April 29, 2024

# Vulnerability Scan Report

prepared by

**HostedScan Security**

hostedscan.com

# Overview

# 1 Executive Summary

Vulnerability scans were conducted on selected servers, networks, websites, and applications. This report contains the discovered potential risks from these scans. Risks have been classified into categories according to the level of threat and degree of potential harm they may pose.

## 1.1 Total Risks

Below is the total number of risks found by severity. High risks are the most severe and should be evaluated first. An accepted risk is one which has been manually reviewed and classified as acceptable to not fix at this time, such as a false positive or an intentional part of the system's architecture.

| Critical | High | Medium | Low | Accepted |
|---|---|---|---|---|
| 0 | 0 | 12 | 6 | 0 |

## 1.2 Report Coverage

This report includes findings for 1 target that were scanned. Each target is a single URL, IP address, or fully qualified domain name (FQDN).

### Vulnerability Categories

| | |
|---|---|
| 8 | Active Web Application Vulnerabilities |
| 8 | Passive Web Application Vulnerabilities |
| 0 | Network Vulnerabilities |
| 0 | SSL/TLS Security |
| 2 | Open TCP Ports |
| 0 | Open UDP Ports |

# 2 Risks By Target

This section contains the vulnerability findings for each target that was scanned. Prioritize the most vulnerable assets first.

## 2.1 Targets Summary

The total number of risks found for each target, by severity.

| Target | ● Critical | ● High | ● Medium | ● Low | ● Accepted |
|---|---|---|---|---|---|
| ● https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 | 0 | 12 | 6 | 0 |

## 2.2 Target Breakdowns

The risks discovered for each target.

Target
https://phamt1042.github.io/Front-End-27-2-Admin/

### Total Risks

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 12 | 6 | 0 |

| Active Web Application Vulnerabilities | Threat Level | First Detected | Last Detected |
|---|---|---|---|
| Absence of Anti-CSRF Tokens | ● Medium | 0 days ago | 0 days ago |
| Cross-Domain Misconfiguration | ● Medium | 0 days ago | 0 days ago |
| Missing Anti-clickjacking Header | ● Medium | 0 days ago | 0 days ago |
| CSP: Wildcard Directive | ● Medium | 0 days ago | 0 days ago |
| CSP: style-src unsafe-inline | ● Medium | 0 days ago | 0 days ago |
| Content Security Policy (CSP) Header Not Set | ● Medium | 0 days ago | 0 days ago |
| X-Content-Type-Options Header Missing | ● Low | 0 days ago | 0 days ago |
| Strict-Transport-Security Header Not Set | ● Low | 0 days ago | 0 days ago |

| Passive Web Application Vulnerabilities | Threat Level | First Detected | Last Detected |
|---|---|---|---|
| Absence of Anti-CSRF Tokens | ● Medium | 0 days ago | 0 days ago |
| Cross-Domain Misconfiguration | ● Medium | 0 days ago | 0 days ago |
| Missing Anti-clickjacking Header | ● Medium | 0 days ago | 0 days ago |
| CSP: Wildcard Directive | ● Medium | 0 days ago | 0 days ago |
| CSP: style-src unsafe-inline | ● Medium | 0 days ago | 0 days ago |
| Content Security Policy (CSP) Header Not Set | ● Medium | 0 days ago | 0 days ago |
| X-Content-Type-Options Header Missing | ● Low | 0 days ago | 0 days ago |

| | | | |
|---|---|---|---|
| Strict-Transport-Security Header Not Set | ● Low | 0 days ago | 0 days ago |

| Open TCP Ports | Threat Level | First Detected | Last Detected |
|---|---|---|---|
| Open TCP Port: 443 | ● Low | 0 days ago | 0 days ago |
| Open TCP Port: 80 | ● Low | 0 days ago | 0 days ago |

