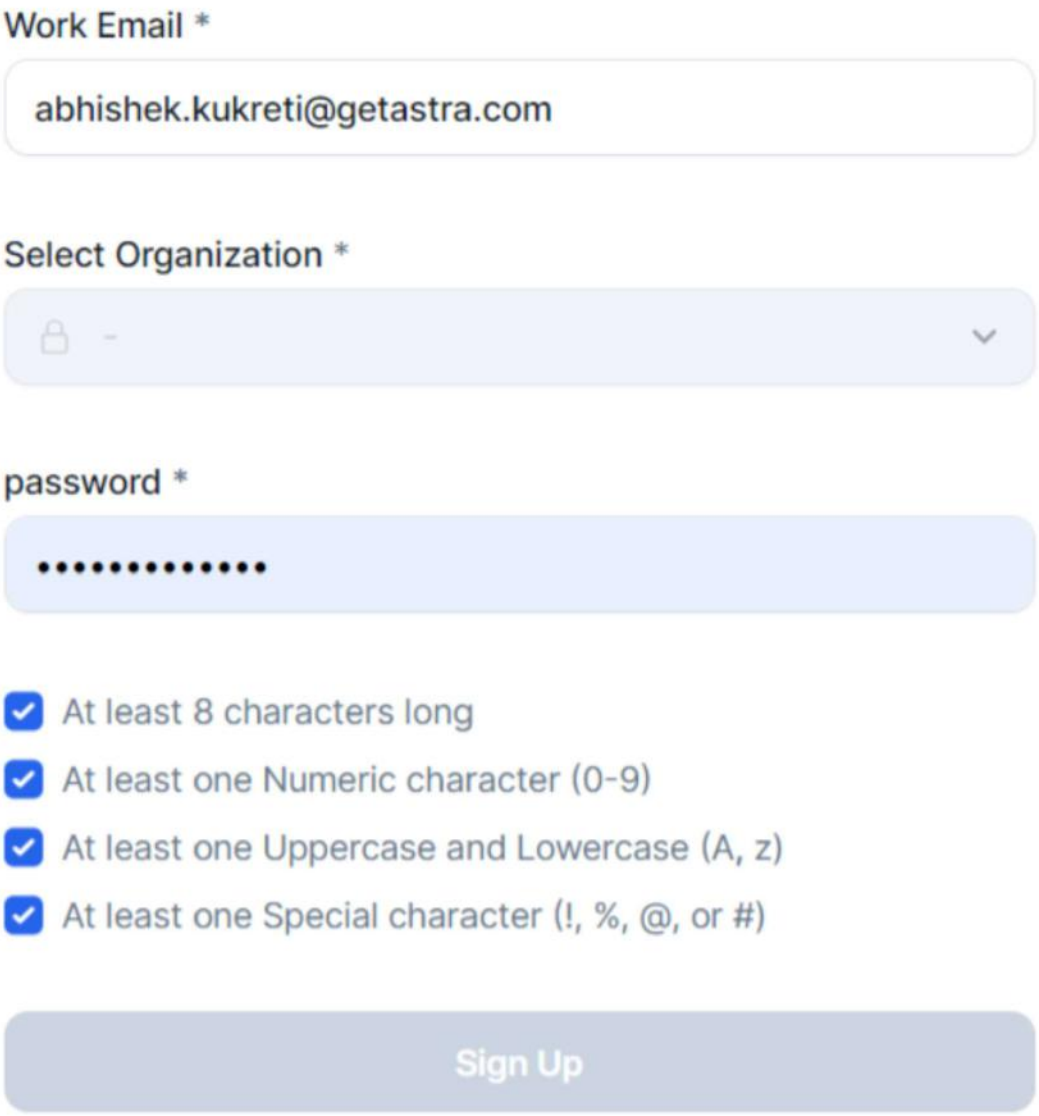
Steps to Reproduce

- Go to the affected component
- If we register using any random Gmail account, the following error is displayed

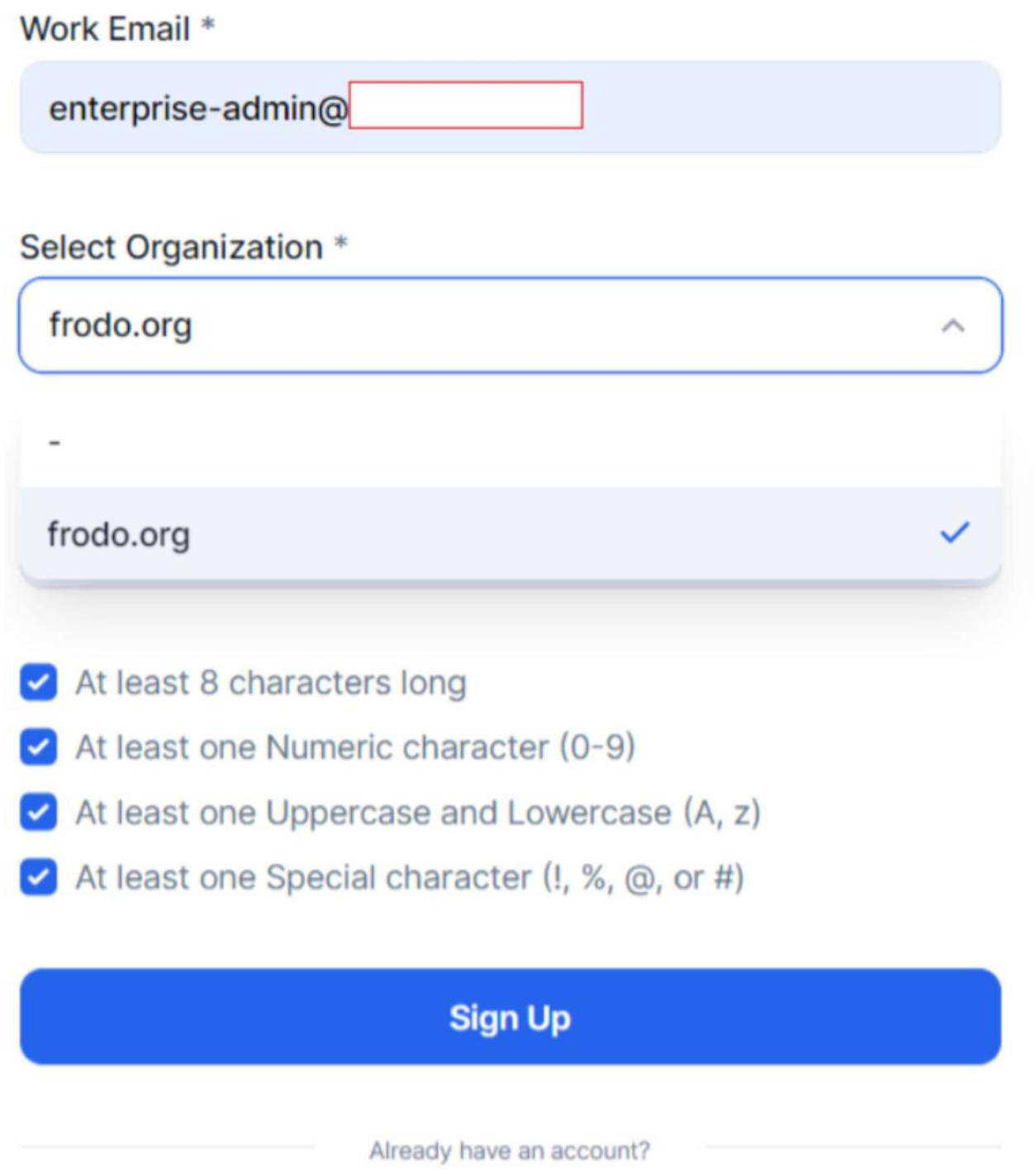
```
Please use your work email to sign up.
```



