CCL SOLUTIONS GROUP

incorporating evidencetalks

Retest of CS11490

IT Health Check Report

Client:	Medigold Health Consultancy Limited
Project:	Retest of CS11490
Document Reference:	CS11702-RPT-01
Prepared for:	Lee Alderdice
Author:	Niall Aiken
Testing Team:	Bianca Napoleonov, Jacob Hudson, Jack Whittington, Niall Aiken, Rehan Bari
Date:	04 November 2025
NCSC Reference Number:	CK2025-X7D3L-3817

Assured Service Provider

in association with National Cyber Security Centre

CHECK Penetration Testing

1. Document Control

1.1. Document Details

Client	Medigold Health Consultancy Limited
Title	Retest of CS11490
Author	Niall Aiken
Version	1.0
Date	04/11/2025
Document Reference	CS11702-RPT-01
Status	Definitive

1.2. Revision History

Version	Date	Author	Comments
0.1	31/10/2025	Niall Aiken	Retest Draft
0.2	04/11/2025	Bianca Napoleonov	Retest QA
1.0	04/11/2025	Niall Aiken	Retest Definitive

1.3. Distribution

Name	Email	Organisation
Lee Alderdice	lee.alderdice@medigold-health.com	Medigold Health Consultancy Limited

CCL Solutions Group
34-36 Cygnet Court, Timothy's Bridge Road, Stratford-upon-Avon, CV37 9NW
+44 (0)1789 261 200 | contact@cclsolutionsgroup.com | cclsolutionsgroup.com
CCL Solutions Group is made up of the following companies: CCL (Solutions) Group Ltd (company number 08128980), CCL-Forensics Ltd (company number 05314495), Evidence Talks Limited (company number 04611669), CCL Cyber Solutions Ltd (company number 11316398), CCL (Computer Consultants) Ltd (company number 02049601)

2. Contents

3. Executive Summary

3.1. Overview

CCL Solutions Group were engaged by Medigold to conduct a retest of their Core, Gateway and Pulse web applications focused on only the remaining low-risk issues. This was conducted between 28th and 31st October 2025. The purpose of this engagement was to verify that all outstanding low-risk findings from the original penetration test had been remediated effectively and that no residual weaknesses remained. During the retesting period, each issue was re-examined using the same techniques and attack paths that originally revealed the vulnerabilities. All fixes implemented by the development team were validated through direct testing, and no instances of the previously observed vulnerable behaviour were reproduced. No regressions or new issues related to the original findings were identified during this verification process. As a result, all outstanding issues are confirmed as successfully remediated, and the application now demonstrates a significantly improved security posture. Continued adherence to secure development practices and periodic security assessments is recommended to maintain this level of assurance.

Below is a summary of the initial issues identified for each of the applications in scope. These have now been resolved.

3.2. Medigold Core Application

Of the issues identified, the most significant related to unsupported or outdated software libraries that were found to be vulnerable to known security flaws. Updates to software libraries often apply fixes to known security vulnerabilities, and as such, should be applied as and when they become available. Additional review of the patching procedures should be undertaken to ensure updates are applied, when required, in a swift manner.

Further comment has been made in relation to missing security controls in place that mitigate against known vulnerabilities and common attacks. Following a secure by design approach implementing security controls on requests and responses sent by both application and API servers helps to improve the overall security posture offered and reduces the likelihood of data leakage or compromise.

Additional note was made in relation to malicious user input being accepted and exported into an excel format using the associated APIs. When the excel export is opened the malicious user inputs are run as a macro which can then perform system commands. In such an instance, service user's information would be at-risk of theft or modification and placing Medigold Health at risk of civil and regulatory financial penalties, in addition to an associated reduction of customer confidence.

3.3. Medigold Gateway Application

The application is generally well-configured, incorporating several security best practices. Common vulnerabilities from malicious input, such as text-based attacks, were effectively mitigated. During testing, workflows were not susceptible to privilege escalation or unauthorised modifications. However, certain areas would benefit from additional security enhancements.

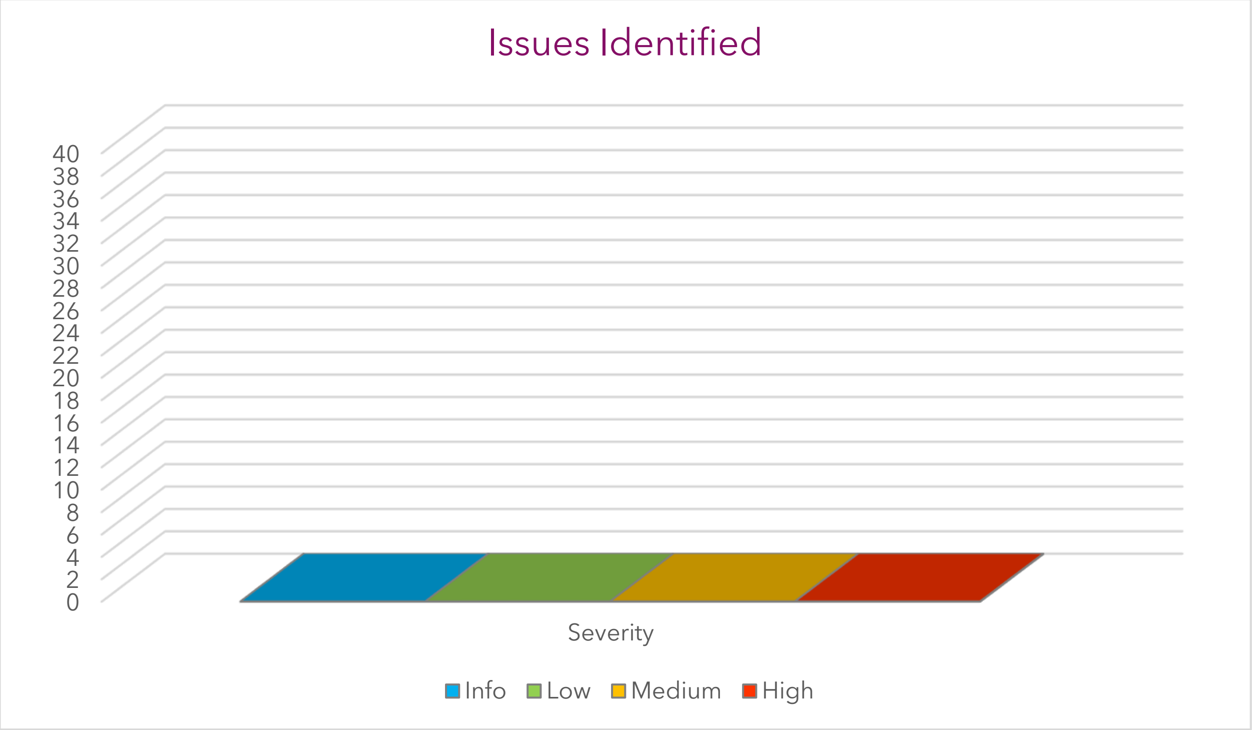
One significant issue is the lack of rate limiting, which allowed over 2,000 requests in a short period without triggering any throttling or blocking mechanisms. Additionally, several low-risk issues were identified, primarily relating to missing security controls in place that mitigate against known vulnerabilities and common attacks. Following a secure by design approach implementing security controls on requests and responses sent by both application and API servers helps to improve the overall security posture offered and reduces the likelihood of data leakage or compromise.

3.4. Medigold Pulse Application

Of the issues identified, the most significant related to an outdated software library identified that was found to have known security flaws. Updates to software libraries often apply fixes to known security vulnerabilities, and as such, should be applied as and when they become available.

Further note has been made in relation to the lack of rate-limiting controls in the application. During the assessment, it was observed that the application did not incorporate rate-limiting, allowing repeated submission of requests without any form of throttling or blocking. This presents a risk of server overload.

The remaining low risk issues identified throughout the assessment were primarily due to configuration that was not in line with security best practice guidelines and required further hardening, while these may not be an immediate risk they should be reviewed and remediated on a case-by-case basis.



4. Next Steps

- Remove accounts created to facilitate the penetration test.
- Remove all data created as part of the assessment where necessary.

5. Technical Information

5.1. Background

CCL (Solutions) Group Ltd. were engaged by Medigold Health to assess a number of applications and associated external infrastructure, to support their ongoing accreditation requirements. This was conducted adhering to the below scope in accordance with the CCL Security Testing Methodology (DOC-10001).

From the original scope of CS11490 only the Core, Gateway and Pulse elements were carried out for retesting.

5.2. Scope

Host	Description
medigoldcore.com	Medigold Core Application
hk.medigoldcore.com	Medigold Nexus application
live.medigoldone.com	Medigold One Application
gateway.medigoldcore.com	Medigold Gateway Application
medigoldpulse.com	Medigold Pulse Application
app-osi-accesshub-prd.azurewebsites.net	Access Hub API
app-osi-address-and-postcode-prd.azurewebsites.net	Address API
app-anubis-prd.azurewebsites.net	Anubis API
app-osi-case-prd.azurewebsites.net	Case API
app-osi-client-prd.azurewebsites.net	Client API
app-osi-clinic-prd.azurewebsites.net	Clinic API
app-osi-comms-prd.azurewebsites.net	Comms API
app-osi-contract-prd.azurewebsites.net	Contract API
app-consultations-prd.azurewebsites.net	Consultation API
app-osi-diary-prd.azurewebsites.net	Diary API
app-osi-employee-prd.azurewebsites.net	Employee API
app-osi-file-management-prd.azurewebsites.net	File Management API
app-osi-integration-prd.azurewebsites.net	Integration API
gateway.api.medigoldcore.com	Gateway API
app-osi-product-prd.azurewebsites.net	Product API

Host	Description
app-osi-signalr-prd.azurewebsites.net	Signalr API
app-osi-utility-prd.azurewebsites.net	Utility API
app-osi-notification-and-consent-prd.azurewebsites.net	Notification and Consent API
mgcorefunctions-prd.azurewebsites.net	Core Functions API
app-osi-logging-prd.azurewebsites.net	Logging API
app-osi-search-prd.azurewebsites.net	Search API
app-aiservice-prd.azurewebsites.net	AI Service API
app-policy-api-prd.azurewebsites.net	Policy API
4.158.63.171 51.11.30.93 20.58.11.223 51.142.251.255 51.104.196.241 51.140.83.70 20.50.107.25 51.11.54.188 51.104.250.166 51.104.249.146 20.50.105.127 20.49.193.217 20.162.233.143 20.58.42.46 51.132.185.208 51.132.210.105 51.104.194.82 20.0.17.75 20.0.184.53	External Infrastructure IPs

5.2.1. User Accounts

User accounts were provided for all external applications within the scope of the engagement with multiple levels of access.

User account	Description
ccl.jacob.hudson.one@medigold-health.com	Technician Core Application

User account	Description
ccl.jacobhudson.two@medigold-health.com	Administrator Core Application
ccl.biancanapoleonov.one@medigold-health.com	Administrator Nexus Application
ccl.biancanapoleonov.two@medigold-health.com	Technician Nexus Application
rehan.bari@cclsolutionsgroup.com	Operator One Application
rehan.bari+1@cclsolutionsgroup.com	Form Manager One Application
rehan.bari+2@cclsolutionsgroup.com	Administrator One Application
ccl.rehanbari.one@medigold-health.com	Global User Gateway Application
ccl.rehanbari.two@medigold-health.com	Standard User Gateway Application
ccl.niallaiken.two@medigold-health.com	Internal Administrator Pulse Application
ccl.niallaiken.one@medigold-health.com	Internal User Pulse Application
niall.aiken@cclsolutionsgroup.com	External Administrator Pulse Application
niall.aiken+1@cclsolutionsgroup.com	External User Pulse Application
ccl.niall.one	Healthcare Surveillance Mobile Application User
ccl.niall.two	Healthcare Surveillance Mobile Application User
jack.whittington@cclsolutionsgroup.com	Administrator account
ccl.jack.whittington@medigold-health.com	Administrator account

5.3. Rules of Engagement

The penetration test was performed in line with the following rules of engagement:

- Grey box testing methodology was used.
- The engagement was conducted remotely.

6. Findings Summary

Below is a list of findings discovered by the security consultants:

6.1. Medigold Core Application Summary

Reference	Vulnerability	Severity	Remediation
1	**Outdated Third-Party Libraries** Review of third-party scripts throughout the application identified instances of libraries that were outdated and affected by software vulnerabilities.	Medium	JavaScript libraries within an enterprise should be identified and included within organisational patching cycles.
2	**Strict Transport Security Header Not Configured** The HTTP Strict Transport Security (HSTS) header was not configured on the web server.	Medium	The HSTS header should be added to all applications to ensure all communications are encrypted.
3	**CSV Injection** The export functionality within the multiple areas of the Core application were identified as being vulnerable to injection attacks.	Medium	Implement appropriate input validation for fields which are later exported to Excel files.
4	**Information Disclosure Via HTTP Header** The application was found to reveal the server technologies and versions within the HTTP headers.	Low	Headers containing information about backend technologies should be removed or obfuscated.
5	**SSL Weak Cipher Suites** Cipher Block Chaining (CBC) are vulnerable to oracle padding attacks. This attack enables a malicious actor to decrypt the contents of the data, without knowing the key.	Low	Weak cipher suites should be disabled for the service.
6	**Missing HTTP Security Headers** HTTP headers which add additional security to applications were not being utilised by the web application.	Low	The recommended HTTP security headers should be configured on the web application.
7	**Insufficient Input Validation** It was possible to bypass the frontend input validation capturing and modifying the requests.	Low	To address the issue of insufficient input validation, implement robust server-side validation to ensure all input conforms to expected formats, types, and ranges before processing.
8	**Sensitive Data in URL** Sensitive information was found to be transmitted within the URL.	Low	Sensitive data should be transmitted within the body of HTTP requests.

Reference	Vulnerability	Severity	Remediation
9	**Deprecated HTTP Security Header** The 'X-XSS-Protection' header was designed to detect and assist in protecting against cross-site scripting ("XSS") attacks; however, this was deprecated.	Low	Remove the 'X-XSS-Protection' header or set to a value of 0.

6.2. Medigold Gateway Application Summary

Reference	Vulnerability	Severity	Remediation
10	**Lack of Rate Limiting** The application lacks rate limiting, allowing users or attackers to send an excessive number of requests in a short period without restriction.	Medium	Implement rate limiting which involves controlling the number of requests that clients can make within a certain time frame.
11	**Outdated Third-Party Libraries** Review of third-party scripts throughout the application identified multiple instances of libraries that were either outdated and affected by software vulnerabilities or no longer supported by the vendor.	Medium	Third-party libraries should be updated to the latest version available.
12	**Missing Content Security Policy Header** The application lacks a Content-Security-Policy (CSP) header, which is a critical defence against cross-site scripting (XSS) and other client-side attacks.	Low	The Content-Security-Policy (CSP) header should be configured to enhance the security of the application, especially in scenarios where user-uploaded files might be served to browsers.
13	**Cookie Weaknesses** Weaknesses were identified with the cookie configuration options set by the application.	Low	All cookies should have both the HTTPOnly and Secure flags set, specifically those which relate to session data.
14	**SSL Weak Cipher Suites** The application supports weak SSL cipher suites, reducing the effectiveness of encryption and exposing secure communications to potential attacks.	Low	Weak methods of encryption should be disabled to ensure only secure cipher suites are used.

6.3. Medigold Pulse Application Summary

Reference	Vulnerability	Severity	Remediation
32	**Lack of Rate Limiting** The application lacks rate limiting, allowing users or attackers to send an excessive number of requests in a short period without restriction.	Medium	Implement rate limiting which involves controlling the number of requests that clients can make within a certain time frame.
33	**Outdated Third-Party Libraries** Review of third-party scripts throughout the application identified multiple instances of libraries that were either outdated and affected by software vulnerabilities or no longer supported by the vendor.	Medium	Third-party libraries should be updated to the latest version available.
34	**Strict Transport Security Header Not Configured** The HTTP Strict Transport Security (HSTS) header was not configured on the web server.	Medium	The HSTS header should be added to all applications to ensure all communications are encrypted.
35	**Username Enumeration** Error messages enabled valid usernames to be enumerated.	Low	The application should display a generic message regardless of the email address validity.
36	**Information Disclosure Via HTTP Header** The application was found to reveal the backend technologies supporting the service within the HTTP banners.	Low	The recommended HTTP security headers should be configured on the web application.
37	**Cross-Origin Resource Sharing: Arbitrary Origin Trusted** Trusting arbitrary origins effectively disables the same-origin policy, allowing two-way interaction by third-party websites. Unless the response consists of only unprotected public content, this policy is likely to present a security risk.	Low	CCL recommends disabling all instances of CORS without a documented business use case and exhausting existing, safer alternatives.
38	**Clickjacking Vulnerability** It was possible to embed the web application within an iframe which makes clickjacking attacks possible.	Low	The frame-ancestors directive of the Content Security Policy (CSP) HTTP response header can be used to restrict which sites if any can embed the application within frame/iframe tags.
39	**Missing HTTP Security Headers** HTTP headers which add additional security to applications were not being utilised by the web application.	Low	The recommended HTTP security headers should be configured on the web application.

Reference	Vulnerability	Severity	Remediation
40	**SSL Weak Cipher Suites** The application supports weak SSL cipher suites, reducing the effectiveness of encryption and exposing secure communications to potential attacks.	Low	Weak methods of encryption should be disabled to ensure only secure cipher suites are used.
41	**Deprecated HTTP Security Header** The 'X-XSS-Protection' header was designed to detect and assist in protecting against cross-site scripting ("XSS") attacks.	Low	Removing or replacing this header with more robust security measures, like Content-Security-Policy (CSP), will improve protection against cross-site scripting (XSS) attacks.

