BITSIGHT®
KNOWLEDGE BASE

Web Application Headers

Publication Date – September 22, 2022

Table of Contents

Web Application Headers Risk Vector

The Web Application Headers risk vector, formerly known as "Application Security," analyzes security-related fields in the header section of communications between users and an application. They contain information about the messages, determine how to receive messages, and how recipients should respond to a message.

Much like a business letterhead, headers explain where the message is going and who it's from, date sent, what type of message it is, and other configuration options. They're included in all back-and-forth communications between applications. Web servers and web-connected applications must conform to a certain set of language (communication) standards when sending information over the Internet. These language definitions are called "protocols."

Web Application Headers cover security risks posed to an organization's application users through Hypertext Transfer Protocols (HTTP) headers. HTTP defines the way a website should respond when it can't find something, if it can find something, or something was temporarily moved. For example, the "404" page (page not found error) can be understood by your web browser thanks to the HTTP standard. Otherwise, web programmers might pick obscure numbers or other ways to tell you that a page is not found. Your browser will then have to guess.

Required headers are important for preventing communication attacks, between applications, from succeeding. Using proper Web Application Headers over the Internet ensure communications are robust against attacks that are designed to take advantage of ambiguity (communication details that are not explicitly defined). Review how to evaluate your hosts for inclusion in Web Application Headers.

- Rating Details
- Data Collection Methods
- Finding Details
- Finding Considerations

Risks
Correctly configured headers protects against malicious behavior, such as man-in-the-middle (MITM) and cross-site scripting (XSS) attacks, and prevents attackers from eavesdropping and capturing sensitive data, such as credentials, corporate email, and customer data.

Remediation
Refer to Web Application Header Finding Details.

- Findings that affect a company's Diligence grades have messages that provide an explanation and remediation. Refer to the Help and Remediation messages for additional details.
- Implement the required headers from the required headers and refer to the configuration requirements.
- Ensure application headers are created correctly and don't contain misspellings (typos).

Recommended Prioritization
HTTP/1.1, 1.0 and transport security (HTTPS or HTTP) are the common combinations of protocol. Starting with implementing the following configurations across your web application headers is recommended to improve "BAD" and "WARN" grades to "FAIR."

- **Content-Security-Policy (CSP):** A properly configured CSP can help prevent XSS attacks by restricting the origins of JavaScript, CSS, and other potentially dangerous resources. For more information about this directive, refer to the W3: Content Security Policy Level 2 and Mozilla: Using Content Security Policy documentation.
- **X-Content-Type-Options:** Setting X-Content-Type-Options to “nosniff” helps to prevent MIME or content sniffing.
- **Strict-Transport-Security:** Required for any HTTPS configuration. Enforces the use of HTTP over TLS/SSL. Properly using this header can help prevent MITM attacks. The Strict-Transport-Security header is defined in RFC-6797.

References
- IANA.org, “Message Headers”
- W3C, “Content Security Policy Level 2”
- Mozilla, “Content Security Policy (CSP)”

Web Application Header Findings

September 22, 2022: Added Assets (Asset, Calculated Importance, & View findings), Comments, Dates (First Seen & Last Seen), Finding Identifier, Finding Grade, Refresh (Refresh Status, Refresh Details, & Refresh Requested), In Remediations (Issue, Details, & Remediation Tip) fields; "Remediation Instructions" renamed to "Remediations."

The Web Application Headers risk vector contains information about the messages, determines how to receive messages, and determines how recipients should respond to a message.

View findings from the Findings [Risks ➔ Findings] page or the BitSight API [GET `/v1/companies/entity_guid/findings?risk_vector=application_security`].

This data is available for download (.csv) via the **Download** button. The download is supported for up to 9000 rows of findings. It includes columns that are currently configured in the table.

Finding Details

The details include the data in Findings, Diligence details, and also the following information:

❖ This field can be included in the table from the Customize Columns option.

† Including this field contains the following details: Cache-Control, Content-Security-Policy, Strict-Transport-Security, X-Content-Type-Options.