- We can also note that we won't be able to enter any organization name if we use any random email



- Now, use any registered email address



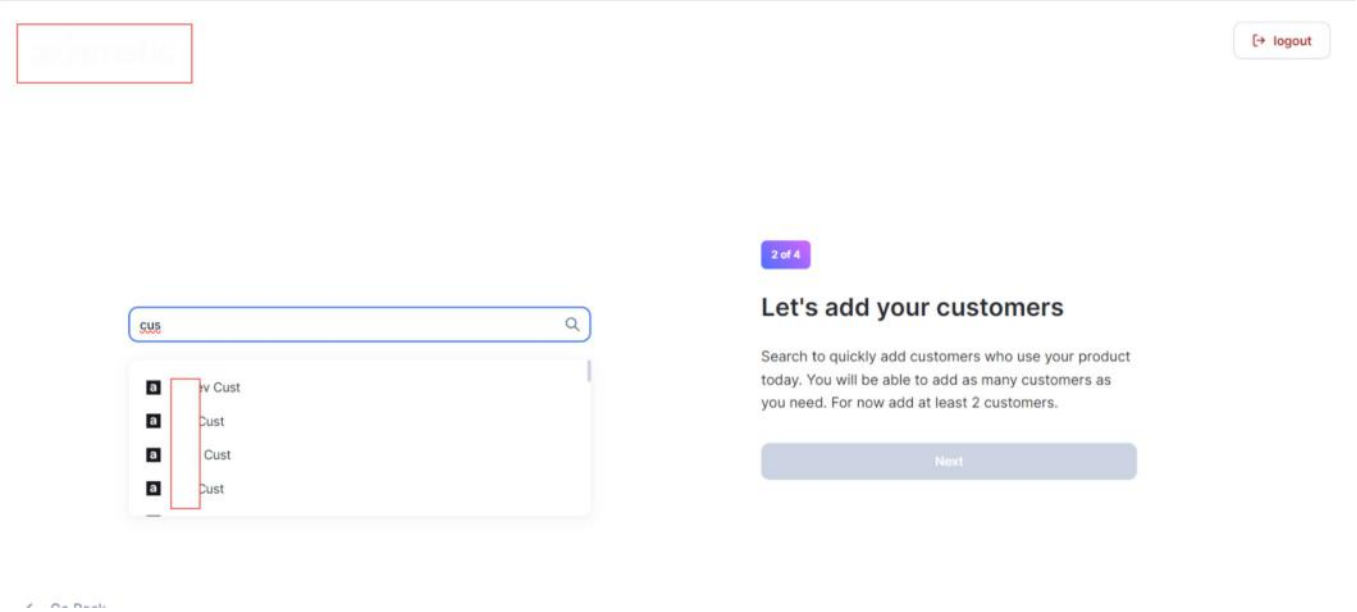
- Capture the request using Burp Proxy and enter your details in the `userName` `userEmail` and the `orgName`

```
POST /onboarding-app/api/v1/tenant/register HTTP/2
Host:
Content-Length: 229
Sec-Ch-Ua: "Chromium";v="121", "Not A(Brand";v="99"
Accept: application/json, text/plain, */*
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.160
Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Origin:
```

```
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

{"userName":"fakeemail51515151@gmail.com","userEmail":"fakeemail51515151@gmail.com","password":"ASDADS@123","publisherId":14697,"orgName":"frodo_org61717z","domain":"axmtestone.ml","mode":"","isCustomer":true,"isVendor":false}
```

- You will now be registered and can access the details of the domain



Suggested Fix

- Ensure that proper server side checks are implemented so users cant use non registered emails
- Ensure that no random individual is able to register using other domain details thats leaked in the register page

7. Possible For Lower Privileged Users To See Details Of Admin Users	
Severity	High
Status	Solved
Risk Score	6.6/10
CWE	284: Improper Access Control
CVSS	5.3 (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N)
Labels	SOC 2, GDPR, SOC 2 - Privacy, OWASP 2021 - A01 - Broken Access Control, ISO 27001, PCI DSS, OWASP 2021, HIPAA

Description

During our pentest, it was discovered that a lower-privileged user can extract information that can only be fetched by the Admin users.