7. Medigold Core Application

This section details the Medigold Core application and associated API issues that were identified during the testing engagement.

7.1. Outdated Third-Party Libraries

Severity	**Medium**
Impact	3
Likelihood	2
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	1

7.1.1. Summary

This finding was found to be remediated after retesting.

Third-party JavaScript libraries, such as jQuery, can be utilised by applications in order to extend the application functionality and reduce development time through means of providing pre-packaged functions, removing the need to code all site content from scratch. In many cases, sites are built around a specific version of a library. The library itself may remain static for fear of introducing compatibility issues, or from fear of impacting application useability. Failure to include JavaScript libraries within organisational patching policy can result in outdated versions being used, leading to the introduction of security vulnerabilities and increasing the possibility of a successful attack being mounted against the application and its users.

7.1.2. Technical Information

The versions of JavaScript libraries currently implemented were found to be outdated and was identified as being vulnerable to attack. The identified versions are potentially vulnerable to cross-site scripting (XSS) attacks. This is highlighted in the table provided below:

Library	Version	Path
Vue	2.7.17	https://medigoldcore.com/libs/vue/2.7.17/vue.min.js

7.1.3. Recommendations

JavaScript libraries within an enterprise should be identified and included within organisational patching cycles. Where a specific, non-current, library is required to provide specific functionality, this should be noted within the appropriate risk registers. If it is not possible to replace outdated libraries, then enhanced monitoring should be considered, or functionality should be recreated within the internal application codebase. Where possible, outdated libraries should be replaced with the most recent, stable, build. At the time of report compilation, these were found to be version 3.5.13 for Vue.

7.1.4. References

We have identified the following locations for further reading:

- **Snyk:** https://security.snyk.io/package/npm/axios/0.21.4
- **Snyk:** https://cwe.mitre.org/data/definitions/1333.html

7.1.5. Affected Systems

Affected Item
https://medigoldcore.com/

7.2. Strict Transport Security Header Not Configured

Severity	**Medium**
Impact	4
Likelihood	2
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	2

7.2.1. Summary

This finding was found to be remediated after retesting.

The HTTP Strict Transport Security (HSTS) header was not configured on the web server. This is a security header which can be sent by a web server to inform the browser to only communicate over HTTPS. Once the browser has received this header it will restrict the user from being able to access HTTP content on the same domain ensuring that all communications are encrypted. Once the configured 'max-age' time period has expired the browser will allow clear-text HTTP communications with the domain again. The 'max-age' countdown resets each time a response is received containing the HSTS header.

7.2.2. Technical Information

The Strict-Transport-Security header was not present in any responses from the web server. The following curl request was used to return the headers:

```
curl -I https://medigoldcore.com
```

The following server response was received, note the missing header:

```
HTTP/2 200
date: Mon, 20 Jan 2025 15:07:57 GMT
content-type: text/html
content-length: 77213
vary: Accept-Encoding
content-md5: qmP+WVoXCjkS7eilplI5nA==
last-modified: Mon, 06 Jan 2025 18:12:02 GMT
etag: "0x8DD2E7D9EDF1D5A"
vary: Origin
x-ms-request-id: f188f6f6-801e-007a-46c1-6a9d20000000
x-ms-version: 2018-03-28
x-azure-ref: 20250120T150757Z-r15774cf85dpf2rdhC1LON7f5c0000000frg0000000187gf
x-fd-int-roxy-purgeid: 0
x-cache: TCP_HIT
x-cache-info: L1_T2
x-frame-options: SAMEORIGIN
accept-ranges: bytes
```

7.2.3. Recommendations

The HSTS header should be added to all applications to ensure all communications are encrypted. As an example, the following header would enforce HTTPS for one year:

```
Strict-Transport-Security: max-age=31536000
```

Note: The ‘max-age’ value is specified in seconds.

7.2.4. References

We have identified the following locations for further reading:

- **OWASP:** https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet

7.2.5. Affected Systems

Affected Item
https://medigoldcore.com/

7.3. CSV Injection

Severity	**Medium**
Impact	3
Likelihood	2
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	3

7.3.1. Summary

This finding was found to be remediated after retesting.

During an application assessment it is often observed that applications include a significant amount of data, which may require export functionality for storing or processing data through other means. Among the available file formats, comma-separated value (CSV) files are extensively used, as they can be opened in various desktop applications and are widely available. It is common for a wide range of application users to generate these file exports.

Most developers overlook export functionality as an attack vector and as such data sanitisation is frequently omitted. Due to this, malicious files can be constructed which could lead to code-execution on end user workstations in the context of the computer account with which the user downloaded the file with.

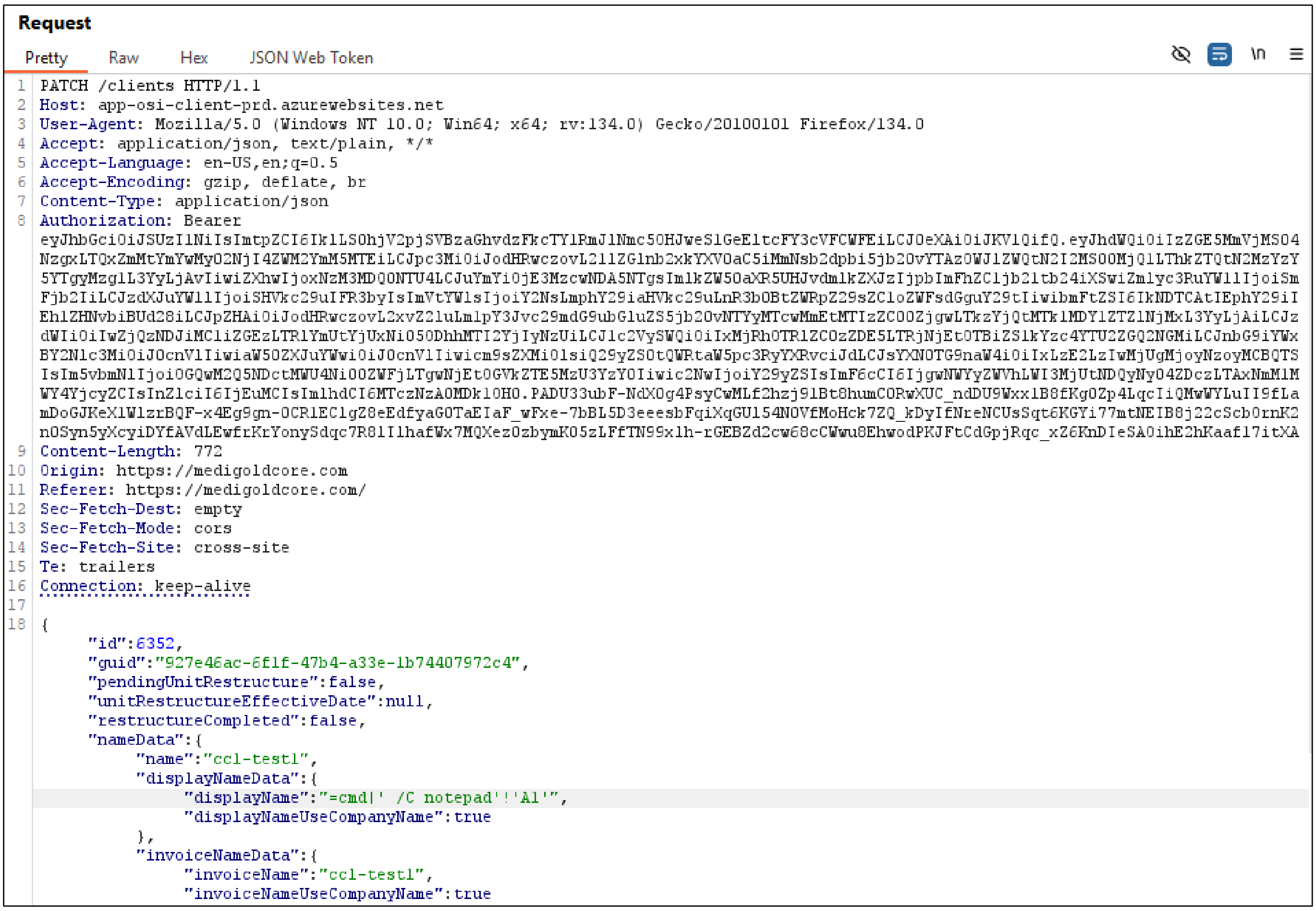
7.3.2. Technical Information

The export functionality within the multiple areas of the Core application were identified as being vulnerable to injection attacks. The payload, shown below, highlights the abuse possible within Microsoft's dynamic data exchange (DDE) which allows for the transfer of data between applications.

The payload below was inserted to the API requests which were automatically imported into fields within the exported document. It should be noted the frontend offered input validation, but this was not applied when capturing and modifying the API requests.

```
=cmd|’ /C notepad’!A1
```

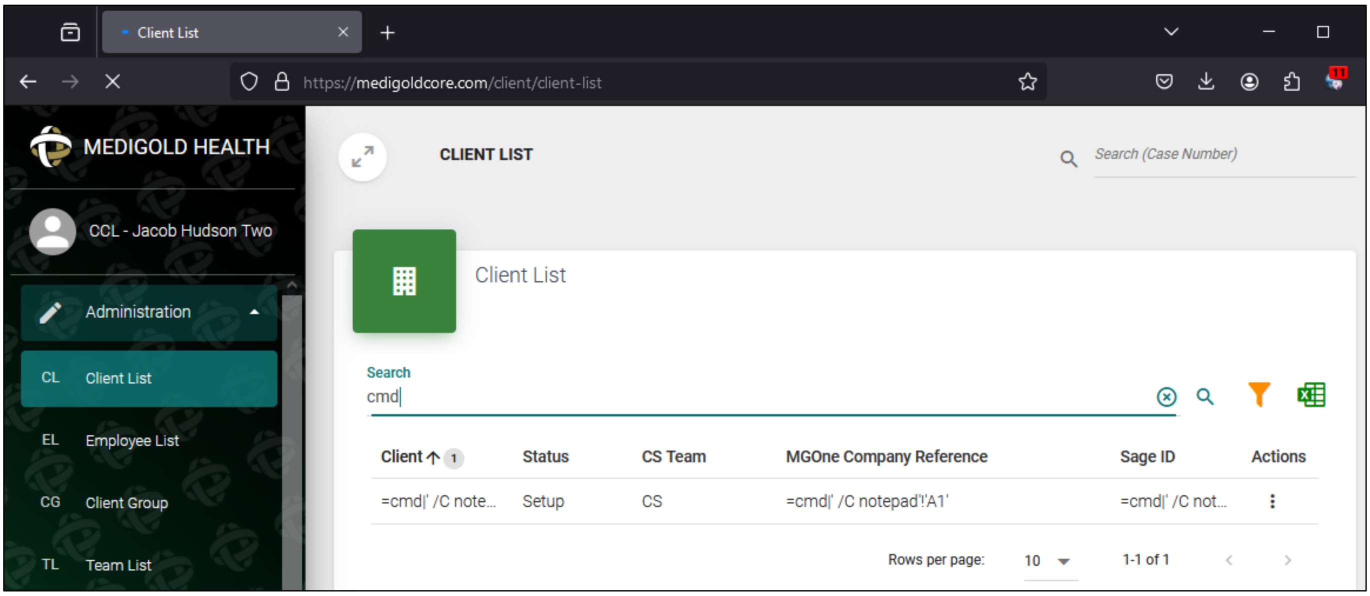
The above payload was used to demonstrate code-execution on a user machine. This can be evidenced by observing that opening the spreadsheet has caused the notepad executable to be launched. It should be noted that Excel, by default, prompts the user with a security warning, prior to executing the embedded script.



```
PATCH /clients HTTP/1.1
Host: app-osi-client-prd.azurewebsites.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101 Firefox/134.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6Ik1LS0hjV2pjSVBzaGhvdzFkcTY1RmJ1Nmc5OHJweS1GeEltcFY3cVFCWFEiLCJ0eXAiOiJKV1QifQ.eyJhdWQiOiIzZGE5MmVjMS04
NzgxLTQxZmMtYmYwMy02NjI4ZWM2YmM5MTEiLCJpc3MiOiJodHRwczovL21lZGlnb2xkYXV0aC5iMmNsb2dpbi5jb20vYTAzOWJlZWQtN2I2MS00MjQ1LThkZTQtN2MzYzY
5YTgyMzg1L3YyLjAvIiwiZXhwIjoxNzM3MDQ0NTU4LCJuYmYiOjE3MzcwNDA5NTgsImlkZW50aXR5UHJvdmlkZXJzIjpbImFhZC1jb21tb24iXSwiZmlyc3RuYW1lIjoiSm
Fjb2IiLCJzdXJuYW1lIjoiSHVkc29uIFR3byIsImVtYWlsIjoiY2NsLmphY29iaHVkc29uLnR3b0BtZWRpZ29sZC1oZWFsdGguY29tIiwibmFtZSI6IkNDTCAtIEphY29iI
Eh1ZHNvbiBUd28iLCJpZHAiOiJodHRwczovL2xvZ2luLm1pY3Jvc29mdG9ubGluZS5jb20vNTYyMTcwMmEtMTIzZC00ZjgwLTkzYjQtMTk1MDY1ZTZlNjMxL3YyLjAiLCJz
dWIiOiIwZjQzNDJiMC1iZGEzLTRlYmUtYjUxNi05ODhhMTI2YjIyNzUiLCJ1c2VySWQiOiIxMjRhOTRlZC0zZDE5LTRjNjEtOTBiZS1kYzc4YTU2ZGQ2NGMiLCJnbG9iYWx
BY2Nlc3MiOiJ0cnVlIiwiaW50ZXJuYWwiOiJ0cnVlIiwicm9sZXMiOlsiQ29yZS0tQWRtaW5pc3RyYXRvciJdLCJsYXN0TG9naW4iOiIxLzE2LzIwMjUgMjoyNzoyMCBQTS
IsIm5vbmNlIjoiOGQwM2Q5NDctMWU4Ni00ZWFjLTgwNjEtOGVkZTE5MzU3YzY0Iiwic2NwIjoiY29yZSIsImF6cCI6IjgwNWYyZWVhLWI3MjUtNDQyNy04ZDczLTAxNmM1M
WY4YjcyZCIsInZlciI6IjEuMCIsImlhdCI6MTczNzA0MDk1OH0.PADU33ubF-NdX0g4PsyCwMLf2hzj91Bt8humC0RwXUC_ndDU9Wxx1B8fKg0Zp4LqcIiQMwWYLuII9fLa
mDoGJKeX1WlzrBQF-x4Eg9gn-0CR1EC1gZ8eEdfyaG0TaEIaF_wFxe-7bBL5D3eeesbFqiXqGU154N0VfMoHck7ZQ_kDyIfNreNCUsSqt6KGYi77mtNEIB8j22cScb0rnK2
n0Syn5yXcyiDYfAVdLEwfrKrYonySdqc7R81I1hafWx7MQXez0zbymK05zLFfTN99x1h-rGEBZd2cw68cCWwu8EhwodPKJFtCdGpjRqc_xZ6KnDIeSA0ihE2hKaaf17itXA
Content-Length: 772
Origin: https://medigoldcore.com
Referer: https://medigoldcore.com/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: cross-site
Te: trailers
Connection: keep-alive

{
    "id":6352,
    "guid":"927e46ac-6f1f-47b4-a33e-1b74407972c4",
    "pendingUnitRestructure":false,
    "unitRestructureEffectiveDate":null,
    "restructureCompleted":false,
    "nameData":{
        "name":"ccl-test1",
        "displayNameData":{
            "displayName":"=cmd|' /C notepad'!'A1'",
            "displayNameUseCompanyName":true
        },
        "invoiceNameData":{
            "invoiceName":"ccl-test1",
            "invoiceNameUseCompanyName":true
```

Figure 1 - Request Modified with the Payload Inserted

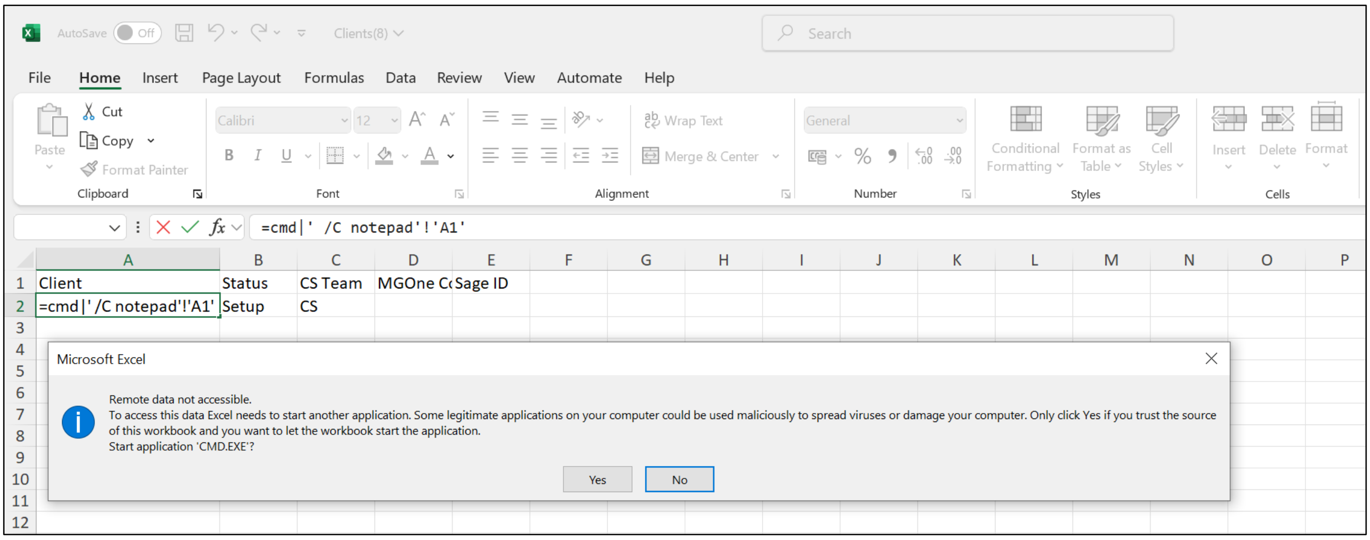


Search
cmd

Client ↑ 1	Status	CS Team	MGOne Company Reference	Sage ID	Actions
=cmd|' /C note...	Setup	CS	=cmd|' /C notepad'!'A1'	=cmd|' /C not...	⋮

