



**Beagle
Security**

Continuous Vulnerability
Assessment Platform



HIPAA Compliance Report

Demo Application

04 Mar 2025



WEB



Copyright

The copyright in this work is vested to Beagle Security and the document is issued in confidence for the purpose to which it is supplied. If you are not the intended recipient, be aware. Any disclosure, copying or distribution of this document without the prior consent of Beagle Security is prohibited. If found so, you will be open to legal actions.

© Beagle Security

info@beaglesecurity.com



Document Details

Action	Date
Completed On	04 Mar 2025
Generated on	06 Oct 2026

Table of Contents

1. Introduction	6
2. Executive Summary	7
3. Application Info	11
4. Technical Summary	12
4.1. HIPAA Compliance Failed Controls	13
4.1.1 Subpart A, HIPAA-164.105: The application must protect private electronic health information.	13
4.1.1.1 Content Security Policy (CSP) header not implemented	13
4.1.2 Subpart C, 164.306(a)(1): Covered entities must ensure the confidentiality, integrity and availability of all electronic protected health information it creates, receives, maintains or transmits	14
4.1.2.1 Virtual hosts found	14
4.1.3 Subpart C, HIPAA-164.308(a)(1)(i): Possible Issue - A covered entity must implement policies and procedures to prevent, detect, contain, and correct security violations.	15
4.1.3.1 X-Frame-Options header not implemented	15
4.1.3.2 X-Content-Type-Options header not implemented	16
4.1.4 Subpart C, 164.312(a)(1): Possible Issue - A covered entity must implement policies and procedures for controlling access to private health information.	18
4.1.4.1 Suspicious Comments Leak	18
4.1.4.2 Cross-Origin-Opener-Policy Misconfiguration	23
4.1.4.3 Base64 Encoded Data Exposure	28
4.1.4.4 Missing Subresource Integrity (SRI) Attribute	33
4.1.5 Subpart C, 164.312(c)(1): Possible Issue - A covered entity must implement policies and procedures to protect private health information from improper alteration or destruction.	36
4.1.5.1 Improper Cache-Control Configuration	36

4.1.6 Subpart C, 164.312(e)(1): A covered entity must implement technical security measures to guard against unauthorized access to private health information that is being transmitted over an electronic communications network.	42
4.1.6.1 SWEET32 attack	42
4.1.6.2 BREACH Attack	44
4.1.6.3 TLS 1.1 outdated version	46
4.1.6.4 TLS 1.0 outdated version	48
4.1.6.5 HTTP Strict Transport Security (HSTS) header not implemented	51
4.1.6.6 Cached Data Retrieved	52
4.1.6.7 Cacheable and Storable Content	57
4.1.7 Subpart C, 164.312(e)(2)(i): Addressable Issue - Implement security measures to ensure that electronically transmitted private health information is not improperly modified without detection until disposed of.	62
4.1.7.1 Web Application Potentially Vulnerable to BEAST attack	62
4.1.7.2 Web Application Potentially Vulnerable to LUCKY13	63
4.2. HIPAA Compliance Passed Controls	63
5. Conclusion	65



Introduction

The data present in this document is the result of the penetration test conducted on "<https://imageresizer.com/>" using Beagle Security. The test generates a checklist of potential compliance issues by checking "**Demo Application**" for vulnerabilities including insecure data collection forms, insecure cookies, third-party links, cross site scripting vulnerabilities, SQL injection, etc.

Using this data, you can then proactively filter and organize identified issues to ensure that your organization's most critical regulatory compliance concerns are addressed as soon as possible. While this information is intended to greatly improve the efficiency with which you can remediate compliance issues, it does not intend to represent the full scope of compliance with the HEALTH INSURANCE PORTABILITY AND ACCOUNTABILITY ACT OF 1996 (HIPAA) regulations.

These results are only a subset of the entire HIPAA compliance. In addition, since these regulations may change with time, this report could have been generated using an older version of the application that has not been updated to reflect those changes. Henceforth, it is solely the user's responsibility to understand and adhere to the most recent rules and regulations published by the authorities.

Beagle Security holds no responsibility for any use or misuse of information presented in this report.

The next section provides the non-technical team with a summary of all key findings and relates the impact it will have on your business. Section 3 provides the technical team with a detailed report of individual vulnerabilities along with its mitigation procedures. This detailed report generated will help your development team to improve the overall security of the system.



Executive Summary

The chart, tables, and graphs displayed below is to provide you the summary of all the vulnerabilities present in "**Demo Application**" based on the status and severity. The severity of each vulnerability is calculated based on its occurrence, frequency, and impact upon the asset.

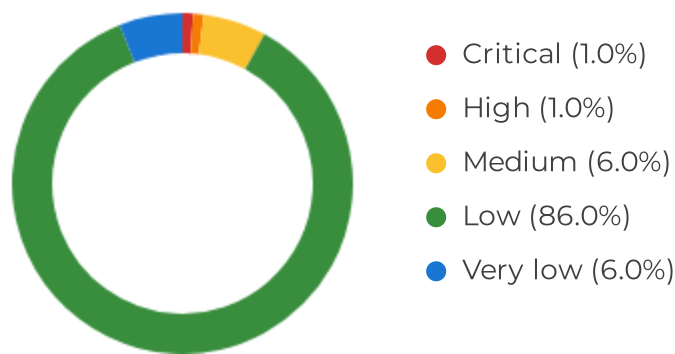
By examining these representations generated by Beagle Security, you can gain insights into state of "**Demo Application**"'s current security posture.

Catalog

Status	Count
New	140
Not Fixed	0
Reopened	0
Fixed	0

Graphical Summary

Vulnerability Distribution



Overall Risk score



Tabular Summary

Category	Count
Critical	2
High	1
Medium	8
Low	120
Very Low	9

HIPAA Compliance Summary

SI No	S.Rule - Part 164	PASS/FAIL
1	Subpart A, HIPAA-164.105: The application must protect private electronic health information.	FAILED
2	Subpart C, 164.306(a)(1): Covered entities must ensure the confidentiality, integrity and availability of all electronic	

	protected health information it creates, receives, maintains or transmits	FAILED
3	Subpart C, HIPAA-164.306(a)(2): Covered entities must protect against any reasonably anticipated threats or hazards to the security or integrity of electronic protected health information.	PASSED
4	Subpart C, HIPAA-164.306(a)(3): Covered entities must protect against any reasonably anticipated uses or disclosures of electronic protected health information.	PASSED
5	Subpart C, HIPAA-164.308(a)(1)(i): Possible Issue - A covered entity must implement policies and procedures to prevent, detect, contain, and correct security violations.	FAILED
6	Subpart C, 164.308(a)(1)(ii)(B): Possible Issue - A covered entity must implement security measures sufficient to reduce risks and vulnerabilities to a reasonable and appropriate level.	PASSED
7	Subpart C, 164.308(a)(5)(ii)(B): Addressable Issue - Implement procedures for guarding against, detecting, and reporting malicious software.	PASSED
8	Subpart C, 164.308(a)(5)(ii)(C): Addressable Issue - Implement procedures for monitoring log-in attempts and reporting discrepancies.	PASSED
9	Subpart C, 164.308(a)(5)(ii)(D): Addressable Issue - Implement procedures for creating, changing, and safeguarding passwords.	PASSED
10	Subpart C, 164.308(a)(7)(i): Possible Issue - Establish (and implement as needed) procedures for responding to an emergency or other occurrence that damages systems that contain electronic protected health information.	PASSED
11	Subpart C, 164.312(a)(1): Possible Issue - A covered entity must implement policies and procedures for controlling access to private health information.	FAILED
12	Subpart C, 164.312(c)(1): Possible Issue - A covered entity must implement policies and procedures to protect private health information from improper alteration or destruction.	FAILED

