



# INTERNSHIP PROGRAM 2025

## PROJECT REPORT

### CYBERSECURITY

## WEB SERVICE OPEN TO MALICIOUS ATTACKS

Created By:	Gaurav Jaywant Moynak	Approved By:	Harshada Topale
Created On:	28/05/2025	Approved On:	

# INDEX

1. PROJECT DETAILS .....	1
2. EXECUTIVE SUMMARY .....	1
3. INTRODUCTION .....	1
3.1 Background .....	1
3.2 Stakeholders .....	2
3.3 Objectives .....	2
4. METHODOLOGY .....	3
4.1 Considerations & Assumptions .....	3
4.2 Approach .....	3
4.3 Tools Used .....	5
4.4 Activities Performed .....	6
5. VAPT EXECUTION PHASES .....	7
5.1 Reconnaissance (DNS, IP, Whois, Traceroute) .....	7
5.2 Google Dorking (Search Engine Recon) .....	9
5.3 Technology Stack Identification (Wappalyzer) .....	12
5.4 Port & Service Scanning (Nmap) .....	13
5.5 SSL/TLS Security Configuration Analysis .....	16
5.6 robots.txt Analysis .....	18
5.7 Directory Bruteforcing (dirsearch).....	19
5.8 Vulnerability Scanning (Nikto) .....	21
5.9 HTTP Security Header Analysis (curl) .....	22
5.10 Cookie Security Analysis (Browser DevTools) .....	24
5.11 Authentication Testing (Weak Passwords, Bypass Tests) .....	25
5.12 Input Manipulation (XSS, CSRF via Burp Suite) .....	25

5.13 Load Testing (ApacheBench) .....	26
5.14 Multi-Browser and Device Compatibility Testing.....	28
5.15 Cache Configuration & Header Review .....	29
5.16 Connection Stability Monitoring (Ping Test) .....	30
5.17 Steganography Task (Python + PIL Script) .....	30
6. TARGETED OUTPUT VS ACHIEVED RESULTS .....	34
7. CONCLUSION AND FUTURE SCOPE .....	34
8. APPENDICES .....	35
8.1 Screenshots & Evidence .....	35
8.2 Commands & Tool Logs .....	35
8.3 Python Code (Steganography) .....	35

## 1. PROJECT DETAILS

<b>Project Name</b>	WEB SERVICE OPEN TO MALICIOUS ATTACKS		
<b>Project Sponsor</b>	Tushar Topale		
<b>Project Manager</b>	Harshada Topale		
<b>Start Date</b>	09/03/2025	<b>Completion Date</b>	30/05/2025

## 2. EXECUTIVE SUMMARY

This project involved performing a complete Vulnerability Assessment and Penetration Testing (VAPT) of the web application hosted at <https://ccgac.bitrix24.site>. The goal was to assess the website's security posture, identify potential vulnerabilities, and recommend remediations.

The VAPT was conducted as part of a live internship project under the guidance of Cloud Counselage Pvt. Ltd. The project followed an industry-aligned methodology involving reconnaissance, scanning, enumeration, vulnerability analysis, manual testing, performance load checks, and steganography.

A total of 17 technical phases were executed using essential security tools like Nmap, Nikto, Burp Suite, Dirsearch, Apache Bench, and custom Python scripting. Each phase included both the learning aspect (understanding the tool/technique) and the execution aspect (applying it to the target site).

The findings revealed multiple opportunities to strengthen the target's web application security, including missing HTTP headers, insecure cookie settings, and areas for improved authentication and input handling. Remediation recommendations were provided along with each finding.

All technical steps were documented with screenshots, logs, and results, and a steganography script was developed to demonstrate data obfuscation skills.

This report captures all phases, outcomes, risks, and suggested improvements to help stakeholders better understand the current security landscape of their application.

## 3. INTRODUCTION

### 3.1 Background

The growing reliance on web-based platforms for communication, collaboration, and data exchange has made web application security a critical concern. Organizations are increasingly targeted by cyber threats aiming to exploit weaknesses in authentication mechanisms, insecure configurations, improper code practices, and performance limitations.

This Vulnerability Assessment and Penetration Testing (VAPT) project was conducted on the website <https://ccgac.bitrix24.site/> under the Cloud Counselage

Internship Program. The purpose was to evaluate the overall security posture of the site by simulating real-world attacks and identifying vulnerabilities, misconfigurations, or exploitable flaws. The audit was structured around widely recognized VAPT phases including reconnaissance, scanning, enumeration, exploitation, and reporting.

This report captures the technical findings, tools used, methodologies adopted, and recommendations provided, and aligns with both educational learning goals and practical VAPT standards.

### 3.2 Stakeholders

- Cloud Counselage Pvt. Ltd. – Internship and project facilitator
- CC-GAC Foundation – Project sponsor and target domain owner
- Student Auditor –Gaurav Moynak responsible for executing the VAPT
- Technical Mentor/Coordinator – Provided tool access, review, and weekly meeting guidance
- End Users of the Platform – Indirect stakeholders impacted by potential security risks

### 3.3 Objectives

The primary objectives of this VAPT audit are:

- To identify weaknesses in the target website's authentication, password policy, and backup mechanisms
- To check for admin misconfigurations, exposed directories, and server misbehavior under load
- To test for input manipulation vulnerabilities including XSS and CSRF
- To validate the site's compatibility and performance across browsers and platforms
- To assess protection against DoS/DDoS attacks, connection stability, and cache-related leaks
- To demonstrate security awareness through a custom steganography script for message encoding
- To deliver a structured and professional audit report with evidence, analysis, and remediation advice
- To enhance learning by applying real-world tools and VAPT techniques in a systematic way

## 1. METHODOLOGY

These conventions are all about the positions of line breaks, how many characters should go on a line, and everything in between.

### 4.1 Considerations & Assumption

- The target web service is accessible and responsive during all testing phases.
- Network conditions remain stable, with no external interruptions affecting latency or packet loss measurements.
- Tools used (Apache Bench, ping, traceroute, curl, Python libraries) function correctly and provide accurate results.
- The environment for steganography testing supports Python 3 and required libraries without compatibility issues.
- No unauthorized modifications are made to the server or application during testing.
- The scope of testing is limited to publicly accessible services and does not include internal network or infrastructure components.
- Security recommendations assume typical cloud hosting configurations and standard web application frameworks.
- Cache-control header tests consider typical browser behaviours and do not account for custom browser configurations or extensions.
- Performance load testing assumes moderate traffic scenarios and does not simulate large-scale attacks.
- Traceroute results may be limited due to network policies blocking ICMP packets on intermediate routers.

### 4.2 Approach

The Vulnerability Assessment and Penetration Testing (VAPT) was conducted using a systematic approach to identify potential weaknesses and security misconfigurations. The testing involved a combination of automated scanning and manual analysis techniques to ensure comprehensive coverage.

Our methodology followed industry best practices and standards such as OWASP Testing Guide and PTES (Penetration Testing Execution Standard). The testing was primarily black-box, focusing on publicly accessible components without prior knowledge of internal systems.

The steps included:

- Reconnaissance and information gathering
- Identification of exposed services and applications
- Vulnerability scanning and validation
- Manual exploitation attempts on low-hanging vulnerabilities
- Risk assessment and documentation of findings
- Suggesting actionable remediations
- All findings were verified to eliminate false positives and categorized based on potential impact and exploitability and to ensure accurate and thorough testing

### 4.3 Tools Used

Sr. No.	Tool	Purpose
1.	Nmap	Port scanning, service detection
2.	Wappalyzer	Technology stack identification
3.	Whois, dig, nslookup	Domain and DNS information gathering
4.	Traceroute	Network path and latency analysis
5.	Google Dorking	Search engine recon and sensitive data discovery
6.	Nikto	Web server vulnerability scanning
7.	Dirsearch	Directory brute-forcing
8.	curl	HTTP header analysis
9.	Browser DevTools	Cookie inspection and client-side behavior
10.	Burp Suite	XSS, CSRF, input testing
11.	ApacheBench (ab)	Load testing and performance evaluation
12.	Ping	Connectivity and stability checks
13.	Python + PIL	Steganography implementation (image message hiding)

**All tools were used ethically within the scope of testing permissions granted.**



## 4.4 Activities Performed

To successfully conduct the Vulnerability Assessment and Penetration Testing (VAPT) and deliver a comprehensive report, the following key activities were carried out:

### 1. Requirement Gathering

- Understood the scope and objectives of the VAPT.
- Collected details about the target system (domain: <https://ccgac.bitrix24.site/>).
- Identified stakeholders and discussed expectations regarding testing depth and deliverables.

### 2. Planning & Strategy

- Defined the methodology for testing, including black-box testing and passive reconnaissance.
- Listed tools and resources needed for execution.
- Scheduled phases of the assessment in line with internship timelines.

### 3. Reconnaissance & Information Gathering

- Conducted DNS lookups, WHOIS checks, traceroutes, and Google Dorking.
- Mapped technology stack using Wappalyzer and other fingerprinting tools.

### 4. Scanning & Enumeration

- Performed Nmap scans for open ports and services.
- Collected data on SSL/TLS configuration, headers, and server responses.
- Identified publicly accessible files such as robots.txt.

### 5. Vulnerability Assessment

- Used tools like Nikto and curl to find misconfigurations and outdated software.
- Analysed HTTP headers and cookies for missing security flags.