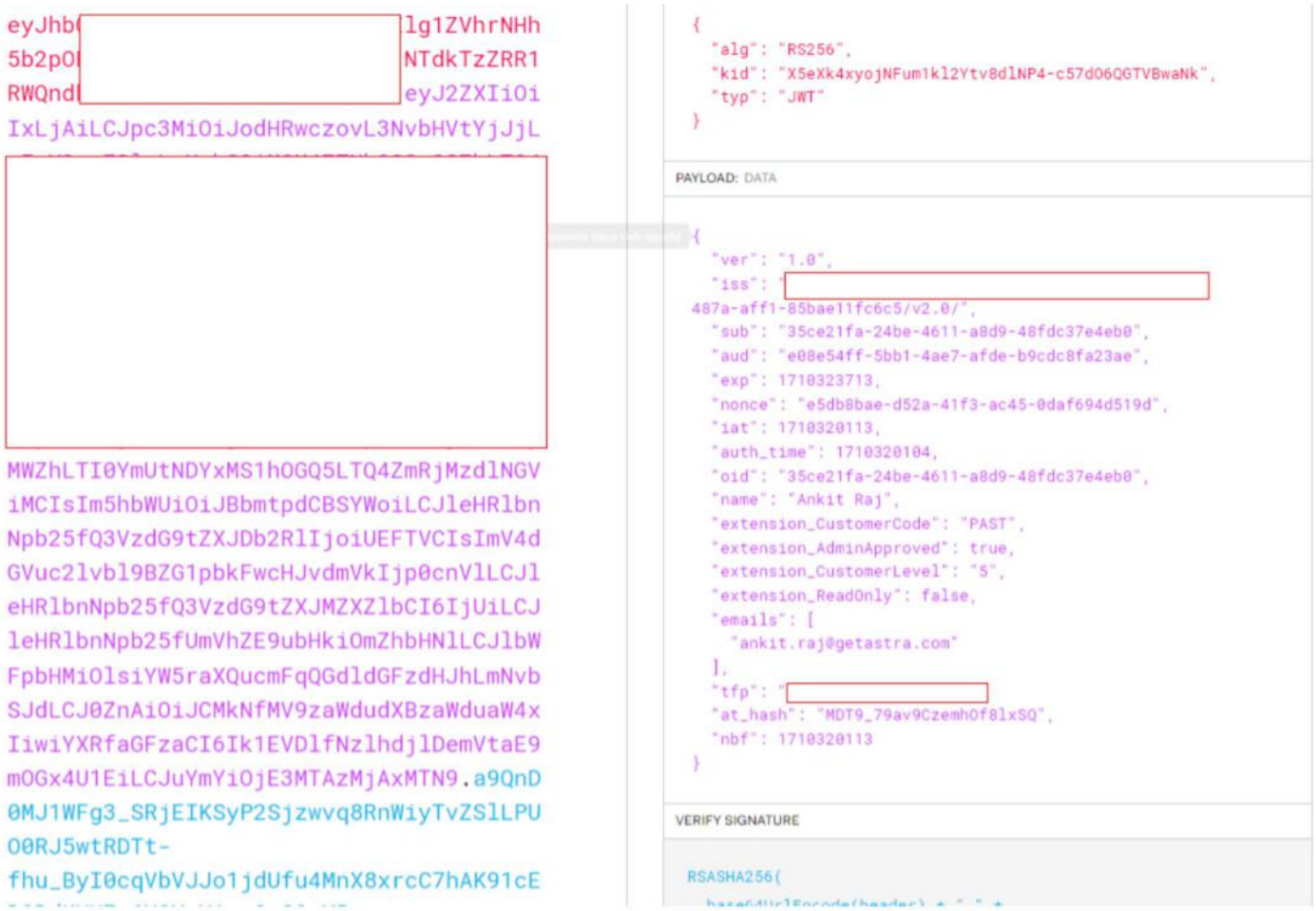
Affected Components

https://getastra.com

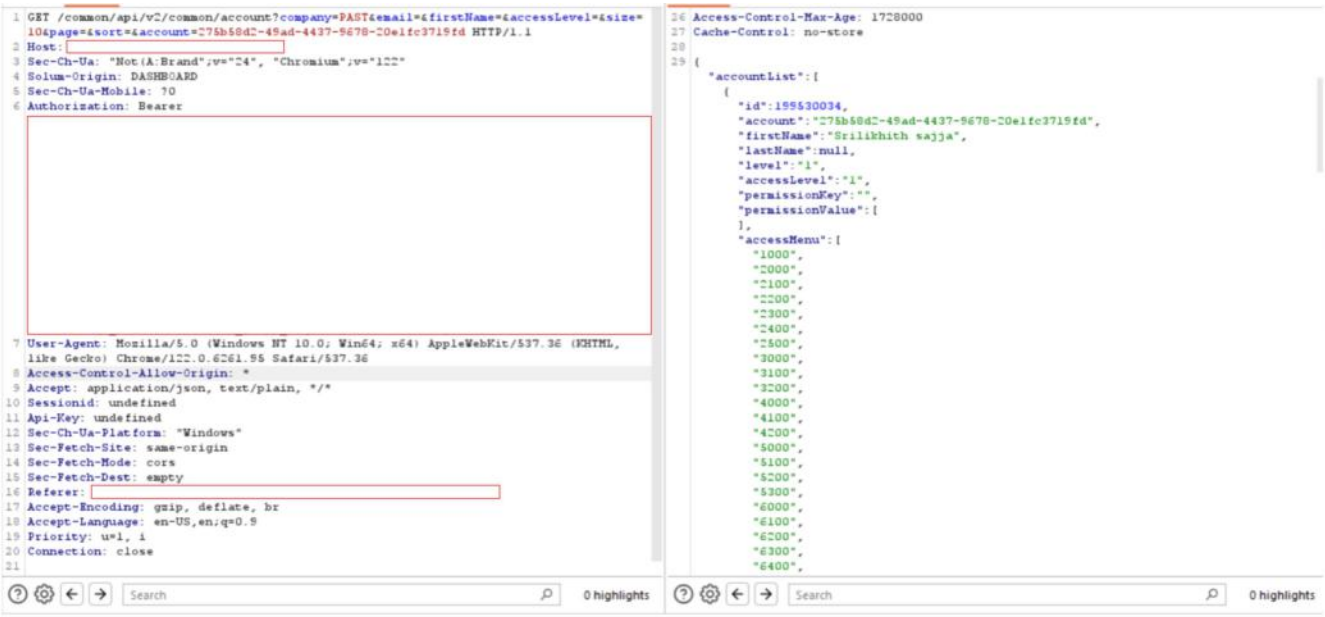
Steps to Reproduce

- We sent the following request using the JWT token of a lesser-privileged user

```
GET /common/api/v2/common/account?company=PAST&email=&firstName=&accessLevel=&size=10&page=&sort=&account=275b58d2-49ad-4437-9678-20e1fc3719fd HTTP/1.1
Host:
Sec-Ch-Ua: "Not(A:Brand";v="24", "Chromium";v="122"
Solum-Origin: DASHBOARD
Sec-Ch-Ua-Mobile: ?0
Authorization: Bearer Token
Access-Control-Allow-Origin: *
Accept: application/json, text/plain, */*
Sessionid: undefined
Api-Key: undefined
Sec-Ch-Ua-Platform: "Windows"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i
Connection: close
```



- We sent the request and were able to extract details of the Admin user



Suggested Fix

- Make it mandatory for developers to declare 'Allowed' access for each resource, and by default, deny it.
- Unless a resource is intended to be publicly accessible, deny access by default.
- Wherever possible, use a single application-wide mechanism for enforcing access controls.
- All load/api calls in the application should check if the logged-in user has permission to access or not.