14	Subpart C, 164.312(e)(1): A covered entity must implement technical security measures to guard against unauthorized access to private health information that is being transmitted over an electronic communications network.	FAILED
15	Subpart C, 164.312(e)(2)(i): Addressable Issue - Implement security measures to ensure that electronically transmitted private health information is not improperly modified without detection until disposed of.	FAILED
16	Subpart C, 164.312(e)(2)(ii): Addressable Issue - Implement a mechanism to encrypt electronic private health information whenever deemed appropriate.	PASSED

Application Info

Project name	ssdlj
Application name	Demo Application
URL	https://imageresizer.com/
Test completed on	04 Mar 2025
Test type	Grey-box pentest

Authentication details

Name	Value
Basic HTTP authentication	Disabled
Test credentials	Login
Scenario recording	Disabled

Technical Summary

Detailed Technical Report

	Open						Fixed
	Critical	High	Medium	Low	Very low	Total	
TLS 1.0 outdated version	0	0	0	0	0	0	0
TLS 1.1 outdated version	0	0	0	0	0	0	0
Missing or Misconfigured DMARC record	0	0	0	0	0	0	0
BREACH Attack	0	0	0	0	0	0	0
Missing Subresource Integrity (SRI) Attribute	0	0	0	0	0	0	0
X-Content-Type-Options header not implemented	0	0	0	0	0	0	0
Improper Cache-Control Configuration	0	0	0	0	0	0	0
HTTP Strict Transport Security (HSTS) header not...	0	0	0	0	0	0	0
X-Frame-Options header not implemented	0	0	0	0	0	0	0
Content Security Policy (CSP) header not...	0	0	0	0	0	0	0
SWEET32 attack	0	0	0	0	0	0	0
Suspicious Comments Leak	0	0	0	0	0	0	0
Cached Data Retrieved	0	0	0	0	0	0	0
Base64 Encoded Data Exposure	0	0	0	0	0	0	0
Cross-Origin-Opener-Policy Misconfiguration	0	0	0	0	0	0	0
Cacheable and Storable Content	0	0	0	0	0	0	0

Web Application Potentially Vulnerable to BEAST attack	0	0	0	0	0	0	0
Virtual hosts found	0	0	0	0	0	0	0
Vulnerable javaScript library	0	0	0	0	0	0	0
Web Application Potentially Vulnerable to LUCKY13	0	0	0	0	0	0	0
SPF Record not Implemented	0	0	0	0	0	0	0

HIPAA Compliance Failed Controls

Subpart A, HIPAA-164.105: The application must protect private electronic health information.

Content Security Policy (CSP) header not implemented

OWASP 2019-API7

OWASP 2013-A5

OWASP 2017-A6

OWASP 2021-A5

PCI v4.0-6.2.4

CWE-16

Subpart A, HIPAA-164.105

ISO27001-A.14.2.5

WASC-15

A.12.1.1

- Likelihood: Medium
- Impact: Low
- Severity: Low

Issue Description:

Content Security Policy (CSP) is a computer security standard. It was introduced to prevent cross-site scripting (XSS), clickjacking and other code injection attacks resulting from execution of malicious content in the trusted web page context. In this application, the Content Security Policy header is not implemented. This leads to vulnerabilities like Cross-site Scripting and related attacks. Not implementing Content Security Policy this application missing out this extra layer of security.

Recommendations:

- The mitigation for this vulnerability is to enable CSP on your website by sending the Content-Security-Policy in HTTP response headers. The header must instruct the browser to apply the policies you specified.

Occurrences:

Occurrence 001

Status: New

URL: <https://imageresizer.com/>

Subpart C, 164.306(a)(1): Covered entities must ensure the confidentiality, integrity and availability of all electronic protected health information it creates, receives, maintains or transmits

Virtual hosts found

OWASP 2013-A5

OWASP 2017-A6

OWASP 2021-A5

PCI v4.0-6.2.4

CWE-200

Subpart C, HIPAA-164.306(a)(1)

WASC-13

WSTG-INPV-17

- Likelihood: Low
- Impact: Low
- Severity: Info

Issue Description:

Virtual hosts allow hosting multiple domain names on a single server. If a vulnerability is found in one virtual host, it could lead to security issues for all other virtual hosts on the same server. Attackers may exploit this shared context to compromise applications hosted on different virtual hosts, potentially leading to unauthorized access, data breaches, and various security risks. Ensuring the security of all virtual hosts is crucial to prevent cross-contamination and protect the integrity of the entire server environment.

Recommendations:

Secure all virtual hosts with regular assessments. For critical applications, opt for a dedicated host to isolate and strengthen security.

Occurrences:

Occurrence 002

Status: New

Test_url: https://dev.imageresizer.com

Vulnerability: Virtual_host

Subpart C, HIPAA-164.308(a)(1)(i): Possible Issue - A covered entity must implement policies and procedures to prevent, detect, contain, and correct security violations.

X-Frame-Options header not implemented

OWASP 2019-API7

OWASP 2013-A5

OWASP 2017-A6

OWASP 2021-A5

PCI v4.0-6.2.4

OWASP PC-C1

CAPEC-103

CWE-693

Subpart C, HIPAA-164.308(a)(1)(i)

ISO27001-A.14.2.5

WASC-14

WSTG-CLNT-09

A.12.1.1

- Likelihood: Medium
- Impact: Medium
- Severity: Medium

Issue Description:

In this webpage, X Frame-Options header is not found. Without X Frame-Options header the browser cannot decide the page to render in <frame> or <iframe> and thus the site cannot ensure that their contents are not embedded in other sites. This vulnerability leads to many attacks like Clickjacking.

Recommendations:

- Implement X-Frame-Options
- The possible types are:-
 - DENY - Prevents any domain from framing the content.
 - SAMEORIGIN - It allows the current site to frame the content.
 - ALLOW-FROM URI - Permits specified URI
- Add HTTP response manually to every page.

Occurrences:

Occurrence 003

Status: **New**

URL: <https://imageresizer.com/>

X-Content-Type-Options header not implemented

OWASP 2019-API7

OWASP 2013-A5

OWASP 2017-A6

OWASP 2021-A5

PCI v4.0-6.2.4

OWASP PC-C1

CWE-16

Subpart C, HIPAA-164.308(a)(1)(i)

ISO27001-A.14.1.2

WASC-15

A.12.1.1

- Likelihood: Low
- Impact: Low
- Severity: Info

Issue Description:

The X-Content-Type-Options response HTTP header is a marker used by the server to indicate that the Multipurpose Internet Mail Extensions types advertised in the Content-Type headers should not be changed and be followed. It is a way to say that the webmasters knew what they were doing. In this webpage, X Content Type Options is not found. This application is vulnerable to Multipurpose Internet Mail Extensions sniffing attacks. These vulnerabilities can occur when a website allows users to upload content to a website. This can give them the opportunity to perform cross-site scripting and compromise the website.

Recommendations:

Step 1: Understand the Vulnerability The X-Content-Type-Options header is a security feature that helps to mitigate MIME type sniffing attacks. These attacks occur when a malicious user tricks a web browser into interpreting a file as a different content type than intended. This can lead to cross-site scripting (XSS) and other types of attacks.

Step 2: Check if the Header is Implemented The first step is to check if the X-Content-Type-Options header is already implemented on your website. This can be done by using a web security scanner or by inspecting the response headers using the developer tools in your browser.

Step 3: Implement the X-Content-Type-Options Header If the header is not already implemented, it can be added to the server configuration or to the code of your website. The implementation will vary depending on the technology stack used. For IIS:

1. Open the Internet Information Services (IIS) Manager.
2. Select the website or application you want to add the header to.
3. In the "HTTP Response Headers" section, click "Add" in the right-hand panel.
4. In the "Name" field, enter "X-Content-Type-Options".
5. In the "Value" field, enter "nosniff".
6. Click "OK" to save the changes.