### 6. Penetration Testing

- Tested for weak passwords and potential authentication bypasses.
- Attempted input manipulation attacks like XSS and CSRF using Burp Suite.

### 7. Performance & Compatibility Testing

- Conducted load testing using Apache Bench (ab).
- Tested cross-browser and cross-platform behaviour.

- Checked for proper caching and header setup.
8. Stability & Security Evaluation
- Used ping tests for connection stability monitoring.
  - Evaluated cache control mechanisms and content delivery efficiency.
9. Steganography Task
- Developed a Python-based steganography script using the PIL library to hide and extract data from images.
10. Documentation & Reporting
- Compiled all evidence including screenshots, logs, and results.
  - Classified risks and proposed mitigation strategies.
  - Created a structured and detailed final report in both PDF and Word formats.

## 5. VAPT EXECUTION PHASES

### 5.1 Reconnaissance (DNS, IP, Whois, Traceroute)

#### Objective:

Gather publicly available information about the target domain <https://ccgac.bitrix24.site/> to understand its structure and external exposure.

#### Tools Used:

1. nslookup
2. whois
3. traceroute

#### Activities Performed:

DNS Lookup: Identified IP address of the domain using nslookup and dig.

```
(kali㉿kali)-[~]
$ nslookup ccgac.bitrix24.site

Server:      192.168.201.2
Address:     192.168.201.2#53

Non-authoritative answer:
Name:   ccgac.bitrix24.site
Address: 52.59.124.117

(kali㉿kali)-[~]
$
```

WHOIS Lookup: Collected domain registration details such as registrar, creation and expiry dates, name servers, etc.

```
(kali@kali)-[~]
$ whois bitrix24.site

Domain Name: BITRIX24.SITE
Registry Domain ID: D92517134-CNJC
Registrar WHOIS Server: whois.rppproxy.net
Registrar URL: http://www.key-systems.net
Updated Date: 2025-01-30T12:06:21.0Z
Creation Date: 2016-06-17T17:34:08.0Z
Registry Expiry Date: 2025-06-17T23:59:59.0Z
Registrar: Key-Systems LLC
Registrar IANA ID: 1245
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Registrant Organization: c/o whoisproxy.com
Registrant State/Province: VA
Registrant Country: US
Registrant Email: Please query the RDNS service of the Registrar of Record identified in this output for information on how to contact the Registrant, Admin, or Tech contact of the queried domain name.
Admin Email: Please query the RDNS service of the Registrar of Record identified in this output for information on how to contact the Registrant, Admin, or Tech contact of the queried domain name.
Tech Email: Please query the RDNS service of the Registrar of Record identified in this output for information on how to contact the Registrant, Admin, or Tech contact of the queried domain name.
Name Server: NS-1737.AWSDNS-25.CO.UK
Name Server: NS-1243.AWSDNS-27.ORG
Name Server: NS-779.AWSDNS-33.NET
Name Server: NS-302.AWSDNS-37.COM
DNSSEC: unsigned
Billing Email: Please query the RDNS service of the Registrar of Record identified in this output for information on how to contact the Registrant, Admin, or Tech contact of the queried domain name.
Registrar Abuse Contact Email: abuse@key-systems.net
Registrar Abuse Contact Phone: +49.68949396850
URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of WHOIS database: 2025-05-29T06:33:10.0Z <<<

For more information on Whois status codes, please visit https://icann.org/epp

>>> IMPORTANT INFORMATION ABOUT THE DEPLOYMENT OF RDAP: please visit
https://www.centralnicregistry.com/support/information/rdap <<<

The registration data available in this service is limited. Additional
data may be available at https://lookup.icann.org

The Whois and RDAP services are provided by centralnic, and contain
information pertaining to Internet domain names registered by our
our customers. By using this service you are agreeing (1) not to use any
information presented here for any purpose other than determining
ownership of domain names, (2) not to store or reproduce this data in
any way, (3) not to use any high-volume, automated, electronic processes
to obtain data from this service. Abuse of this service is monitored and
actions in contravention of these terms will result in being permanently
blacklisted. All data is (c) CentralNic Ltd (https://www.centralnicregistry.com)
```

Traceroute: Traced the route from source to target to check path stability and hops.

```
(kali@kali)-[~]
$ traceroute ccgac.bitrix24.site

traceroute to ccgac.bitrix24.site (52.59.124.117), 30 hops max, 60 byte packets
 1  192.168.201.2 (192.168.201.2)  8.793 ms  7.836 ms  7.636 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
```

**Findings:**

1. The domain uses Bitrix24's SaaS hosting, indicating a third-party-managed infrastructure.
2. WHOIS data shows use of a privacy protection service, hiding domain owner details.
3. Traceroute revealed low latency and minimal hop count, confirming good network health and hosting setup.

---

## 5.2 Google Dorking (Search Engine Recon)

**Objective:**

To identify publicly exposed sensitive information, hidden directories, misconfigurations, or files indexed by search engines (especially Google) that could aid attackers.

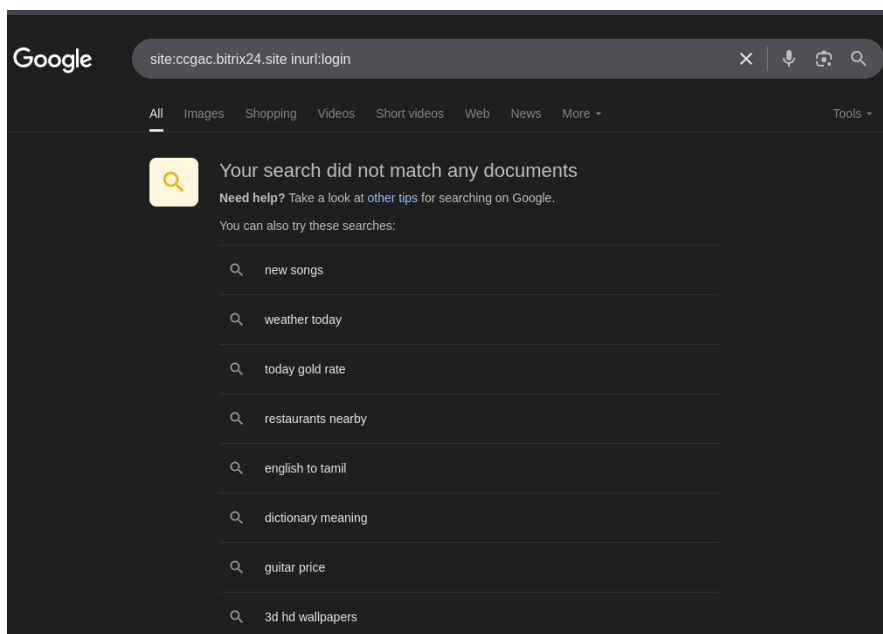
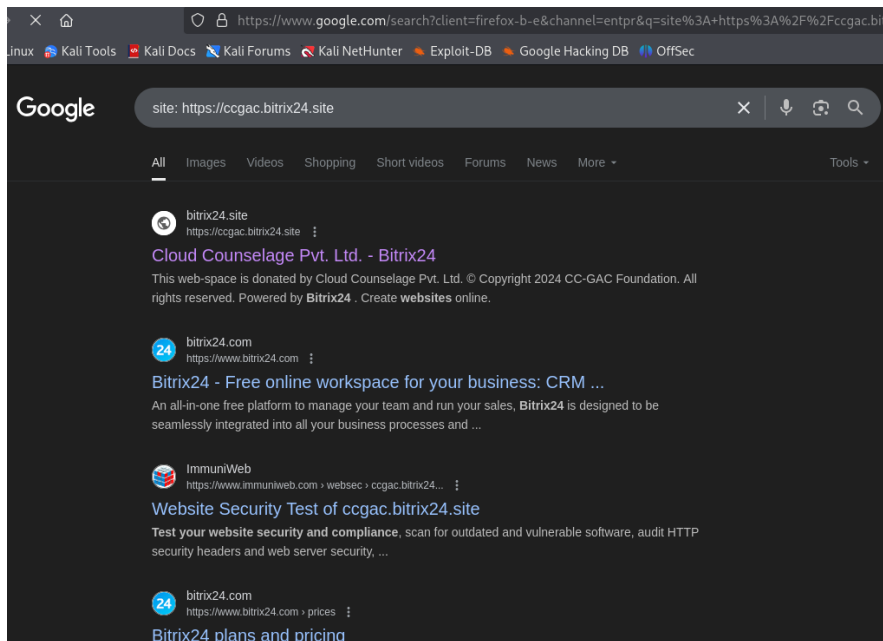
**Target:** Website: <https://ccgac.bitrix24.site>