Additional References

<https://www.hacksplaining.com/prevention/broken-access-control>;

8. Outdated and Vulnerable Components In Use	
Severity	Medium
Status	Solved
Risk Score	5/10
CWE	362: Information Disclosure
CVSS	6.3 (CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:L)

Description

During our pentest, we found that the application is using an older version of

1. PHP (7.1.17)
2. Nginx (1.12.2)
3. Wordpress (4.9.9) [Blog]
4. Magento (1.9.x)

   These versions are outdated and should be updated as soon as possible as using an outdated version of any software with unpatched security issues can enable an attacker to exploit them and perform various malicious actions.

   The Nginx version used is known to have exploitable vulnerabilities as shown below.+ nginx before versions 1.15.6 and 1.14.1 has a vulnerability in the implementation of HTTP/2 that can allow for excessive CPU usage. This issue affects nginx compiled with the ngx_http_v2_module (not compiled by default) if the 'http2' option of the 'listen' directive is used in a configuration file.+ nginx before versions 1.15.6 and 1.14.1 has a vulnerability in the implementation of HTTP/2 that can allow for excessive memory consumption. This issue affects nginx compiled with the ngx_http_v2_module (not compiled by default) if the 'http2' option of the 'listen' directive is used in a configuration file.

Affected Components

https://examplemag.com/

https://examplemag.com/blog

https://examplemag.com/ https://examplemag.com/blog

Steps to Reproduce

- We were able to use Wappalyzer extension on Google Chrome to find the WordPress and Nginx version in use

- The PHP version was revealed on running the tool Nikto
- The Magento version was revealed inside the[Magmi.ini configuration file]()
- The Nginx version was revealed in 404 pages as well server response headers, as shown below

```
 HTTP/1.1 200 OK Server: nginx/1.12.2 Date: Tue, 08 Jan 2019 05:59:55 GMT Content-Type: text/html; charset=UTF-8
Connection: close Expires: Thu, 19 Nov 1981 08:52:00 GMT Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache Set-Cookie: frontend=ebc4a6e23fc9afc001c8ddb7de8ddcdc; expires=Tue, 08-Jan-2019 15:59:54 GMT;
Max-Age=36000; path=/; domain=bilablau.dk X-Frame-Options: SAMEORIGIN Content-Length: 320280
```

Suggested Fix

- It is recommended to upgrade or update to the latest stable version of the affected component that is currently available
- It is always highly recommended to hide version numbers of software used, as this can make the attack easier for hackers.

9. Reverse Tabnabbing	
Severity	Medium
Status	Solved
Risk Score	4.8/10
CWE	1022: Use of Web Link to Untrusted Target with window.opener Access
CVSS	4.3 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N)
Labels	OWASP 2021, OWASP 2021 - A05 - Security Misconfiguration, SOC 2, SOC 2 - Security

Description

During our pentest, we discovered various Reverse Tabnabbing pages.

In Reverse Tabnabbing, when you open a link in a new tab (target="_blank"), the page that opens in a new tab can access the initial tab and change its location using the `window.opener` property.

It is an attack where a page linked from the target page is able to rewrite that page, for example, to replace it with a phishing site. As the user was originally on the correct page they are less likely to notice that it has been changed to a phishing site, especially if the site looks the same as the target. If the user authenticates to this new page then their credentials (or other sensitive data) are sent to the phishing site rather than the legitimate one.

As well as the target site being able to overwrite the target page, any HTTP link can be spoofed to overwrite the target page if the user is on an unsecured network, for example, a public wifi hotspot. The attack is possible even if the target site is only available via HTTPS as the attacker only needs to spoof the HTTP site that is being linked to. The attack is typically possible when the source site uses a target instruction in an HTML link to specify a target loading location that does not replace the current location and then lets the current window/tab available and does not include any of the preventative measures detailed below.

The attack is also possible for links opened via the `window.open` javascript function.

Here is a video showing an example of the Reverse Tabnabbing attack: https://drive.google.com/file/d/1wxYtasfo73HmXI-btHoLospXKN8J4Prm/preview

Affected Components

https://xyz.com

Steps to Reproduce

- Visit the above URL and right-click to select View page source
- On the page source, search for `_blank`
- Check if `noopener` and `noreferrer` keywords are set in the `rel` attribute
- One of the evidence we found is:

Suggested Fix

- Wherever target=_blank is used, it is highly recommended to add the attribute: rel="noopener noreferrer".
- Remember, that every time you open a new window via `window.open();` you're also vulnerable to this, so always reset the "opener" property

```
var newWnd = window.open();
newWnd.opener = null;
```

Additional References

- https://www.owasp.org/index.php/Reverse_Tabnabbing
- https://mathiasbynens.github.io/rel-noopener/ (DEMO)
- https://dev.to/ben/the-targetblank-vulnerability-by-example
- https://mathiasbynens.github.io/rel-noopener/
- https://medium.com/@jitbit/target-blank-the-most-underestimated-vulnerability-ever-96e328301f4c

10. Insecure HTTP Cookies	
Severity	Medium
Status	Solved
Risk Score	4.8/10
CWE	614: Sensitive Cookie in HTTPS Session Without 'Secure' Attribute
CVSS	4.3 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N)

Description

During our pentest, we discovered some cookies without HTTPOnly and Secure flags.

SSL cookie without Secure flag and HttpOnly set was found on this website. If the secure flag is set on a cookie, then browsers will not submit the cookie in any requests that use an unencrypted HTTP connection, thereby preventing the cookie from being trivially intercepted by an attacker monitoring network traffic. If the secure flag is not set, then the cookie will be transmitted in clear text if the user visits any HTTP URLs within the cookie's scope.

If the HttpOnly attribute is set on a cookie, then the cookie's value cannot be read or set by client-side JavaScript. This measure makes certain client-side attacks, such as cross-site scripting, slightly harder to exploit by preventing them from trivially capturing the cookie's value via an injected script.