Rows per page: 10 1-1 of 1

Figure 2 - Input Successfully Rendered in the Application



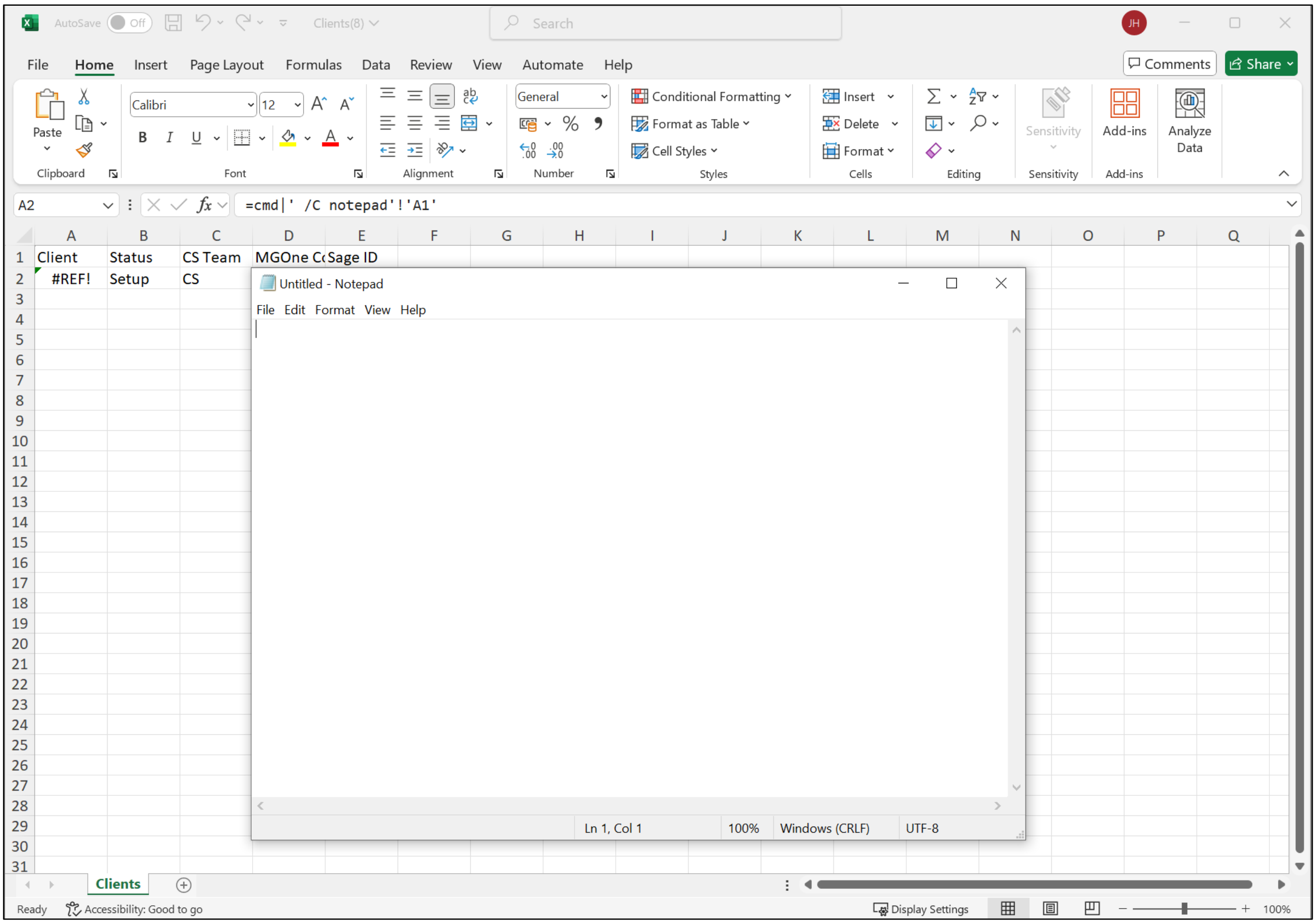
```
=cmd|' /C notepad'!'A1'
```

	A	B	C	D	E
1	Client	Status	CS Team	MGOne C	Sage ID
2	=cmd|'/C notepad'!'A1'	Setup	CS		

Microsoft Excel

Remote data not accessible.
To access this data Excel needs to start another application. Some legitimate applications on your computer could be used maliciously to spread viruses or damage your computer. Only click Yes if you trust the source of this workbook and you want to let the workbook start the application.
Start application 'CMD.EXE'?

Figure 3 - Microsoft Excel Warning Message



```
=cmd|' /C notepad'!'A1'
```

	A	B	C	D	E
1	Client	Status	CS Team	MGOne C	Sage ID
2	#REF!	Setup	CS		

Figure 4 – Notepad.exe Executing Successfully

The following table details the areas of the applications and inputs affected by this issue.

Frontend Endpoint	API Endpoint	Parameters
https://medigoldcore.com/client/client-list	https://app-osi-client-prd.azurewebsites.net/clients	Company Name Display Name MGOne Company Reference Sage ID
https://medigoldcore.com/employee/employee-list	https://app-osi-employee-prd.azurewebsites.net/employee/create	Surname forename
https://medigoldcore.com/clientgroup/group-admin/	https://app-osi-client-prd.azurewebsites.net/group	Group Name
https://medigoldcore.com/team/team-list	https://app-osi-client-prd.azurewebsites.net/teams	Team Name
https://medigoldcore.com/clinic/clinician-list	https://app-osi-clinic-prd.azurewebsites.net/clinician	Forename Surname
https://medigoldcore.com/clinic/clinic-list	https://app-osi-clinic-prd.azurewebsites.net/clinic	Clinic Name
https://medigoldcore.com/clinic/equipment-list	https://app-osi-clinic-prd.azurewebsites.net/equipment	Equipment Name
https://medigoldcore.com/product/product-list	https://app-osi-product-prd.azurewebsites.net/product/create	Product Name
https://medigoldcore.com/product/element-list	https://app-osi-product-prd.azurewebsites.net/productElement/create	Product Element Name
https://medigoldcore.com/contract/contract-list	https://app-osi-contract-prd.azurewebsites.net/contract/create	Contract Name
https://medigoldcore.com/systemadmin/session-cancellation-reasons	https://app-osi-diary-prd.azurewebsites.net/sessioncancellationreason/create	Reason Name
https://medigoldcore.com/product/group-list	https://app-osi-product-prd.azurewebsites.net/productGroup/create	Product Group Name
https://medigoldcore.com/systemadmin/letter-templates-list	https://app-osi-product-prd.azurewebsites.net/letter-template	Letter Template Name
https://medigoldcore.com/systemadmin/email-templates-list	https://app-osi-product-prd.azurewebsites.net/email-template	Email Template Name

7.3.3. Recommendations

Implement appropriate input validation for fields which are later exported to Excel files. A technique commonly used by developers is to prefix cells with an apostrophe (‘) as this directs Excel to interpret values as text, however this only works for the XLS format and could negatively impact the interpretation of data.

Where possible, utilise a data whitelist or regex to reject input that does not conform, or is not required, to the required characters for the field(s) in question. For example, an account number field only requires numeric characters and there is no reason to permit special (£$%&*()) characters.

7.3.4. References

We have identified the following locations for further reading:

- **Contextis:** https://www.contextis.com/en/blog/comma-separated-vulnerabilities

7.3.5. Affected Systems

Affected Item
https://medigoldcore.com
https://app-osi-client-prd.azurewebsites.net
https:// app-osi-employee-prd.azurewebsites.net
https://app-osi-clinic-prd.azurewebsites.net
https://app-osi-product-prd.azurewebsites.net
https://app-osi-contract-prd.azurewebsites.net
https://app-osi-diary-prd.azurewebsites.net

7.4. Information Disclosure Via HTTP Header

Severity	Low
Impact	1
Likelihood	2
Fix Effort	Intermediate
Status	**Resolved**
Reference	4

7.4.1. Summary

This finding was found to be remediated after retesting.

The ability to accurately identify the vendor and version number of software running on a server is of high importance to an attacker. Knowing the exact type of server in use allows an attacker to determine any known vulnerabilities that may be used against it and will significantly aid in the selection of any appropriate exploits. A number of application servers will, by default, respond to a client request with data regarding the software running on the server, in the form of an HTTP response header.

7.4.2. Technical Information

During routine browsing of the Core application, it was found that API responses returned provided positive identification of the technologies in use. Furthermore, the specific version of the application was identified. This is demonstrated in the evidence provided below.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Mon, 13 Jan 2025 15:47:48 GMT
Server: Microsoft-IIS/10.0
Access-Control-Allow-Origin: https://medigoldcore.com
Set-Cookie:
ARRAffinity=f754d78ff09c216acc98b80e5772ef221e45af540bfcb723ed3c28909580ebc6;Path=/;HttpOnly;Secure;Dom
ain=app-osi-logging-prd.azurewebsites.net
Set-Cookie:
ARRAffinitySameSite=f754d78ff09c216acc98b80e5772ef221e45af540bfcb723ed3c28909580ebc6;Path=/;HttpOnly;Sa
meSite=None;Secure;Domain=app-osi-logging-prd.azurewebsites.net
Vary: Accept-Encoding
x-ms-middleware-request-id: 00000000-0000-0000-0000-000000000000
Request-Context: appId=cid-v1:ed511f6b-bc68-4757-9861-29b859eb38be
X-Powered-By: ASP.NET
Content-Length: 43
```

7.4.3. Recommendations

The IIS server should be reconfigured to prevent software version information from being returned in application responses. This can be performed using the “URL Rewrite” HTTP module provided by Microsoft for this purpose:

```
<rewrite>
       <outboundRules rewriteBeforeCache=”true”>
          <rule name=”Remove Server Header”>
              <match serverVariable=”RESPONSE_Server” pattern=”.+” />
              <action type=”Rewrite” value=”” />
          </rule>
```

```
        <ouboundRules>
</rewrite>
```

The ASP.net version number can be removed using the below configuration within the “web.config” file:

```
<system.web>
        <httpRuntime enableVersionHeader=”false”> />
</system.web>
```

Whilst the ”X-Powered-By” header can be removed with the following configuration, also within the “web.config” file:

```
<htmlProtocol>
        <customHeaders>
                <remove name=”X-Powered-By” />
        </customHeaders>
</htmlProtocol>
```

7.4.4. References

We have identified the following locations for further reading:

- **Microsoft:** https://docs.microsoft.com/en-gb/archive/blogs/benjaminperkins/change-or-modify-a-response-header-value-using-url-rewrite
- **CWE-200:** https://cwe.mitre.org/data/definitions/200.html
- **OWASP:** https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html#server

7.4.5. Affected Systems

Affected Item	Path
https://app-osi-accesshub-prd.azurewebsites.net	All Paths
https://app-osi-logging-prd.azurewebsites.net	/logInformation
https://app-policy-api-prd.azurewebsites.net	All Paths
https://app-osi-employee-prd.azurewebsites.net	/employee/unit-link/a1c9d4c2-bb52-4b40-8931-eb0685cbf7e8 /employees/unit/927e46ac-6f1f-47b4-a33e-1b74407972c4/a1c9d4c2-bb52-4b40-8931-eb0685cbf7e8
https://app-osi-utility-prd.azurewebsites.net	/iddCodes

7.5. SSL Weak Cipher Suites

Severity	**Low**
Impact	2
Likelihood	2
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	5

7.5.1. Summary

This finding was found to be remediated after retesting.

SSL certificates are used to encrypt communications and verify identity. To transfer data securely TLS/SSL uses one or more cipher suites, these are used to establish a secure connection. If the connection permits, an attacker could purposely specify to downgrade a connection to use the weakest supported version. Weaknesses in either the protocol or cipher suite, could allow a well-position attacker to Man-in-the-middle (MiTM) traffic, to derive clear-text data from the HTTPS communications which could include cookies, usernames and passwords.

7.5.2. Technical Information

The following cipher suites used Cipher Block Chaining (CBC). These are vulnerable to oracle padding attacks. This attack enables a malicious actor to decrypt the contents of the data, without knowing the key.

Key Exchange	Encryption	Bits	Cipher Suite Name (IANA/RFC)
ECDH 256	AES	128	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDH 256	AES	256	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

7.5.3. Recommendations

Weak methods of encryption should be disabled to ensure only secure cipher suites are used. This can be achieved, by removing the outdated ciphers and using secure alternatives. An example of a secure cipher configuration list can be found below.

```
ssl_ciphers           ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-
AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;
```

7.5.4. References

We have identified the following locations for further reading:

- **Mozilla:** https://ssl-config.mozilla.org/
- **Github:** https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.md

7.5.5. Affected Systems

Affected Item
https://medigoldcore.com/

7.6. Missing HTTP Security Headers

Severity	**Low**
Impact	2
Likelihood	1
Fix Effort	**Easy**
Status	**Resolved**
Reference	6

7.6.1. Summary

This finding was found to be remediated after retesting.

HTTP headers which add additional security to applications were not being utilised by the web application. Browsers support a number of HTTP headers which inform them what resources they should trust and whether or not to automatically sanitise/block potentially malicious code. These features help further enhance the security of web applications and should be employed where possible. Below is an overview of each of these security headers:

Content-Security-Policy - This header allows web site administrators to control resources the user agent is allowed to load for a given page. With a few exceptions, policies mostly involve specifying server origins and script endpoints. This helps guard against cross-site scripting attacks (XSS).

X-Content-Type-Options - This header is a marker used by the server to indicate that the MIME types advertised in the Content-Type headers should not be changed and be followed. This was introduced to block content sniffing that was happening and could transform non-executable MIME types into executable MIME types.

X-Frame-Options - This header can be used to indicate whether or not a browser should be allowed to render a page in a <frame>, <iframe>, <embed> or <object>. Sites can use this to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites.

7.6.2. Technical Information

The following headers were not configured on the web application:

- Content-Security-Policy
- X-Content-Type-Options

```
HTTP/2 200
date: Mon, 20 Jan 2025 15:07:57 GMT
content-type: text/html
content-length: 77213
vary: Accept-Encoding
content-md5: qmP+WVoXCjkS7eilplI5nA==
last-modified: Mon, 06 Jan 2025 18:12:02 GMT
etag: "0x8DD2E7D9EDF1D5A"
vary: Origin
x-ms-request-id: f188f6f6-801e-007a-46c1-6a9d20000000
x-ms-version: 2018-03-28
x-azure-ref: 20250120T150757Z-r15774cf85dpf2rdhC1LON7f5c0000000frg0000000187gf
x-fd-int-roxy-purgeid: 0
x-cache: TCP_HIT
```

```
x-cache-info: L1_T2
x-frame-options: SAMEORIGIN
accept-ranges: bytes
```

7.6.3. Recommendations

The recommended HTTP security headers should be configured on the web application and API.

The Content-Security-Policy header should be configured. This header allows very granular options to be configured, the OWASP document within references can help determine a suitable value.

The X-Content-Type-Options header should be configured with a value of 'nosniff' to instruct the browser to honour the specified content-type. E.g.

```
X-Content-Type-Options: nosniff
```

The X-Frame-Options header should be configured with a value of 'deny' to prevent the application being rendered within an iframe etc., E.g.