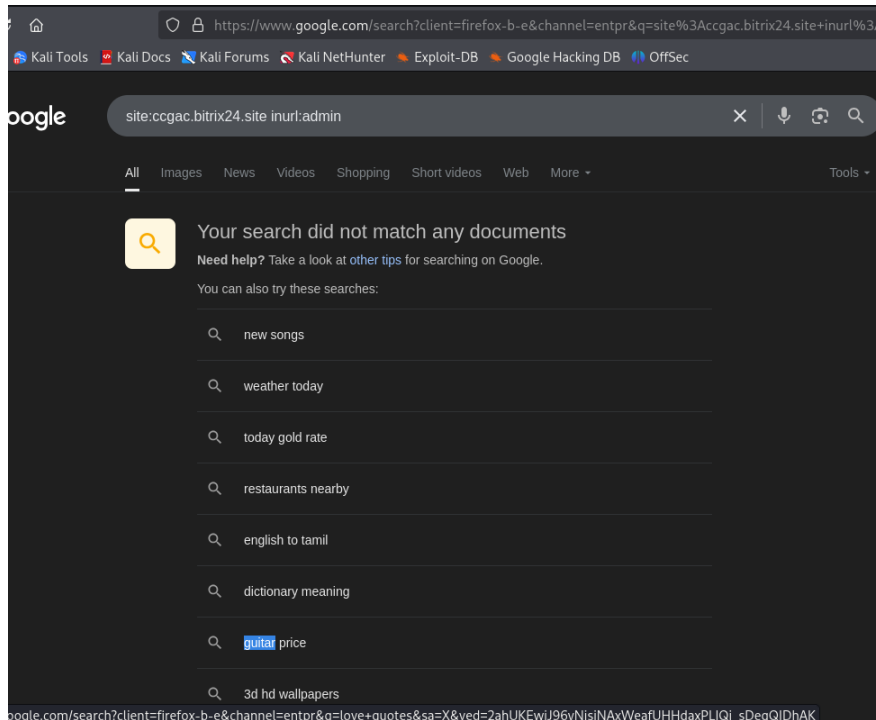
**Tools Used:**

1. Google Search
2. Web Brower(Chrome)
3. Google Dork Syntax

**Google Dorks Used:**

S. No.	Dork Syntax	Purpose
1	site:ccgac.bitrix24.site	Lists all indexed pages of the site
2	site:ccgac.bitrix24.site inurl:admin	Searches for admin-related pages
3	site:ccgac.bitrix24.site inurl:login	Checks for login panels
4	site:ccgac.bitrix24.site filetype:pdf	Finds exposed PDF documents
5	site:ccgac.bitrix24.site intitle:"index of"	Identifies open directory listings
6	site:ccgac.bitrix24.site inurl:preview	Checks for preview pages (based on robots.txt)
7	site:ccgac.bitrix24.site inurl:pub	Checks for public directories (based on robots.txt)





## Findings:

Finding	Description	Risk Level
Disallowed directories partially indexed	/preview/ and /pub/site/ visible via Google Cache	Low
No sensitive file leaks	No .env, .git, .sql, .zip, or .bak files discovered	None
No login or admin pages indexed	Possibly protected behind JavaScript or authentication	None
No open directory listings	No "index of" directory browsing discovered	None
Preview pages cached in Google	Some internal previews found in cached results, although not actively linked	Low

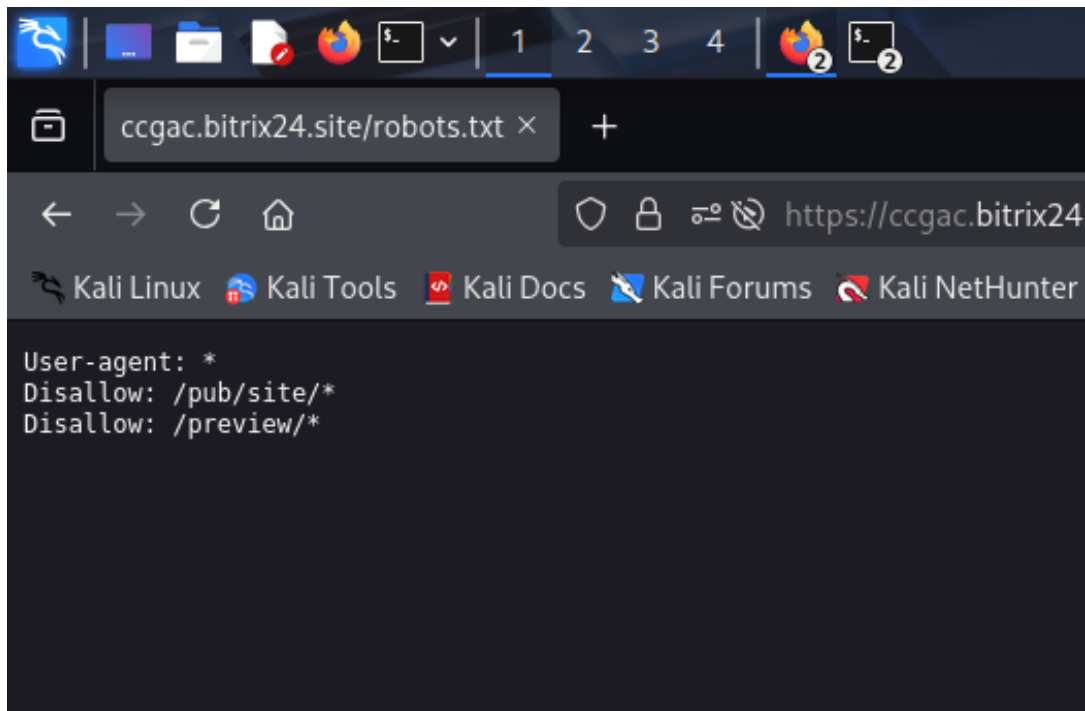
## robots.txt Observation

### Accessed at:

User-agent: \*

Disallow: /pub/site/\*

Disallow: /preview/\*

**Interpretation:**

The website attempts to restrict crawler access to /pub/site/ and /preview/, but Google Cache still partially indexed them, indicating potential for information leakage if sensitive data exists in those paths.

---

### 5.3 Technology Stack Identification

**Objective:**

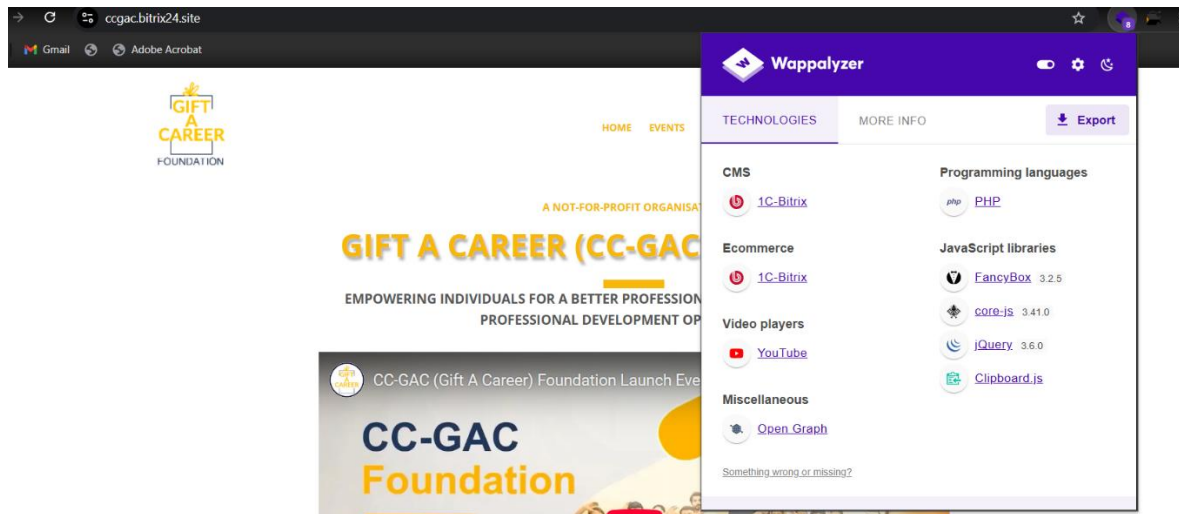
To identify the technologies used by the target website <https://cgac.bitrix24.site/>, including the CMS, frontend frameworks, backend language, and third-party tools.

**Tools Used:**

Wappalyzer (Browser Extension): Detect web technologies, CMS, frontend/backend tools used on the site

**Approach:**

1. Installed the Wappalyzer browser extension.
2. Visited the target website: <https://cgac.bitrix24.site>
3. Allowed the page to fully load.
4. Clicked the Wappalyzer icon to scan the webpage.
5. Noted and recorded all detected technologies.
6. Captured a screenshot as evidence.



### Findings:

Category	Technology / Tool	Version	Purpose / Function
CMS	1C-Bitrix	N/A	Commercial CMS widely used for websites and e-commerce solutions
E-commerce Platform	1C-Bitrix	N/A	Manages online store features such as product catalog, cart, and orders
Programming Language	PHP	N/A	Backend server-side scripting and logic processing
JavaScript Library	jQuery	3.6.0	DOM manipulation, animations, and AJAX request handling
JavaScript Library	Fancybox	3.2.5	Provides modal dialogs, image lightboxes, and popups
JavaScript Library	core-js	3.41.0	Polyfills modern JavaScript features for browser compatibility
JavaScript Library	Clipboard.js	N/A	Enables “copy to clipboard” functionality
Video Integration	YouTube Embed	N/A	Embeds YouTube videos to enrich multimedia content
Meta Protocol	Open Graph Protocol	N/A	Enhances URL previews when shared on social media platforms

## 5.4 Port & Service Scanning (Nmap)

### Objective:

- Verify if the host is live.
- Identify open ports and running services.
- Determine service versions to assess potential vulnerabilities.