For Apache:

1. Open the .htaccess file in the root directory of your website.
2. Add the following line: Header set X-Content-Type-Options "nosniff"
3. Save the changes.

For Nginx:

1. Open the nginx.conf file.
2. Add the following line in the server block: `add_header X-Content-Type-Options nosniff;`
3. Save the changes.

For Node.js:

1. Install the helmet package using npm: `npm install helmet --save`
2. In your server.js file, add the following code: `var helmet = require('helmet');
app.use(helmet());`
3. Save the changes.

Step 4: Test the Implementation After implementing the X-Content-Type-Options header, it is important to test if it is working correctly. This can be done by using a web security scanner or by manually testing the website for any MIME type sniffing vulnerabilities.

Step 5: Repeat for all Technologies The above steps should be repeated for all technologies used in the website, including PHP, Python, Ruby, Java, JSP, ASP, JavaScript, Perl, and others.

Step 6: Regularly Update and Monitor It is important to regularly update the server and application configurations to ensure that the X-Content-Type-Options header is still implemented and working correctly. This should also be monitored regularly to detect any potential vulnerabilities.

In addition to implementing the X-Content-Type-Options header, it is also recommended to follow secure coding practices and perform regular security audits to ensure the overall security of the web application.

Occurrences:

Occurrence 004

Status: **New**

URL: <https://imageresizer.com/>

Subpart C, 164.312(a)(1): Possible Issue - A covered entity must implement policies and procedures for controlling access to private health information.

Suspicious Comments Leak

OWASP 2017-A3

OWASP 2021-A1

CWE-200

Subpart C, HIPAA-164.312(a)(1)

ISO27001-A.14.1.3

WASC-13

WSTG-INFO-05

- Likelihood: Medium
- Impact: Low
- Severity: Low

Issue Description:

A potential security threat arises when suspicious comments are revealed in the response, potentially aiding an attacker's endeavors. This issue is particularly relevant for script blocks or files where matches are made against entire content.

Recommendations:

To mitigate this risk, eliminate all comments that could furnish information helpful to attackers and rectify any underlying issues they may refer to.

Occurrences:

Occurrence 005

Status: **New**

Evidence: [query](#)

Method: GET

URL: <https://imageresizer.com/en>

Occurrence 006

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/>

Occurrence 007

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/fr>

Occurrence 008

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/flip-image>

Occurrence 009

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/ja>

Occurrence 010

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/bulk-resize>

Occurrence 011

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/heic-to-jpg>

Occurrence 012

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/color-picker>

Occurrence 013

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/bn>

Occurrence 014

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/image-enlarger>

Occurrence 015

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/about>

Occurrence 016

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/image-compressor>

Occurrence 017

Status: **New**

Evidence: todo

Method: GET

URL: <https://imageresizer.com/es>

Occurrence 018

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/jpg-to-png>

Occurrence 019

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/crop-image>

Occurrence 020

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/it>

Occurrence 021

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/id>

Occurrence 022

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/de>

Occurrence 023

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/changelogs>

Occurrence 024

Status: **New**

Evidence: query

Method: GET

URL: <https://imageresizer.com/image-converter>

Cross-Origin-Opener-Policy Misconfiguration

OWASP 2017-A3

OWASP 2021-A4

OWASP PC-C10

CWE-693

Subpart C, HIPAA-164.312(a)(1)

WASC-14

WSTG-INFO-09

- Likelihood: Medium
- Impact: Low
- Severity: Low

Issue Description:

Inadequate protection of browsing context can compromise sensitive data by allowing untrusted documents to share the same context, leading to potential data leaks.

Recommendations:

To prevent unauthorized access, configure your web server/application to set the Cross-Origin-Opener-Policy header with a value of 'same-origin'. Consider using modern browsers that support this feature for enhanced security.

Occurrences:

Occurrence 025

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: <https://imageresizer.com/de>

Occurrence 026

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: <https://imageresizer.com/id>

Occurrence 027

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: <https://imageresizer.com/crop-image>

Occurrence 028

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: <https://imageresizer.com/en>

Occurrence 029

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: https://imageresizer.com/_next/static/chunks/pages/crop-image-8c61f299065bcb9e.js

Occurrence 030

Status: **New**

Method:	GET
Parameter:	Cross-Origin-Resource-Policy
URL:	https://imageresizer.com/color-picker

Occurrence 031

Status: **New**

Method:	GET
Parameter:	Cross-Origin-Resource-Policy
URL:	https://imageresizer.com/image-converter

Occurrence 032

Status: **New**

Method:	GET
Parameter:	Cross-Origin-Resource-Policy
URL:	https://imageresizer.com/changelogs

Occurrence 033

Status: **New**

Method:	GET
Parameter:	Cross-Origin-Resource-Policy
URL:	https://imageresizer.com/image-enlarger

Occurrence 034

Status: **New**

Method:	GET
Parameter:	Cross-Origin-Resource-Policy

URL: <https://imageresizer.com/favicon.ico>

Occurrence 035

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: <https://imageresizer.com/fr>

Occurrence 036

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: <https://imageresizer.com/bulk-resize>

Occurrence 037

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: <https://imageresizer.com/>

Occurrence 038

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: <https://imageresizer.com/es>

Occurrence 039

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: https://imageresizer.com/heic-to-jpg

Occurrence 040

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: https://imageresizer.com/flip-image

Occurrence 041

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: https://imageresizer.com/_next/static/css/fccf139d35fdf47a.css

Occurrence 042

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: https://imageresizer.com/bn

Occurrence 043

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: https://imageresizer.com/image-compressor

Occurrence 044

Status: **New**

Method: GET

Parameter: Cross-Origin-Resource-Policy

URL: https://imageresizer.com/about

Base64 Encoded Data Exposure

OWASP 2017-A3

OWASP 2021-A4

OWASP PC-C8

CAPEC-170

CWE-200

Subpart C, HIPAA-164.312(a)(1)

ISO27001-A.14.1.3

WASC-13

WSTG-INFO-05

- Likelihood: Medium
- Impact: Low
- Severity: Low

Issue Description:

A potential security issue arises when Base64 encoded data is inadvertently disclosed by an application or web server, which could potentially lead to sensitive information being exposed. A thorough analysis of the response is required.

Recommendations:

Manually verify that the Base64 encoded data does not contain sensitive information and cannot be aggregated or exploited by other vulnerabilities.

Occurrences:

Occurrence 045

Status: **New**

Evidence: 2B3NaQEXXpoEzqvgnhY5tA3QtHNDfv

Method: GET

URL: <https://imageresizer.com/es>

Occurrence 046

Status: **New**

Evidence: [/_next/static/6pm7C4aGi_iWz87U253RO/_buildManifest](#)

Method: GET

URL: <https://imageresizer.com/bulk-resize>

Occurrence 047

Status: **New**

Evidence: [2FcORKNM6zBmYIHkvnNI9F1b69n0ao5CAc0ltwZ0d0YiczX](#)

Method: GET

URL: <https://imageresizer.com/color-picker>

Occurrence 048

Status: **New**

Evidence: [2FaVzhclgIM2rrSWm8iUWpz3E8197jK6Bw](#)

Method: GET

URL: <https://imageresizer.com/about>

Occurrence 049

Status: **New**

Evidence: [/_next/static/6pm7C4aGi_iWz87U253RO/_buildManifest](#)

Method: GET

URL: <https://imageresizer.com/de>

Occurrence 050

Status: **New**

Evidence: 2BzQ9RMddcgYvDoGLJtdPFWo1Gi4bMH6nbvLlj5jvB9BwVptNM0

Method: GET

URL: https://imageresizer.com/image-compressor

Occurrence 051

Status: New

Evidence: 2BzYXplh7l5EljyLTs85g6Njl6SVFDq0c3VpccFSs7j9lZwq0u0boQ1YxWIYjnqT

Method: GET

URL: https://imageresizer.com/id

Occurrence 052

Status: New

Evidence: 2FfQ3wSdg6PDL8Pkrtj7JwKFYfcUsfLM0lEmqN5dr6hmYfMs2skWa9HZ7Y
DqOEbuRRUC8oThxlaTQ4mTWb5F