Field	Description
Assets	Asset details.
Asset	The asset name.
Calculated Importance	The BitSight-calculated asset importance.
View findings	Filter findings by the asset.
Cache-Control ❖ †	Indicates if the Cache-Control header is missing.
Comments	Finding comments for describing the status of resolution or validity of findings to external stakeholders and other interested parties.
Content-Security-Policy ❖ †	Indicates if the Content-Security-Policy header is missing.
Dates	Observation dates.
First Seen	The date when the finding was first observed.
Last Seen	The date when the finding was last observed.
Destination Port ❖	The destination port number identified in the finding.

Final Location ❖	URL where headers were observed.
Finding Identifier	An ID for the finding.
Finding Grade	The finding grade.
HTTP Headers	HTTP header details.
Last Seen IP:Port	The most recently observed IP:Port pair.
Observed IPs ❖	The IP address where the certificate was seen, on the most recent day.
Optional HTTP Header Fields	Optional HTTP header records and issues.
Refresh	Refresh details.
Refresh Status	The status of a user-requested refresh of a finding.
Refresh Details	Clarification on remaining issues, such as if the issue is still present or further developments.
Refresh Requested	The date when a refresh was requested.
Remediations ❖	Information for how to resolve a negative finding.
Issue	The finding name.
Details	A description of the finding.
Remediation Tip	Tips for remediating the finding.
Required HTTP Header Fields	Required HTTP header records and issues.
Strict-Transport-Security ❖ †	Indicates whether the Strict-Transport-Security header is missing.
URL	The URL of the web page.
X-Content-Type-Options ❖ †	Indicates if the X-Content-Type-Options header is missing.

Web Application Header Finding Considerations

To reduce the population of “noisy” Web Application Headers findings that are not valid risk indicators, the following criteria is applied:

Item	Criteria	Why?
Hostname	The host is part of the company's infrastructure.	Countless hosts, including subdomains (mail.google.com), are tallied by BitSight. This is to detect Web Application Headers across the vast BitSight inventory. The company's domains are used as the criteria for identifying hosts. Related findings are matched and assigned to a company. **Example:** If the company has a subset domain (e.g., saperix.com) of the host specified in the finding (www.saperix.com), the observation is recorded.
Port	Must be 80 or 443.	These ports are the most likely to host content that are of interest. Non-standard ports (8000, 8080, 8443, etc.) are often web management interfaces for software and hardware platforms.
Hardware Appliances	Must not be common hardware appliances, i.e., Cisco ASA, Sonicwall firewalls, etc.	These are the devices that general users cannot address and fix.
Content-Type	Must be “text/html.”	If the Content-Type is “application/json,” this typically means the host is a HTTP-based API. **Example:** An image is not a useful web application. The finding will be dropped.
Content-Length	Must be absent or greater than 0.	If present and is specified as “0” (no HTML returned), the finding is not generated. If the header is missing from the response, this check is skipped.

Response	Must end in “200.”	The data provider normally follows all 3xx-based redirects, but if the 3xx redirect causes the protocol to change (such as from HTTP to HTTPS or vice versa), the finding ends and a new finding is created for the new protocol. If the response ends in anything else, such as: ● 3xx redirect - The headers received from the web application are not necessarily the headers that a normal user would observe. ● 4xx or 5xx HTTP status code - Something went wrong during the request. The ultimate resource was not returned.
Redirects	Redirects to a different hostname (i.e., saperix.io) will result in a finding being created only for saperix.io, assuming saperix.io is part of the company’s infrastructure.	● Findings are based on the original hostname. When subdomains with different URLs are shortcuts to the same web application, the number of findings may be excessive. If a finding is based on the terminal hostname, it will have rapid fluctuations in finding grades. This is because a different page is evaluated each day. ● When a subdomain of a company redirects to a domain (and infrastructure) that belongs to a different company, the company that the finding should be assigned to is ambiguous.
Protocol Changes	Can only change from HTTP to HTTPS.	Redirects from HTTPS to HTTP is a misconfiguration and should be addressed.