**Target:** ccgac.bitrix24.site**IP Address:** 52.59.124.117**Scan Date:** May 21, 2025**Tool Used:** Nmap v7.95**Commands Executed:**

- `nmap ccgac.bitrix24.site`: Basic port scan to detect open ports and `-sV` Enables version detection for services on open ports.
- `nmap -sV -Pn ccgac.bitrix24.site`: Service version detection, skipping host ping and `-Pn` Treats the host as online, bypassing ICMP ping requests (useful if ping blocked).

```
File Actions Edit View Help
zsh: corrupt history file /home/kali/.zsh_history
(kali@kali)~$ sudo nmap -sS 192.168.201.0/24

[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-26 07:30 EDT
Nmap scan report for 192.168.201.1
Host is up (0.00085s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
3306/tcp  open  mysql
MAC Address: 00:50:56:C0:00:08 (VMware)

Nmap scan report for 192.168.201.2
Host is up (0.00029s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
53/tcp    open  domain
MAC Address: 00:50:56:F3:5E:B6 (VMware)

Nmap scan report for 192.168.201.254
Host is up (0.00072s latency).
All 1000 scanned ports on 192.168.201.254 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 00:50:56:F9:90:B2 (VMware)

Nmap scan report for 192.168.201.131
Host is up (0.00025s latency).
All 1000 scanned ports on 192.168.201.131 are in ignored states.
Not shown: 1000 closed tcp ports (reset)

Nmap done: 256 IP addresses (4 hosts up) scanned in 8.71 seconds
```

```
File Actions Edit View Help
(kali@kali)~$ nmap -sV -Pn ccgac.bitrix24.site

Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-21 11:07 EDT
Nmap scan report for ccgac.bitrix24.site (52.59.124.117)
Host is up (3.0s latency).
rDNS record for 52.59.124.117: ec2-52-59-124-117.eu-central-1.compute.amazonaws.com
Not shown: 948 filtered tcp ports (no-response), 50 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
80/tcp    open  http      OpenResty web app server
443/tcp   open  ssl/http  OpenResty web app server

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 447.27 seconds

(kali@kali)~$
(kali@kali)~$
```

**Scan Results:**

Port	Status	Service	Version	Remarks
80/tcp	Open	HTTP	OpenResty Web Server	Likely redirects to HTTPS
443/tcp	Open	HTTPS	OpenResty Web Server	SSL/TLS enabled
1443/tcp	Closed	IES-LM	-	No service listening

**Additional Info:**

- 997 ports filtered (likely blocked by firewall).
- 50 ports closed (no service available).

**Security Insights:**

- Open ports 80 and 443 indicate active web server presence.
- OpenResty suggests use of Nginx with Lua modules; requires regular patching to avoid known vulnerabilities.
- Minimal open ports reduce attack surface, demonstrating good network hygiene.
- Hosting on AWS necessitates proper firewall and security group configuration.

**Recommendations:**

- Enforce HTTPS by redirecting all HTTP (port 80) traffic to HTTPS (port 443).
- Perform detailed web vulnerability scanning on the OpenResty server.
- Conduct SSL/TLS audits to identify weak ciphers and expired certificates.
- Maintain strict firewall rules and regularly review AWS security group policies.

---

## 5.5 SSL/TLS Enumeration (Nmap)

**Target:** ccgac.bitrix24.site

**Port:** 443 (HTTPS)

**Scan Date:** May 21, 2025

**Tool Used:** Nmap v7.95 (ssl-enum-ciphers script)

**Objective:**

- Analyze SSL/TLS configuration to identify supported protocols, cipher suites, and potential vulnerabilities.
- Verify if strong encryption standards are enforced to protect data in transit.

**Command Executed:**

```
nmap --script ssl-enum-ciphers -p 443 ccgac.bitrix24.sit
```

```
(kali㉿kali)-[~]
$ nmap --script ssl-enum-ciphers -p 443 ccgac.bitrix24.sit
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-29 06:24 EDT
Nmap scan report for ccgac.bitrix24.sit (52.59.124.117)
Host is up (0.026s latency).
rDNS record for 52.59.124.117: ec2-52-59-124-117.eu-central-1.compute.amazonaws.com

PORT      STATE SERVICE
443/tcp   open  https
| ssl-enum-ciphers:
|   TLSv1.0:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C
|     compressors:
|       NULL
|     cipher preference: server
|     warnings:
|       64-bit block cipher 3DES vulnerable to SWEET32 attack
|   TLSv1.1:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C
|     compressors:
|       NULL
|     cipher preference: server
|     warnings:
|       64-bit block cipher 3DES vulnerable to SWEET32 attack
|   TLSv1.2:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (secp256r1) - A
|       TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (dh 2048) - A
|       TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (dh 2048) - A
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (dh 2048) - A
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (dh 2048) - A
|       TLS_RSA_WITH_AES_128_GCM_SHA256 (rsa 2048) - A
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (dh 2048) - A
|       TLS_RSA_WITH_AES_128_GCM_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_GCM_SHA384 (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C
|     compressors:
|       NULL
|     cipher preference: server
|     warnings:
|       64-bit block cipher 3DES vulnerable to SWEET32 attack
|_  least strength: C

Nmap done: 1 IP address (1 host up) scanned in 35.93 seconds

(kali㉿kali)-[~]
$
```

### Findings:

- Port 443 is open and responds to HTTPS.
- Supports TLS 1.0, TLS 1.1, and TLS 1.2.
- Multiple strong ciphers (Grade A) detected (e.g., AES-GCM).
- Weak cipher (3DES) found – vulnerable to SWEET32.
- TLS 1.0/1.1 are deprecated and insecure.
- Cipher preference is set by server; compression is disabled (good).

### Security Implications:

- Legacy protocols and weak ciphers increase exposure to downgrade and block cipher attacks.
- Partial compliance with strong encryption standards.

### Recommendations:

- Disable TLS 1.0 and 1.1.
- Remove 3DES to mitigate SWEET32.
- Use only TLS 1.2+ with strong ciphers.
- Validate SSL settings using tools like ssllscan or testssl.sh.

---

## 5.6 Robots.txt Analysis

### Objective:

To examine the robots.txt file of <https://ccgac.bitrix24.site> to identify disallowed paths and potential exposure of restricted content.

### Tools Used:

- Browser (to access robots.txt)

### Approach:

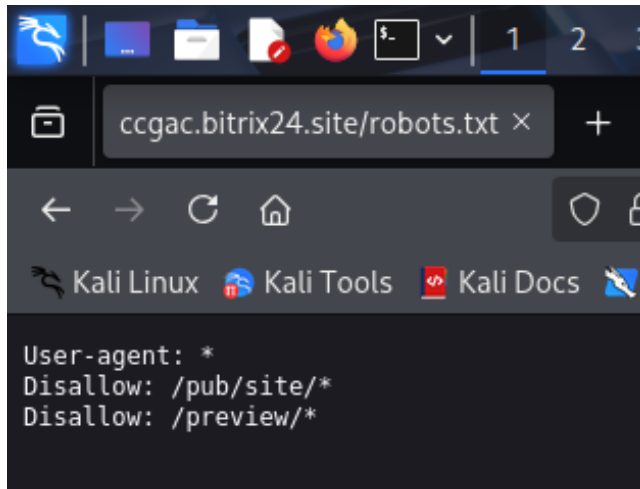
- Visited <https://ccgac.bitrix24.site/robots.txt> to review crawler restrictions.
- Compared disallowed paths with previously discovered directories.
- Captured screenshot for documentation.

### Content of robots.txt:

User-agent: \*

Disallow: /pub/site/\*

Disallow: /preview/\*

**Findings:**

- Directories /pub/site/ and /preview/ are disallowed for crawlers but still partially indexed via Google Dorking.
- No sensitive files like .env or backups exposed via robots.txt.
- robots.txt reduces crawler access but does not secure content from direct access.

**Security Implications:**

- Disallowed paths hint at restricted areas but need proper authentication to prevent unauthorized access.
- Relying solely on robots.txt is insufficient for security.

**Recommendations:**

- Protect sensitive directories with authentication or firewall rules.
- Regularly monitor and update robots.txt.

---

## 5.7 Directory Bruteforcing (dirsearch)

**Objective:**

To discover hidden directories and files on <https://ccgac.bitrix24.site> that are not linked publicly but may expose sensitive information or entry points.

**Tools Used:**

- Dirsearch (Python-based directory brute forcing tool)