```
X-Frame-Options: deny
```

7.6.4. References

We have identified the following locations for further reading:

- **Mozilla:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
- **Mozilla:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection
- **Microsoft:** https://msdn.microsoft.com/en-us/library/gg622941%28v=vs.85%29.aspx
- **Mozilla:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options
- **Mozilla:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy
- **OWASP:** https://www.owasp.org/index.php/OWASP_Secure_Headers_Project

7.6.5. Affected Systems

Affected Item
https://medigoldcore.com/

7.7. Insufficient Input Validation

Severity	**Low**
Impact	2
Likelihood	2
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	7

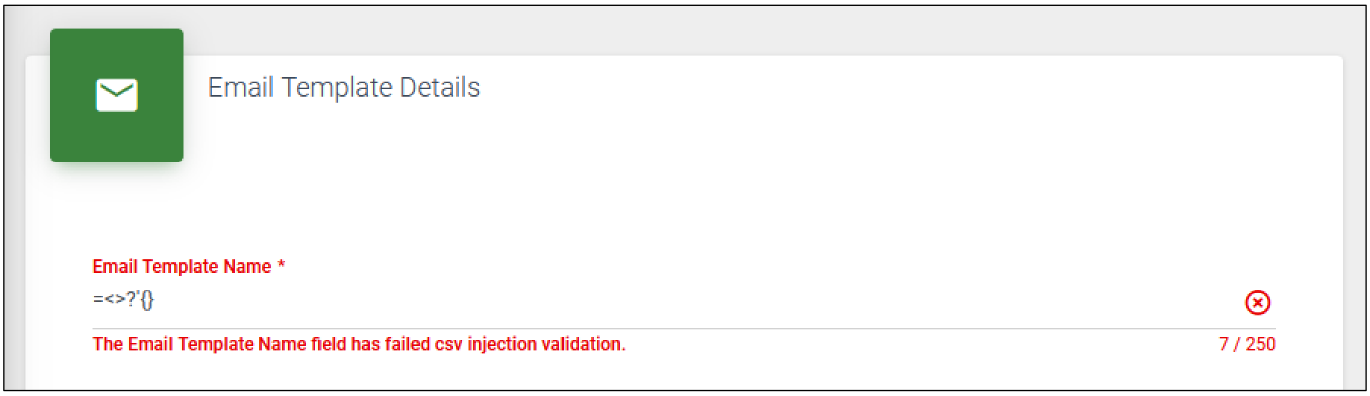
7.7.1. Summary

This finding was found to be remediated after retesting.

Input validation is the testing of any input supplied by a user or application. Input validation prevents improperly formed data from entering an application. Input validation should happen as early as possible in the data flow, ideally as soon as the data is received. Failure to enforce sufficient input validation would enable an attacker to input malicious data which could lead to further vulnerabilities.

7.7.2. Technical Information

The application was found to implement client-side validation for user input protecting inputs from the use of special characters often used in injections attacks.



Email Template Details

Email Template Name *

=<>?'{}

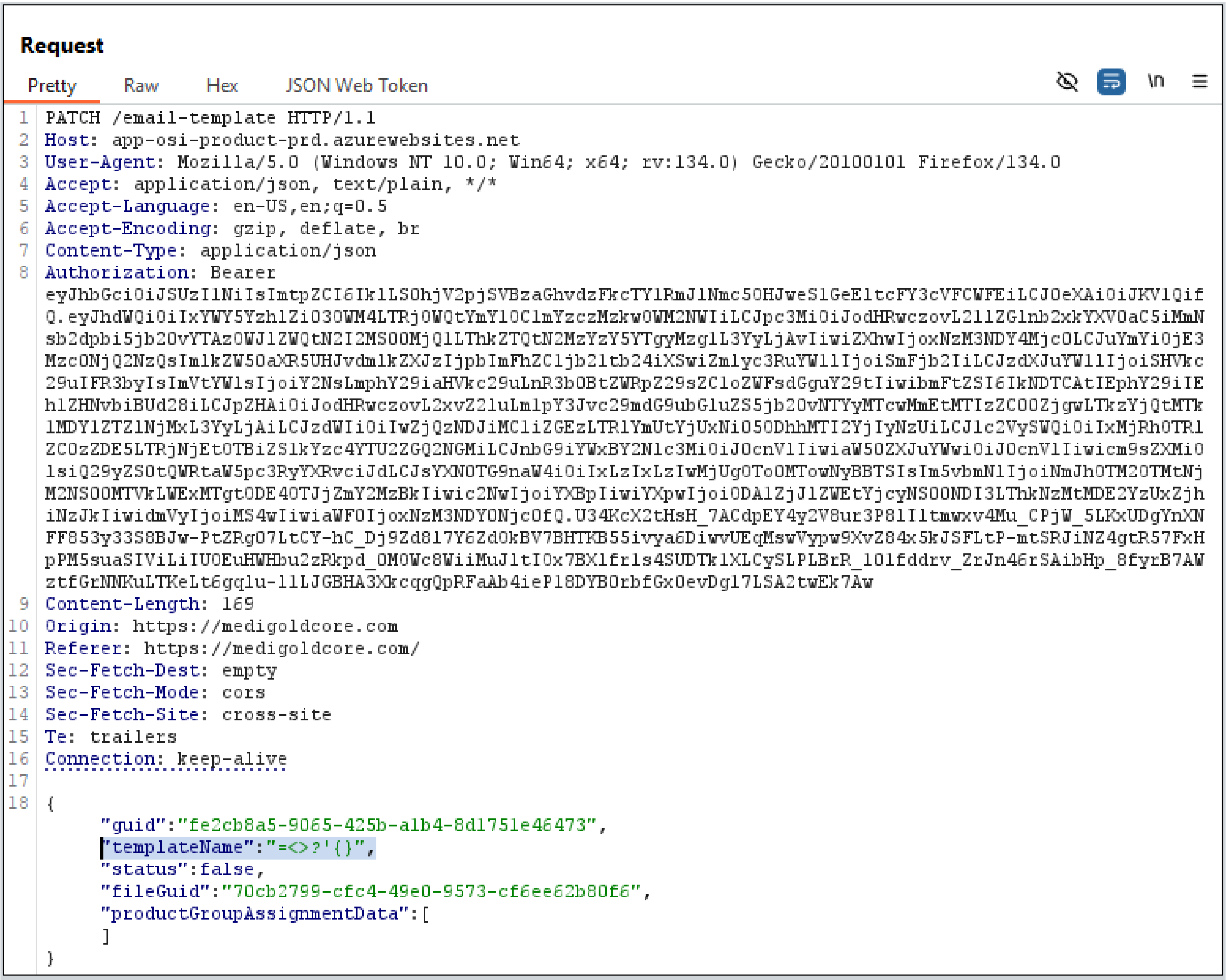
The Email Template Name field has failed csv injection validation.

7 / 250

Figure 5 - Client-Side Input Validation

It was possible to bypass restrictions on user input by capturing and modifying the API request to either create or modify a record. The issue was found to be prevalent across the entire application and associated APIs when creating records.

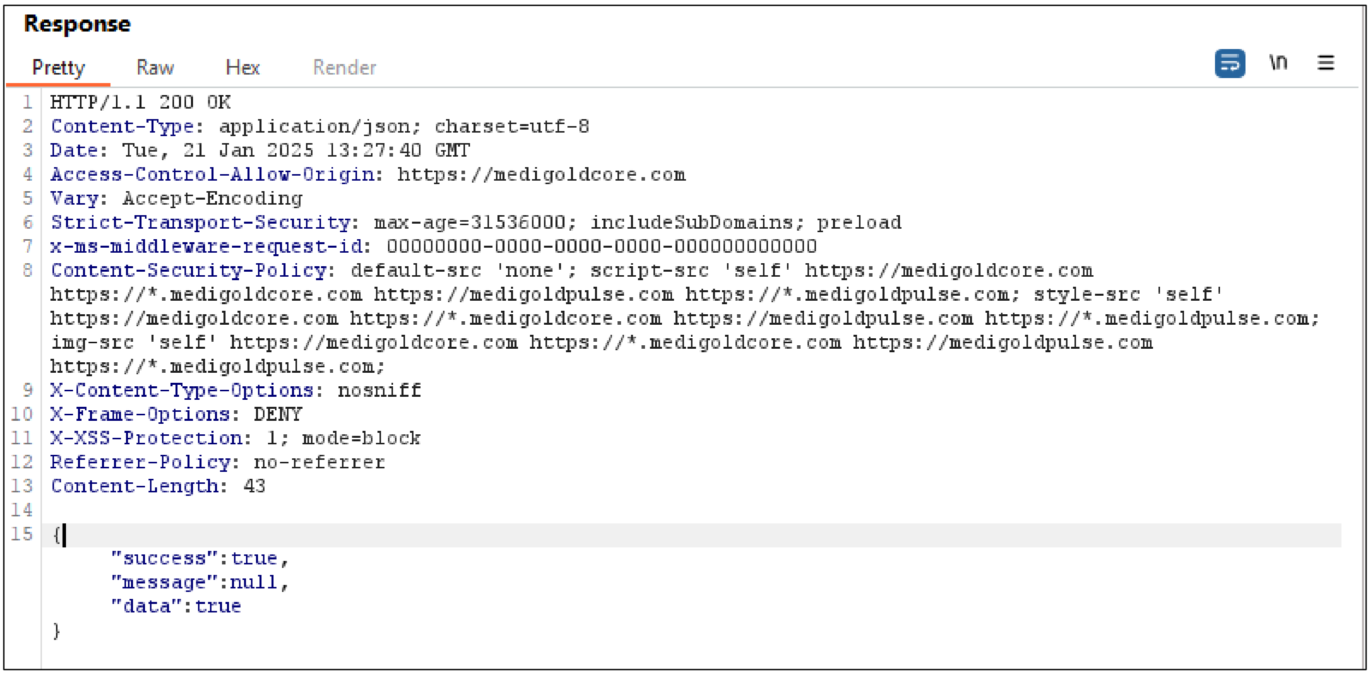
The example demonstrates the issue when creating an email template:



```
PATCH /email-template HTTP/1.1
Host: app-osi-product-prd.azurewebsites.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101 Firefox/134.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6Ik1LSOhjV2pjSVBzaGhvdzFkcTY1RmJ1Nmc5OHJweS1GeE1tcFY3cVFCWFEiLCJ0eXAiOiJKV1Qif
Q.eyJhdWQiOiIxYWY5YzhlZi03OWM4LTRjOWQtYmY1OC1mYzczMzkwOWM2NWIiLCJpc3MiOiJodHRwczovL2llZGlnb2xkYXV0aC5iMmN
sb2dpbi5jb20vYTAzOWJlZWQtN2I2MS00MjQ1LThkZTQtN2MzYzY5YTgyMzglL3YyLjAvIiwiZXhwIjoxNzM3NDY4MjcOLCJuYmYiOjE3
MzcONjQ2NzQsImlkZW5OaXR5UHJvdmlkZXJzIjpbImFhZC1jb21tb24iXSwiZmlyc3RuYW1lIjoiSmFjb2IiLCJzdXJuYW1lIjoiSHVkc
29uIFR3byIsImVtYWlsIjoiY2NsLmphY29iaHVkc29uLnR3bOBtZWRpZ29sZC1oZWFsdGguY29tIiwibmFtZSI6IkNDTCAtIEphY29iIE
hlZHNvbiBUd28iLCJpZHAiOiJodHRwczovL2xvZ2luLmlpY3Jvc29mdG9ubGluZS5jb20vNTYyMTcwMmEtMTIzZC00ZjgwLTkzYjQtMTk
lMDYlZTZlNjMxL3YyLjAiLCJzdWIiOiIwZjQzNDJiMC1iZGEzLTRlYmUtYjUxNi05ODhhMTI2YjIyNzUiLCJlc2VySWQiOiIxMjRhOTRl
ZCOzZDE5LTRjNjEtOTBiZS1kYzc4YTU2ZGQ2NGMiLCJnbG9iYWxBY2Nlc3MiOiJ0cnVlIiwiaW5OZXJuYWwiOiJ0cnVlIiwicm9sZXMiO
lsiQ29yZSOtQWRtaW5pc3RyYXRvciJdLCJsYXNOTG9naW4iOiIxLzIxLzIwMjUgOToOMTowNyBBTSIsIm5vbmNlIjoiNmJhOTM2OTMtNj
M2NSOOMTVkLWExMTgtODE4OTJjZmY2MzBkIiwic2NwIjoiYXBpIiwiYXpwIjoiODAlZjJlZWEtYjcyNSOONDI3LThkNzMtMDE2YzUxZjh
iNzJkIiwidmVyIjoiMS4wIiwiaWFOIjoxNzM3NDYONjcOfQ.U34KcX2tHsH_7ACdpEY4y2V8ur3P8lIltmwxv4Mu_CPjW_5LKxUDgYnXN
FF853y33S8BJw-PtZRg07LtCY-hC_Dj9Zd817Y6Zd0kBV7BHTKB55ivya6DiwvUEqMswVypw9XvZ84x5kJSFLtP-mtSRJiNZ4gtR57FxH
pPM5suaSIViLiIU0EuHWHbu2zRkpd_OM0Wc8WiiMuJltI0x7BXlfrls4SUDTk1XLCySLPLBrR_1O1fddrv_ZrJn46rSAibHp_8fyrB7AW
ztfGrNNKuLTKeLt6gqlu-l1LJGBHA3XkcqgQpRFaAb4ieP18DYB0rbfGx0evDg17LSA2twEk7Aw
Content-Length: 169
Origin: https://medigoldcore.com
Referer: https://medigoldcore.com/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: cross-site
Te: trailers
Connection: keep-alive

{
    "guid":"fe2cb8a5-9065-425b-a1b4-8d1751e46473",
    "templateName":"=<>?'{}",
    "status":false,
    "fileGuid":"70cb2799-cfc4-49e0-9573-cf6ee62b80f6",
    "productGroupAssignmentData":[
    ]
}
```

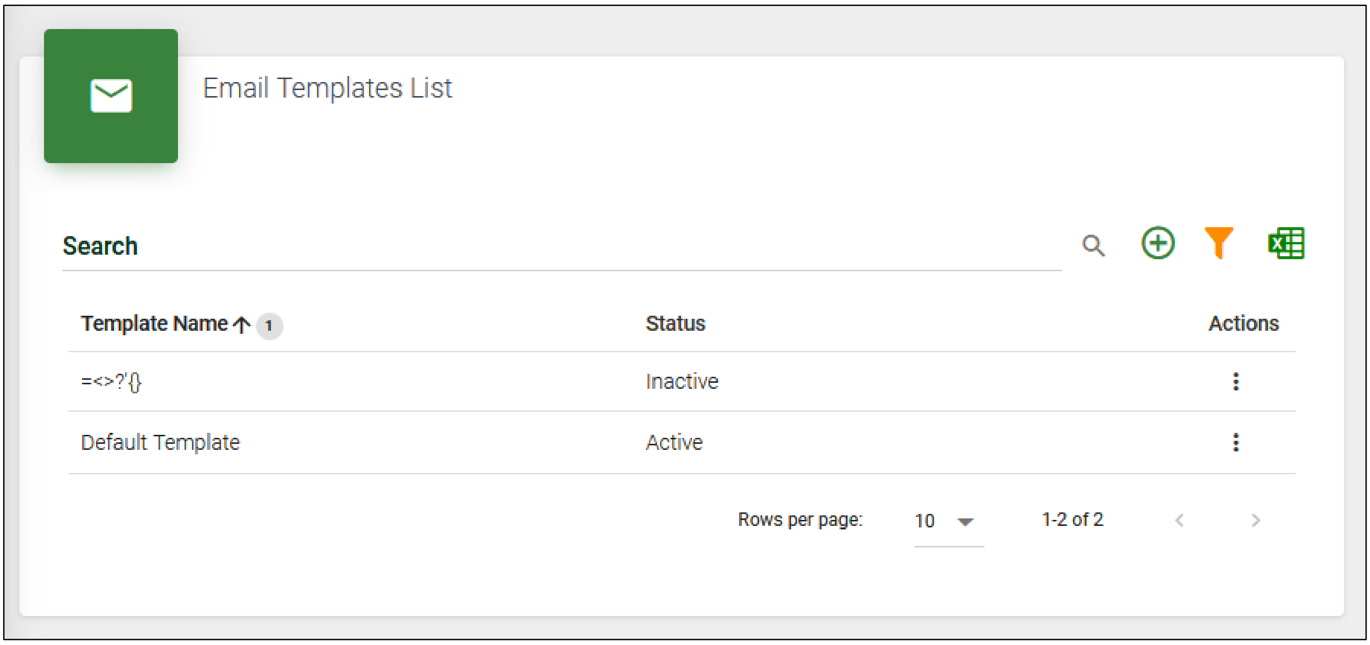
Figure 6 - Modified API Request to Bypass Character Restrictions



```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 21 Jan 2025 13:27:40 GMT
Access-Control-Allow-Origin: https://medigoldcore.com
Vary: Accept-Encoding
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
x-ms-middleware-request-id: 00000000-0000-0000-0000-000000000000
Content-Security-Policy: default-src 'none'; script-src 'self' https://medigoldcore.com
https://*.medigoldcore.com https://medigoldpulse.com https://*.medigoldpulse.com; style-src 'self'
https://medigoldcore.com https://*.medigoldcore.com https://medigoldpulse.com https://*.medigoldpulse.com;
img-src 'self' https://medigoldcore.com https://*.medigoldcore.com https://medigoldpulse.com
https://*.medigoldpulse.com;
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Referrer-Policy: no-referrer
Content-Length: 43

{
     "success":true,
     "message":null,
     "data":true
}
```

Figure 7 - 200 Response Received



Template Name ↑ 1	Status	Actions
=<>?'{}	Inactive	⋮
Default Template	Active	⋮

Rows per page: 10 ▾ 1-2 of 2 ‹ ›

Figure 8 - Special Characters Rendered on Application

7.7.3. Recommendations

To address the issue of insufficient input validation, implement robust server-side validation to ensure all input conforms to expected formats, types, and ranges before processing. Define clear validation rules for each input field, such as restricting length, disallowing special characters if unnecessary, or using whitelists for acceptable input patterns. Employ structured validation libraries where available to minimise human error. Additionally, ensure the validation process rejects unexpected or malformed data with informative error messages for the user. Validate all inputs at the point of entry, regardless of the source, including API endpoints and external integrations, to mitigate the risk of injecting unintended or harmful data into the system. By implementing and adhering to these practices, you enhance the reliability and security of your application, reducing vulnerabilities stemming from improperly sanitised input.

7.7.4. References

We have identified the following locations for further reading:

- **OWASP:** https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

7.7.5. Affected Systems

Affected Item
https://medigoldcore.com
https://app-osi-client-prd.azurewebsites.net
https:// app-osi-employee-prd.azurewebsites.net
https://app-osi-clinic-prd.azurewebsites.net
https://app-osi-product-prd.azurewebsites.net
https://app-osi-contract-prd.azurewebsites.net
https://app-osi-diary-prd.azurewebsites.net

7.8. Sensitive Data in URL

Severity	**Low**
Impact	2
Likelihood	2
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	8

7.8.1. Summary

This finding was found to be remediated after retesting.

Sensitive information was found to be transmitted within the URL. It is not best practice to transmit sensitive data within URL's as they are often recorded my intermediate devices such as proxies. In addition, if an external web application was accessed from the affected URL, it is highly likely that the sensitive data would be sent to the external application via the Referrer HTTP header.

7.8.2. Technical Information

The access_token was sent in a GET request which would be then stored in browser history. This gives an attacker the opportunity to try reuse the token. Not only this the token contains sensitive information about the user such as First name, Last name, Email and UserId.