How to Evaluate a Host for Web Application Headers Inclusion

Review the criteria for generating Web Application Header findings. To determine if a finding is created, use the following procedure to run a `curl -IL` command on a host:

1. Use your browser's developer tool to copy the cURL command to your clipboard. Refer to the following instructions for your browser:
   - Chrome
   - Safari
   - Firefox
   - Internet Explorer
   - Microsoft Edge
2. Go to the "Network" tab of the developer tool.
3. Right-click "www.bitsighttech.com" and then click **Copy ➔ Copy** as cURL.
4. Paste this into your Terminal and include "-v" at the end of your query.
5. Run the cURL command.

Example Response

Refer to the following response, located before the encrypted data.

```
HTTP/1.1 301 Moved Permanently
Location: https://www.google.com/
Content-Type: text/html; charset=UTF-8
Date: Tue, 28 Aug 2018 17:17:40 GMT
Expires: Thu, 27 Sep 2018 17:17:40 GMT
Cache-Control: public, max-age=2592000
Server: gws
Content-Length: 220
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alt-Svc: quic=":443"; ma=2592000; v="44,43,39,35"

HTTP/1.1 200 OK
Date: Tue, 28 Aug 2018 17:17:40 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2018-08-28-17; expires=Thu, 27-Sep-2018 17:17:40 GMT;
path=/; domain=.google.com
Set-Cookie:
NID=137=CQFYBttoqB9c2BYrozSME9mnGMSygLp6PKHEj2mj5vXWAEfWgdb5AsiPDumeyvo6sX5OBT
ULQOdT9drlusQQu-6KhGfqrGHgRfSbUpkRwCjxXkltT8Varb4m_rM9Nyba; expires=Wed,
27-Feb-2019 17:17:40 GMT; path=/; domain=.google.com; HttpOnly
Transfer-Encoding: chunked
```

```
Alt-Svc: quic=":443"; ma=2592000; v="44,43,39,35"
Accept-Ranges: none
Vary: Accept-Encoding
```

Additional Help

If a finding is missing:

- Verify the host against the criteria for generating Web Application Header findings.
- Ensure your block lists are not preventing lawful, periodic internet scanners from visiting your web properties.
- Check your firewall settings and ensure friendly scanners are able to connect.

If a header is missing:

- Ensure the domain is in your company's infrastructure.
- If the port has been closed, but 60 days hasn't elapsed for the finding to fall off your rating, select the refresh button to update the port.

If findings are not updating from your changes:

- Please allow 2-3 days for Web Application Header findings to update.
- If the changes do not meet the criteria, no findings were created.

Please contact BitSight Support for additional help.

How is the Web Application Headers Risk Vector Assessed?

A variety of HTTP headers are assessed to determine if security best practices are being followed. Only the HTTP headers of hosts that return HTTP 200 responses are assessed. Learn why HTTPS is preferred over HTTP:

- National Cyber Security Centre: Serve websites over HTTPS (always)
- Troy Hunt: Here's Why Your Static Website Needs HTTPS

Overview

Field	Description	Details & Values
Finding Behavior	How findings behave, depending on the action taken.	• New findings immediately impact the grade. • Remediated findings: ◦ The newest finding replaces the past finding and impacts the grade for 60 days, as it completes its lifetime. ◦ The previous finding is replaced and stops impacting the grade.
Lifetime	The number of days a finding will impact the risk vector grade, assuming nothing changes in the future and the finding is not updated with new information.	60 Days
No Findings	The letter grade if there are no findings for this risk vector.	C – “C” Letter Grade

		This is set in the center of the grading scale for computing into security ratings. Some findings cannot be traced back to specific companies due to the use of third party systems; such as web filters and Content Delivery Networks (CDN), that are capable of redirecting and encapsulating network traffic. Some firewalls might also be detecting and blocking external scanning tools from getting any data.
Refresh	The BitSight platform regularly checks for new observations. BitSight findings are updated as these observations change, e.g., newly observed Diligence findings or an existing finding was remediated.	
Automated Scan Duration	The duration of a regularly scheduled finding refresh, as the BitSight platform checks for new observations.	60 Days
User-Requested Refresh Duration	The duration of a user-requested refresh, which initiates a refresh of eligible findings upon request. This is recommended when a change in the finding is expected, such as when a finding has been remediated.	3 Business Days
Weight	Out of 40% in Diligence.	3%

Findings

Remediation Instructions

Web Application Header findings that affect a company's Diligence grades have messages that provide a brief description and remediation instructions (if any). They are specific to a field or value in an application header.