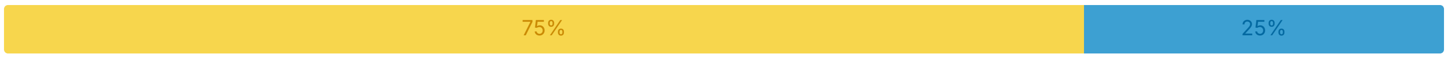
# 3 Active Web Application Vulnerabilities

The OWASP ZAP active web application scan crawls the pages of a web application. It scans for all of the passive scan checks and additionally makes requests and submits forms to actively test an application for even more vulnerabilities. The active scan checks for vulnerabilities such as SQL injection, remote command execution, XSS, and more.

## 3.1 Total Risks

Total number of risks found by severity.

| Critical | High | Medium | Low | Accepted |
|---|---|---|---|---|
| 0 | 0 | 6 | 2 | 0 |



## 3.2 Risks Breakdown

Summary list of all detected risks.

| Title | Threat Level | Open | Accepted |
|---|---|---|---|
| Absence of Anti-CSRF Tokens | ● Medium | 1 | 0 |
| Cross-Domain Misconfiguration | ● Medium | 1 | 0 |
| Missing Anti-clickjacking Header | ● Medium | 1 | 0 |
| CSP: Wildcard Directive | ● Medium | 1 | 0 |
| CSP: style-src unsafe-inline | ● Medium | 1 | 0 |
| Content Security Policy (CSP) Header Not Set | ● Medium | 1 | 0 |
| X-Content-Type-Options Header Missing | ● Low | 1 | 0 |
| Strict-Transport-Security Header Not Set | ● Low | 1 | 0 |

## 3.3 Full Risk Details

Detailed information about each risk found by the scan.

### Absence of Anti-CSRF Tokens

● Medium

#### Description

No Anti-CSRF tokens were found in a HTML submission form.

A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.

CSRF attacks are effective in a number of situations, including:

* The victim has an active session on the target site.

* The victim is authenticated via HTTP auth on the target site.

* The victim is on the same local network as the target site.

CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.

#### Solution

Phase: Architecture and Design

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

For example, use anti-CSRF packages such as the OWASP CSRFGuard.

Phase: Implementation

Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script.

Phase: Architecture and Design

Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330).

Note that this can be bypassed using XSS.

Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation.

Note that this can be bypassed using XSS.

Use the ESAPI Session Management control.

This control includes a component for CSRF.

Do not use the GET method for any request that triggers a state change.

Phase: Implementation

Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.

### Instances (1 of 1)

uri: https://phamt1042.github.io/Front-End-27-2-Admin/
method: GET
evidence: <form action="" method="POST" class="form" id="form-login">
otherinfo: No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token, data[_Token][key]] was found in the following HTML form: [Form 1: "fullname-login" "password-login" ].

### References

https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
https://cwe.mitre.org/data/definitions/352.html

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## Cross-Domain Misconfiguration

● Medium

### Description

Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server

### Solution

Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance).

Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.

#### Instances (1 of 2)

uri: https://phamt1042.github.io/Front-End-27-2-Admin/

method: GET

evidence: Access-Control-Allow-Origin: *

otherinfo: The CORS misconfiguration on the web server permits cross-domain read requests from arbitrary third party domains, using unauthenticated APIs on this domain. Web browser implementations do not permit arbitrary third parties to read the response from authenticated APIs, however. This reduces the risk somewhat. This misconfiguration could be used by an attacker to access data that is available in an unauthenticated manner, but which uses some other form of security, such as IP address white-listing.

#### References

https://vulncat.fortify.com/en/detail?id=desc.config.dotnet.html5_overly_permissive_cors_policy

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## Missing Anti-clickjacking Header

● Medium

### Description

The response does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks.

### Solution

Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app.

If you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

#### Instances (1 of 1)

uri: https://phamt1042.github.io/Front-End-27-2-Admin/
method: GET
param: x-frame-options

#### References

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## CSP: Wildcard Directive

● Medium

### Description

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

### Solution

Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.

