# Wapiti vulnerability report

## Target: http://google-gruyere.appspot.com/366974477833105008145397588284477084980/

Date of the scan: Mon, 14 Aug 2023 06:48:19 +0000. Scope of the scan: folder

## Summary

| Category | Number of vulnerabilities found |
|---|---|
| Backup file | 0 |
| Blind SQL Injection | 0 |
| Weak credentials | 0 |
| CRLF Injection | 0 |
| Content Security Policy Configuration | 1 |
| Cross Site Request Forgery | 0 |
| Potentially dangerous file | 0 |
| Command execution | 0 |
| Path Traversal | 0 |
| Htaccess Bypass | 0 |
| HTTP Secure Headers | 4 |
| HttpOnly Flag cookie | 0 |
| Open Redirect | 0 |
| Secure Flag cookie | 0 |
| SQL Injection | 0 |
| Server Side Request Forgery | 0 |
| Cross Site Scripting | 2 |
| XML External Entity | 0 |
| Internal Server Error | 2 |
| Resource consumption | 0 |

| Category | Number of vulnerabilities found |
|---|---|
| Fingerprint web technology | 0 |

## Content Security Policy Configuration

### Description

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks.

### Vulnerability found in /366974477833105008145397588284477084980/

Description HTTP Request cURL command line

```
CSP is not set
```

### Solutions

Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control what resources the user agent is allowed to load for that page.

### References

- Mozilla: Content Security Policy (CSP)
- OWASP: Content Security Policy Cheat Sheet
- OWASP: How to do Content Security Policy (PDF)

## HTTP Secure Headers

### Description

HTTP security headers tell the browser how to behave when handling the website's content.

### Vulnerability found in /366974477833105008145397588284477084980/

Description HTTP Request cURL command line

```
X-Frame-Options is not set
```

### Vulnerability found in /366974477833105008145397588284477084980/

Description HTTP Request cURL command line

```
X-XSS-Protection is not set
```

### Vulnerability found in /366974477833105008145397588284477084980/

Description HTTP Request cURL command line

```
X-Content-Type-Options is not set
```

### Vulnerability found in /366974477833105008145397588284477084980/

Description HTTP Request cURL command line

```
Strict-Transport-Security is not set
```

**Solutions**

Use the recommendations for hardening your HTTP Security Headers.

**References**

- Netsparker: HTTP Security Headers: An Easy Way to Harden Your Web Applications
- KeyCDN: Hardening Your HTTP Security Headers
- OWASP: HTTP SECURITY HEADERS (Protection For Browsers) (PDF)

## Cross Site Scripting

**Description**

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in web applications which allow code injection by malicious web users into the web pages viewed by other users. Examples of such code include HTML code and client-side scripts.

### Vulnerability found in /366974477833105008145397588284477084980/deletesnippet

Description HTTP Request cURL command line

```
XSS vulnerability found via injection in the parameter index
```

## Vulnerability found in /366974477833105008145397588284477084980/snippets.gtl

Description HTTP Request cURL command line

```
XSS vulnerability found via injection in the parameter uid
```

### Solutions

The best way to protect a web application from XSS attacks is ensure that the application performs validation of all headers, cookies, query strings, form fields, and hidden fields. Encoding user supplied output in the server side can also defeat XSS vulnerabilities by preventing inserted scripts from being transmitted to users in an executable form. Applications can gain significant protection from javascript based attacks by converting the following characters in all generated output to the appropriate HTML entity encoding:<, >, &, ', (, ), #, %, ; , +, -

### References

- OWASP: Cross Site Scripting (XSS)
- Wikipedia: Cross-site scripting
- CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

# Internal Server Error

### Description

An error occurred on the server's side, preventing it to process the request. It may be the sign of a vulnerability.

## Anomaly found in /366974477833105008145397588284477084980/saveprofile

Description HTTP Request cURL command line

```
The server responded with a 500 HTTP error code while attempting to inject a payload in the parameter
```

## Anomaly found in /366974477833105008145397588284477084980/saveprofile

Description HTTP Request cURL command line

```
The server responded with a 500 HTTP error code while attempting to inject a payload in the parameter
```

### Solutions

More information about the error should be found in the server logs.

### References

- Wikipedia: List of 5xx HTTP status codes
- OWASP: Improper Error Handling