Finding Grading

Since Web Application Header findings are based on the entire header configuration and not on individual errors, finding grades can't be pre-assigned without evaluating the entire finding. See how Web Application Header findings are graded.

Content Checks

- Websites with mixed HTTP and HTTPS content.
- Intra-site URLs are evaluated for HTTPS protocol use.
- Redirects from HTTPS to HTTP.
- Check if the “WWW-Authenticate” is contained in an HTTP 401 response from non-HTTPS events.

Assessed Headers

- Access-Control-Allow-Origin
- Cache-Control
- Content-Security-Policy
- Expires
- HTTP Strict-Transport-Security

- Set-Cookie
- X-Content-Type-Options
- X-Frame-Options (Frame-Options)
- X-XSS-Protection

Required Headers

These are important for preventing attacks and are checked for usage and correct configurations. If an application header exists and the required header is not found in the findings, the company is penalized on missing headers. The penalties are described below under "Configuration Requirements."

Header	Required For
Cache-Control • Overview • Implementation	HTTP/1.1
Content-Security-Policy • Overview • Implementation	• HTTP/1.1 • HTTP/1.0
Expires • Overview • Implementation	HTTP/1.0
HTTP Strict-Transport-Security (HSTS) • Overview • Implementation	• HTTP/1.1 • HTTP/1.0
X-Content-Type-Options • Overview • Implementation	• HTTP/1.1 • HTTP/1.0
X-Frame-Options • Overview • Implementation	HTTP/1.0

Optional Headers

Optional headers may be present, in addition to required headers.

- If present, optional headers are verified that they are configured correctly and go towards the requirements as a whole for a GOOD or FAIR finding grade.
- If not present, companies are not penalized since they are unnecessary for preventing malicious actions.

Header	Optional For
Access-Control-Allow-Origin • Overview • Implementation	• HTTP/1.0 • HTTP/1.1
Location • Overview • Implementation	• HTTP/1.0 • HTTP/1.1

Set-Cookie • Overview • Implementation	• HTTP/1.0 • HTTP/1.1
WWW-Authenticate • Overview • Implementation	• HTTP/1.0 • HTTP/1.1
X-XSS-Protection • Overview • Implementation	• HTTP/1.0 • HTTP/1.1

Configuration Requirements

Requirements for **GOOD** grade: No misconfigured headers (required or optional) are present.

Requirements for **FAIR** grade: No more than 50% distinct misconfigured headers can be present (required and optional)

For HTTP connections, no headers are graded unless Set-Cookie is defined. The finding grade will default to NEUTRAL.

Required HTTP 1.1 (HTTPS):

- Content-Security-Policy
- HTTP Strict-Transport-Security
- X-Content-Type-Options
- Cache-Control

Required HTTP 1.1 (non-HTTPS):

- Content-Security-Policy
- X-Content-Type-Options
- Cache-Control
- Set-Cookie

Required HTTP 1.0 (HTTPS):

- Content-Security-Policy
- HTTP Strict-Transport-Security
- X-Content-Type-Options
- Expires
- X-Frame-Options

Required HTTP 1.0 (non-HTTPS):

- Content-Security-Policy
- X-Content-Type-Options
- Expires
- X-Frame-Options
- Set-Cookie

Responses

The following errors downgrade the response from HTTPS to HTTP:

- 200 responses

- 30X responses
- 401 responses

HTTP 1.1 (HTTPS)

Response	Description
200	We validate that no hyperlinks in the HTML for the web page downgrade the user inside the site and the domain of the site. We also validate and ensure the HTML of the webpage does not import resources (such as scripts and images) from outside the site using HTTP instead of HTTPS. The finding is graded BAD if these resources are present.
30x (301, 302, 307)	Any HTTPS finding that immediately downgrades the user to an HTTP connection using a redirect is graded as BAD.