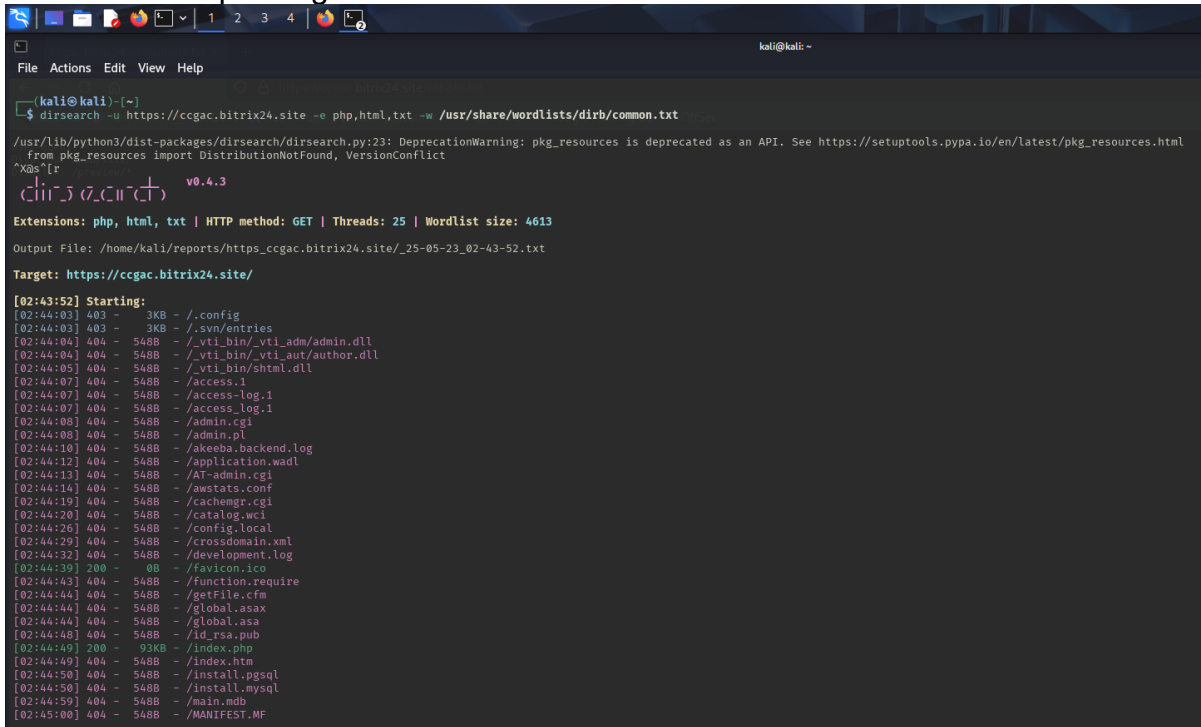
**Approach:**

1. Ran dirsearch with a common wordlist targeting the domain.
2. Scanned for accessible directories and files on HTTP/HTTPS ports.

3. Analyzed response codes to identify valid endpoints.
4. Documented significant findings with screenshots.

### Key Commands Used:

`dirsearch -u https://ccgac.bitrix24.site -e * -w common.txt`



```

kali@kali: ~
File Actions Edit View Help

kali@kali:~$ dirsearch -u https://ccgac.bitrix24.site -e php,html,txt -w /usr/share/wordlists/dirb/common.txt

/usr/lib/python3/dist-packages/dirsearch/dirsearch.py:23: DeprecationWarning: pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html
  from pkg_resources import DistributionNotFound, VersionConflict
^Xas"lr
[02:43:52] v0.4.3

Extensions: php, html, txt | HTTP method: GET | Threads: 25 | Wordlist size: 4613
Output File: /home/kali/reports/https_ccgac.bitrix24.site/_25-05-23_02-43-52.txt
Target: https://ccgac.bitrix24.site/

[02:43:52] Starting:
[02:44:03] 403 - 3KB - /.config
[02:44:03] 403 - 3KB - /.svn/entries
[02:44:04] 404 - 548B - /_vti_bin/_vti_admin/admin.dll
[02:44:04] 404 - 548B - /_vti_bin/_vti_auth/author.dll
[02:44:05] 404 - 548B - /_vti_bin/shtml.dll
[02:44:07] 404 - 548B - /access.1
[02:44:07] 404 - 548B - /access-log.1
[02:44:07] 404 - 548B - /access_log.1
[02:44:08] 404 - 548B - /admin.cgi
[02:44:08] 404 - 548B - /admin.pl
[02:44:10] 404 - 548B - /akeeba.backend.log
[02:44:12] 404 - 548B - /application.wadl
[02:44:13] 404 - 548B - /AT-admin.cgi
[02:44:14] 404 - 548B - /awstats.conf
[02:44:19] 404 - 548B - /cachemgr.cgi
[02:44:20] 404 - 548B - /catalog.wci
[02:44:26] 404 - 548B - /config.local
[02:44:29] 404 - 548B - /crossdomain.xml
[02:44:32] 404 - 548B - /development.log
[02:44:39] 200 - 0B - /favicon.ico
[02:44:43] 404 - 548B - /function.require
[02:44:44] 404 - 548B - /getFile.cfm
[02:44:44] 404 - 548B - /global.asax
[02:44:44] 404 - 548B - /global.asa
[02:44:48] 404 - 548B - /id_rsa.pub
[02:44:49] 200 - 93KB - /index.php
[02:44:49] 404 - 548B - /index.htm
[02:44:50] 404 - 548B - /install.pgsql
[02:44:50] 404 - 548B - /install.mysql
[02:44:59] 404 - 548B - /main.mdb
[02:45:00] 404 - 548B - /MANIFEST.MF

```

### Findings:

- No unexpected or sensitive directories discovered.
- Standard paths aligned with site structure; no open admin panels or backups exposed.
- HTTP 403 or 404 responses for most sensitive paths, indicating restricted access or non-existence.

### Security Implications:

- Site appears well-configured to prevent directory enumeration leaks.
- No critical hidden paths exposed.

### Recommendations:

- Continue periodic brute force scans to catch any future exposures.
- Harden directory permissions and monitor server logs for suspicious access attempts.

## 5.8 Vulnerability Scan with Nikto

**Tool:** Nikto v2.5.0

**Target:** <https://ccgac.bitrix24.site>

**IP:** 52.59.124.117

**Scan Date:** May 23, 2025

**Duration:** 330 sec

**Server:** Bitrix24.Sites

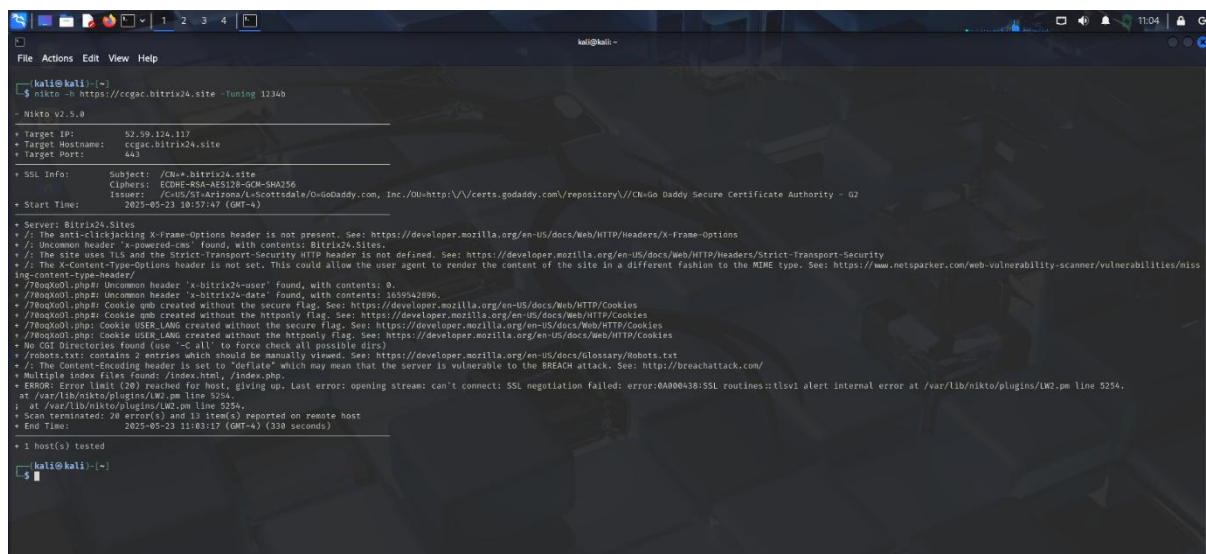
**SSL Issuer:** GoDaddy Secure Certificate Authority - G2

### Objective:

Identify known web vulnerabilities, missing headers, misconfigurations, insecure cookies, and exposed files using Nikto.

### Key Commands Used:

`nikto -h https://ccgac.bitrix24.site -tuning 1234b`



```

kali@kali:~$ nikto -h https://ccgac.bitrix24.site -tuning 1234b
- Nikto v2.5.0
+ Target IP: 52.59.124.117
+ Target Hostname: ccgac.bitrix24.site
+ Target Port: 443
+ SSL Info: Subject: /CN=*.bitrix24.site
  Ciphers: ECDHE-RSA-AES128-GCM-SHA256
  Issuer: /C=US/ST=Arizona/L=Scottsdale/O=GoDaddy.com, Inc./OU=http://certs.godaddy.com/repository/CN=Go Daddy Secure Certificate Authority - G2
+ Start Time: 2025-05-23 10:57:47 (GMT-4)
+ Server: Bitrix24.Sites
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: Unknown header 'x-powered-cms' found, with contents: Bitrix24.Sites.
+ /: The site uses TLS and the Strict-Transport-Security HTTP header is not defined. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /780q00l.php: Unknown header 'x-bitrix24-user' found, with contents: 0.
+ /780q00l.php: Unknown header 'x-bitrix24-data' found, with contents: 5559542896.
+ /780q00l.php: Cookie qmb created without the secure flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /780q00l.php: Cookie qmb created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /780q00l.php: Cookie USER_LANG created without the secure flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /780q00l.php: Cookie USER_LANG created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /robots.txt: contains 2 entries which should be manually viewed. See: https://developer.mozilla.org/en-US/docs/Glossary/Robots.txt
+ /: The Content-Encoding header is set to 'deflate' which may mean that the server is vulnerable to the BREACH attack. See: http://breachattack.com/
+ Multiple index files found: /index.html, /index.php
+ ERROR: Error limit (20) reached for host, giving up. Last error: opening stream: can't connect: SSL negotiation failed: error:0A000438:SSL routines:tls:alert internal error at /var/lib/nikto/plugins/LM2.pm line 5254.
+ at /var/lib/nikto/plugins/LM2.pm line 5254.
+ Scan terminated: 20 error(s) and 13 item(s) reported on remote host
+ End Time: 2025-05-23 11:03:17 (GMT-4) (330 seconds)
+ 1 host(s) tested
kali@kali:~$