Method: GET

URL: https://imageresizer.com/favicon.ico

Occurrence 053

Status: New

Evidence: /_next/static/6pm7C4aGi_iWz87U253RO/_buildManifest

Method: GET

URL: https://imageresizer.com/en

Occurrence 054

Status: New

Evidence: 2FmoXmeYlFttGP4xELQnYFOrwoE62jEKuvcnPD

Method: GET

URL: https://imageresizer.com/_next/static/chunks/pages/crop-image-8c61f299065bcb9e.js

Occurrence 055

Status: **New**

Evidence: 2FwVmJ8WoNTrBy9G2Fa2G8cUK6esQjxvQpe5F4Wn5lrN3ZEJ85R9023NJJo
CYXWHbTmigfSPk0AXxiPYNxwamJgfG0uo

Method: GET

URL: <https://imageresizer.com/flip-image>

Occurrence 056

Status: **New**

Evidence: 2BS4otl0umUShndpWFobfQrRFeTTOaq

Method: GET

URL: <https://imageresizer.com/bn>

Occurrence 057

Status: **New**

Evidence: JwblfGR0rjz6se4hLxHTgp45sFxbyv3e2sMe9PWhyCvv

Method: GET

URL: <https://imageresizer.com/fr>

Occurrence 058

Status: **New**

Evidence: 2BvmBHf4sY3PP1bxSA14WjOoNk0uVLTkDUAKP4rA

Method: GET

URL: <https://imageresizer.com/crop-image>

Occurrence 059

Status: **New**

Evidence: 2Fk5iiousc8JUHPaLQOEeNg7G7Teepjv0npEi60DELbX0pImCmPvZLdCLS
ROpcjEmGQWshX

Method: GET

URL: <https://imageresizer.com/>

Occurrence 060

Status: **New**

Evidence: 1Gjpv5XUbf4GEfyOLj2ZPnXZgdH1DKsGojtmDugAZqfWym8thOTK50INi5id
0oq1WiOT0lzAPdUD0xssKmMXBt7E0k0CMSsU40TrTy1kYdOv2jk0zBT1565
UTE9ILH0mYk0

Method: GET

URL: <https://imageresizer.com/heic-to-jpg>

Occurrence 061

Status: **New**

Evidence: 2FDCET2SQzkHyR5NB7Xx1N7zqP6LZGzJyZhXrLDvTV3OzJ7bQ8UT4To6F0v
TnUXFVn5a2dzCwxZTeN333twvTU

Method: GET

URL: https://imageresizer.com/_next/static/css/fccf139d35fdf47a.css

Occurrence 062

Status: **New**

Evidence: 2FCr4UBxVx07opKtaO8pyCbf7WWgs0JkqrpDF8T

Method: GET

URL: <https://imageresizer.com/image-converter>

Occurrence 063

Status: **New**

Evidence: 2BIHJAtNJ2T9lOXLOqLSSYa50EJmILwv85WqljTnXX2irm48EzoFyCuA0QQl
KlfPjLo7uD028l8fUw

Method: GET

URL: <https://imageresizer.com/changelogs>

Missing Subresource Integrity (SRI) Attribute

OWASP 2017-A6

OWASP 2021-A5

OWASP PC-C8

CWE-345

Subpart C, HIPAA-164.312(a)(1)

ISO27001-A.14.1.3

WASC-15

WSTG-CONF-02

- Likelihood: Medium
- Impact: Medium
- Severity: Medium

Issue Description:

An external server is serving a script or link without an integrity attribute, making it vulnerable to tampering and potential injection of malicious content.

Recommendations:

Step 1: Understand Subresource Integrity (SRI) Subresource Integrity (SRI) is a security feature that allows web developers to ensure that the external resources (such as scripts, stylesheets, and fonts) loaded on their website are not tampered with by a third party. It works by adding a cryptographic hash to the resource's URL, which is then compared to the hash of the resource when it is loaded on the website. If the hashes do not match, the resource is not loaded, preventing any potential malicious code from being executed on the website.

Step 2: Identify the Resources The first step in mitigating the Missing Subresource Integrity (SRI) Attribute vulnerability is to identify all the external resources that are being loaded on the website. This includes scripts, stylesheets, fonts, and any other external resources.

Step 3: Generate Hashes for Resources Next, you will need to generate a cryptographic hash for each of the external resources. This can be done using a tool such as SRI Hash Generator (<https://www.srihash.org/>). Simply enter the URL of the

resource and click on "Generate Hash". This will generate a unique hash for the resource.

Step 4: Add SRI Attribute to Resource Links Once you have generated the hashes for all the external resources, you will need to add the SRI attribute to the links of these resources. This can be done by adding the "integrity" attribute to the link tag and the corresponding hash in the "sha256" attribute. For example:

```
<script src="https://example.com/script.js" integrity="sha256-2s7z3gVbQd3gVbQd3gVbQd3gVbQd3gVbQd3g="></script>
```

Step 5: Test and Verify After adding the SRI attribute to all the external resource links, it is important to test and verify that the resources are still loading correctly on the website. You can use a tool such as SRI Test (<https://sritest.io/>) to check if the resources are being loaded with the correct hashes.

Step 6: Implement Content Security Policy (CSP) In addition to SRI, it is recommended to also implement a Content Security Policy (CSP) on the website. This will further enhance the security of the website by specifying which external resources are allowed to be loaded. Any external resources that do not match the specified policy will not be loaded, preventing potential attacks.

Step 7: Regularly Monitor and Update Hashes It is important to regularly monitor and update the hashes for external resources as they may change over time. This can be done by using automated tools or manually checking and updating the hashes.

Step 8: Educate Developers It is crucial to educate all developers on the importance of implementing SRI and CSP on their code. They should be aware of the potential risks of not using these security features and should be trained on how to properly implement them in their code.

Step 9: Use a Content Delivery Network (CDN) Consider using a Content Delivery Network (CDN) to host external resources. CDNs often have built-in support for SRI, making it easier to implement on the website.

Step 10: Regular Security Audits Perform regular security audits on the website to identify any new external resources that may have been added and ensure they are properly secured with SRI and CSP.

Conclusion By following these steps, you can mitigate the Missing Subresource Integrity (SRI) Attribute vulnerability on your web application. It is important to regularly monitor and update the security measures in place to ensure the website remains secure.

Occurrences:

Occurrence 064

Status: **New**

Evidence: `<link rel="preconnect" href="https://fonts.googleapis.com"/>`

Method: GET

URL: https://imageresizer.com/bn

Occurrence 065

Status: New

Evidence: <link rel="preconnect" href="https://fonts.googleapis.com/">

Method: GET

URL: https://imageresizer.com/

Occurrence 066

Status: New

Evidence: <link rel="preconnect" href="https://fonts.googleapis.com/">

Method: GET

URL: https://imageresizer.com/crop-image

Occurrence 067

Status: New

Evidence: <link rel="preconnect" href="https://fonts.googleapis.com/">

Method: GET

URL: https://imageresizer.com/changelogs

Occurrence 068

Status: New

Evidence: <link rel="preconnect" href="https://fonts.googleapis.com/">

Method: GET

URL: https://imageresizer.com/color-picker

Occurrence 069

Status: **New**

Evidence: <link rel="preconnect" href="https://fonts.googleapis.com/">

Method: GET

URL: https://imageresizer.com/bulk-resize

Occurrence 070

Status: **New**

Evidence: <link rel="preconnect" href="https://fonts.googleapis.com/">

Method: GET

URL: https://imageresizer.com/about

Subpart C, 164.312(c)(1): Possible Issue - A covered entity must implement policies and procedures to protect private health information from improper alteration or destruction.