HTTP 1.0 (HTTPS)

Response	Description
200	We validate that no hyperlinks in the HTML for the web page downgrade the user inside the site and the domain of the site. We also validate and ensure the HTML of the webpage does not import resources (such as scripts and images) from outside the site using HTTP instead of HTTPS. The finding is graded BAD if these resources are present.
30x (302, 307)	Any HTTPS finding that immediately downgrades the user to an HTTP connection using a redirect is graded as BAD.

Web Application Header Finding Grades

Web Application Header findings are graded as GOOD, FAIR, WARN, BAD, or NEUTRAL.

Methodology Overview

Scenario	Methodology
Content-Security-Policy is missing.	For just the headers, any missing Content-Security-Policy is penalized.
There's an HTTPS to HTTP redirect. It is changed to be an HTTPS to HTTPS redirect.	HTTPS to HTTPS redirects are graded as NEUTRAL. Since NEUTRAL Web Application Header findings do not impact the risk vector grade, they do not replace the HTTPS to HTTP redirect BAD finding grade.
There's an HTTP request in a redirect chain.	Once there has been any HTTP request in a redirect chain, the security of the chain is potentially compromised. For example, an attacker can intercept the HTTP request (man-in-the-middle) and then redirect the destination. Even having the final URL requested via HTTPS doesn't protect against this.[1] [2] [3]
A required header is not present.	The company is penalized on missing required headers, as described in the configuration requirements.
Optional headers are present.	Optional headers are verified that they are configured correctly and go towards the requirements as a whole for a GOOD or FAIR finding grade, as described in the configuration requirements.
Optional headers are not present.	Since optional headers are unnecessary for preventing malicious actions (as described in the configuration requirements), there's no penalty.
Set-Cookie header is not set.	HTTP findings are not graded unless the Set-Cookie header is set.
The presence of any HTTP links within an HTTPS page if upgrade-insecure-requests is present.	Any check for the presence of mixed content is skipped; there's no penalty on the presence of any HTTP links within an HTTPS page if upgrade-insecure-requests is present.

An HTTP Strict-Transport-Security (HSTS) header on an HTTP response.	An HTTP Strict-Transport-Security (HSTS) header on an HTTP response is ignored.
Redirects to a deep link of a hostname.	Redirects to a deep link of a hostname are graded on the contents of that particular page. Example: The following redirect is considered to be targeting the same hostname and to a deep link of that same hostname: example.com redirects to example.com/app/settings
Wildcard DNS	The presence of wildcards in DNS records can have an unnecessary magnification of the number of Web Application Header findings. These repeated findings are handled as a single finding.

Finding Weights

See the relative weights of Web Application Header findings:

Type	Weight
HTTPS to HTTP Redirect	Heavy
WWW-Authenticate (Error #401)	Medium
Mixed HTTP & HTTPS Content	Medium
HTTP Header	Light

Finding Grades

GOOD

Findings are graded as GOOD if HTTPS connections are present and Set-Cookie is secure.

BAD

- The presence of any HTTP links or references embedded in an HTTPS website. See content checks.
- Any request for credentials that uses the WWW-Authenticate header.
- We validate that no hyperlinks in the HTML for the web page downgrade the user inside the site and the domain of the site and ensure the HTML of the webpage does not import resources (such as scripts and images) from outside the site using HTTP instead of HTTPS. The finding is graded BAD if these resources are present.
- Any HTTPS finding that immediately downgrades the user to an HTTP connection using a redirect.

NEUTRAL

NEUTRAL Web Application Header findings do not negatively impact the risk vector grade.

- No headers are graded unless Set-Cookie is defined and the finding grade will default to NEUTRAL.