```
GET
/hub/access?access_token=eyJhbGciOiJSUzI1NiIsImtpZCI6IklLS0hjV2pjSVBzaGhvdzFkcTY1RmJlNmc5OHJweS1GeEltcF
Y3cVFCWFEiLCJ0eXAiOiJKV1QifQ.eyJhdWQiOiJjZTQ4MWQ5ZC05ZGE4LTRjYzAtYWVjYy04ZGZhZDBkNWUwY2QiLCJpc3MiOiJodH
RwczovL21lZGlnb2xkYXV0aC5iMmNsb2dpbi5jb20vYTAzOWJlZWQtN2I2MS00MjQ1LThkZTQtN2MzYzY5YTgyMzg1L3YyLjAvIiwiZ
XhwIjoxNzM3MTE3OTExLCJuYmYiOjE3MzcxMTQzMTEsImlkZW50aXR5UHJvdmlkZXJzIjpbImFhZC1jb21tb24iXSwiZmlyc3RuYW1l
IjoiSmFjb2IiLCJzdXJuYW1lIjoiSHVkc29uIFR3byIsImVtYWlsIjoiY2NsLmphY29iaHVkc29uLnR3b0BtZWRpZ29sZC1oZWFsdGg
uY29tIiwibmFtZSI6IkNDTCAtIEphY29iIEh1ZHNvbiBUd28iLCJpZHAiOiJodHRwczovL2xvZ2luLm1pY3Jvc29mdG9ubGluZS5jb2
0vNTYyMTcwMmEtMTIzZC00ZjgwLTkzYjQtMTk1MDY1ZTZlNjMxL3YyLjAiLCJzdWIiOiIwZjQzNDJiMC1iZGEzLTRlYmUtYjUxNi05O
DhhMTI2YjIyNzUiLCJ1c2VySWQiOiIxMjRhOTRlZC0zZDE5LTRjNjEtOTBiZS1kYzc4YTU2ZGQ2NGMiLCJnbG9iYWxBY2Nlc3MiOiJ0
cnVlIiwiaW50ZXJuYWwiOiJ0cnVlIiwicm9sZXMiOlsiQ29yZS0tQWRtaW5pc3RyYXRvciJdLCJsYXN0TG9naW4iOiIxLzE3LzIwMjU
gODo1NzoyOCBBTSIsIm5vbmNlIjoiMDA0ZjY4ZWMtMjRjOS00YWQ4LTkwMjAtZTkzNDI0YTc3YzYwIiwic2NwIjoiYXBpIiwiYXpwIj
oiODA1ZjJlZWEtYjcyNS00NDI3LThkNzMtMDE2YzUxZjhiNzJkIiwidmVyIjoiMS4wIiwiaWF0IjoxNzM3MTE0MzExfQ.VMtvmEU-
YZqSoswIVZ8Zs8XYLrbP3o1slzPiLyja8pSwpuTQqIKDcd9aQou61TMA2JrhoKkXpkhxBZ5QBvnceqIdjgmxhy_4iV7aZvuHY2fbjvn
M6BxTB4k9A_mg1KYcZ_FyQLoWceNcrajGmLuwsJ4NSkdn6LZi5p0uS52BWLKjgNVLACvaFJNl6wUIHBa9w9wFyIprw5ZTozi8O_Z6vf
q4wm6Gfvcxpvs-Vjcii35kNPDYJo6zjWeQJpKLnSIsaeVdmMFKHLpQhKIm0AzkEzKIetVj2Nb2gwfVRuBFn8GCMBOhf-
ODWiLMWwdFW9uZNn02xqOUoZMs4bbbhMMMuQ HTTP/1.1
Host: app-osi-accesshub-prd.azurewebsites.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101 Firefox/134.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Sec-WebSocket-Version: 13
Origin: https://medigoldcore.com
Sec-WebSocket-Key: /Z1KlAoJKA7HRhNWodXDdw==
Connection: keep-alive, Upgrade
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: websocket
Sec-Fetch-Site: cross-site
Pragma: no-cache
Cache-Control: no-cache
```

```
Upgrade: websocket
```

7.8.3. Recommendations

Sensitive data should be transmitted within the body of HTTP requests. This could include HTTP headers such as cookies, however for significant amounts of data, it should be sent within the request body, e.g. via a POST request.

7.8.4. References

We have identified the following locations for further reading:

- **OWASP:** https://www.owasp.org/index.php/Information_exposure_through_query_strings_in_url

7.8.5. Affected Systems

Affected Item
https://app-osi-accesshub-prd.azurewebsites.net

7.9. Deprecated HTTP Security Header

Severity	**Low**
Impact	2
Likelihood	1
Fix Effort	**Easy**
Status	**Resolved**
Reference	9

7.9.1. Summary

This finding was found to be remediated after retesting.

The 'X-XSS-Protection' header is designed to detect and assist in protecting against cross-site scripting ("XSS") attacks. However, most modern browsers either do not implement or ignore this header value. Browser versions which do interpret the header have been found to introduce new security vulnerabilities into otherwise secure websites, due to poor implementation of XSS protection measures. It is no longer recommended to set the X-XSS Protection header, as to avoid these vulnerabilities.

As a result, the X-XSS-Protection header was deprecated in most modern web browsers, and other measures such as Content Security Policy (CSP) and input validation and sanitisation should be used to provide a more comprehensive defence against XSS attacks.

7.9.2. Technical Information

The following response was received from the API note the X-XSS-Protection header is incorrectly configured in line with recent security best practices.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Mon, 20 Jan 2025 11:10:46 GMT
Access-Control-Allow-Origin: https://medigoldcore.com
Vary: Accept-Encoding
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
x-ms-middleware-request-id: 72052ae3-1dbf-4d6f-aa70-dc0a21f9d78c
Request-Context: appId=cid-v1:2a08c510-bf67-4b3e-ace1-7c3fb54c72dd
Content-Security-Policy: default-src 'none'; script-src 'self' https://medigoldcore.com
https://*.medigoldcore.com https://medigoldpulse.com https://*.medigoldpulse.com; style-src 'self'
https://medigoldcore.com https://*.medigoldcore.com https://medigoldpulse.com
https://*.medigoldpulse.com; img-src 'self' https://medigoldcore.com https://*.medigoldcore.com
https://medigoldpulse.com https://*.medigoldpulse.com;
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Referrer-Policy: no-referrer
Content-Length: 85
```

7.9.3. Recommendations

The 'X-XSS Protection' Header was deprecated to encourage developers to adopt more robust XSS protection measures, such as Content Security Policy (CSP), which offers more comprehensive protection against XSS attacks and provides greater flexibility in configuring security policies for web applications

The ‘X-XSS-Protection’ header has been deprecated and should be either removed, or set to a value of ‘0’ e.g.:

```
X-XSS-Protection: 0;
```

7.9.4. References

We have identified the following locations for further reading:

- **Mozilla:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection

7.9.5. Affected Systems

Affected Item
https://app-osi-case-prd.azurewebsites.net
https://app-osi-client-prd.azurewebsites.net
https://app-osi-clinic-prd.azurewebsites.net
https://app-osi-contract-prd.azurewebsites.net
https://app-osi-diary-prd.azurewebsites.net
https://app-osi-employee-prd.azurewebsites.net
https://app-osi-file-management-prd.azurewebsites.net
https://app-osi-notification-and-consent-prd.azurewebsites.net
https://app-osi-product-prd.azurewebsites.net
https://app-osi-utility-prd.azurewebsites.net
https://app-policy-api-prd.azurewebsites.net

8. Medigold Gateway Application

This section details the web application issues that were identified during the testing engagement.

8.1. Lack of Rate Limiting

Severity	**Medium**
Impact	3
Likelihood	3
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	10

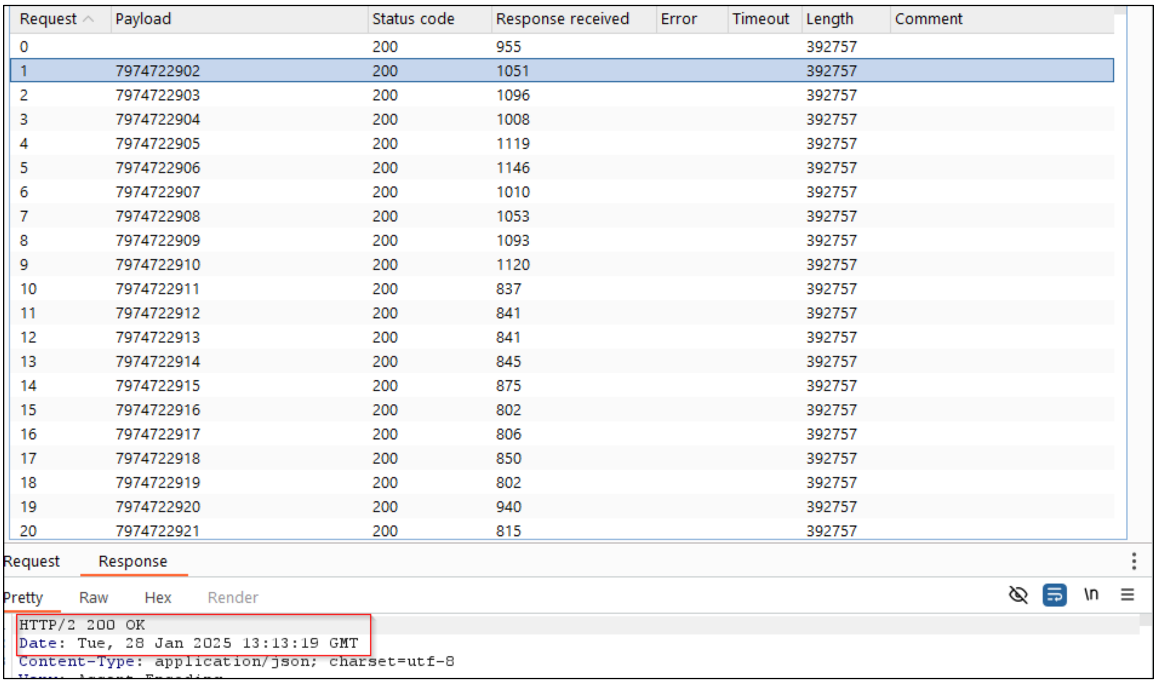
8.1.1. Summary

This finding was found to be remediated after retesting.

Rate-limiting is used in web applications and APIs to limit the number of requests that can be made to the endpoint, once this limit is reached service should be denied to the user. With a lack of rate-limiting an attacker is able to repeat requests an unlimited number of times leading to an associated drain on the resources of the server, potentially resulting in a denial-of-service ("DoS") attack.

8.1.2. Technical Information

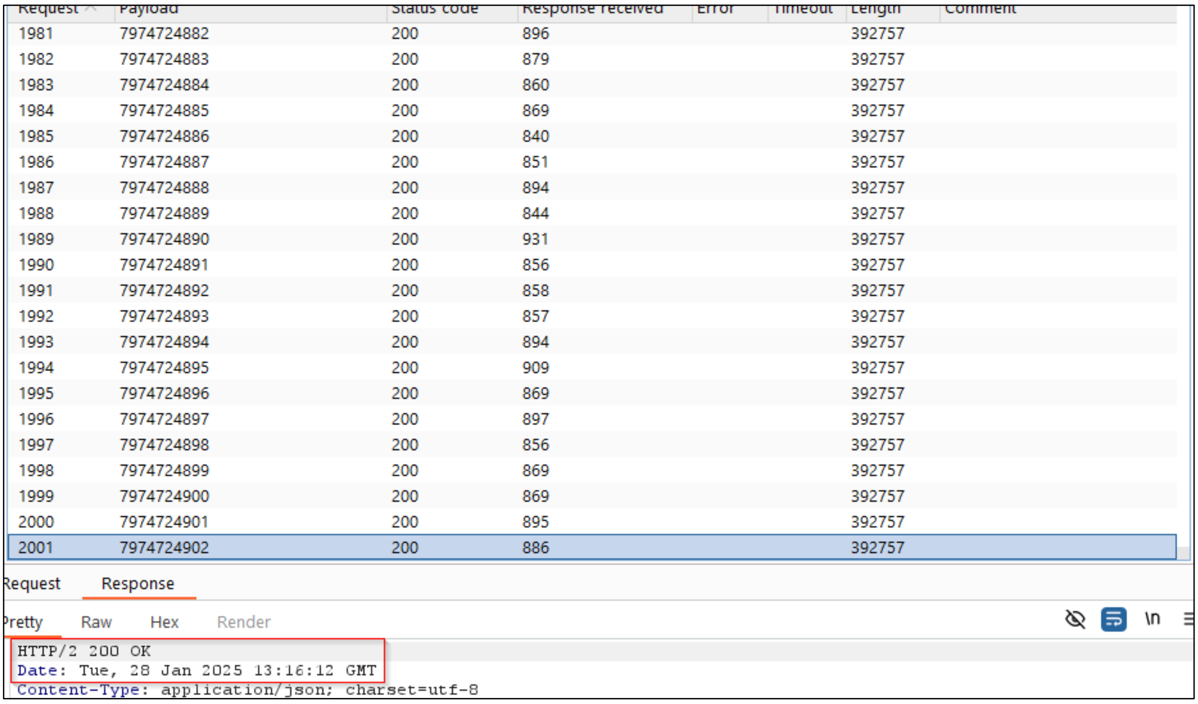
The web application did not appear to utilise any rate-limiting techniques. It was possible to repeat requests to the web application with no throttling. If exploited, this could fully compromise availability to the web application for all end-users. Over 2000 requests were sent in the space of approximately 180 seconds, an example of this can be seen below.



Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	955			392757	
1	7974722902	200	1051			392757	
2	7974722903	200	1096			392757	
3	7974722904	200	1008			392757	
4	7974722905	200	1119			392757	
5	7974722906	200	1146			392757	
6	7974722907	200	1010			392757	
7	7974722908	200	1053			392757	
8	7974722909	200	1093			392757	
9	7974722910	200	1120			392757	
10	7974722911	200	837			392757	
11	7974722912	200	841			392757	
12	7974722913	200	841			392757	
13	7974722914	200	845			392757	
14	7974722915	200	875			392757	
15	7974722916	200	802			392757	
16	7974722917	200	806			392757	
17	7974722918	200	850			392757	
18	7974722919	200	802			392757	
19	7974722920	200	940			392757	
20	7974722921	200	815			392757	

```
HTTP/2 200 OK
Date: Tue, 28 Jan 2025 13:13:19 GMT
Content-Type: application/json; charset=utf-8
```

Figure 9 - Start Time for Requests Sent



Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
1981	7974724882	200	896			392757	
1982	7974724883	200	879			392757	
1983	7974724884	200	860			392757	
1984	7974724885	200	869			392757	
1985	7974724886	200	840			392757	
1986	7974724887	200	851			392757	
1987	7974724888	200	894			392757	
1988	7974724889	200	844			392757	
1989	7974724890	200	931			392757	
1990	7974724891	200	856			392757	
1991	7974724892	200	858			392757	
1992	7974724893	200	857			392757	
1993	7974724894	200	894			392757	
1994	7974724895	200	909			392757	
1995	7974724896	200	869			392757	
1996	7974724897	200	897			392757	
1997	7974724898	200	856			392757	
1998	7974724899	200	869			392757	
1999	7974724900	200	869			392757	
2000	7974724901	200	895			392757	
2001	7974724902	200	886			392757	

```
HTTP/2 200 OK
Date: Tue, 28 Jan 2025 13:16:12 GMT
Content-Type: application/json; charset=utf-8
```

Figure 10 - 2000 Requests Successfully Sent

8.1.3. Recommendations

Implementing rate limiting involves controlling the number of requests that clients can make within a certain time frame. This helps prevent abuse, misuse, and denial-of-service attacks. There are a number of methods for introducing rate limiting to an application, these include:

- Token Bucket: Clients consume tokens at a fixed rate, and requests are only served if there are available tokens.
- Fixed Window: Counts the number of requests within a fixed time window (e.g., per second, per minute).
- Sliding Window: Maintains a rolling window of requests within a specified time period.

The chosen method should be enforced by the web server and when the rate limit is exceeded an appropriate error code should be provided to the user, for example ""HTTP status code 429 - Too Many Requests".

8.1.4. References

We have identified the following locations for further reading:

- **OWASP:** https://www.owasp.org/index.php/Denial_of_Service

8.1.5. Affected Systems

Affected Item
https://gateway.medigoldcore.com

8.2. Outdated Third-Party Libraries

Severity	**Medium**
Impact	3
Likelihood	3
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	11

8.2.1. Summary

This finding was found to be remediated after retesting.

Third-party JavaScript libraries, such as jQuery, can be utilised by applications in order to extend the application functionality and reduce development time through means of providing pre-packaged functions, removing the need to code all site content from scratch. In many cases, sites are built around a specific version of a library. The library itself may remain static for fear of introducing compatibility issues, or from fear of impacting application useability. Failure to include JavaScript libraries within organisational patching policy can result in outdated versions being used, leading to the introduction of security vulnerabilities and increasing the possibility of a successful attack being mounted against the application and its users.