#### Instances (1 of 6)

uri: https://phamt1042.github.io/
method: GET
param: content-security-policy
evidence: default-src 'none'; style-src 'unsafe-inline'; img-src data:; connect-src 'self'
otherinfo: The following directives either allow wildcard sources (or ancestors), are not defined, or are overly broadly defined: form-action The directive(s): form-action are among the directives that do not fallback to default-src, missing/excluding them is the same as allowing anything.

#### References

https://www.w3.org/TR/CSP/
https://caniuse.com/#search=content+security+policy
https://content-security-policy.com/
https://github.com/HtmlUnit/htmlunit-csp
https://developers.google.com/web/fundamentals/security/csp#policy_applies_to_a_wide_variety_of_resources

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## CSP: style-src unsafe-inline

● Medium

### Description

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

### Solution

Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.

#### Instances (1 of 3)

uri: https://phamt1042.github.io/
method: GET
param: Content-Security-Policy
evidence: default-src 'none'; style-src 'unsafe-inline'; img-src data:; connect-src 'self'
otherinfo: style-src includes unsafe-inline.

#### References

https://www.w3.org/TR/CSP/
https://caniuse.com/#search=content+security+policy
https://content-security-policy.com/
https://github.com/HtmlUnit/htmlunit-csp
https://developers.google.com/web/fundamentals/security/csp#policy_applies_to_a_wide_variety_of_resources

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## Content Security Policy (CSP) Header Not Set

● Medium

### Description

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

### Solution

Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

#### Instances (1 of 1)

uri: https://phamt1042.github.io/Front-End-27-2-Admin/
method: GET

#### References

https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy
https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
https://www.w3.org/TR/CSP/
https://w3c.github.io/webappsec-csp/
https://web.dev/articles/csp
https://caniuse.com/#feat=contentsecuritypolicy
https://content-security-policy.com/

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## X-Content-Type-Options Header Missing

● Low

### Description

The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

### Solution

Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages.

If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.

#### Instances (1 of 2)

uri: https://phamt1042.github.io/Front-End-27-2-Admin/
method: GET
param: x-content-type-options
otherinfo: This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scan rule will not alert on client or server error responses.

#### References

https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941(v=vs.85)
https://owasp.org/www-community/Security_Headers

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## Strict-Transport-Security Header Not Set

● Low

### Description

HTTP Strict Transport Security (HSTS) is a web security policy mechanism whereby a web server declares that complying user agents (such as a web browser) are to interact with it using only secure HTTPS connections (i.e. HTTP layered over TLS/SSL). HSTS is an IETF standards track protocol and is specified in RFC 6797.

### Solution

Ensure that your web server, application server, load balancer, etc. is configured to enforce Strict-Transport-Security.

#### Instances (1 of 4)

uri: https://phamt1042.github.io/
method: GET

#### References

https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html
https://owasp.org/www-community/Security_Headers
https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security
https://caniuse.com/stricttransportsecurity
https://datatracker.ietf.org/doc/html/rfc6797

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

# 4 Passive Web Application Vulnerabilities

The OWASP ZAP passive web application scan crawls the pages of a web application. It inspects the web pages as well as the requests and responses sent between the server. The passive scan checks for vulnerabilities such as cross-domain misconfigurations, insecure cookies, vulnerable js dependencies, and more.

## 4.1 Total Risks

Total number of risks found by severity.

| Critical | High | Medium | Low | Accepted |
|---|---|---|---|---|
| 0 | 0 | 6 | 2 | 0 |

## 4.2 Risks Breakdown

Summary list of all detected risks.

| Title | Threat Level | Open | Accepted |
|---|---|---|---|
| Absence of Anti-CSRF Tokens | ● Medium | 1 | 0 |
| Cross-Domain Misconfiguration | ● Medium | 1 | 0 |
| Missing Anti-clickjacking Header | ● Medium | 1 | 0 |
| CSP: Wildcard Directive | ● Medium | 1 | 0 |
| CSP: style-src unsafe-inline | ● Medium | 1 | 0 |
| Content Security Policy (CSP) Header Not Set | ● Medium | 1 | 0 |
| X-Content-Type-Options Header Missing | ● Low | 1 | 0 |
| Strict-Transport-Security Header Not Set | ● Low | 1 | 0 |

## 4.3 Full Risk Details

Detailed information about each risk found by the scan.