Improper Cache-Control Configuration

OWASP PC-C8

CWE-525

Subpart C, HIPAA-164.312(c)(1)

ISO27001-A.14.1.3

WASC-13

WSTG-ATHN-06

- Likelihood: Medium
- Impact: Low
- Severity: Low

Issue Description:

Inadequate caching settings can lead to sensitive information being stored locally on clients or proxies. Ensure that the cache-control header is properly configured for all resources.

Recommendations:

To prevent unauthorized caching, set the cache-control HTTP header to 'no-cache, no-store, must-revalidate'. For public assets, consider setting 'public, max-age, immutable' instead.

Occurrences:

Occurrence 071

Status: New

Evidence:	max-age=3600
Method:	GET
Parameter:	cache-control
URL:	https://imageresizer.com/heic-to-jpg

Occurrence 072

Status: New

Evidence:	max-age=3600
Method:	GET
Parameter:	cache-control
URL:	https://imageresizer.com/it

Occurrence 073

Status: New

Evidence:	max-age=3600
Method:	GET
Parameter:	cache-control
URL:	https://imageresizer.com/en

Occurrence 074

Status: New

Evidence: max-age=3600

Method: GET

Parameter: cache-control

URL: https://imageresizer.com/de

Occurrence 075

Status: New

Evidence: max-age=3600

Method: GET

Parameter: cache-control

URL: https://imageresizer.com/fr

Occurrence 076

Status: New

Evidence: max-age=3600

Method: GET

Parameter: cache-control

URL: https://imageresizer.com/es

Occurrence 077

Status: New

Evidence: max-age=3600

Method: GET

Parameter: cache-control

URL: https://imageresizer.com/bn

Occurrence 078

Status: **New**

Evidence: max-age=3600

Method: GET

Parameter: cache-control

URL: <https://imageresizer.com/image-converter>

Occurrence 079

Status: **New**

Evidence: max-age=3600

Method: GET

Parameter: cache-control

URL: <https://imageresizer.com/crop-image>

Occurrence 080

Status: **New**

Evidence: max-age=3600

Method: GET

Parameter: cache-control

URL: <https://imageresizer.com/about>

Occurrence 081

Status: **New**

Evidence: max-age=3600

Method: GET

Parameter: cache-control

URL: <https://imageresizer.com/bulk-resize>

Occurrence 082

Status: **New**

Evidence: max-age=3600

Method: GET

Parameter: cache-control

URL: <https://imageresizer.com/>

Occurrence 083

Status: **New**

Evidence: max-age=3600

Method: GET

Parameter: cache-control

URL: <https://imageresizer.com/jpg-to-png>

Occurrence 084

Status: **New**

Evidence: max-age=3600

Method: GET

Parameter: cache-control

URL: <https://imageresizer.com/image-enlarger>

Occurrence 085

Status: **New**

Evidence: max-age=3600

Method:	GET
Parameter:	cache-control
URL:	https://imageresizer.com/flip-image

Occurrence 086

Status: **New**

Evidence:	max-age=3600
Method:	GET
Parameter:	cache-control
URL:	https://imageresizer.com/color-picker

Occurrence 087

Status: **New**

Evidence:	max-age=3600
Method:	GET
Parameter:	cache-control
URL:	https://imageresizer.com/changelogs

Occurrence 088

Status: **New**

Evidence:	max-age=3600
Method:	GET
Parameter:	cache-control
URL:	https://imageresizer.com/ja

Occurrence 089

Status: **New**

Evidence:	max-age=3600
Method:	GET
Parameter:	cache-control
URL:	https://imageresizer.com/id

Occurrence 090Status: New

Evidence:	max-age=3600
Method:	GET
Parameter:	cache-control
URL:	https://imageresizer.com/image-compressor

Subpart C, 164.312(e)(1): A covered entity must implement technical security measures to guard against unauthorized access to private health information that is being transmitted over an electronic communications network.

SWEET32 attack

OWASP 2013-A9

OWASP 2017-A9

OWASP 2021-A6

PCI V3.2-6.5.4

PCI v4.0-6.2.4

OWASP PC-C1

CAPEC-217

CWE-326

Subpart C, HIPAA-164.312(e)(1)

ISO27001-A.14.1.3

WASC-04

WSTG-CRYP-01

A.12.3.1

- Likelihood: Medium
- Impact: High
- Severity: High

Issue Description:

The Sweet32 attack is an SSL/TLS vulnerability that allows attackers to compromise HTTPS connections using 64-bit block ciphers. An attacker can use this to run JavaScript in a browser and send enormous quantities of traffic over a TLS connection, causing a collision. The attacker can extract information from a session cookie via this collision.

Recommendations:

Step 1: Understand the SWEET32 Attack The SWEET32 attack is a vulnerability that affects the use of the Triple DES (3DES) cipher in TLS and SSL protocols. This vulnerability allows an attacker to decrypt sensitive information such as login credentials, credit card numbers, and other confidential data that is encrypted using 3DES.

Step 2: Identify Affected Systems The first step in mitigating the SWEET32 attack is to identify which systems are vulnerable. In this case, we need to check if any of the web applications listed in the vulnerability report use the 3DES cipher in their TLS or SSL protocols.

Step 3: Update TLS/SSL Protocols The most effective way to mitigate the SWEET32 attack is to disable the use of 3DES cipher in TLS and SSL protocols. This can be achieved by updating the TLS and SSL protocols to use more secure ciphers such as AES or ChaCha20.

For PHP: To update the TLS and SSL protocols in PHP, you will need to edit the `php.ini` file and add the following line: `openssl_cipher_list = "DEFAULT:!3DES"`

For Python: To update the TLS and SSL protocols in Python, you will need to use the `ssl` module and specify the ciphers to be used. For example: `ssl.wrap_socket(sock, ciphers='ECDHE-RSA-AES256-GCM-SHA384')`

For Ruby: To update the TLS and SSL protocols in Ruby, you will need to use the OpenSSL library and specify the ciphers to be used. For example:

```
OpenSSL::SSL::SSLContext.new.tap do |ctx| ctx.ciphers = 'TLSv1.2:!aNULL:!eNULL' end
```

For Java: To update the TLS and SSL protocols in Java, you will need to use the `SSLContext` class and specify the ciphers to be used. For example:

```
SSLContext sslContext = SSLContext.getInstance("TLSv1.2"); sslContext.init(null, null, null);  
SSLParameters params = sslContext.getDefaultSSLParameters();  
params.setCipherSuites(new String[]
```

```
{"TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",  
"TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256"});
```

```
sslContext.getDefaultSSLParameters().setCipherSuites(params.getCipherSuites());
```

For JSP: To update the TLS and SSL protocols in JSP, you will need to configure the server to use more secure ciphers. This can be achieved by editing the `server.xml` file and specifying the ciphers to be used.

For ASP: To update the TLS and SSL protocols in ASP, you will need to edit the server configuration to use more secure ciphers. This can be achieved by editing the `web.config` file and specifying the ciphers to be used.

For JavaScript: To update the TLS and SSL protocols in JavaScript, you will need to use the TLS/SSL API and specify the ciphers to be used. For example: `var tls = require('tls'); var options = { ciphers: 'DEFAULT:!3DES' }; var secureContext = tls.createSecureContext(options);`

For Perl: To update the TLS and SSL protocols in Perl, you will need to use the Net::SSL module and specify the ciphers to be used. For example: `use Net::SSL; my $ctx = Net::SSL::CTX_new();`

`Net::SSL::CTX_set_cipher_list($ctx, "DEFAULT:!3DES");`

For Node.js: To update the TLS and SSL protocols in Node.js, you will need to use the tls module and specify the ciphers to be used. For example: `const tls = require('tls'); const options = { ciphers: 'DEFAULT:!3DES' }; const secureContext = tls.createSecureContext(options);`

Step 4: Update Web Frameworks In addition to updating the TLS and SSL protocols, it is also important to update the web frameworks used in the web applications to the latest versions. This will ensure that any known vulnerabilities are patched and the applications are more secure.