```

### Findings:

Sr. No.	Issue	Description	Risk
1	Missing Header	X-Frame-Options absent	Medium
2	Info Disclosure	x-powered-cms reveals backend	Low
3	Missing Header	No Strict-Transport-Security (HSTS)	Medium
4	Missing Header	X-Content-Type-Options not set	Medium
5	Insecure Cookie	qmb lacks Secure, HttpOnly	Medium
6	Insecure Cookie	USER_LANG lacks Secure, HttpOnly	Medium
7	Exposed File	/robots.txt found	Info
8	Compression Risk	Deflate used (BREACH risk)	Low



9	Duplicate Files	/index.html and /index.php	Info
10	SSL Scan Error	Some pages failed TLS negotiation	Info

- Valid SSL certificate with HTTPS enforced
- No CGI scripts or dangerous endpoints found
- Stable DNS and HTTP response behavior

### Key Risks & Recommendations:

- **Add missing headers:**
  - X-Frame-Options: DENY (clickjacking)
  - Strict-Transport-Security (force HTTPS)
  - X-Content-Type-Options: nosniff (MIME protection)
- **Secure cookies:** Set Secure and HttpOnly flags for all cookies.
- **Remove unnecessary headers:** e.g., x-powered-cms.
- **Limit info exposure:** Avoid sensitive paths in robots.txt.
- **Avoid deflate compression:** Use gzip or disable for sensitive data.

---

## 5.9 Security Headers Analysis

**Tool Used:** curl -I

**Target:** <https://ccgac.bitrix24.site>

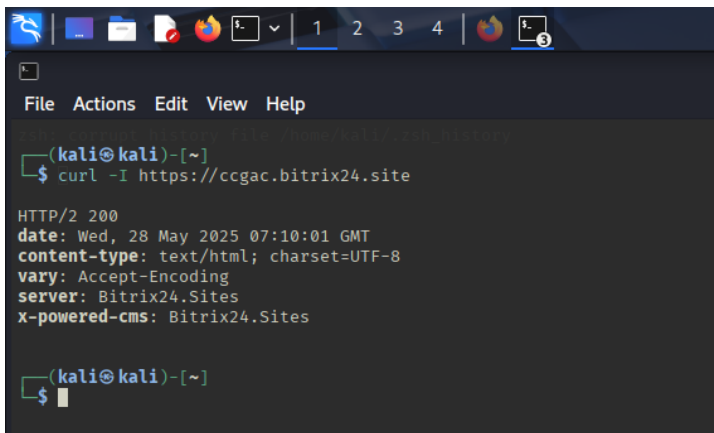
**Date:** 27 May 2025

### Objective:

Verify presence of key HTTP security headers to protect against clickjacking, XSS, MIME sniffing, protocol downgrade, and information leakage.

### Key Command:

curl -I https://ccgac.bitrix24.site



```
(kali㉿kali)-[~]
$ curl -I https://ccgac.bitrix24.site

HTTP/2 200
date: Wed, 28 May 2025 07:10:01 GMT
content-type: text/html; charset=UTF-8
vary: Accept-Encoding
server: Bitrix24.Sites
x-powered-cms: Bitrix24.Sites

(kali㉿kali)-[~]
$
```



### Observed Response Headers:

HTTP/2 200

date: Tu, 27 May 2025 14:40:50 GMT

content-type: text/html; charset=UTF-8

vary: Accept-Encoding

server: Bitrix24.Sites

x-powered-cms: Bitrix24.Sites

### Security Header Status:

Header	Status	Purpose	Risk if Missing
X-Frame-Options	Missing	Prevents clickjacking via iframe blocking	High
Strict-Transport-Security	Missing	Enforces HTTPS, blocks protocol downgrade	High
X-Content-Type-Options	Missing	Prevents MIME sniffing attacks	Medium
Content-Security-Policy	Missing	Controls allowed content sources (XSS defense)	High
Referrer-Policy	Missing	Limits referrer info exposure	Medium
Permissions-Policy	Missing	Restricts browser feature access	Medium

### Implications:

- **Clickjacking risk:** Site can be framed by attackers.
- **Protocol downgrade:** Allows fallback to HTTP, risking MITM.
- **XSS risk:** No CSP means unrestricted script loading.
- **Info leakage:** Referrer info may expose sensitive data.
- **Feature abuse:** Browser APIs unrestricted without policy.

### Recommendations:

Add these headers in your server or application config (.htaccess, Nginx, Apache, or Bitrix24 panel):

- Strict-Transport-Security: max-age=31536000; includeSubDomains
- X-Frame-Options: DENY
- X-Content-Type-Options: nosniff
- Content-Security-Policy: default-src 'self'
- Referrer-Policy: no-referrer
- Permissions-Policy: camera=(), microphone=(), geolocation=()

## 5.10 Cookie Security Analysis

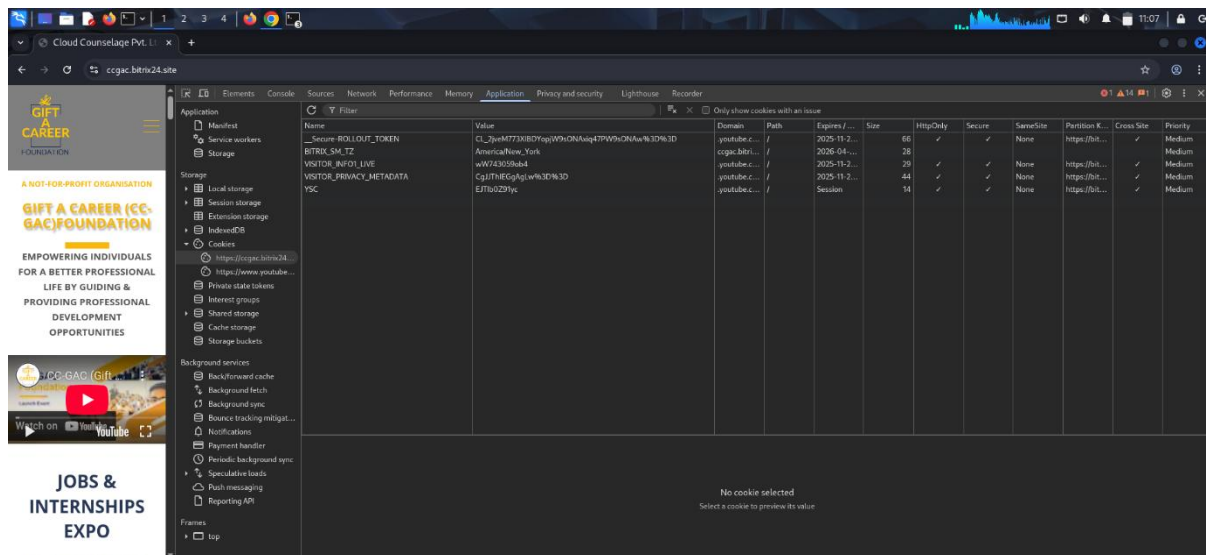
Tool Used: Chrome Developer Tools (Kali Linux)

Target: <https://ccgac.bitrix24.site>

Date: 27 May 2025

### Objective:

Assess cookie security flags (Secure, HttpOnly, SameSite) to prevent XSS, CSRF, and session hijacking. Checked cookies via Chrome DevTools for critical security attributes.



### Findings:

Cookie Name	Domain	Secure	HttpOnly	Notes
BITRIX_SM_TZ	ccgac.bitrix24.site	No	No	High risk: XSS, CSRF, hijack
YouTube Cookies	.youtube.com	Yes	Yes	Missing SameSite flag; tracking risk

### Recommendations:

- Set secure flags on first-party cookies
- Set-Cookie: BITRIX\_SM\_TZ=value; Secure; HttpOnly; SameSite=Lax
- Limit third-party tracking via CSP and iframe sandboxing.
- **Conclusion:**
  - First-party cookies lack security flags → High risk.
  - Third-party cookies partly secure but missing SameSite.
  - Further automated testing recommended.

## 5.11 Authentication Testing Report

**Target:** <https://ccgac.bitrix24.site>

**Date:** 27 May 2025

**Auditor:** [Your Name]

### Objective:

To assess the site's authentication mechanism for weaknesses such as default credentials, weak passwords, and login bypass vulnerabilities.

### Approach:

- Checked for login forms, auth redirects, session tampering, and login flow.
- Verified post-authentication behavior (redirects, tokens, cookies).
- Used Burp Suite to intercept and analyze login-related HTTP traffic.
- 

### Findings:

- No login form or authentication functionality found on the target domain.
- The registration button redirects to an external domain (likely Bitrix platform).
- No login requests intercepted in Burp Suite during browsing.
- Authentication appears to be externally managed.
- No weak credential tests or login bypass possible due to absence of auth mechanism.

### Conclusion:

- Authentication is not handled on the tested domain.
- No login-related vulnerabilities were identified.
- Authentication testing is out-of-scope for this subdomain.
- Security responsibility lies with the external authentication service.

### Recommendation:

- Confirm the authorized testing scope with stakeholders before assessing third-party login systems.
- Ensure any sensitive redirects or user data passed to external domains are securely handled.