### Absence of Anti-CSRF Tokens

● Medium

#### Description

No Anti-CSRF tokens were found in a HTML submission form.

A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.

CSRF attacks are effective in a number of situations, including:

* The victim has an active session on the target site.

* The victim is authenticated via HTTP auth on the target site.

* The victim is on the same local network as the target site.

CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.

#### Solution

Phase: Architecture and Design

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

For example, use anti-CSRF packages such as the OWASP CSRFGuard.

Phase: Implementation

Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script.

Phase: Architecture and Design

Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330).

Note that this can be bypassed using XSS.

Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation.

Note that this can be bypassed using XSS.

Use the ESAPI Session Management control.

This control includes a component for CSRF.

Do not use the GET method for any request that triggers a state change.

Phase: Implementation

Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.

### Instances (1 of 1)

uri: https://phamt1042.github.io/Front-End-27-2-Admin/
method: GET
evidence: <form action="" method="POST" class="form" id="form-login">
otherinfo: No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token, data[_Token][key]] was found in the following HTML form: [Form 1: "fullname-login" "password-login" ].

### References

https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
https://cwe.mitre.org/data/definitions/352.html

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## Cross-Domain Misconfiguration

● Medium

### Description

Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server

### Solution

Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance).

Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.

#### Instances (1 of 2)

uri: https://phamt1042.github.io/Front-End-27-2-Admin/

method: GET

evidence: Access-Control-Allow-Origin: *

otherinfo: The CORS misconfiguration on the web server permits cross-domain read requests from arbitrary third party domains, using unauthenticated APIs on this domain. Web browser implementations do not permit arbitrary third parties to read the response from authenticated APIs, however. This reduces the risk somewhat. This misconfiguration could be used by an attacker to access data that is available in an unauthenticated manner, but which uses some other form of security, such as IP address white-listing.

#### References

https://vulncat.fortify.com/en/detail?id=desc.config.dotnet.html5_overly_permissive_cors_policy

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## Missing Anti-clickjacking Header

● Medium

### Description

The response does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks.

### Solution

Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app.

If you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

#### Instances (1 of 1)

uri: https://phamt1042.github.io/Front-End-27-2-Admin/
method: GET
param: x-frame-options

#### References

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## CSP: Wildcard Directive

● Medium

### Description

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

### Solution

Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.

#### Instances (1 of 6)

uri: https://phamt1042.github.io/
method: GET
param: content-security-policy
evidence: default-src 'none'; style-src 'unsafe-inline'; img-src data:; connect-src 'self'
otherinfo: The following directives either allow wildcard sources (or ancestors), are not defined, or are overly broadly defined: form-action The directive(s): form-action are among the directives that do not fallback to default-src, missing/excluding them is the same as allowing anything.

#### References

https://www.w3.org/TR/CSP/
https://caniuse.com/#search=content+security+policy
https://content-security-policy.com/
https://github.com/HtmlUnit/htmlunit-csp
https://developers.google.com/web/fundamentals/security/csp#policy_applies_to_a_wide_variety_of_resources

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## CSP: style-src unsafe-inline

● Medium

### Description

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

### Solution

Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.

#### Instances (1 of 3)

uri: https://phamt1042.github.io/
method: GET
param: Content-Security-Policy
evidence: default-src 'none'; style-src 'unsafe-inline'; img-src data:; connect-src 'self'
otherinfo: style-src includes unsafe-inline.