Step 5: Disable 3DES Cipher in Databases If any of the databases listed in the vulnerability report use the 3DES cipher, it is important to disable it. This can be achieved by updating the database configuration to use more secure ciphers.

For Microsoft SQL Server: To disable the 3DES cipher in Microsoft SQL Server, you will need to edit the registry and add the following key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SC HANNEL\Ciphers\Triple DES 168

For PostgreSQL: To disable the 3DES cipher in PostgreSQL, you will need to edit the postgresql.conf file and add the following line: `ssl_ciphers = 'DEFAULT:!3DES'`

For Oracle: To disable the 3DES cipher in Oracle, you will need to edit the

Occurrences:

Occurrence 091

Status: **New**

URL: <https://imageresizer.com/>

BREACH Attack

OWASP 2019-API7

OWASP 2013-A9

OWASP 2017-A9

OWASP 2021-A6

PCI v4.0-6.2.4

OWASP PC-C1

CWE-310

Subpart C, HIPAA-164.312(e)(1)

WASC-04

WSTG-CLNT-10

A.12.3.1

- Likelihood: Low

- Impact: Low
- Severity: Info

Issue Description:

The BREACH (Browser Reconnaissance & Exfiltration via Adaptive Compression of Hypertext) attack is a security vulnerability that specifically targets websites using HTTPS with compression enabled. This attack exploits the compression algorithms employed in HTTPS to extract sensitive information, such as authentication tokens or session cookies, from encrypted traffic.

For a website to be vulnerable to the BREACH attack, the following conditions must be met:

- HTTP Compression Enabled**: The website must use HTTP compression.
- User Input Reflected in Response Body**: The application must reflect user input in its HTTP response body.
- Exposure of Secrets in Response Body**: The application must expose sensitive information (e.g., CSRF tokens) in the response body.

Recommendations:

- Disable Compression at the HTTP Level: Ensure that HTTP compression is turned off to prevent attackers from leveraging the compression side-channel.
- Obfuscate Traffic Length: Introduce variability in response lengths by appending a random number of bytes to each response. This randomness makes it harder for attackers to infer sensitive information based on response size.
- Implement Rate Limiting: Enforce rate limits to restrict the number of times a page can be accessed within a certain period. For example, limit access to a maximum of five times per minute to reduce the attacker's ability to gather enough data for a successful exploit.

Occurrences:

Occurrence 092

Status: [New](#)

URL: <https://imageresizer.com/>

TLS 1.1 outdated version

OWASP 2013-A5

OWASP 2017-A6

OWASP 2021-A5

PCI v4.0-6.2.4

OWASP PC-C1

CAPEC-217

CWE-326

Subpart C, HIPAA-164.312(e)(1)

WSTG-CRYP-01

A12.3.1

- Likelihood: High
- Impact: High
- Severity: Critical

Issue Description:

This site has TLS 1.1 outdated version. The pseudo-random function (PRF) in TLS 1.1 is based on a combination of MD5 and SHA-1. There is no known weakness in the PRF of TLS 1.1. Nevertheless, MD5 and SHA-1 are "bad press". Therefore TLS 1.1 is vulnerable to poodle and beast attacks.

Recommendations:

Step 1: Identify and Confirm TLS 1.1 Outdated Version The first step in mitigating the TLS 1.1 outdated version vulnerability is to identify and confirm that the web application is indeed using TLS 1.1. This can be done by running a scan using a vulnerability scanner or by checking the server configuration.

Step 2: Disable TLS 1.1 The most effective way to mitigate the TLS 1.1 outdated version vulnerability is to disable TLS 1.1 on the server. This can be done by modifying the server configuration for each of the technologies listed in the vulnerability report. Below are the steps to disable TLS 1.1 for some of the commonly used technologies:

Apache:

1. Open the Apache configuration file (httpd.conf).
2. Look for the line "SSLProtocol" and add "-TLSv1.1" to the end of it. It should look like this: SSLProtocol all -SSLv2 -SSLv3 -TLSv1.1
3. Save the changes and restart Apache.

Nginx:

1. Open the Nginx configuration file (nginx.conf).
2. Look for the line "ssl_protocols" and add "TLSv1.1" to the list. It should look like this: ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
3. Save the changes and restart Nginx.

IIS:

1. Open the IIS Manager.
2. Select the server and click on "SSL Settings" under "IIS" section.
3. Uncheck the box next to "TLS 1.1" under "Protocols" section.
4. Click "Apply" to save the changes.

Step 3: Update TLS Version in Code In addition to disabling TLS 1.1 on the server, it is also important to update the TLS version in the code of the web application. This will ensure that the web application is not using TLS 1.1 even if it is enabled on the server. Below are the steps to update the TLS version for some of the commonly used programming languages and frameworks:

PHP:

1. Open the php.ini file.
2. Look for the line "openssl.default_tls_version" and change the value to "TLSv1.2".
3. Save the changes and restart the web server.

Python:

1. Open the code for the web application.
2. Look for the line "ssl.PROTOCOL_TLSv1_1" and change it to "ssl.PROTOCOL_TLSv1_2".
3. Save the changes and restart the web server.

Java:

1. Open the code for the web application.
2. Look for the line "SSLContext.getInstance("TLSv1.1")" and change it to "SSLContext.getInstance("TLSv1.2")".
3. Save the changes and restart the web server.

Step 4: Test and Verify After making the above changes, it is important to test and verify that TLS 1.1 is disabled and the web application is using a secure version of TLS. This can be done by running a scan using a vulnerability scanner or by using an online SSL checker tool.

Step 5: Update Server Operating System and Software In addition to the above steps, it is also important to keep the server operating system and software up to date with the latest security patches. This will ensure that any known vulnerabilities in the server software are patched and the server is secure.

Step 6: Regularly Monitor and Update It is important to regularly monitor and update the server and web application to ensure that TLS 1.1 is not re-enabled and that the web application is always using the latest secure version of TLS. This can be

done by setting up regular vulnerability scans and keeping track of updates for the server and web application technologies.

In conclusion, the TLS 1.1 outdated version vulnerability can be effectively mitigated by disabling TLS 1.1 on the server, updating the TLS version in the code, and keeping the server and web application up to date. It is important to regularly monitor and update to ensure that the web application is always using a secure version of TLS.

Occurrences:

Occurrence 093

Status: New

URL: <https://imageresizer.com/>

TLS 1.0 outdated version

OWASP 2013-A5

OWASP 2017-A6

OWASP 2021-A5

PCI v4.0-6.2.4

OWASP PC-C1

CAPEC-217

CWE-326

Subpart C, HIPAA-164.312(e)(1)

WSTG-CRYP-01

A.12.3.1

- Likelihood: High
- Impact: High
- Severity: Critical

Issue Description:

TLS(Transport Layer Security) ensures that communications are encrypted for security and privacy. There are various problems with TLS 1.0. An attacker might create connection failures and exploit vulnerabilities like BEAST (Browser Exploit Against SSL/TLS) by using TLS 1.0. Also Attackers can perform man-in-the-middle attacks and observe the encryption traffic between your website and its visitors.

Recommendations:

Step 1: Disable TLS 1.0 on the Web Server The first step in mitigating the TLS 1.0 outdated version vulnerability is to disable TLS 1.0 on the web server. This can be done by modifying the server's configuration file for each technology stack. Below are the steps to disable TLS 1.0 for the most common web servers:

IIS:

1. Open the Registry Editor and navigate to
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.0\Server.
2. Right-click on the Server key and select New > DWORD (32-bit) Value.
3. Name the new value Enabled and set its value to 0.
4. Repeat steps 2 and 3 for the Client key.
5. Restart the IIS server for the changes to take effect.