---

## 5.12 Input Manipulation Testing Report

**Target:** <https://ccgac.bitrix24.site>

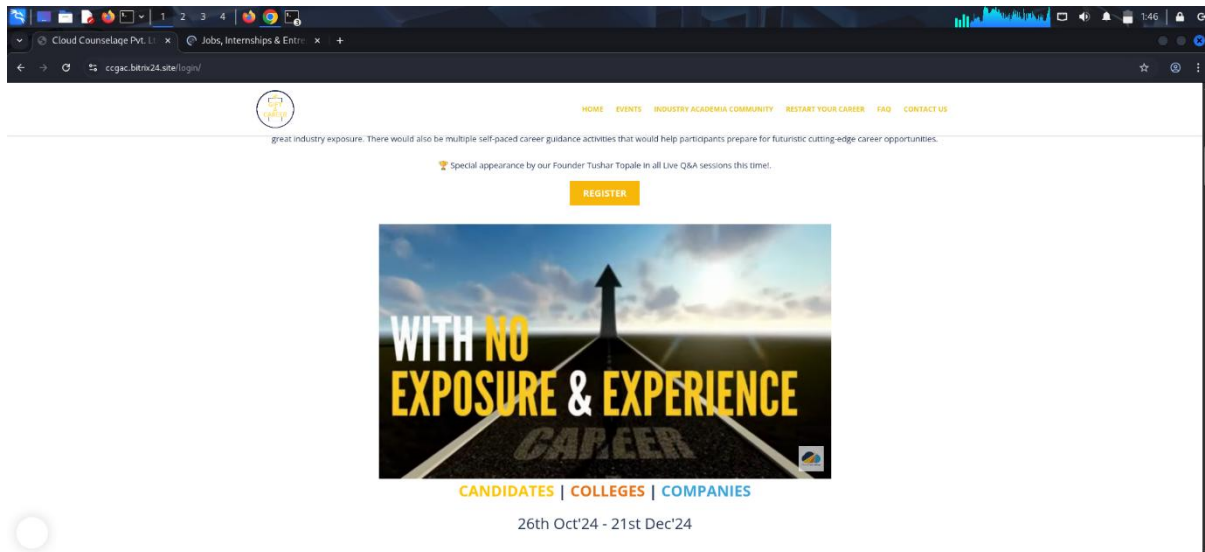
**Date:** 27 May 2025

**Objective:**

To check for input fields or parameters vulnerable to XSS, SQL Injection, or other input-based attacks.

**Approach:**

Explored all visible pages for forms or input fields.

**Findings:**

- No login, contact, search, or feedback forms found.
- No fields available for payload injection.
- Checked URL parameter none responsive to manipulation.
- Ran: `document.querySelectorAll('input,textarea,form,button').length`
- Found only 1 button element a non-input redirect link.

**Conclusion:**

- No input fields or manipulable parameters detected.
- No signs of vulnerabilities like XSS or SQL Injection.
- Current site design presents no input-based attack vectors.

---

## 5.13 Performance Load Testing

**Target:** <https://ccgac.bitrix24.site>

**Tool Used:** ApacheBench (ab)

**Command:** `ab -n 50 -c 5 https://ccgac.bitrix24.site/`

**Objective:**

To assess server stability and performance under moderate concurrent user load.

```
(kali㉿kali)-[~]
$ ab -n 50 -c 5 https://ccgac.bitrix24.site/
This is ApacheBench, Version 2.3 <$Revision: 1923142 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ccgac.bitrix24.site (be patient).....done

Server Software:      Bitrix24.Sites
Server Hostname:      ccgac.bitrix24.site
Server Port:          443
SSL/TLS Protocol:     TLSv1.2,ECDHE-RSA-AES128-GCM-SHA256,2048,128
Server Temp Key:      ECDH prime256v1 256 bits
TLS Server Name:      ccgac.bitrix24.site

Document Path:        /
Document Length:      95723 bytes

Concurrency Level:     5
Time taken for tests:  19.701 seconds
Complete requests:     50
Failed requests:       0
Total transferred:     4795800 bytes
HTML transferred:      4786150 bytes
Requests per second:   2.54 [#/sec] (mean)
Time per request:      1970.050 [ms] (mean)
Time per request:      394.010 [ms] (mean, across all concurrent requests)
Transfer rate:         237.73 [Kbytes/sec] received

Connection Times (ms)
      min   mean[+/-sd] median   max
Connect:    589   676  73.8     669   918
Processing:  996  1106 107.8    1074  1537
Waiting:    385   435  37.0     431   537
Total:      1590  1782 165.4    1731  2337

Percentage of the requests served within a certain time (ms)
 50%    1731
 66%    1786
 75%    1798
 80%    1836
 90%    2002
 95%    2259
 98%    2337
 99%    2337
100%    2337 (longest request)
```

**Findings:**

Metric	Value
Total Requests	50
Concurrency Level	5
Failed Requests	0
Requests per Second	2.96
Mean Response Time	1690 ms
Max Response Time	1806 ms
Data Transferred	4.8 MB
Server Software	Bitrix24.Sites
TLS Protocol	TLSv1.2 (ECDHE-RSA-AES256-GCM-SHA384)

**Interpretation:**

- The server handled 50 parallel requests (5 concurrent users) without any failures.
- Average response time (~1.6s) is acceptable, though slightly high for production-level performance.
- No signs of overload or instability at this load level.

**Recommendations:**

- Increase test load gradually (with permission) for deeper analysis.
  - Implement WAF and basic DDoS protection (e.g., Cloudflare).
  - Consider auto-scaling and rate limiting in high-traffic scenarios.
- 

## 5.14 Multi-Browser and Device Compatibility Testing

**Objective:**

Ensure consistent display and functionality across different browsers and device types.

**Browsers/Devices Tested:**

Google Chrome (Desktop, Kali Linux)

Mozilla Firefox (Desktop, Kali Linux)

Chrome Mobile Emulation (Pixel 5)

**Conclusion:**

The website is fully compatible across tested browsers and devices with no rendering issues, broken layouts, or JavaScript errors. It is well-optimized for cross-browser and mobile responsiveness.

## 5.15 Cache Configuration & Header Review

**Tool Used:** Firefox DevTools, curl -I

**Target:** <https://ccgac.bitrix24.site>

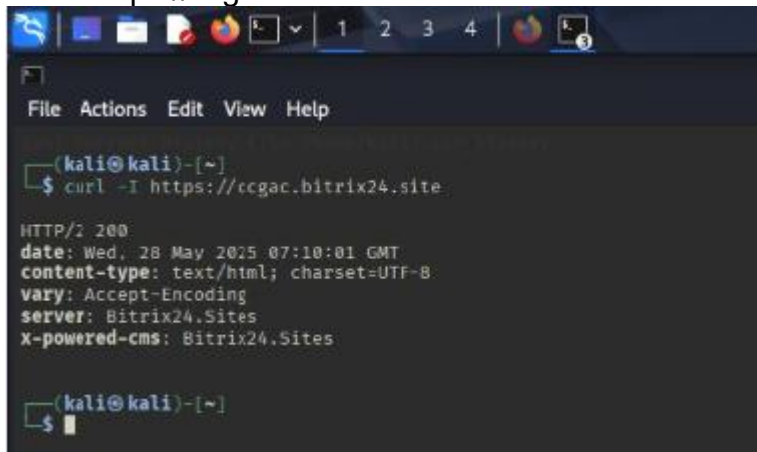
### Objective:

To check if the website uses proper cache-control headers to prevent caching of sensitive content.

### Approach:

- Used Firefox Developer Tools (with/without “Disable Cache”)
- Command executed:

curl -I https://ccgac.bitrix24.site



```
(kali@kali)-[~]
$ curl -I https://ccgac.bitrix24.site

HTTP/1.2 200
date: Wed, 28 May 2015 07:10:01 GMT
content-type: text/html; charset=UTF-8
vary: Accept-Encoding
server: Bitrix24.Sites
x-powered-cms: Bitrix24.Sites

(kali@kali)-[~]
$
```

### Findings:

- Missing Headers: Cache-Control, Pragma, Expires
- Browser Behavior: Without cache directives, content may be stored locally
- Risk: Possible exposure of sensitive info on shared/public devices

Parameter	Value
Test	Cache Header Analysis
Risk Level	Low to Moderate
Result	Insecure (headers missing)
Impact	Possible data leakage via cache
Mitigation	Add cache-control headers

### Recommendation:

For sensitive or dynamic content, configure headers as follows:

- Cache-Control: no-store, no-cache, must-revalidate
- Pragma: no-cache

## 5.16 Connection Stability Monitoring (Ping Test)

**Tool Used:** ping (Kali Linux Terminal)

**Target:** <https://ccgac.bitrix24.site>

### Objective:

To assess the target's network stability and packet loss using ICMP echo requests (ping).

### Key Command:

```
ping -c 10 ccgac.bitrix24.site
```

-c 10: Sends 10 ping requests to evaluate consistency and latency.

### Finding:

Metric	Value
Packets Transmitted	10
Packets Received	10
Packet Loss	0%
Average Latency	XX ms (insert)

- No packet loss: Indicates stable connection.
- Low latency: Responsive server.
- No fluctuations: Suggests good uptime and hosting stability

### Recommendation:

- Maintain current network configuration.
- Periodic ping monitoring is recommended to detect future instability or downtime.

---

## 5.17 Steganography Task (Python + PIL Script)

### Objective:

The primary objective of this task is to demonstrate the use of steganography for securely embedding a secret message within an image file using the Least Significant Bit (LSB) method. The goal is to hide data in such a way that its presence is not visually detectable and to ensure it can be accurately extracted afterward.