#### References

https://www.w3.org/TR/CSP/
https://caniuse.com/#search=content+security+policy
https://content-security-policy.com/
https://github.com/HtmlUnit/htmlunit-csp
https://developers.google.com/web/fundamentals/security/csp#policy_applies_to_a_wide_variety_of_resources

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## Content Security Policy (CSP) Header Not Set

● Medium

### Description

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

### Solution

Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

#### Instances (1 of 1)

uri: https://phamt1042.github.io/Front-End-27-2-Admin/
method: GET

#### References

https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy
https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
https://www.w3.org/TR/CSP/
https://w3c.github.io/webappsec-csp/
https://web.dev/articles/csp
https://caniuse.com/#feat=contentsecuritypolicy
https://content-security-policy.com/

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## X-Content-Type-Options Header Missing

● Low

### Description

The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

### Solution

Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages.

If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.

#### Instances (1 of 2)

uri: https://phamt1042.github.io/Front-End-27-2-Admin/
method: GET
param: x-content-type-options
otherinfo: This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scan rule will not alert on client or server error responses.

#### References

https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941(v=vs.85)
https://owasp.org/www-community/Security_Headers

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## Strict-Transport-Security Header Not Set

● Low

### Description

HTTP Strict Transport Security (HSTS) is a web security policy mechanism whereby a web server declares that complying user agents (such as a web browser) are to interact with it using only secure HTTPS connections (i.e. HTTP layered over TLS/SSL). HSTS is an IETF standards track protocol and is specified in RFC 6797.

### Solution

Ensure that your web server, application server, load balancer, etc. is configured to enforce Strict-Transport-Security.

#### Instances (1 of 4)

uri: https://phamt1042.github.io/
method: GET

#### References

https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html
https://owasp.org/www-community/Security_Headers
https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security
https://caniuse.com/stricttransportsecurity
https://datatracker.ietf.org/doc/html/rfc6797

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

# 5 SSL/TLS Security

The SSLyze security scan checks for misconfigured SSL/TLS certificates, expired certificates, weak ciphers, and SSL/TLS vulnerabilities such as Heartbleed.

## 5.1 Total Risks

Total number of risks found by severity.

| Critical | High | Medium | Low | Accepted |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |

## 5.2 Risks Breakdown

Summary list of all detected risks.

| Title | Threat Level | Open | Accepted |
|---|---|---|---|
| No risks detected | | | |

# 6 Network Vulnerabilities

The OpenVAS network vulnerability scan tests servers and internet connected devices for over 50,000 vulnerabilities. OpenVAS uses the Common Vulnerability Scoring System (CVSS) to quantify the severity of findings. 0.0 is the lowest severity and 10.0 is the highest.

## 6.1 Total Risks

Total number of risks found by severity.

| Critical | High | Medium | Low | Accepted |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |

## 6.2 Risks Breakdown

Summary list of all detected risks.

| Title | Threat Level | CVSS Score | Open | Accepted |
|---|---|---|---|---|
| No risks detected | | | | |

# 7 Open TCP Ports

The NMAP TCP port scan discovers open TCP ports with a complete scan of ports 0 to 65535.

## 7.1 Total Risks

Total number of risks found by severity.

| Critical | High | Medium | Low | Accepted |
|---|---|---|---|---|
| 0 | 0 | 0 | 2 | 0 |

100%

## 7.2 Risks Breakdown

Summary list of all detected risks.

| Title | Threat Level | Open | Accepted |
|---|---|---|---|
| Open TCP Port: 443 | ● Low | 1 | 0 |
| Open TCP Port: 80 | ● Low | 1 | 0 |

## 7.3 Full Risk Details

Detailed information about each risk found by the scan.

### Open TCP Port: 443

● Low

#### Description

An open port may be an expected configuration. For example, web servers use port 80 to serve websites over http and port 443 to serve websites over https. For a list of commonly used ports see https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers.

An unexpected open port could give unintended access to applications, data, and private networks. Open ports can also be dangerous when expected services are out of date and exploited through security vulnerabilities.