8.2.2. Technical Information

The following vulnerable third-party libraries were identified on the web application:

Library	Version	Path
Bootstrap	4.3.1	/lib/bootstrap/dist/js/bootstrap.bundle.min.js
jQuery-Validation	1.17.0	/lib/jquery-validation/dist/jquery.validate.min.js
jQuery.datatables	1.10.23	/lib/datatables/datatables.min.js

8.2.3. Recommendations

Third-party libraries should be updated to the latest version available. Additionally, libraries utilised by the application should be incorporated into the organisation patching policy to ensure future updates are applied consistently and within a timely manner.

Upgrades should be implemented in a test environment prior to production to ensure no functionality faces compatibility issues.

8.2.4. References

We have identified the following locations for further reading:

- **Datatables:** https://datatables.net/
- **Jqueryvalidation:** https://jqueryvalidation.org/
- **Bootstrap:** https://getbootstrap.com/docs/versions/

8.2.5. Affected Systems

Affected Item
https://gateway.medigoldcore.com

8.3. Missing HTTP Security Headers

Severity	**Low**
Impact	2
Likelihood	1
Fix Effort	**Easy**
Status	**Resolved**
Reference	12

8.3.1. Summary

This finding was found to be remediated after retesting.

HTTP headers which add additional security to applications were not being utilised by the web application. Browsers support a number of HTTP headers which inform them what resources they should trust and whether or not to automatically sanitise/block potentially malicious code. These features help further enhance the security of web applications and should be employed where possible. Below is an overview of each of these security headers:

Content-Security-Policy - This header allows web site administrators to control resources the user agent is allowed to load for a given page. With a few exceptions, policies mostly involve specifying server origins and script endpoints. This helps guard against cross-site scripting attacks (XSS).

X-Content-Type-Options - This header is a marker used by the server to indicate that the MIME types advertised in the Content-Type headers should not be changed and be followed. This was introduced to block content sniffing that was happening and could transform non-executable MIME types into executable MIME types.

X-Frame-Options - This header can be used to indicate whether or not a browser should be allowed to render a page in a <frame>, <iframe>, <embed> or <object>. Sites can use this to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites.

8.3.2. Technical Information

The following headers were not configured on the web application:

- Content-Security-Policy
- X-Content-Type-Options
- X-Frame-Options

```
HTTP/2 200 OK
Date: Mon, 27 Jan 2025 10:45:24 GMT
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Strict-Transport-Security: max-age=3153600
Request-Context: appId=cid-v1:470354d8-6f09-4ac0-badd-80c93bb2721a
accept-ranges: bytes
```

8.3.3. Recommendations

The recommended HTTP security headers should be configured on the web application and API.

The Content-Security-Policy header should be configured. This header allows very granular options to be configured, the OWASP document within references can help determine a suitable value.

The X-Content-Type-Options header should be configured with a value of 'nosniff' to instruct the browser to honour the specified content-type. E.g.

```
X-Content-Type-Options: nosniff
```

The X-Frame-Options header should be configured with a value of 'deny' to prevent the application being rendered within an iframe etc., E.g.

```
X-Frame-Options: deny
```

8.3.4. References

We have identified the following locations for further reading:

- **Mozilla:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
- **Mozilla:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection
- **Microsoft:** https://msdn.microsoft.com/en-us/library/gg622941%28v=vs.85%29.aspx
- **Mozilla:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options
- **Mozilla:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy
- **OWASP:** https://www.owasp.org/index.php/OWASP_Secure_Headers_Project

8.3.5. Affected Systems

Affected Item
https://gateway.medigoldcore.com

8.4. Cookie Weaknesses

Severity	**Low**
Impact	2
Likelihood	1
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	13

8.4.1. Summary

This finding was found to be remediated after retesting.

Weaknesses were identified with the cookie configuration options set by the application. Cookies are small pieces of data sent between a user's browser and the backend webserver. One of the most common uses for cookies is to track state, i.e. a user's session. This is because HTTP is a stateless protocol and without additional information (e.g. a cookie), the server would be unable to differentiate between different users.

Cookies are set within a user's browser upon receiving a "Set-Cookie" HTTP header within a response. The "Set-Cookie" header enables the server to inform the browser of the cookies name and value. In addition to this required information, there are additional flags which can be set to inform the browser how to handle the cookie. Two common security flags are:

HTTPOnly – This flag informs the browser to not allow client-side scripts (such as JavaScript) to be able to access cookies. This reduces the risk that a malicious script could steal cookies and thus minimises the chance of user sessions being hijacked.

Secure – This flag informs the browser to only ever send the cookie when communication is occurring over an encrypted channel (HTTPS). This reduces the risk that the cookie will become known to local threat actors eavesdropping on communications.

8.4.2. Technical Information

The following table details the affected cookies as well as what security flags were missing:

Cookie	HttpOnly flag	Secure flag
ApplicationGatewayAffinityCORS	No	Yes
ApplicationGatewayAffinity	No	No

8.4.3. Recommendations

All cookies should have both the HTTPOnly and Secure flags set, specifically those which relate to session data. Additionally, it is essential to address the issue of duplicate cookies. It is recommended to ensure that each cookie is uniquely defined, and any duplicates should be eliminated to avoid confusion and reduce the risk of session hijacking or other related attacks. Proper cookie management practices should be implemented, ensuring that only one version of each cookie is set, and redundant or conflicting cookies are removed.

8.4.4. References

We have identified the following locations for further reading:

- **CWE-1004:** https://cwe.mitre.org/data/definitions/1004.html
- **CWE-614:** https://cwe.mitre.org/data/definitions/614.html
- **OWASP:** https://www.owasp.org/index.php/HttpOnly
- **OWASP:** https://www.owasp.org/index.php/SecureFlag

8.4.5. Affected Systems

Affected Item
https://gateway.medigoldcore.com

8.5. SSL Weak Cipher Suites

Severity	**Low**
Impact	2
Likelihood	2
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	14

8.5.1. Summary

This finding was found to be remediated after retesting.

SSL certificates are used to encrypt communications and verify identity. To transfer data securely TLS/SSL uses one or more cipher suites, these are used to establish a secure connection. If the connection permits, an attacker could purposely specify to downgrade a connection to use the weakest supported version. Weaknesses in either the protocol or cipher suite, could allow a well-position attacker to Man-in-the-middle (MiTM) traffic, to derive clear-text data from the HTTPS communications which could include cookies, usernames and passwords.

8.5.2. Technical Information

The following cipher suites used Cipher Block Chaining (CBC). These are vulnerable to oracle padding attacks. This attack enables a malicious actor to decrypt the contents of the data, without knowing the key.

Key Exchange	Encryption	Bits	Cipher Suite Name (IANA/RFC)
ECDH 253	AES	256	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDH 253	AES	128	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

8.5.3. Recommendations

Weak methods of encryption should be disabled to ensure only secure cipher suites are used. This can be achieved, by removing the outdated ciphers and using secure alternatives. An example of a secure cipher configuration list can be found below.

```
ssl_ciphers          ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;
```

8.5.4. References

We have identified the following locations for further reading:

- **Microsoft:** https://learn.microsoft.com/en-us/dotnet/standard/security/vulnerabilities-cbc-mode
- **Github:** https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.md

8.5.5. Affected Systems

Affected Item
https://gateway.medigoldcore.com

9. Medigold Pulse Application

This section details the web application issues that were identified during the testing engagement.

9.1. Lack of Rate Limiting

Severity	**Medium**
Impact	3
Likelihood	3
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	15

9.1.1. Summary

This finding was found to be remediated after retesting.

Rate-limiting is used in web applications and APIs to limit the number of requests that can be made to the endpoint, once this limit is reached service should be denied to the user. With a lack of rate-limiting an attacker is able to repeat requests an unlimited number of times leading to an associated drain on the resources of the server, potentially resulting in a denial-of-service ("DoS") attack.

9.1.2. Technical Information

Rate-limiting is used in APIs to limit the number of requests that can be made to the endpoint, once this limit is reached service should be denied to the user. With a lack of rate-limiting an attacker is able to repeat requests an unlimited number of times leading to an associated drain on the resources of the server, potentially resulting in a denial-of-service ("DoS") attack.

The CCL team Identified two points of functionality lacking Rate limiting implementation.

The first element was the creating of a user:

```
Request:
POST /apps/user/provision HTTP/1.1
Host: gateway.api.medigoldcore.com
--Redacted--
Accept-Language: en-GB,en;q=0.9
Sec-Ch-Ua: "Chromium";v="131", "Not_A Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.6778.86 Safari/537.36
Accept: application/json, text/plain, */*
Content-Type: application/json
Origin: https://medigoldpulse.com
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://medigoldpulse.com/
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
Connection: keep-alive
```

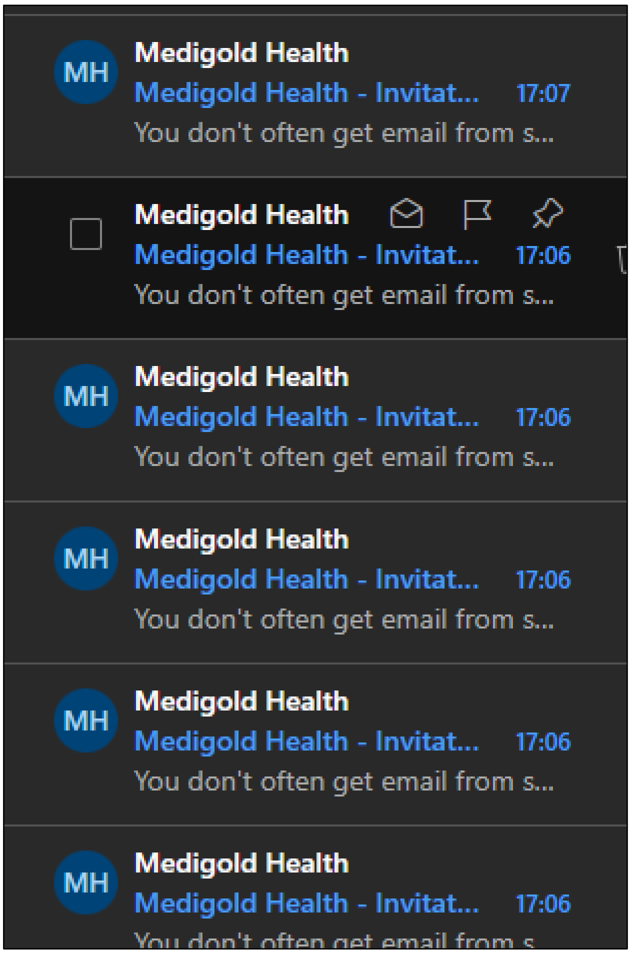
```
{"firstName":"CCL","lastName":"TESTING","email":"niall.aiken+20@cclsolutionsgroup.com","units":["939E6B
90-5A46-4D33-BD2C-555F75224EBE"],"client":"88AAE309-DAD2-46EF-B0FD-
8F4C5913E388","applicationName":"Consultation","roleNk":"LU","sendInvite":true,"inviteUrl":"management-
referrals"}
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8
Date: Tue, 14 Jan 2025 17:06:58 GMT
Access-Control-Allow-Origin: https://medigoldpulse.com
Set-Cookie:
ARRAffinity=6f77be28b3004d4cc1569dedf410c508dec950fcbe1d403138b14e18b993b928;Path=/;HttpOnly;Secure;Dom
ain=gateway.api.medigoldcore.com
Set-Cookie:
ARRAffinitySameSite=6f77be28b3004d4cc1569dedf410c508dec950fcbe1d403138b14e18b993b928;Path=/;HttpOnly;Sa
meSite=None;Secure;Domain=gateway.api.medigoldcore.com
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
x-ms-middleware-request-id: 00000000-0000-0000-0000-000000000000
Content-Security-Policy: default-src 'none'; script-src 'self' https://medigoldcore.com
https://*.medigoldcore.com https://medigoldpulse.com https://*.medigoldpulse.com; style-src 'self'
https://medigoldcore.com https://*.medigoldcore.com https://medigoldpulse.com
https://*.medigoldpulse.com; img-src 'self' https://medigoldcore.com https://*.medigoldcore.com
https://medigoldpulse.com https://*.medigoldpulse.com;
X-Content-Type-Options: nosniff
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
X-XSS-Protection: 1; mode=block
Referrer-Policy: no-referrer
Referrer-Policy: no-referrer
X-Permitted-Cross-Domain-Policies: none
Content-Length: 43

{"success":true,"message":null,"data":null}
```

Each request generates an email invite which could cause an overload of spam and draining of resources.



Medigold Health
Medigold Health - Invitat... 17:07
You don't often get email from s...

Medigold Health
Medigold Health - Invitat... 17:06
You don't often get email from s...

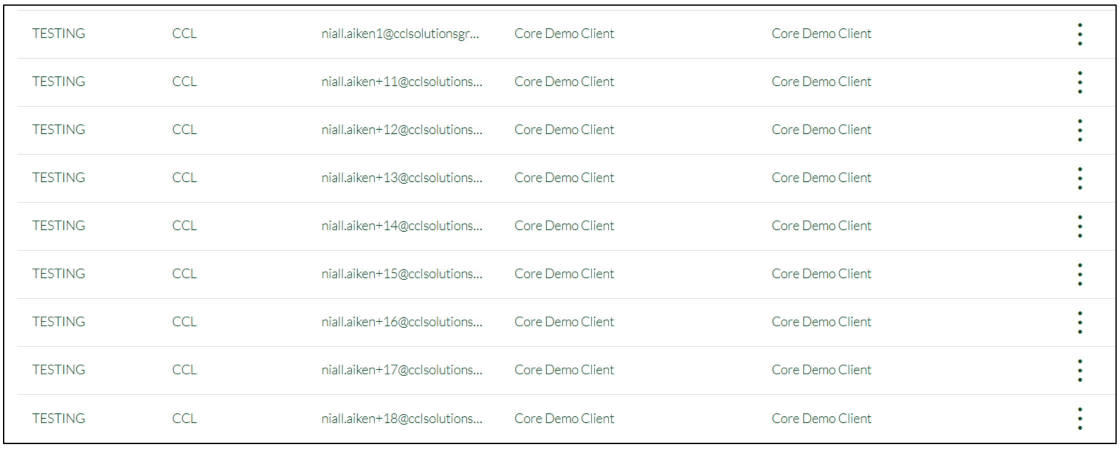
Medigold Health
Medigold Health - Invitat... 17:06
You don't often get email from s...

Medigold Health
Medigold Health - Invitat... 17:06
You don't often get email from s...

Medigold Health
Medigold Health - Invitat... 17:06
You don't often get email from s...

Medigold Health
Medigold Health - Invitat... 17:06
You don't often get email from s

Figure 11 – Email Spam Received



TESTING	CCL	niall.aiken1@cclsolutionsgr...	Core Demo Client	Core Demo Client	⋮
TESTING	CCL	niall.aiken+11@cclsolutions...	Core Demo Client	Core Demo Client	⋮
TESTING	CCL	niall.aiken+12@cclsolutions...	Core Demo Client	Core Demo Client	⋮
TESTING	CCL	niall.aiken+13@cclsolutions...	Core Demo Client	Core Demo Client	⋮
TESTING	CCL	niall.aiken+14@cclsolutions...	Core Demo Client	Core Demo Client	⋮
TESTING	CCL	niall.aiken+15@cclsolutions...	Core Demo Client	Core Demo Client	⋮
TESTING	CCL	niall.aiken+16@cclsolutions...	Core Demo Client	Core Demo Client	⋮
TESTING	CCL	niall.aiken+17@cclsolutions...	Core Demo Client	Core Demo Client	⋮
TESTING	CCL	niall.aiken+18@cclsolutions...	Core Demo Client	Core Demo Client	⋮

Figure 12- Confirmation of Accounts Created on The User Panel

The second instance of lack of rate limiting was on the forgot password functionality

Request:

```
POST
/medigoldauth.onmicrosoft.com/B2C_1A_Signin_HRD_Saml_Extensions/SelfAsserted?tx=StateProperties=eyJUSUQ
iOiJhN2Y1YjVmOC05MTc4LTQwN2MtYWFmMy1jZmEwOWUwNWIzZGYifQ&p=B2C_1A_Signin_HRD_Saml_Extensions HTTP/1.1

Host: medigoldauth.b2clogin.com
--redact--
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Csrf-Token:
QTdVV0ROakFndW1oR0IyY2N0byt2MWhwVjZuMEtnZTRNclNvUHdRUmM4bFh2OWtNaEowUGV2Ky9UTW9VTUF2WWp4eWxoQzFZbnVnTzJ
LcEFYR3p6OHc9PTsyMDI1LTAxLTE1VDEwOjA4OjI1LjI3MTg5NDZaO1JUcjJOQXAzZlJ5b1VWL0lhWWQzTnc9PTt7IlRhcmdldEVudG
l0eSI6IkZvcmdvdFBhc3N3b3JkRXhjaGFuZ2UiLCJPcmNoZXN0cmF0aW9uU3RlcCI6MX0=
X-Requested-With: XMLHttpRequest
Content-Length: 101
Origin: https://medigoldauth.b2clogin.com
Referer:
https://medigoldauth.b2clogin.com/medigoldauth.onmicrosoft.com/B2C_1A_Signin_HRD_Saml_Extensions/api/Co
mbinedSigninAndSignup/unified?claimsexchange=ForgotPasswordExchange&csrf_token=dXRiUFI2RUx2cXlDSmZqMlFo
OWtNbWRsUUxhb1hEUEV5ZW1sSUhET2E1V3cweEc0VCtUMktTWElWMEY4N1g4SWpXMHpzQlNuNWptQ0VBS0J3WDJFT3c9PTsyMDI1LTA
xLTE1VDEwOjA4OjIxLjk1MTI3MzZaO0J3cURJLzA3NHpZRDJ4MGw1YmszeWc9PTt7Ik9yY2hlc3RyYXRpb25TdGVwIjo1fQ==&tx=St
ateProperties=eyJUSUQiOiJhN2Y1YjVmOC05MTc4LTQwN2MtYWFmMy1jZmEwOWUwNWIzZGYifQ&p=B2C_1A_Signin_HRD_Saml_E
xtensions&hint=Niall.aiken+1@cclsolutionsgroup.com
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
Te: trailers
Connection: keep-alive

&request_type=VERIFICATION_REQUEST&claim_id=email&claim_value=Niall.aiken%2B1%40cclsolutionsgroup.com
```