Even if the domain that issued the cookie does not host any content that is accessed over HTTP, an attacker may be able to use links of the form http://example.com:443/ to perform the same attack.

Affected Components

https://example.com

Steps to Reproduce

- Verifying that a web site sets this flag on any particular cookie can be done using an intercepting proxy, like ZAP. You can capture each response from the server and examine any Set-Cookie headers it includes to see if the secure flag or HttpOnly flag is set on the cookie.

Suggested Fix

- **Set Secure Flag:** The secure flag should be set on all cookies that are used for transmitting sensitive data when accessing content over HTTPS. If cookies are used to transmit session tokens, then areas of the application that are accessed over HTTPS should employ their own session handling mechanism, and the session tokens used should never be transmitted over unencrypted communications.
- **Set HttpOnly flag:** There is usually no good reason not to set the HttpOnly flag on all cookies. Unless you specifically require legitimate client-side scripts within your application to read or set a cookie's value, you should set the HttpOnly flag by including this attribute within the relevant Set-cookie directive.

  For more info:

  OWASP - How to set the SecureFlag on cookies

  PHP - Setting a secure session cookie

  OWASP - HttpOnly

11. Missing API Security Headers	
Severity	Medium
Status	Solved
Risk Score	4.8/10
CWE	0: API Security Headers
CVSS	4.3 (CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N)
Labels	OWASP 2021, OWASP 2021 - A05 - Security Misconfiguration

Description

During our pentest, we detected that the following API security headers are missing

1. Content Security Policy
2. Strict Transport Security
3. X-Content-Type-Option

1. Content Security Policy:A CSP is an important standard by the W3C which prevents a broad range of content injection attacks such as cross-site scripting (XSS), data injection attacks, packet sniffing attacks, etc. It is a declarative policy that informs the user agent what are valid sources to load resources from.

2. Strict Transport Security Header: Missing the Strict Transport Security header means that the application allows users to connect over unencrypted networks. As a result, an attacker can modify a legitimate user's network traffic, could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The sslstrip tool automates this process.

3. X-Content-Type-Option: Missing Content-Type header means that this website could be at risk of MIME-sniffing attacks.

Impact

Missing Strict Transport Security header means that the application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The sslstrip tool automates this process

Missing Content-Type header means that this website could be at risk of a MIME-sniffing attacks.

Affected Components

Sitewide

Steps to Reproduce

We scanned the website using ZAP Proxy which alerted us of these missing headers

Suggested Fix

The recommended configuration for API endpoints is:

```
Content-Security-Policy: default-src 'none'; frame-ancestors 'none'
Strict-Transport-Security: max-age=63072000
X-Content-Type-Options: nosniff
```

12. Possible To Prevent Normal Users From Booking Tickets By Performing Large Number Of False Pre-Bookings	
Severity	Medium
Status	Solved
Risk Score	4.7/10
CWE	665: Vulnerability
CVSS	6.3 (CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:L)

Description

During our pentest, we found that the application allows anonymous users to make pre-bookings for any number of available seats for any trip by entering a random email ID and phone number and choosing PayX as a payment option

This facility of allowing Pre-bookings for n number of tickets (seats) in a specific trip can be misused by attackers to keep making false pre-bookings often using any email id and phone number, and thus making ticket bookings always unavailable for normal users resulting in financial loss for the company.

Affected Components

https://example.com

Steps to Reproduce

1. Access URL - `https://www.example.com/` and Choose City of Departure and City of Arrival and Date of Trip and click on Search and we will get the search results like - `https://www.example.com/ticket/Show?DepartId=11&ArriveeId=15&dateDepart=2019-01-02`
2. Click on See Seats on any one of the available trips and then select all free seats available for booking and click on the Book option
3. The user can give any anonymous email id, like `dubuzuba@cliptik.net` and phone number such as `9999999999` and then choose `PayX` as payment option for pre-booking all available free seats for any trip
4. The same malicious user can keep pre-booking for n number of trips to make seats unavailable for normal users.

Payload:

```
 POST /order/Step?order=f4fdb7fee17c HTTP/1.1
 Host: www.example.com
 User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
 Accept: text/html,application/xhtml
xml,application/xml;q=0.9,*/*;q=0.8
 Accept-Language: en-US,en;q=0.5
 Accept-Encoding: gzip, deflate
 Content-Type: application/x-www-form-urlencoded
 Content-Length: 288
 Connection: close
 Cookie: Abp.Localization.CultureName=en; _ga=GA1.2.478555404.1545993401; cookieconsent_status=dismiss;
ASP.NET_SessionId=0gyxcafwrppqaoubpq5m0tpt;
__RequestVerificationToken=5aw7mC4RZpnRP68Z1NmtmkSP1XQ1gt9I0CA0PSugi6oePJIxTMfutRNIuCy5jWde1ioEJEjUGookl-
nK7wxoglTcGFTfZkEyIj5GZBhKQiY1; XSRF-TOKEN=e00DKqcAbZMxkNiPbTRGjqgKFRwMbGbctGxkP-Q-hZOlhBTli6-
as48AAz0f1Dab4GL__GJPMZUcwb6czlBbpoG_R1QHabFEeeCArOD3ebE1; TawkConnectionTime=0; _gid=GA1.2.1972097347.1546234622;
_fbp=fb.1.1546234623401.1297126354
 Upgrade-Insecure-Requests: 1

__RequestVerificationToken=TUZNZzqhg3_7uiOOBgLCYLL2tEroW-
bGIb6rB8siDogAj3rho3YdASbThzsoCRkPlu3_T_svjNcITnjTCm2HSuOqQEBwQdZcSvy9mO9UsiA1&couponCode=&clt.email=dubuzuba%40cliptik
.net&clt.t

 POST /order/Step?order=f4fdb7fee17c HTTP/1.1
 Host: www.example.com
 User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
/
;q=0.8
 Accept-Language: en-US,en;q=0.5
 Accept-Encoding: gzip, deflate
 Content-Type: application/x-www-form-urlencoded
 Content-Length: 288
```

```
 Connection: close
 Cookie: Abp.Localization.CultureName=en; _ga=GA1.2.478555404.1545993401; cookieconsent_status=dismiss;
ASP.NET_SessionId=0gyxcafwrppqaoubpq5m0tpt;
__RequestVerificationToken=5aw7mC4RZpnRP68Z1NmtmkSP1XQ1gt9I0CA0PSugi6oePJIxTMfutRNIuCy5jWde1ioEJEjUGookl-
nK7wxoglTcGFTfZkEyIj5GZBhKQiY1; XSRF-TOKEN=e00DKqcAbZMxkNiPbTRGjqgKFRwMbGbctGxkP-Q-hZOlhBTli6-
as48AAz0f1Dab4GL__GJPMZUcwb6czlBbpoG_R1QHabFEeeCArOD3ebE1; TawkConnectionTime=0; _gid=GA1.2.1972097347.1546234622;
_fbp=fb.1.1546234623401.1297126354
 Upgrade-Insecure-Requests: 1

 __RequestVerificationToken=TUZNZzqhg3_7uiOOBgLCYLL2tEroW-
bGIb6rB8siDogAj3rho3YdASbThzsoCRkPlu3_T_svjNcITnjTCm2HSuOqQEBwQdZcSvy9mO9UsiA1&couponCode=&clt.email=dubuzuba%40cliptik
.net&clt.t
```