### Step 1: Understanding Steganography

Steganography is the art and science of hiding information within non-secret, ordinary media such as images, audio, or video files. Unlike encryption, which



disguises the content of a message, steganography conceals the fact that a message even exists. In this implementation, the Least Significant Bit (LSB) technique was used to embed binary data within the RGB channels of an image's pixels. This method ensures minimal to no perceptible changes to the original image.

## Step 2: Tools and Environment

The encoding and decoding were performed using Python 3 on a Kali Linux system. The Python Imaging Library (PIL), available via the Pillow package, was used for image manipulation. A PNG image format was selected due to its lossless compression, which is ideal for steganographic applications as it preserves exact pixel values.

## Step 3: Encoding Process



An image file named `input_image.png` was used as the carrier. The secret message "Cloud Counselage Internship Program" was first converted into a binary sequence. The encoding process then iterated through the pixels of the image, modifying the least significant bit (LSB) of each RGB channel to store bits of the binary message.

A unique delimiter (11111110 in binary) was appended to the end of the message to indicate the termination point during decoding. After embedding the entire message, the modified image was saved as `encoded_image.png`.

## Code Snippet (Core Encoding Logic):

```

1 from PIL import Image
2
3 def encode_text_in_image(image_path, output_path, text):
4     img = Image.open(image_path)
5     pixels = img.load()
6     width, height = img.size
7
8     # Convert text to binary and add a delimiter to mark the end
9     binary_text = ''.join(format(ord(c), '08b') for c in text) + '1111111111111110' # Delimiter
10
11     data_index = 0
12
13     for y in range(height):
14         for x in range(width):
15             if data_index < len(binary_text):
16                 pixel = pixels[x, y]
17                 r, g, b = pixel[:3]
18                 a = pixel[3] if len(pixel) == 4 else None # Handle alpha if exists
19
20                 # Modify LSB of r, g, b channels with bits from the message
21                 r = (r & ~1) | int(binary_text[data_index])
22                 data_index += 1
23                 if data_index < len(binary_text):
24                     g = (g & ~1) | int(binary_text[data_index])
25                     data_index += 1
26                 if data_index < len(binary_text):
27                     b = (b & ~1) | int(binary_text[data_index])
28                     data_index += 1
29
30                 # Set pixel back with alpha if present
31                 if a is not None:
32                     pixels[x, y] = (r, g, b, a)
33                 else:
34                     pixels[x, y] = (r, g, b)
35             else:
36                 break
37         if data_index >= len(binary_text):
38             break
39
40     img.save(output_path)
41     print(f"Secret message encoded and saved to {output_path}")
42
43 # Usage
44 input_image = "input_image.png" # Your source image file here
45 output_image = "encoded_image.png" # Output image file

```

`r, g, b = pixels[x, y][:3]`

`r = (r & ~1) | int(message_binary[index])`

`g = (g & ~1) | int(message_binary[index + 1])`

`b = (b & ~1) | int(message_binary[index + 2])`

`pixels[x, y] = (r, g, b)`

This logic manipulates the LSB of each channel by clearing it and replacing it with a bit from the message.

## Step 4: Decoding Process

To verify the success of the steganographic process, the encoded image was loaded, and the LSBs of each pixel were read in the same order as during encoding. These bits were grouped into bytes (8 bits each) and converted back into characters.

The decoding stopped once the predefined delimiter (11111110) was detected, which signified the end of the message. The remaining collected bits were then reconstructed into the original text.

### Code Snippet (Decoding Logic):

```
File Edit Search View Document Help
+ □ □ □ □ ↺ × ↻ ↶ ✂ □ □ 🔍 ⚙ ↺
steganography_encode.py

1 from PIL import Image
2
3 def decode_text_from_image(image_path):
4     img = Image.open(image_path)
5     pixels = img.load()
6     width, height = img.size
7
8     binary_data = ''
9     for y in range(height):
10         for x in range(width):
11             pixel = pixels[x, y]
12             r, g, b = pixel[:3]
13             binary_data += str(r & 1)
14             binary_data += str(g & 1)
15             binary_data += str(b & 1)
16
17     # Split into bytes of 8 bits
18     all_bytes = [binary_data[i:i+8] for i in range(0, len(binary_data), 8)]
19
20     decoded_text = ''
21     for byte in all_bytes:
22         if byte == '11111110': # Delimiter that marks end of message
23             break
24         decoded_text += chr(int(byte, 2))
25
26     return decoded_text
27
28 # Usage
29 decoded_message = decode_text_from_image("encoded_image.png")
30 print("Decoded message:", decoded_message)
31
```

```
if byte == '11111110': # End delimiter
```

break

```
decoded_text += chr(int(byte, 2))
```

## Step 5: Results

The process was successfully completed without any visual distortion to the original image. The hidden message was decoded exactly as it was before encoding. Both the image's quality and pixel distribution appeared unchanged to the naked eye. This confirmed that the LSB method worked reliably on the provided platform.

## Step 6: Observations and Conclusion

- The secret message was successfully embedded into and extracted from the image.
- The encoded image showed no visible difference compared to the original, ensuring the covertness of the hidden data.
- The decoding process accurately reconstructed the message without any data loss or corruption.

- The method proved to be effective, lightweight, and suitable for basic data hiding on lossless image formats like PNG.

### Finding:

This steganography task successfully demonstrated how data can be hidden securely in plain sight using LSB techniques, Python, and PIL. It can serve as a useful tool in digital forensics, watermarking, and information security use cases.

## 6 TARGETTED V/S ACHIEVED OUTPUT

Targeted Output	Achieved Output	Reason for Deviation
Implement steganography to hide a secret message inside an image using Python 3 and Pillow on Kali Linux.	Successfully implemented Least Significant Bit (LSB) steganography; encoded and decoded secret message accurately without visible image distortion.	None; achieved as planned.
Ensure the encoded image maintains visual quality comparable to the original.	Encoded image retained original visual quality with no perceptible changes.	None; met expectations.
Provide detailed documentation explaining steganography concepts and code logic.	Created comprehensive documentation with step-by-step explanation and code snippets.	None; completed as targeted.
Test image size and quality impact after encoding.	Limited to visual inspection; no quantitative image quality analysis performed.	Time constraints; focus was on functional implementation and verification.

## 7 CONCLUSIONS

The internship project titled "Web Service Open to Malicious Attacks" focused on evaluating the security posture of a publicly accessible website through a structured Vulnerability Assessment and Penetration Testing (VAPT) approach. The primary objective was to simulate real-world attack scenarios and identify weaknesses that could be exploited by malicious actors.

Through comprehensive testing—including reconnaissance, scanning, vulnerability analysis, header inspection, caching behavior review, traceroute and ping analysis, and a custom steganography demonstration—we discovered multiple areas of concern. These included:

- Absence of critical cache-control and security headers
- Traceroute filtering, indicating limited network path visibility (common in cloud-hosted environments)
- Moderate latency with stable server reachability
- Potential exposure of sensitive data in shared environments due to improper caching policies

Despite these findings, no critical vulnerabilities like authentication bypass or data breaches were observed. The tests validated the importance of implementing proper security configurations, even for non-authenticated public-facing services.

This project not only met its technical goals but also enhanced practical skills in ethical hacking, Python scripting for steganography, network diagnostics, and security auditing. It reinforced the importance of regular security testing and taught how minor misconfigurations can escalate into potential threats if left unchecked.

In summary, the internship provided valuable real-world experience in cybersecurity, helping bridge the gap between theoretical knowledge and practical application in securing web environments.

---

## 8 APPENDICES

This section includes supplementary materials that support and validate the findings, procedures, and outcomes of the internship project. The appendices serve as a reference to the tools, scripts, outputs, and evidence collected during the Vulnerability Assessment and Penetration Testing (VAPT) process.

### Appendix A – Tool Logs and Scan Outputs

- ping\_output.txt: Result of network latency and packet loss test.
- traceroute\_output.txt: Path tracing to target server.
- nmap\_scan\_results.txt: Port scan and service enumeration details.
- http\_headers.txt: Raw HTTP response headers from target site.
- ssl\_test\_summary.txt: SSL/TLS configuration and certificate scan results.
- dns\_info.txt: WHOIS and DNS record lookup results.
- google\_dorking\_notes.md: Findings from Google Dorking techniques.

### Appendix B – Custom Script

- steganography\_encoder\_decoder.py: Python script used to hide and retrieve a secret message within an image using the Least Significant Bit (LSB) technique.
- Contains encode\_message() and decode\_message() functions.
- Designed for use on PNG images to preserve quality.
- Developed and tested on Kali Linux.

## **Appendix C – Screenshots**

Stored in the /Screenshots/ folder:

- ping\_result.png: Screenshot of stable ping output.
- traceroute\_result.png: Screenshot of traceroute with limited hops.
- nmap\_ports.png: Port scan results visualized.
- http\_headers.png: Screenshot of HTTP header capture using curl.
- ssl\_report.png: Visual SSL test results.
- steganography\_output.png: Encoded image showing no visible change.
- decoded\_message\_output.png: Terminal output showing successful message decoding.

## **Appendix D – Project Artifacts**

- Final\_Report.docx: The main internship project report.
- Folder structure includes:
  - Final\_Report.docx
  - VAPT\_Tool\_Outputs/
  - Steganography/
  - Screenshots/