- For HTTP connections, no headers are graded unless Set-Cookie is defined. The finding grade will default to NEUTRAL.
- Redirects to the root of a different hostname are graded as NEUTRAL.
  Redirects to a deep link of a hostname are graded on the contents of that particular page.
- Remediated findings result in a NEUTRAL grade, which squashes the previous BAD finding.

Resources:

1. Owasp, "Testing for Sensitive Information Sent via Unencrypted Channels"
2. Owasp, "Testing for Credentials Transported over an Encrypted Channel"
3. EnableSecurity, "Surf Jacking - HTTPS will not save you"

Web Application Header Findings and Remediation

Web Application Header findings that affect a company's Diligence grades have messages that provide a brief description and remediation instructions (if any). They are specific to a field or value in an application header. It's displayed in the details view.

If the response is empty, ensure the header is not empty and values are set.

- Evaluated
- Not Evaluated
- Messages

Evaluated

Links are evaluated to determine if any link results in a downgrade of the recipient, from an HTTPS connection to an HTTP connection.

Message	Description	Remediation Instructions
HTTPS to HTTP link (intra-domain)	HTTPS webpage with internal HTTP link to same domain.	Avoid using HTTP links on HTTPS webpages.
HTTPS to HTTP link (intra-site)	HTTPS webpage with internal HTTP link to same site.	Avoid using HTTP links on HTTPS webpages.

Sites with a confirmed immediate redirect (using a redirect code 301, 302, 307) are graded as "NEUTRAL."

Resources are evaluated to check if any external dependency is used through HTTP (non-HTTPS) that might leave the application users at risk.

Message	Description	Remediation Instructions
HTTP external resource on HTTPS.	HTTPS webpage with an external HTTP resource.	Avoid using HTTP external resources on HTTPS webpages.

Not Evaluated

Findings with HTTP are not graded unless the Set-cookie header is set.

Messages

Message	Description	Remediation Instructions
Header is missing	This required header was not found.	Ensure your policy correctly implements the required headers. Refer to the required headers.
Ineffective headers:	The implementation of these header(s) do not follow security best practices.	Ensure your headers are implemented correctly, as outlined in RFC-7231. Your headers should not permit caching of encrypted content. They should also have specific permissions (as opposed to using wildcards or other generalizations) and be formatted properly.
Invalid character	This response contains invalid characters.	Responses may only include any ASCII character, except control characters, and allowed separator characters, as specified in RFC-2616 (section 4.2).
Invalid URL	The URL specified by this directive is not valid.	Ensure the URL is correctly formatted and is a valid and existing URL.
Missing directive	A required directive cannot be found.	Ensure your policy correctly implements the required headers. Refer to the required headers.
Missing required headers	One or more required security headers are not set.	Ensure your policy correctly implements the required headers. Refer to the required headers.
Missing URL	There is no URL specified by this directive.	Include a valid and existing URL. Ensure it is correctly formatted.
Must be a valid integer	This value must be a valid integer between -2^31 and 2^31 -1.	Ensure the value is a valid integer and does not contain any other characters, aside from numbers.
No security headers are set	None of the security headers are set.	Set your security headers. Refer to the Veracode: Guidelines for Setting Security Headers and the required headers.
No value set	A value is expected for this directive, but none are set.	Ensure that you have set a value for this directive.
Optional headers ineffective: [HTML_REMOVED]	The following number of headers are formatted in a way that makes them ineffective.	Format your headers correctly, as outlined in RFC-7230 (section 3.2).

Redirect	The page redirected to a different hostname or IP using a 301, 302, or 307 status code.	
Required headers ineffective: [HTML_REMOVED]	The following number of headers are formatted in a way that makes them ineffective.	Format your headers correctly, as outlined in RFC-7230 (section 3.2).
Required headers not set: [HTML_REMOVED]	The following number of required headers are not set.	Ensure your policy correctly implements the required headers. Refer to the required headers.
Value not allowed	No value is allowed for this directive.	Do not include a value along with this directive; it is directive-only.

Proper Access-Control-Allow-Origin Implementation

Optional for both HTTP/1.0 and HTTP/1.1

This field specifies which domains can share resources. Setting this field to an asterisk (*) allows any domain to use these resources. For more information about this directive, see the W3 Access Control for Cross-Site Requests.