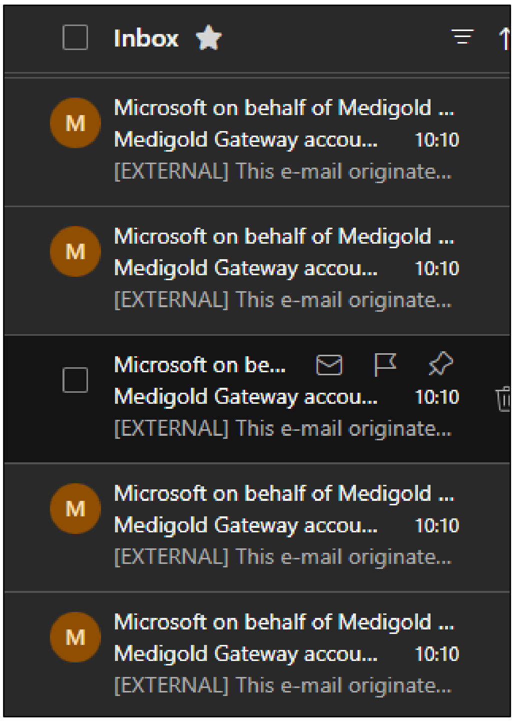
Response:

```
HTTP/1.1 200 OK

Cache-Control: no-store, must-revalidate, no-cache
Content-Type: text/json; charset=utf-8
Vary: Accept-Encoding
x-ms-gateway-requestid: 5493db88-8e62-4ea0-acc3-6797d2c65202
X-Frame-Options: DENY
Public: OPTIONS,TRACE,GET,HEAD,POST
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
--redacted–
Allow: OPTIONS
Allow: TRACE
Allow: GET
Allow: HEAD
Allow: POST
Date: Wed, 15 Jan 2025 10:10:13 GMT
```

```
Content-Length: 27


{"status":"200","result":0}
```



□ Inbox ★
M
Microsoft on behalf of Medigold ...
Medigold Gateway accou... 10:10
[EXTERNAL] This e-mail originate...
M
Microsoft on behalf of Medigold ...
Medigold Gateway accou... 10:10
[EXTERNAL] This e-mail originate...
□
Microsoft on be...
Medigold Gateway accou... 10:10
[EXTERNAL] This e-mail originate...
M
Microsoft on behalf of Medigold ...
Medigold Gateway accou... 10:10
[EXTERNAL] This e-mail originate...
M
Microsoft on behalf of Medigold ...
Medigold Gateway accou... 10:10
[EXTERNAL] This e-mail originate...

Figure 13- Password Reset Emails Received

9.1.3. Recommendations

Implementing rate limiting on an API involves controlling the number of requests that clients can make within a certain time frame. This helps prevent abuse, misuse, and denial-of-service attacks. There are a number of methods for introducing rate limiting to an application, these include:

- Token Bucket: Clients consume tokens at a fixed rate, and requests are only served if there are available tokens.
- Fixed Window: Counts the number of requests within a fixed time window (e.g., per second, per minute).
- Sliding Window: Maintains a rolling window of requests within a specified time period.

he chosen method should be enforced by the web server and when the rate limit is exceeded an appropriate error code should be provided to the user, for example "HTTP status code 429 - Too Many Requests".

9.1.4. References

We have identified the following locations for further reading:

- **OWASP:** https://www.owasp.org/index.php/Denial_of_Service

9.1.5. Affected Systems

Hostname
https://medigoldpulse.com/
gateway.api.medigoldcore.com

9.2. Outdated Third-Party Libraries

Severity	**Medium**
Impact	3
Likelihood	3
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	16

9.2.1. Summary

This finding was found to be remediated after retesting.

Third-party JavaScript libraries, such as jQuery, can be utilised by applications in order to extend the application functionality and reduce development time through means of providing pre-packaged functions, removing the need to code all site content from scratch. In many cases, sites are built around a specific version of a library. The library itself may remain static for fear of introducing compatibility issues, or from fear of impacting application useability. Failure to include JavaScript libraries within organisational patching policy can result in outdated versions being used, leading to the introduction of security vulnerabilities and increasing the possibility of a successful attack being mounted against the application and its users.

9.2.2. Technical Information

The following vulnerable third-party libraries were identified on the web application

Software	Current Version	Latest Version	Path
Axios	1.6.5	1.7.9	/libs/axios/1.6.5/axios.min.js

9.2.3. Recommendations

Third-party libraries should be updated to the latest version available. Additionally, libraries utilised by the application should be incorporated into the organisation patching policy to ensure future updates are applied consistently and within a timely manner.

Upgrades should be implemented in a test environment prior to production to ensure no functionality faces compatibility issues.

9.2.4. References

We have identified the following locations for further reading:

- **OWASP:** https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities.html
- **Snyk**: https://security.snyk.io/package/npm/axios/1.6.5

9.2.5. Affected Systems

Hostname
https://medigoldpulse.com/

9.3. Strict Transport Security Header Not Configured

Severity	**Medium**
Impact	4
Likelihood	2
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	17

9.3.1. Summary

This finding was found to be remediated after retesting.

The HTTP Strict Transport Security (HSTS) header was not configured on the web server. This is a security header which can be sent by a web server to inform the browser to only communicate over HTTPS. Once the browser has received this header it will restrict the user from being able to access HTTP content on the same domain ensuring that all communications are encrypted. Once the configured 'max-age' time period has expired the browser will allow clear-text HTTP communications with the domain again. The 'max-age' countdown resets each time a response is received containing the HSTS header.

9.3.2. Technical Information

The Strict-Transport-Security header was not present in any responses from the web server.

```
GET / HTTP/2
Host: medigoldpulse.com
Sec-Ch-Ua: "Chromium";v="131", "Not_A Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-GB,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.6778.86 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applic
ation/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
Connection: keep-alive
```

Response:

```
HTTP/2 200 OK
Date: Mon, 13 Jan 2025 09:14:45 GMT
Content-Type: text/html
Vary: Accept-Encoding
Last-Modified: Fri, 03 Jan 2025 16:00:49 GMT
Etag: W/"0x8DD2C0FCA94DF33"
X-Ms-Request-Id: 928fb88a-a01e-0066-129b-65fd25000000
```

```
X-Ms-Version: 2018-03-28
Access-Control-Expose-Headers:  Accept-Ranges,Content-Length,Content-Range,Content-Type,Date,ETag,Last-
Modified,Server,x-ms-request-id,x-ms-version
Access-Control-Allow-Origin: *
X-Azure-Ref: 20250113T091445Z-r1d8dc5d8764vm5dhC1LONhyen00000011sg00000000ktqu
X-Fd-Int-Roxy-Purgeid: 81807916
X-Cache: TCP_HIT
X-Cache-Info: L1_T2
```

9.3.3. Recommendations

The HSTS header should be added to all applications to ensure all communications are encrypted. As an example, the following header would enforce HTTPS for one year:

```
Strict-Transport-Security: max-age=31536000
```

Note: The ‘max-age’ value is specified in seconds.

9.3.4. References

We have identified the following locations for further reading:

- **OWASP:** https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet

9.3.5. Affected Systems

Hostname
https://medigoldpulse.com/
https://app-consultations-prd.azurewebsites.net/

9.4. Username Enumeration

Severity	**Low**
Impact	1
Likelihood	3
Fix Effort	**Easy**
Status	**Resolved**
Reference	18

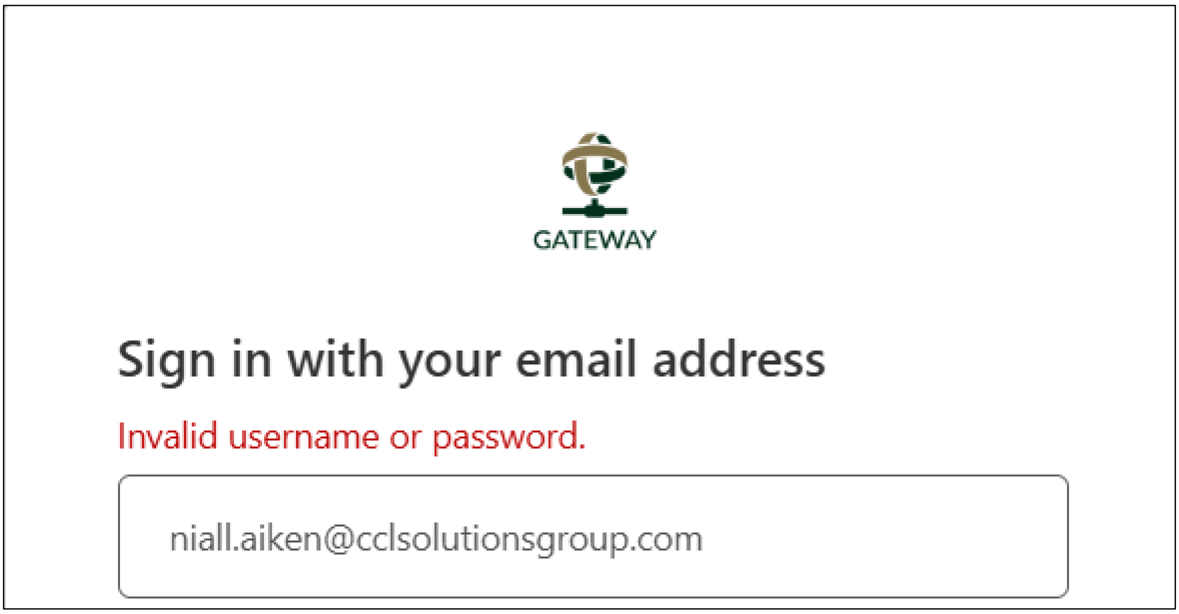
9.4.1. Summary

This finding was found to be remediated after retesting.

Error messages enabled valid registered email addresses to be enumerated. These are of key interest to attackers as they are required when performing common password attacks. Additionally, registered email addresses can be used to target the user themselves in social engineering attacks.

9.4.2. Technical Information

When a valid username is entered with an invalid password the following response is returned



GATEWAY

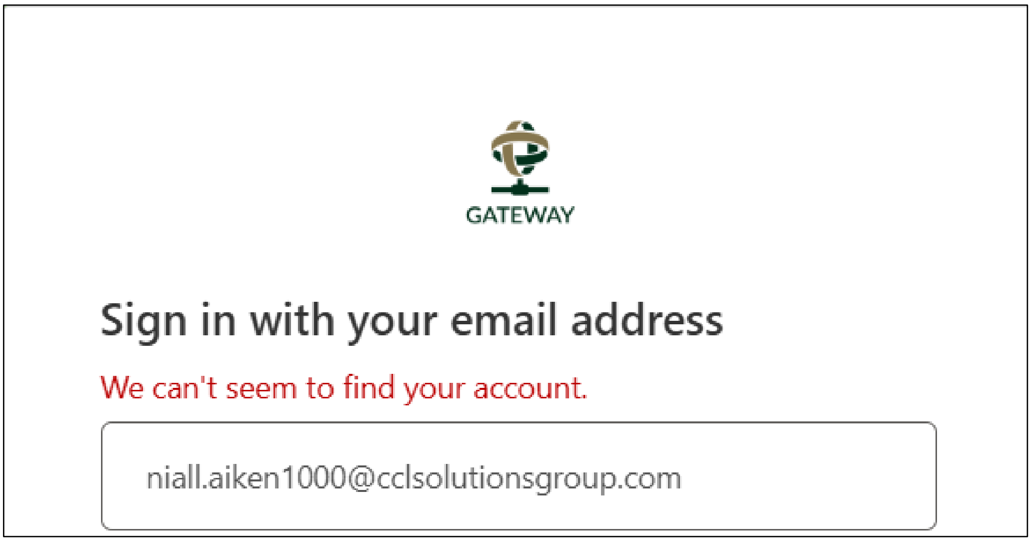
Sign in with your email address

Invalid username or password.

niall.aiken@cclsolutionsgroup.com

Figure 14 - Valid Account Message

In comparison to the following error message when an invalid is entered.



Sign in with your email address

We can't seem to find your account.

niall.aiken1000@cclsolutionsgroup.com

Figure 15 - Invalid Account Error Message

9.4.3. Recommendations

The application should display a generic message regardless of the email address validity. The application should also handle the invalid submissions in the same way, i.e. redirect to the same page either way.

9.4.4. References

We have identified the following locations for further reading:

- **CWE-203:** https://cwe.mitre.org/data/definitions/203.html

9.4.5. Affected Systems

Hostname
https://medigoldpulse.com/
https://medigoldauth.b2clogin.com/

9.5. Information Disclosure Via HTTP Header_ASP

Severity	**Low**
Impact	1
Likelihood	2
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	19

9.5.1. Summary

This finding was found to be remediated after retesting.

The ability to accurately identify the vendor and version number of software running on a server is of high importance to an attacker. Knowing the exact type of server in use allows an attacker to determine any known vulnerabilities that may be used against it and will significantly aid in the selection of any appropriate exploits. A number of application servers will, by default, respond to a client request with data regarding the software running on the server, in the form of an HTTP response header.

9.5.2. Technical Information

All requests to the application were found to return the “Server” and several ASP HTTP headers, which contain the web server and versions of ASP.NET used.

Response:

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
Date: Wed, 15 Jan 2025 09:05:57 GMT
Server: Microsoft-IIS/10.0
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://medigoldpulse.com
Set-Cookie:
ARRAffinity=c3cea541a79e79d9b830bd0962230a90a73c7e966a667c7e488559acce1b68d5;Path=/;HttpOnly;Secure;Dom
ain=app-policy-api-prd.azurewebsites.net
Set-Cookie:
ARRAffinitySameSite=c3cea541a79e79d9b830bd0962230a90a73c7e966a667c7e488559acce1b68d5;Path=/;HttpOnly;Sa
meSite=None;Secure;Domain=app-policy-api-prd.azurewebsites.net
Vary: Accept-Encoding
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
X-Ms-Middleware-Request-Id: 00000000-0000-0000-0000-000000000000
Content-Security-Policy:    default-src    'none';    script-src    'self'    https://medigoldcore.com
https://*.medigoldcore.com  https://medigoldpulse.com  https://*.medigoldpulse.com;  style-src  'self'
https://medigoldcore.com              https://*.medigoldcore.com              https://medigoldpulse.com
https://*.medigoldpulse.com;   img-src   'self'   https://medigoldcore.com   https://*.medigoldcore.com
https://medigoldpulse.com https://*.medigoldpulse.com;
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-Xss-Protection: 1; mode=block
Referrer-Policy: no-referrer
X-Powered-By: ASP.NET
```

9.5.3. Recommendations

The IIS server should be reconfigured to prevent software version information from being returned in application responses. This can be performed using the “URL Rewrite” HTTP module provided by Microsoft for this purpose:

```
<rewrite>
<outboundRules rewriteBeforeCache=”true”>
   <rule name=”Remove Server Header”>
<match serverVariable=”RESPONSE_Server” pattern=”.+” />
<action type=”Rewrite” value=”” />
   </rule>
<ouboundRules>
</rewrite>
```

The ASP.net version number can be removed using the below configuration within the “web.config” file:

```
<system.web>
<httpRuntime enableVersionHeader=”false”> />
</system.web>
```

Whilst the “X-Powered-By” header can be removed with the following configuration, also within the “web.config” file:

```
<htmlProtocol>
<customHeaders>
<remove name=”X-Powered-By” />
</customHeaders>
</htmlProtocol>
```

9.5.4. References

We have identified the following locations for further reading:

- **Microsoft:** https://docs.microsoft.com/en-gb/archive/blogs/benjaminperkins/change-or-modify-a-response-header-value-using-url-rewrite
- **CWE-200:** https://cwe.mitre.org/data/definitions/200.html
- **OWASP:** https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html#server

9.5.5. Affected Systems

Hostname
https://app-consultations-prd.azurewebsites.net/

9.6. Cross-Origin Resource Sharing: Arbitrary Origin Trusted

Severity	**Low**
Impact	1
Likelihood	1
Fix Effort	**Easy**
Status	**Resolved**
Reference	20

9.6.1. Summary

This finding was found to be remediated after retesting.

Trusting arbitrary origins effectively disables the same-origin policy, allowing two-way interaction by third-party websites. Unless the response consists of only

An HTML5 CORS policy controls whether (and how) content running on other domains can perform two-way interaction with the domain that publishes the policy. The policy is fine-grained and can apply access controls per request based on the URL and other features of the request.

Trusting arbitrary origins effectively disables the same-origin policy, allowing two-way interaction by third-party websites. Unless the response consists of only unprotected public content, this policy is likely to present a security risk.

If the site specifies the header Access-Control-Allow-Credentials: true, third-party sites may be able to carry out privileged actions and retrieve sensitive information. Even if it does not, attackers may be able to bypass any IP-based access controls by proxying through users' browsers.

9.6.2. Technical Information

Request to application, including arbitrary Origin header:

```
 GET / HTTP/2
Host: medigoldpulse.com
Accept-Encoding: gzip, deflate, br
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/129.0.6668.71 Safari/537.36
Cache-Control: max-age=0
Origin: https://evilsite.com
```

Response from application:

```
HTTP/2 200 OK

Date: Mon, 13 Jan 2025 10:05:58 GMT
Content-Type: text/html
Vary: Accept-Encoding
Last-Modified: Fri, 03 Jan 2025 16:00:49 GMT
Etag: W/"0x8DD2C0FCA94DF33"
X-Ms-Request-Id: 928fb88a-a01e-0066-129b-65fd25000000
X-Ms-Version: 2018-03-28
```

```
Access-Control-Expose-Headers:  Accept-Ranges,Content-Length,Content-Range,Content-Type,Date,ETag,Last-
Modified,Server,x-ms-request-id,x-ms-version
Access-Control-Allow-Origin: *
X-Azure-Ref: 20250113T100558Z-r1d8dc5d876bjvg2hC1LONa4h4000000028g000000008bwf
X-Fd-Int-Roxy-Purgeid: 81807916
X-Cache: TCP_HIT
X-Cache-Info: L1_T2
```

9.6.3. Recommendations

CCL recommends disabling all instances of CORS without a documented business use case and exhausting existing, safer alternatives.

Should a business use case be documented, and no viable alternatives are present, establish an allow list that consists of authorised and expected domains or IPs to allow additional CORS access.

9.6.4. References

We have identified the following locations for further reading:

- **Mozilla:** https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

9.6.5. Affected Systems

Hostname
https://medigoldpulse.com/

9.7. Clickjacking Vulnerability

Severity	**Low**
Impact	1
Likelihood	1
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	21

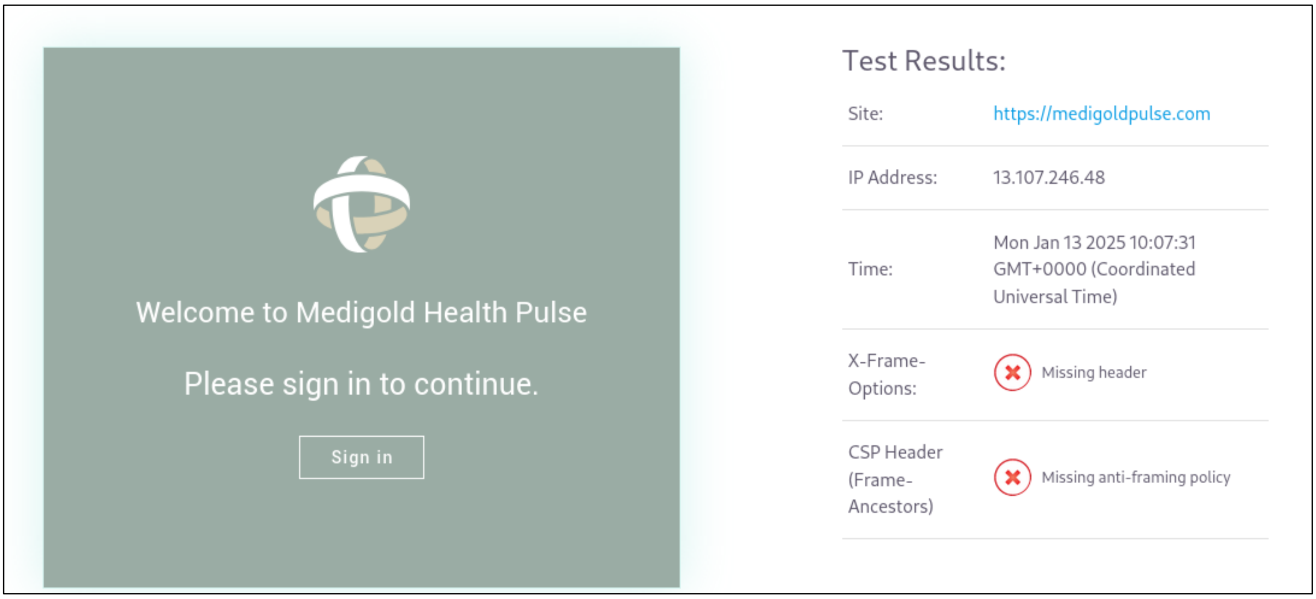
9.7.1. Summary

This finding was found to be remediated after retesting.

It was possible to embed the web application within an iframe which makes clickjacking attacks possible. Clickjacking is an attack against an end-user whereby a threat actor can mislead the victim into performing actions on the affected with their knowledge. This is commonly done by overlaying the affected application in an invisible iframe on top of a video/game etc. The user believes they are interacting with the video or game however, they are performing actions on the affected web application. Attackers use this type of vulnerability for many nefarious purposes, e.g. get the user to change their password, modify a user's details, buy a product, etc.

9.7.2. Technical Information

Using an iframe, the application could be embedded within other websites, as shown in the following screenshot:



Welcome to Medigold Health Pulse

Please sign in to continue.

Test Results:

Site:	https://medigoldpulse.com
IP Address:	13.107.246.48
Time:	Mon Jan 13 2025 10:07:31 GMT+0000 (Coordinated Universal Time)
X-Frame-Options:	☒ Missing header
CSP Header (Frame-Ancestors)	☒ Missing anti-framing policy

Figure 16 - Login Embedded in iframe

9.7.3. Recommendations

The frame-ancestors directive of the Content Security Policy (CSP) HTTP response header can be used to restrict which sites if any can embed the application within frame/iframe tags. For example, to prevent embedding of the application for all domains, the following CSP should be sent:

```
Content-Security-Policy: frame-ancestors ‘none’
```

Unfortunately, not all browsers support this HTTP header at present, therefore it is recommended that the X-Frame-Options response header is also set. The equivalent header for this to the CSP example above is:

```
X-Frame-Options: DENY
```

The OWASP reference below contains more information as to other options for these headers if frames are required in certain circumstances.

9.7.4. References

We have identified the following locations for further reading:

- **OWASP:** https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet
- **CWE-1021:** https://cwe.mitre.org/data/definitions/1021.html

9.7.5. Affected Systems

Hostname
https://medigoldpulse.com/

9.8. Missing HTTP Security Headers

Severity	**Low**
Impact	2
Likelihood	1
Fix Effort	**Easy**
Status	**Resolved**
Reference	22

9.8.1. Summary

This finding was found to be remediated after retesting.

HTTP headers which add additional security to applications were not being utilised by the web application. Browsers support a number of HTTP headers which inform them what resources they should trust and whether or not to automatically sanitise/block potentially malicious code. These features help further enhance the security of web applications and should be employed where possible. Below is an overview of each of these security headers:

Content-Security-Policy - This header allows web site administrators to control resources the user agent is allowed to load for a given page. With a few exceptions, policies mostly involve specifying server origins and script endpoints. This helps guard against cross-site scripting attacks (XSS).

X-Content-Type-Options - This header is a marker used by the server to indicate that the MIME types advertised in the Content-Type headers should not be changed and be followed. This was introduced to block content sniffing that was happening and could transform non-executable MIME types into executable MIME types.

9.8.2. Technical Information

The following headers were not configured on the web application

- Content-Security-Policy
- X-Content-Type-Options

```
HTTP/2 200 OK
Date: Wed, 15 Jan 2025 09:34:18 GMT
Content-Type: text/html
Vary: Accept-Encoding
Last-Modified: Mon, 13 Jan 2025 17:55:42 GMT
Etag: W/"0x8DD33FB7F3975F5"
X-Ms-Request-Id: b131e8ab-b01e-0050-5f2d-675077000000
X-Ms-Version: 2018-03-28
Access-Control-Expose-Headers:  Accept-Ranges,Content-Length,Content-Range,Content-Type,Date,ETag,Last-
Modified,Server,x-ms-request-id,x-ms-version
Access-Control-Allow-Origin: *
X-Azure-Ref: 20250115T093418Z-r1d8dc5d876wdmzthC1LONxee40000000hpg00000000yfkb
X-Fd-Int-Roxy-Purgeid: 82319555
X-Cache: TCP_HIT
```

9.8.3. Recommendations

The recommended HTTP security headers should be configured on the web application.

The Content-Security-Policy header should be configured. This header allows very granular options to be configured, the OWASP document within references can help determine a suitable value.

The X-Content-Type-Options header should be configured with a value of 'nosniff' to instruct the browser to honour the specified content-type. E.g.

```
X-Content-Type-Options: nosniff
```

The X-Frame-Options header should be configured with a value of 'deny' to prevent the application being rendered within an iframe etc., E.g.

```
X-Frame-Options: deny
```

9.8.4. References

We have identified the following locations for further reading:

- **Mozilla:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options
- **Mozilla:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy
- **OWASP:** https://www.owasp.org/index.php/OWASP_Secure_Headers_Project

9.8.5. Affected Systems

Hostname
https://medigoldpulse.com/
gateway.api.medigoldcore.com

9.9. SSL Weak Cipher Suites

Severity	**Low**
Impact	2
Likelihood	2
Fix Effort	**Intermediate**
Status	**Resolved**
Reference	23

9.9.1. Summary

This finding was found to be remediated after retesting.

SSL certificates are used to encrypt communications and verify identity. To transfer data securely TLS/SSL uses one or more cipher suites, these are used to establish a secure connection. If the connection permits, an attacker could purposely specify to downgrade a connection to use the weakest supported version. Weaknesses in either the protocol or cipher suite, could allow a well-position attacker to Man-in-the-middle (MiTM) traffic, to derive clear-text data from the HTTPS communications which could include cookies, usernames and passwords.

9.9.2. Technical Information

The following cipher suites used Cipher Block Chaining (CBC). These are vulnerable to oracle padding attacks. This attack enables a malicious actor to decrypt the contents of the data, without knowing the key.

Key Exchange	Encryption	Bits	Cipher Suite Name (IANA/RFC)
ECDHE	AES	256	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE	AES	128	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

9.9.3. Recommendations

Weak methods of encryption should be disabled to ensure only secure cipher suites are used. This can be achieved, by removing the outdated ciphers and using secure alternatives. An example of a secure cipher configuration list can be found below.

```
ssl_ciphers          ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-
AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384
```

9.9.4. References

We have identified the following locations for further reading:

- **Mozilla:** https://ssl-config.mozilla.org/
- **Github:** https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.md
- **Microsoft:** https://learn.microsoft.com/en-us/dotnet/standard/security/vulnerabilities-cbc-mode

9.9.5. Affected Systems

Hostname
https://medigoldpulse.com/

9.10. Deprecated HTTP Security Header

Severity	**Low**
Impact	1
Likelihood	1
Fix Effort	**Easy**
Status	**Resolved**
Reference	24

9.10.1. Summary

This finding was found to be remediated after retesting.

The 'X-XSS-Protection' header was designed to detect and assist in protecting against cross-site scripting ("XSS") attacks. However, most modern browsers either do not implement or ignore this header value. Browser versions which do interpret the header have been found to introduce new security vulnerabilities into otherwise secure websites, due to poor implementation of XSS protection measures. It is no longer recommended to set the X-XSS-Protection header, as to avoid these vulnerabilities.

9.10.2. Technical Information

The 'X-XSS-Protection' header was deprecated because modern browsers have moved towards more advanced and standardised mechanisms for preventing cross-site scripting (XSS) attacks. Originally, the X-XSS-Protection header was introduced by Microsoft's Internet Explorer and later adopted by other browsers as a non-standard feature.

However, over time, it became apparent that relying solely on this header for XSS protection wasn't sufficient. It had limitations and could sometimes interfere with legitimate scripts, leading to false positives or other issues. Additionally, browsers started implementing more effective XSS protection mechanisms as part of their core security features. The following sample application response shows the affected header set with a value of '1; mode=block':

Response:

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
Date: Wed, 15 Jan 2025 09:05:57 GMT
Server: Microsoft-IIS/10.0
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://medigoldpulse.com
Set-Cookie:
ARRAffinity=c3cea541a79e79d9b830bd0962230a90a73c7e966a667c7e488559acce1b68d5;Path=/;HttpOnly;Secure;Dom
ain=app-policy-api-prd.azurewebsites.net
Set-Cookie:
ARRAffinitySameSite=c3cea541a79e79d9b830bd0962230a90a73c7e966a667c7e488559acce1b68d5;Path=/;HttpOnly;Sa
meSite=None;Secure;Domain=app-policy-api-prd.azurewebsites.net
Vary: Accept-Encoding
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
X-Ms-Middleware-Request-Id: 00000000-0000-0000-0000-000000000000
Content-Security-Policy:    default-src    'none';    script-src    'self'    https://medigoldcore.com
https://*.medigoldcore.com  https://medigoldpulse.com  https://*.medigoldpulse.com;  style-src  'self'
https://medigoldcore.com             https://*.medigoldcore.com             https://medigoldpulse.com
```

```
https://*.medigoldpulse.com; img-src 'self' https://medigoldcore.com https://*.medigoldcore.com
https://medigoldpulse.com https://*.medigoldpulse.com;
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-Xss-Protection: 1; mode=block
Referrer-Policy: no-referrer
X-Powered-By: ASP.NET
```

9.10.3. Recommendations

The ‘X-XSS Protection’ Header was deprecated to encourage developers to adopt more robust XSS protection measures, such as Content Security Policy (CSP), which offers more comprehensive protection against XSS attacks and provides greater flexibility in configuring security policies for web applications.

The ‘X-XSS-Protection’ header has been deprecated and should be either removed, or set to a value of ‘0’

e.g.: X-XSS-Protection: 0;

9.10.4. References

We have identified the following locations for further reading:

- **Mozilla:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection

9.10.5. Affected Systems

Hostname
https://medigoldpulse.com/

Appendix A - Testing Team

This section provides information on the consultants involved during this engagement.

Name	Email	Qualifications
Bianca Napoleonov	bianca.napoleonov@cclsolutiosngroup.com	CSTL Applications
Jack Whittington	jack.whittington@cclsolutionsgroup.com	CTM
Jacob Hudson	jacob.hudson@cclsolutionsgroup.com	CSTL Infrastructure and Applications
Niall Aiken	niall.aiken@cclsolutionsgroup.com	CTM
Rehan Bari	rehan.bari@cclsolutionsgroup.com	CSTL Applications

Appendix B - Reporting Metrics

B.1. Risk Ratings

This section provides information on how risk ratings are calculated within the report.

The following table describes with examples what each level of impact and likelihood represents.

Impact (1-5)	Likelihood (1-5)
1 - This is the lowest level of impact and generally represents information disclosure. For example, a service which discloses software version numbers or other minor information which could aid an attacker in further attacks. However, does not on its own allow a threat actor to launch an attack.	**1** - A likelihood of 1 would represent the lowest risk of an issue being exploited. This could be because of the high complexity of the exploit. For example, a vulnerability which would require a highly skilled attacker to exploit. Alternatively, a low likelihood could be due to significant access requirements to leverage the exploit. It could also be a combination of these two things. For example, a vulnerability which can only be exploited from an air-gapped network.
2 - This level of impact refers to; more severe instances of information disclosure, minor configuration weaknesses that could be leveraged by attackers who already have a level of access to systems, or configuration weaknesses which would not directly result in compromise but could reduce the time requirements or difficulty of attacks.	**2** - This level of likelihood refers to vulnerabilities that would require a high level of access to exploit, and / or a high level of skill. For example, administrative access may be required to the system, and / or the vulnerability may not be publicly available.
3 - The level of impact refers to vulnerabilities that would give a moderate level of access to systems, or significant information disclosure. For example, gaining access to a system with user-level credentials. Or a system that discloses usernames, emails, or sensitive technical information to unauthenticated attackers.	**3** - This level regards vulnerabilities that may require a level of access, either to networks or systems, for an attacker to be suitably positioned to exploit. For example, user-level credentials may be required to perform the exploit. Alternatively, or conjunctively, the vulnerability may require insider knowledge to exploit. At this level of likelihood, the exploit would require a reasonably skilled attacker.
4 - This level of impact represents vulnerabilities that give an attacker a high level of access to systems. For example, gaining administrative level access via privilege escalation to a server, or admin access to an application. Allowing an attacker to affect changes to the system / application.	**4** -This level represents vulnerabilities that have a high likelihood of being exploited. For example, a server on an organisations standard corporate network with a vulnerability for which there is publicly available exploit tools.
5 - This is the highest level and would represent an impact which allowed complete compromise of the system. For example, bypassing the login mechanism to access a system as an administrative user. Allowing an attacker to access all data within that system.	**5** - This is the highest level of likelihood and refers to vulnerabilities which are trivial to exploit. For example, a vulnerability which is exploitable using automated tools, with very little skill. This could also be due to ease of access. For example, a vulnerability that can be exploited remotely over the Internet using publicly available tools.

The risk rating of an issue is then calculated using the following formula:

Impact + Likelihood = Overall Risk

The table below is used to calculate the severity for that issue:

Overall Risk	Severity
2-4	Low
5-6	Medium
7-10	High

B.2. Fix Effort

The following table describes with examples how the "Fix Effort" is calculated for issues.

Fix Effort	Description
Easy	This represents issues which require minimal effort to remediate. For example, installing security updates, or changing settings.
Intermediate	This represents issues that require a moderate amount of effort to resolve. For example, modifying configuration files, or enabling settings which could cause disruption. For example, settings which provide security, but disable support for legacy clients.
Hard	A fix effort of high relates to vulnerabilities that require the considerable effort to resolve. Potentially requiring work on behalf of the software vendor or in-house web / software development work. This could also relate to issues which require network redesign.

Please note, the fix effort cannot always factor in organisation specific rationale as to the work required to remediate issues. For example, in the case of critical infrastructure. Missing patches may be assigned a "Fix Effort" of Low. However, systems may need to be patched out-of-hours, as they cannot be taken offline during the day. Consequently, the fix effort is calculated solely on the technical complexity of remediating an issue.