Suggested Fix

- It is recommended to implement Captcha like (Google reCAPTCHA v3) on Affected URL - https://www.example.com/order/Step?order=bb901cc878e8 and allow user to confirm payment method only after performing server side validation of CAPTCHA value entered by user. This will help prevent the use of bots for such attacks
- In case of bus ticket booking by anonymous users, kindly send OTP to phone number mentioned by user in https://www.example.com/order/Step?order=bb901cc878e8 and allow user to confirm payment via payx only on server side verification of OTP value associated with specifc order number and phone number
- Allow PayX as Payment option only for registered users who are already Logged In to their user account
- New users could be asked for a verification by KYC document submission upon registration. This will help identifying anyone misusing the web services and will prevent people from creating fake profiles to perform such attacks
- Another option that can be considered is that users could be asked for the payment of a small advance, for completing the pre-booking
- Limiting the complete pre-booking feature to only the users who has used the web services previously and paid for it could also help in separating legitimate users from the illegitimate ones.

Additional References

<https://swcregistry.io/docs/SWC-118>

13. Cross Domain Referrer Leakage	
Severity	Medium
Status	Solved
Risk Score	4.3/10
CWE	0: Cross Domain Information Leakage
CVSS	4.7 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:N/A:_)

Description

During our pentest, we were able to detect Cross-Domain Referer Leakage vulnerability on the website. This could result in sensitive information like the Order ID of a user, being disclosed.

When a web browser makes a request for a resource, it typically adds an HTTP header, called the "Referer" header, indicating the URL of the resource from which the request originated. This occurs in numerous situations, for example when a web page loads an image or script, or when a user clicks on a link or submits a form.

If the resource being requested resides on a different domain, then the Referer header is still generally included in the cross-domain request. If the originating URL contains any sensitive information within its query string, such as a session token, then this information will be transmitted to the other domain. If the other domain is not fully trusted by the application, then this may lead to a security compromise.

Affected Components

https://www.example.com/index.php?route=account/order/info&order_id=16752

Steps to Reproduce

- Visit the affected URL or likewise, and click on the Instagram link at the bottom of the page
- Using the Developer Tools in the browser, one can see that the below Request Header is sent when clicking the link

```
 GET /example.com/ HTTP/1.1 Host: www.instagram.com User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:63.0) Gecko/20100101 Firefox/63.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: https://www.example.com/index.php?route=account/order/info&order_id=16752 Connection: close Cookie: urlgen="{}:1g9ScV:-ZXSsn88EApkJCBfctnMETBnvfg"; rur=FTW; mid=W7srMAALAAEhi7oqf54d-6xt_X_o; mcd=3;
csrftoken=Nzs54oxlKBxf35f6RpKeEjE7ZHib7ZX8 Upgrade-Insecure-Requests: 1
```

- We were able to use Burp Suite Proxy to find that Cross Domain Referer Leakage vulnerability exists in the Affected URL, when we clicked on one of the Social Media links on the page.

Suggested Fix

- Applications should never transmit any sensitive information within the URL query string. In addition to being leaked in the Referer header, such information may be logged in various locations and maybe visible on-screen to untrusted parties. If placing sensitive information in the URL is unavoidable, consider using the Referer-Policy HTTP header to reduce the chance of it being disclosed to third parties
- The following code can be added to the `httpd.conf`file following which Apache should be restarted

```
<IfModule headers_module>
RequestHeader set X-HTTPS
```

```
Header always set Referrer-Policy: "same-origin"
</IfModule>
```

14. Secure SSH Access	
Severity	Low
Status	Solved
Risk Score	2.8/10
CWE	1125: Excessive Attack Surface
CVSS	6.3 (CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:L)

Description

During our pentest, we discovered that the server allows SSH connections from ANY IP address. It might be possible for hackers to brute-force credentials.

Affected Components

SSH

Suggested Fix

1. Speak to your host and only allow whitelisted/trusted IP addresses to login to the server via SSH. Your developers may have to whitelist their own IP every time for SSH access if they do not have a static IP
2. Use fail2ban to prevent brute-force: https://www.digitalocean.com/community/tutorials/how-to-protect-ssh-with-fail2ban-on-ubuntu-14-0
3. Use key based authentication for ALL SSH users rather than password based logic
4. [Optional] Enable two factor authentication for SSH via Duo Security: https://duo.com/docs/loginduo

15. No CAPTCHA Implemented	
Severity	Low
Status	Solved
Risk Score	2.5/10
CWE	693: Misconfiguration
CVSS	3.1 (CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N)

Description

During our pentest, it was found that certain pages on the website, which could be vulnerable to automated attacks did not have CAPTCHA implemented.

CAPTCHA needs to be implemented in public pages of website to prevent brute force attacks against the application, which could cause Denial of Service attacks.

Affected Components

https://xyz.com/module/giftchecks/useCheck

Steps to Reproduce

- Visit the affected Links to find no Captcha present to prevent brute force attacks.

Suggested Fix

- It is recommended to implement CAPTCHA to prevent brute force attacks.

Appendix

APPENDIX A — MEASUREMENT SCALES

Astra determines severity ratings using in-house expertise and industry-standard rating methodologies such as the Open Web Application Security Project (OWASP) and the Common Vulnerability Scoring System (CVSS).