Apache:

1. Open the Apache configuration file (httpd.conf) and search for the line
"SSLProtocol".
2. Replace "SSLProtocol all -SSLv3 -TLSv1" with "SSLProtocol all -SSLv3 -TLSv1 -
TLSv1.1".
3. Save the file and restart Apache for the changes to take effect.

Nginx:

1. Open the Nginx configuration file (nginx.conf) and search for the line
"ssl_protocols".
2. Replace "ssl_protocols TLSv1 TLSv1.1 TLSv1.2" with "ssl_protocols TLSv1.2".
3. Save the file and restart Nginx for the changes to take effect.

Step 2: Update SSL/TLS Libraries The next step is to update the SSL/TLS libraries used by the web server. This will ensure that the server is using the latest and most secure versions of these libraries. Below are the steps to update the SSL/TLS libraries for the most common web servers:

IIS:

1. Download and install the latest version of the Microsoft Schannel Security Support Provider (SSP) update.
2. Restart the IIS server for the changes to take effect.

Apache:

1. Download and install the latest version of the OpenSSL library.
2. Restart Apache for the changes to take effect.

Nginx:

1. Download and install the latest version of the OpenSSL library.
2. Rebuild Nginx with the new library using the following command: "nginx -V".
3. Restart Nginx for the changes to take effect.

Step 3: Configure TLS 1.2 as the Default Protocol The next step is to configure TLS 1.2 as the default protocol for all web server technologies. This will ensure that all connections to the server are using the most secure version of TLS. Below are the steps to configure TLS 1.2 as the default protocol for each technology stack:

PHP:

1. Open the php.ini file and search for the line "openssl.default_tls_version".
2. Replace "openssl.default_tls_version = TLSv1" with "openssl.default_tls_version = TLSv1.2".
3. Save the file and restart the web server for the changes to take effect.

Python:

1. Open the Python configuration file (python.ini) and search for the line "ssl_version".
2. Replace "ssl_version = PROTOCOL_TLSv1" with "ssl_version = PROTOCOL_TLSv1_2".
3. Save the file and restart the web server for the changes to take effect.

Ruby:

1. Open the Ruby configuration file (ruby.ini) and search for the line "ssl_version".
2. Replace "ssl_version = :TLSv1" with "ssl_version = :TLSv1_2".
3. Save the file and restart the web server for the changes to take effect.

Java:

1. Open the Java configuration file (java.ini) and search for the line "jdk.tls.client.protocols".
2. Replace "jdk.tls.client.protocols=TLSv1" with "jdk.tls.client.protocols=TLSv1.2".
3. Save the file and restart the web server for the changes to take effect.

ASP:

1. Open the ASP configuration file (asp.ini) and search for the line "sslProtocol".
2. Replace "sslProtocol = TLSv1" with "sslProtocol = TLSv1.2".
3. Save the file and restart the web server for the changes to take effect.

JavaScript:

1. Open the JavaScript configuration file (javascript.ini) and search for the line "tls_version".
2. Replace "tls_version = TLSv1" with "tls_version = TLSv1.2".
3. Save the file and restart the web server for the changes to take effect.

Perl:

1. Open the Perl configuration file (perl.ini) and search for the line "ssl_version".
2. Replace "ssl_version = TLSv

Occurrences:

Occurrence 094

Status: **New**

URL: <https://imageresizer.com/>

HTTP Strict Transport Security (HSTS) header not implemented

OWASP 2019-API7

OWASP 2013-A5

OWASP 2017-A6

OWASP 2021-A5

PCI v4.0-6.2.4

OWASP PC-C1

CAPEC-217

CWE-523

Subpart C, HIPAA-164.312(e)(1)

ISO27001-A.14.1.2

WASC-4

WSTG-CONF-07

A.12.1.1

- Likelihood: Medium
- Impact: Low
- Severity: Low

Issue Description:

In this website, the HTTP Strict Transport Security (HSTS) policy is not implemented. This website is being served from not only HTTP but also HTTPS and it lacks HSTS policy implementation. HTTP Strict Transport Security is a web security policy mechanism to interact with complying user agents such as a web browser using only secure HTTP connections.

Recommendations:

- Configure your web server to redirect HTTP requests to HTTPS.

Occurrences:

Occurrence 095

Status: **New**

URL: https://imageresizer.com/

Cached Data Retrieved

OWASP 2017-A3

OWASP 2021-A2

OWASP PC-C8

CAPEC-170

CWE-524

Subpart C, HIPAA-164.312(e)(1)

WASC-10

WSTG-ATHN-06

- Likelihood: Medium
- Impact: Low
- Severity: Low

Issue Description:

The content was retrieved from a shared cache. If the response contains sensitive, personal, or user-specific information, it could lead to information leakage or unauthorized session access. This is particularly an issue with proxy caches in corporate or educational environments.

Recommendations:

Ensure that responses do not contain sensitive data by implementing HTTP response headers such as Cache-Control: no-cache, must-revalidate, private, Pragma: no-cache, and Expires: 0. These configurations prevent caching servers from storing and retrieving the content without validation.

Occurrences:

Occurrence 096		Status: New
Evidence:	Age: 8269	
Method:	GET	
URL:	https://imageresizer.com/favicon.ico	
Occurrence 097		Status: New

Evidence:	Age: 5410
Method:	GET
URL:	https://imageresizer.com/crop-image

Occurrence 098

Status: New

Evidence:	Age: 10047
Method:	GET
URL:	https://imageresizer.com/_next/static/css/fccf139d35fdf47a.css

Occurrence 099

Status: New

Evidence:	Age: 8778
Method:	GET
URL:	https://imageresizer.com/image-compressor

Occurrence 100

Status: New

Evidence:	Age: 8317
Method:	GET
URL:	https://imageresizer.com/fr

Occurrence 101

Status: New

Evidence:	Age: 14267
Method:	GET

URL: <https://imageresizer.com/en>

Occurrence 102

Status: **New**

Evidence: Age: 8318

Method: GET

URL: <https://imageresizer.com/bn>

Occurrence 103

Status: **New**

Evidence: Age: 13557

Method: GET

URL: <https://imageresizer.com/changelogs>

Occurrence 104

Status: **New**

Evidence: Age: 8929

Method: GET

URL: <https://imageresizer.com/image-converter>

Occurrence 105

Status: **New**

Evidence: Age: 8317

Method: GET

URL: <https://imageresizer.com/de>

Occurrence 106

Status: **New**

Evidence:	Age: 8316
Method:	GET
URL:	https://imageresizer.com/id

Occurrence 107

Status: **New**

Evidence:	Age: 10065
Method:	GET
URL:	https://imageresizer.com/

Occurrence 108

Status: **New**

Evidence:	Age: 8318
Method:	GET
URL:	https://imageresizer.com/es

Occurrence 109

Status: **New**

Evidence:	Age: 8317
Method:	GET
URL:	https://imageresizer.com/about

Occurrence 110

Status: **New**

Evidence:	Age: 14330
Method:	GET

URL: <https://imageresizer.com/color-picker>

Occurrence 111

Status: **New**

Evidence: Age: 8316

Method: GET

URL: <https://imageresizer.com/heic-to-jpg>

Occurrence 112

Status: **New**

Evidence: Age: 2095

Method: GET

URL: <https://imageresizer.com/bulk-resize>

Occurrence 113

Status: **New**

Evidence: Age: 11194

Method: GET

URL: <https://imageresizer.com/flip-image>

Occurrence 114

Status: **New**

Evidence: Age: 9901

Method: GET

URL: https://imageresizer.com/_next/static/chunks/pages/crop-image-8c61f299065bcb9e.js

Cacheable and Storable Content

OWASP 2017-A6

OWASP 2021-A5

OWASP PC-C8

CWE-524

Subpart C, HIPAA-164.312(e)(1)

WASC-13

WSTG-ATHN-06

- Likelihood: Medium
- Impact: Low
- Severity: Low

Issue Description:

A caching vulnerability occurs when sensitive information is stored and retrieved by caching components such as proxy servers, potentially leaking user data or enabling session hijacking attacks. This issue arises in environments with shared caching servers, commonly found in corporate or educational networks.