Message	Description	Remediation Instructions
"null" must be lowercase	A "null" setting must be lowercase.	Use only lowercase letters for the "null" value.
Can only be used for first setting	A "*" or "null" can only be used for the first setting.	Ensure the first values in the setting are either "*" or "null" (lowercase only).
Duplicate entries	There are duplicate entries.	Remove duplicate entries and ensure the remaining values are formatted correctly.
Incompatible setting	This setting is incompatible with earlier settings in this header.	Ensure your settings do not conflict with each other, as specified in W3C Access Control for CSR.

Proper Cache-Control Implementation

Required for HTTP/1.1

This field sets conditions for storing data in the browser cache. The cache-control header is defined in RFC-7234. For cache-response verification, a no-cache directive is sufficient for a good response.

See the Google: Best Practice Guide for more information about this header.

Message	Description	Remediation Instructions
Cannot be negative	This directive must be set to a value greater than or equal to zero.	When specifying max-age or any other number in your Cache-Control header, it must be an integer greater than or equal to 0.
Directive used multiple times	This directive can only be used once.	Remove duplicate directives from your policy.
Insecure configuration	This cache-control configuration is insecure.	To satisfy this requirement, there are two choices. The first choice is to use the directive "max-age=N" with Cache-Control where N is an integer greater than or equal to zero. The second choice is to set the Expires header. For example, you can set Expires to “0,” or if desired, set to a negative value to disable caching. See RFC-7234 (sections 5.2 and 5.3) for more info.
Must be a quoted string	This value must be a string contained within double quotes.	Ensure the attribute begins and ends with double quotation marks (") and (") and does not contain smart typographer's quotes (“) and (”).
Must be an integer	The max-age value must be an integer.	Max-age must be an integer between -2 ^ 31 and 2 ^ 31 - 1 and cannot contain any other characters aside from numbers.
Public and private directives	Public and private directives cannot be set simultaneously.	Choose either the “public” or “private” value in your directive, but not both.
Satisfied by Expires header	Cache-Control is effectively implemented by the presence of the Expires header.	
Value is missing	The cache-control value is missing.	Implement Cache-Control correctly, according to RFC-7234.

Value not allowed	This cache-control value is not allowed for this directive.	Use correct values. See RFC-7234 for a comprehensive overview of cache-control.

What is Content-Security-Policy (CSP)?

Content-Security-Policy (CSP) directives are in-depth controls that can be used to protect against code injection. It requires a website to be designed or refactored with CSP in mind.

When used, their presence indicates a company has a good cyber security posture. A properly configured Content-Security-Policy (CSP) can help prevent cross-site scripting (XSS) attacks by restricting the origins of JavaScript, CSS, and other potentially dangerous resources.

The absence of CSP directives does not automatically make a website or service exploitable.

Required for:

- HTTP/1.1
- HTTP/1.0

Resources

- Assessed Directives
- Goals
- Implementation

Goals of Content-Security-Policy (CSP)

The deployment of Content-Security-Policy (CSP) helps organizations with the following goals, as defined in the Mozilla: Content-Security-Policy Level 3 (section 1.2):

- Mitigate the risk of content-injection attacks by giving developers fairly granular control over.
    - Resources can be requested (and subsequently embedded or executed) on behalf of a specific Document or Worker.
    - The execution of inline scripts.
    - Dynamic code execution (via `eval()` and similar constructs).
    - The application of inline style.
- Mitigate the risk of attacks that require a resource to be embedded in a malicious context (the "Pixel Perfect" attack described in `[TIMING]`, for example) by giving developers granular control over the origins that can embed a given resource.
- Provide a policy framework that allows developers to reduce the privilege of their applications.
- Provide a reporting mechanism that allows developers to detect flaws being exploited in the wild.