The severity of each finding in this report was determined independently of the severity of other findings. Vulnerabilities assigned a higher severity have more significant technical and business impact and achieve that impact through fewer dependencies on other flaws.

Critical: Vulnerability is an otherwise high-severity issue with additional security implications that could lead to exceptional business impact. Findings are marked as critical severity to communicate an exigent need for immediate remediation. Examples include threats to human safety, permanent loss or compromise of business-critical data, and evidence of prior compromise.

High: Vulnerability introduces significant technical risk to the system that is not contingent on other issues being present to exploit. Examples include creating a breach in the confidentiality or integrity of sensitive business data, customer information, or administrative and user accounts.

Medium: Vulnerability does not in isolation lead directly to the exposure of sensitive business data. However, it can be leveraged in conjunction with another issue to expose business risk. Examples include insecurely storing user credentials, transmitting sensitive data unencrypted, and improper network segmentation.

Low: Vulnerability may result in limited risk or require the presence of multiple additional vulnerabilities to become exploitable. Examples include overly verbose error messages, insecure TLS configurations, and detailed banner information disclosure.

Informational: Finding does not have a direct security impact but represents an opportunity to add an additional layer of security, is a deviation from best practices, or is a security-relevant observation that may lead to exploitable vulnerabilities in the future. Examples include vulnerable yet unused source code and missing HTTP security headers.

APPENDIX B - RESOLUTION STATUS

Unsolved: This status indicates that the security team has reported an issue or vulnerability to the customer, but it is yet to be resolved by the customer. Further actions are required to address the reported security concern.

Under Review: This status is assigned when the customer has fixed the reported issue or vulnerability. The security team will now evaluate and validate the fix to ensure it has been implemented correctly and effectively mitigates the identified risk.

Accepted Risk: This status reflects the customer's decision not to address or resolve the reported issue or vulnerability. By accepting the associated risk, the customer has chosen not to pursue any further action in mitigating the identified security concern.

Help Wanted: When assigned this status, it indicates that the customer has requested additional clarification or assistance from the security team. This could involve seeking further guidance, recommendations, or expertise to better understand and address the reported security issue.

Solved: This status signifies that the customer has successfully resolved the reported vulnerability, and it has been verified by the security team. The necessary measures have been taken to mitigate the risk, ensuring that the identified security concern is no longer present.

APPENDIX C — RISK SCORE

Security recommendations/long-term tasks are marked as "Unsolved" or "Accepted Risk". Astra has evaluated the risk based on information provided and has provided feedback on a case-to-case basis as requested.

Security grades are assigned for vulnerabilities needing immediate attention and does not include security best practices, although reported. For each vulnerability, a risk score is assigned which signifies real world possibility of an attack being orchestrated using the vulnerability. The risk score is calculated using a correlation of multiple factors including potential loss value of a vulnerability, CVSS score, historic data about attacks performed using similar vulnerability & severity of such vulnerabilities assigned by our security engineers in the past.

APPENDIX D — TEST CASES

The following lists of tests are suggestive & not limited to the ones listed. Most importantly, every test case has multiple sub-test cases ranging from a few to sometimes 1000+ sub tests. Additional test cases will be performed based on factors such as:

1. Business Logic
2. Technology Stack
3. Framework/CMS/APIs
4. Application specific features

OWASP Top 10

#	For Applications
1	Broken Access Control
2	Cryptographic Failures
3	Injection
4	Insecure Design
5	Security Misconfiguration
6	Vulnerable and Outdated Components
7	Identification and Authentication Failures
8	Software and Data Integrity Failures
9	Security Logging and Monitoring Failuresies
10	Server-Side Request Forgery

SANS 25 Software Errors/Tests

#	SANS 25
1	Improper Restriction of Operations within the Bounds of a Memory Buffer
2	Improper Neutralization of Input During Web Page Generation ('XSS')
3	Improper Input Validation
4	Information Exposure
5	Out-of-bounds Read
6	Improper Neutralization of Special Elements used in an SQL Command (SQLi)
7	Use After Free
8	Integer Overflow or Wraparound
9	Cross-Site Request Forgery (CSRF)ies
10	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
11	Improper Neutralization of Special Elements used in an OS Command
12	Out-of-bounds Write
13	Improper Authentication
14	NULL Pointer Dereference
15	Incorrect Permission Assignment for Critical Resource
16	Unrestricted Upload of File with Dangerous Type
17	Improper Restriction of XML External Entity Reference
18	Improper Control of Generation of Code ('Code Injection')
19	Use of Hard-coded Credentials
20	Uncontrolled Resource Consumption
21	Missing Release of Resource after Effective Lifetime
22	Untrusted Search Path
23	Deserialization of Untrusted Data
24	Improper Privilege Management
25	Improper Certificate Validation

174 Other Test Cases

#	Test Performed	Typical Severity
1	OS Command Injection	High
2	SQL Injection (Second Order)	High
3	XML External Entity Injection	High
4	LDAP Injection	High
5	XPath Injection	High
6	XML Injection	High
7	ASP.NET Debugging Enabled	High
8	DoS Locking Customer Accounts	Medium
9	DoS Buffer Overflows	Medium
10	Storing too much data in session (DoS)	High
11	Writing user-provided data to disk (DoS)	High
12	HTTP Insecure methods available on Server	High
13	Out of band resource load (HTTP)	High
14	File path manipulation	High
15	Server-site JavaScript code injection	High
16	Perl code injection	High
17	Ruby code injection	High
18	Python code injection	High
19	Expression Language injection	High
20	Unidentified code injection	High
21	Server-side template injection	High
22	SSL injection	High
23	Stored XSS	High
24	HTTP response header injection	High
25	Reflected XSS	High
26	Client-side template injection	High
27	DOM-based XSS	High
28	Reflected DOM-based XSS	High
29	Stored DOM-based XSS	High
30	DOM-based JavaScript Injection	High
31	Reflected DOM-based JavaScript Injection	High
32	Stored DOM-based JavaScript Injection	High
33	Path-relative style sheet import	Information
34	Client-side SQLi (DOM-based)	High
35	Client-side SQLi (Reflected DOM-based)	High
36	Client-side SQLi (Stored DOM-based)	High
37	WebSocket Hijacking (DOM-based)	High
38	WebSocket Hijacking (Reflected DOM-based)	High
39	WebSocket Hijacking (Stored DOM-based)	High
40	Local Path Manipulation (DOM-based)	High
41	Local Path Manipulation (Reflected DOM)	High
42	Local Path Manipulation (Stored DOM-based)	High
43	Client-side XPATH Injection (DOM-based)	Low
44	Client-side XPATH Injection (Reflected DOM)	Low
45	Client-side XPATH Injection (Stored DOM)	Low