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

## Open TCP Port: 80

● Low

### Description

An open port may be an expected configuration. For example, web servers use port 80 to serve websites over http and port 443 to serve websites over https. For a list of commonly used ports see https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers.

An unexpected open port could give unintended access to applications, data, and private networks. Open ports can also be dangerous when expected services are out of date and exploited through security vulnerabilities.

| Vulnerable Target | First Detected | Last Detected |
|---|---|---|
| https://phamt1042.github.io/Front-End-27-2-Admin/ | 0 days ago | 0 days ago |

# 8 Open UDP Ports

The NMAP UDP port scan discovers open ports of common UDP services

## 8.1 Total Risks

Total number of risks found by severity.

| Critical | High | Medium | Low | Accepted |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |

## 8.2 Risks Breakdown

Summary list of all detected risks.

| Title | Threat Level | Open | Accepted |
|---|---|---|---|
| No risks detected | | | |

# 9 Glossary

### Accepted Risk

An accepted risk is one which has been manually reviewed and classified as acceptable to not fix at this time, such as a false positive or an intentional part of the system's architecture.

### Active Web Application Vulnerabilities

The OWASP ZAP active web application scan crawls the pages of a web application. It scans for all of the passive scan checks and additionally makes requests and submits forms to actively test an application for even more vulnerabilities. The active scan checks for vulnerabilities such as SQL injection, remote command execution, XSS, and more.

### Fully Qualified Domain Name (FQDN)

A fully qualified domain name is a complete domain name for a specific website or service on the internet. This includes not only the website or service name, but also the top-level domain name, such as .com, .org, .net, etc. For example, 'www.example.com' is an FQDN.

### Passive Web Application Vulnerabilities

The OWASP ZAP passive web application scan crawls the pages of a web application. It inspects the web pages as well as the requests and responses sent between the server. The passive scan checks for vulnerabilities such as cross-domain misconfigurations, insecure cookies, vulnerable js dependencies, and more.

### Network Vulnerabilities

The OpenVAS network vulnerability scan tests servers and internet connected devices for over 50,000 vulnerabilities. OpenVAS uses the Common Vulnerability Scoring System (CVSS) to quantify the severity of findings. 0.0 is the lowest severity and 10.0 is the highest.

### Open TCP Ports

The NMAP TCP port scan discovers open TCP ports with a complete scan of ports 0 to 65535.

### Open UDP Ports

The NMAP UDP port scan discovers open ports of common UDP services

### Risk

A risk is a finding from a vulnerability scan. Each risk is a potential security issue that needs review. Risks are assigned a threat level which represents the potential severity.

### SSL/TLS Security

The SSLyze security scan checks for misconfigured SSL/TLS certificates, expired certificates, weak ciphers, and SSL/TLS vulnerabilities such as Heartbleed.

### Target

A target represents target is a single URL, IP address, or fully qualified domain name (FQDN) that was scanned.

### Threat Level

The threat level represents the estimated potential severity of a particular risk. Threat level is divided into 4 categories: High, Medium, Low and Accepted.

### Threat Level

The threat level represents the estimated potential severity of a particular risk. Threat level is divided into 5 categories: Critical, High, Medium, Low and Accepted.

### CVSS Score

The CVSS 3.0 score is a global standard for evaluating vulnerabilities with a 0 to 10 scale. CVSS maps to threat levels: 0.1 - 3.9 = Low, 4.0 - 6.9 = Medium, 7.0 - 8.9 = High, 9.0 - 10.0 = Critical

This report was prepared using

## HostedScan Security ®

For more information, visit **hostedscan.com**

Founded in Seattle, Washington in 2019, HostedScan, LLC. is dedicated to making continuous vulnerability scanning and risk management much more easily accessible to more businesses.

HostedScan, LLC.

2212 Queen Anne Ave N
Suite #521
Seattle, WA 98109

Terms & Policies
hello@hostedscan.com