Recommendations:

To mitigate this risk, ensure that responses do not contain sensitive information. If necessary, set HTTP response headers like Cache-Control: no-cache, no-store, must-revalidate, private and Pragma: no-cache to prevent caching and retrieval of the content without validation.

Occurrences:

Occurrence 115

Status: **New**

Evidence: max-age=3600

Method: GET

URL: https://imageresizer.com/bulk-resize

Occurrence 116

Status: **New**

Evidence: max-age=3600

Method: GET

URL: <https://imageresizer.com/flip-image>

Occurrence 117

Status: **New**

Evidence: max-age=3600

Method: GET

URL: <https://imageresizer.com/color-picker>

Occurrence 118

Status: **New**

Evidence: max-age=3600

Method: GET

URL: <https://imageresizer.com/en>

Occurrence 119

Status: **New**

Evidence: max-age=3600

Method: GET

URL: <https://imageresizer.com/about>

Occurrence 120

Status: **New**

Evidence: max-age=3600

Method: GET

URL: <https://imageresizer.com/es>

Occurrence 121

Status: **New**

Evidence:	max-age=3600
Method:	GET
URL:	https://imageresizer.com/_next/static/chunks/pages/crop-image-8c61f299065bcb9e.js

Occurrence 122

Status: **New**

Evidence:	max-age=3600
Method:	GET
URL:	https://imageresizer.com/de

Occurrence 123

Status: **New**

Evidence:	max-age=3600
Method:	GET
URL:	https://imageresizer.com/id

Occurrence 124

Status: **New**

Evidence:	max-age=3600
Method:	GET
URL:	https://imageresizer.com/image-enlarger

Occurrence 125

Status: **New**

Evidence:	max-age=3600
Method:	GET

URL: <https://imageresizer.com/>

Occurrence 126

Status: **New**

Evidence: max-age=3600

Method: GET

URL: <https://imageresizer.com/heic-to-jpg>

Occurrence 127

Status: **New**

Evidence: max-age=3600

Method: GET

URL: https://imageresizer.com/_next/static/css/fccf139d35fdf47a.css

Occurrence 128

Status: **New**

Evidence: max-age=3600

Method: GET

URL: <https://imageresizer.com/bn>

Occurrence 129

Status: **New**

Evidence: max-age=3600

Method: GET

URL: <https://imageresizer.com/fr>

Occurrence 130

Status: **New**

Evidence: max-age=3600

Method: GET

URL: https://imageresizer.com/image-compressor

Occurrence 131

Status: New

Evidence: max-age=3600

Method: GET

URL: https://imageresizer.com/favicon.ico

Occurrence 132

Status: New

Evidence: max-age=3600

Method: GET

URL: https://imageresizer.com/image-converter

Occurrence 133

Status: New

Evidence: max-age=3600

Method: GET

URL: https://imageresizer.com/crop-image

Occurrence 134

Status: New

Evidence: max-age=3600

Method: GET

Subpart C, 164.312(e)(2)(i): Addressable Issue - Implement security measures to ensure that electronically transmitted private health information is not improperly modified without detection until disposed of.

Web Application Potentially Vulnerable to BEAST attack

OWASP 2019-API7

OWASP 2013-A9

OWASP 2017-A9

OWASP 2021-A6

PCI v4.0-6.2.4

OWASP PC-C1

Subpart C, HIPAA-164.312(e)(2)(i)

WASC-04

WSTG-CLNT-10

A.12.3.1

- Likelihood: Low
- Impact: Low
- Severity: Info

Issue Description:

The website is found to be potentially vulnerable to BEAST attack. This vulnerability exists in all cipher block chaining (CBC) ciphers in SSL V3/TLS 1.0. BEAST is a plaintext attack that obtains the victim's cookie value from an SSL/TLS encrypted session (by conducting a session hijacking attack), and then tampering with an encrypted CBC (cipher block chain)) Mode to implement the attack catalogue, which is mainly aimed at the CBC mode of the symmetric encryption algorithm in TLS1.0 and earlier versions of the protocol.

Recommendations:

We recommend you to

- Reduce the lifespan of the SSL session
- Support TLS 1.1 onward on the server.

Occurrences:

Occurrence 135

Status: [New](#)

URL: https://imageresizer.com/

Web Application Potentially Vulnerable to LUCKY13

OWASP 2019-API7

OWASP 2013-A9

OWASP 2017-A9

OWASP 2021-A6

PCI v4.0-6.2.4

OWASP PC-C1

Subpart C, HIPAA-164.312(e)(2)(i)

WASC-04

WSTG-CLNT-10

A.12.3.1

- Likelihood: Low
- Impact: Low
- Severity: Info

Issue Description:

The website is found to be potentially vulnerable to LUCKY13. LUCKY13 is a timing attack that can be used against implementations of the TLS protocol using the cipher block chaining mode of operation. All TLS and DTLS ciphersuites which include CBC-mode encryption are potentially vulnerable. Using a man-in-the-middle attack, an attacker can obtain sensitive data.

Recommendations:

- It is recommended to disable lower versions of TLS
- Upgrade OpenSSL
- Remove all cipher block chaining ciphers, AEAD cipher suites such as AES-GCM are recommended.

Occurrences:

Occurrence 136

Status: [New](#)

URL: https://imageresizer.com/

HIPAA Compliance Passed Controls

Subpart C, HIPAA-164.306(a)(2): Covered entities must protect against any reasonably anticipated threats or hazards to the security or integrity of electronic protected health information.

Subpart C, HIPAA-164.306(a)(3): Covered entities must protect against any reasonably anticipated uses or disclosures of electronic protected health information.

Subpart C, 164.308(a)(1)(ii)(B): Possible Issue - A covered entity must implement security measures sufficient to reduce risks and vulnerabilities to a reasonable and appropriate level.

Subpart C, 164.308(a)(5)(ii)(B): Addressable Issue - Implement procedures for guarding against, detecting, and reporting malicious software.

Subpart C, 164.308(a)(5)(ii)(C): Addressable Issue - Implement procedures for monitoring log-in attempts and reporting discrepancies.

Subpart C, 164.308(a)(5)(ii)(D): Addressable Issue - Implement procedures for creating, changing, and safeguarding passwords.

Subpart C, 164.308(a)(7)(i): Possible Issue - Establish (and implement as needed) procedures for responding to an emergency or other occurrence that damages systems that contain electronic protected health information.

Subpart C, 164.312(d): Possible Issue - A covered entity must implement procedures to verify that a person or entity seeking access to private health information is the one claimed.

Subpart C, 164.312(e)(2)(ii): Addressable Issue - Implement a mechanism to encrypt electronic private health information whenever deemed appropriate.



Conclusion

This application complies with 9 out of 16 controls in the HIPAA scan.

This HIPAA compliance report is prepared following the penetration test on Demo Application and the result will only cover as far as the “black-box”, or “grey-box” testing methods allow.

Visit the US Government Printing Office website for a complete copy of the HEALTH INSURANCE PORTABILITY AND ACCOUNTABILITY ACT OF 1996: <https://www.govinfo.gov/content/pkg/PLAW-104publ191/pdf/PLAW-104publ191.pdf>

The sub-requirements associated with the HIPAA guidelines on the application security standards determine the passing or failure of tests mentioned in this document. All other sub-requirements that do not belong to the criteria are not considered in this report.