#	Test Performed	Typical Severity
46	Client-side JSON Injection (DOM-based)	Low
47	Client-side JSON Injection (Reflected DOM)	Low
48	Client-side JSON Injection (Stored DOM-based)	Low
49	Flash cross-domain policy	High
50	Cross-origin resource sharing	
51	Cross-origin resource sharing (arbitrary)	High
52	Cross-origin resource sharing (encrypted)	Low
53	Cross-origin resource sharing (all sub-domains)	Low
54	Cross-site Request Forgery (CSRF)	Medium
55	SMTP header injection	Medium
56	Cleartext submission of password	High
57	External service interaction (DNS)	High
58	External service interaction (HTTP)	High
59	External service interaction (SMTP)	Information
60	Referrer dependent response	Information
61	Spoofable client IP address	Information
62	User-agent dependent response	Information
63	Password returned in a later response	Medium
64	Password submitted using GET method	Low
65	Password returned in URL query string	Low
66	SQL statement in request parameter	Medium
67	Cross-domain POST	Information
68	ASP.NET ViewState without MAC Enabled	
69	XML entity expansion	Medium
70	Long redirection response	Information
71	Serialized object in HTTP message	
72	Duplicate cookies set	Information
73	WebSocket Hijacking (DOM-based)	High
74	WebSocket Hijacking (Reflected DOM-based)	High
75	WebSocket Hijacking (Stored DOM-based)	High
76	Local Path Manipulation (DOM-based)	High
77	Local Path Manipulation (Reflected DOM)	High
78	Local Path Manipulation (Stored DOM-based)	High
79	Client-side XPATH Injection (DOM-based)	Low
80	Client-side XPATH Injection (Reflected DOM)	Low
81	Client-side XPATH Injection (Stored DOM)	Low
82	Client-side JSON Injection (DOM-based)	Low
83	Client-side JSON Injection (Reflected DOM)	Low
84	Client-side JSON Injection (Stored DOM-based)	Low
85	Flash cross-domain policy	High
86	Cross-origin resource sharing	
87	Cross-origin resource sharing (arbitrary)	High
88	Cross-origin resource sharing (encrypted)	Low
89	Cross-origin resource sharing (all sub-domains)	Low
90	Cross-site Request Forgery (CSRF)	Medium
91	SMTP header injection	Medium
92	Cleartext submission of password	High
93	External service interaction (DNS)	High

#	Test Performed	Typical Severity
94	External service interaction (HTTP)	High
95	External service interaction (SMTP)	Information
96	Referrer dependent response	Information
97	Spoofable client IP address	Information
98	User-agent dependent response	Information
99	Password returned in a later response	Medium
100	Password submitted using GET method	Low
101	Password returned in URL query string	Low
102	SQL statement in request parameter	Medium
103	Cross-domain POST	Information
104	ASP.NET ViewState without MAC Enabled	
105	XML entity expansion	Medium
106	Long redirection response	Information
107	Serialized object in HTTP message	
108	Duplicate cookies set	Information
109	Input returned in response (stored)	Information
110	Input returned in response (reflected)	Information
111	Suspicious input transformation (reflected)	Information
112	Suspicious input transformation (stored)	Information
113	Open redirection (stored)	Low
114	Open redirection (reflected)	Medium
115	Open redirection (DOM-based)	Low
116	Open redirection (Stored DOM-based)	Low
117	Open redirection (Reflected DOM-based)	Medium
118	SSL cookie without secure flag set	Medium
119	Cookie scoped to parent domain	Low
120	Cross-domain referrer leakage	Information
121	Cross-domain script include	Information
122	Cookie without HTTPOnly flag set	
123	Session token in URL	
124	Password field with autocomplete enabled	
125	Password value set in cookie	Medium
126	Browser cross-site scripting disabled	Information
127	HTTP TRACE method is enabled	Information
128	Cookie manipulation (DOM-based)	Low
129	Cookie manipulation (reflected DOM-based)	Low
130	Cookie manipulation (DOM-based)	Low
131	Ajax request header manipulation (DOM-based)	Low
132	Ajax request header manipulation (reflected)	Low
133	Ajax request header manipulation (stored DOM)	Low
134	Denial of service (DOM-based)	Information
135	Denial of service (reflected DOM-based)	Information
136	Denial of service (stored DOM-based)	Low
137	HTML5 web message manipulation DOM-based	Information
138	HTML5 web message manipulation (reflected)	Information
139	HTML5 web message manipulation (stored DOM)	Information
140	HTML5 storage manipulation (DOM-based)	Information
141	HTML5 storage manipulation (reflected DOM)	Information

#	Test Performed	Typical Severity
142	HTML5 storage manipulation (stored DOM)	Information
143	Link manipulation (DOM-based)	Low
144	Link manipulation (reflected DOM-based)	Low
145	Link manipulation (stored DOM-based)	Low
146	Link manipulation (reflected & stored)	Information
147	Document domain manipulation (DOM-based)	Medium
148	Document domain manipulation reflected DOM	Medium
149	Document domain manipulation (stored DOM)	Medium
150	DOM data manipulation (DOM-based)	Information
151	CSS Injection (reflected & stored)	Medium
152	Client-side HTTP parameter pollution (reflected)	Low
153	Client-side HTTP parameter pollution (Stored)	Low
154	Form action hijacking (reflected)	Medium
155	Form action hijacking (stored)	Medium
156	Database connection string disclosed	Medium
157	Source code disclosure	
158	Directory listing	Information
159	Email addresses disclosed	Information
160	Private IP addresses disclosed	Information
161	Social security numbers disclosed	Information
162	Credit card numbers disclosed	Information
163	Private key disclosed	Information
164	Cacheable HTTPS response	Information
165	Base64 encoded data in parameter	Information
166	Multiple content types specified	Information
167	HTML does not specify charset	Information
168	HTML uses unrecognized charset	Information
169	Content type incorrectly stated	Low
170	Content type is not specified	Information
171	SSL certificate	Medium
172	Unencrypted communications	Low
173	Strict transport security not enforced	Low
174	Mixed content	Low