Goals

The grading of the deployment of CSP should try to assess the fulfillment of each of the following goals:

- Prevent Content Injection Attacks
- Prevent Resource Embedding Attacks
- Prevent the Reduction of Application Privileges
- Detect Attack Attempts via CSP Reporting

Prevent Content Injection Attacks

Effective implementation requires an application to be designed or refactored in a way that eliminates inline Javascript or adds specific controls to its execution via the hash or nonce values.

The following checks can be made:

- Presence of `unsafe-inline` or `unsafe-eval` in `script-src` and `unsafe-inline` in `style-src`. Policies using these values reflects a website that is not optimized to take advantage of CSP and does not protect the user from code-injection attacks.
- Absence of host specific `script-src`, `style-src` and `object-src?` directives. A CSP that does not specify at least one authorized host for its script and style directives does not protect against code injection attacks.

Prevent Resource Embedding Attacks

Websites should specify the origins that are allowed to embed resources in the website's document to prevent attacks that take advantage of embedding malicious resources.

The following checks can be made:

- Using the `frame-ancestors` directive to prevent the website resources from being loaded inside an iframe prevents resource embedding attacks such as the pixel perfect attack or clickjacking.

- For `X-Frame-Options`, verify the header is not set. They achieve the same objective.
- If `upgrade-insecure-requests` is present, there's no penalty on the presence of any HTTP links within an HTTPS page. Any check for the presence of mixed content is skipped.

Prevent the Reduction of Application Privileges

Reduction of application privileges follows the security principle of least privilege and directs that any application should not be given access to processes or files in excess of what the application needs, at minimum, to complete its tasks.

The following checks can be made:

- Assess the level of restriction of all the fetch directives: All the fetch directives either through the `default-src` or specifically defining each one of them should contain at least one host specific value, or none, and no wildcards.
- Limit the use of objects and plugins that can be used by the application (`plugin-types`).
- Limit the actions of the current document (form-action).

Detect Attack Attempts via CSP Reporting

One of the features of CSP is its ability to report violations of the policy to the website owner. These reports may represent simple errors in the application or in the policy but they may also represent real attack attempts against the application, such as XSS vulnerabilities being exploited or malicious software modifying user's browsers to target the application. This feature should be used and the reports monitored for events that may require action (e.g., correcting an XSS vulnerability). The following checks should be made:

- **Reporting policy violations:** Ensure the `report-to` header being used when reporting policy violations.
- **Check if the policy is report only:** If the `policy` is `report only`, it is not being enforced and consequently not protecting the application and its users directly. However, this demonstrates that violations are being monitored for response or for debugging of a future implementation of an enforcement policy. This effort is relevant and should be considered.
- **X-Frame-Options:** Prevents resource embedding attacks. Use only when `X-Frame-Options` is present and correctly set.
- **No reporting**: Use only when there is no place to log violations.
- **Content-Security-Policy record is report only:** For `report-only` records. If there is no regular CSP header, there should already be an annotation indicating that the required header is missing.

References

- Mozilla: Content-Security-Policy Level 3 (Section 1.2)
- WHATWG "DOM Living Standard" (Section 4.5)
- WHATWG "HTML Living Standard" (Section 10.2.6.3)
- ECMAScript "ECMA-262" (Section 18.2.1)
- WC3 "Content Security Policy Level 3" (Informative References Section)
- Mozilla "CSP: upgrade-insecure-requests"

What Content-Security-Policy (CSP) Directives are Assessed?

Content-Security-Policy (CSP) directives are in-depth controls, and the absence of those directives does not automatically make a website or service exploitable. However, when used, their presence indicates a company has a good cyber security posture.

Objectives

- **Application privilege limitation** - The CSP uses the form-action directive to limit application privilege.
- **Application privilege limitation** - The CSP uses the plugin-types or object-src directives to limit application privilege.
- **Code Injection Prevention** - This requires the website to be designed or refactored with CSP in mind. The user specifies explicit hosts for source-list directives and does not use the unsafe-inline or unsafe-eval keywords. This objective is not met if any of the source-list directives are improperly set or incomplete.
- **Reporting** - The CSP specifies a reporting location through a reporting directive. If either the report-to or report-uri directive is present, the reporting objective is considered to be satisfied.
- **Resource Embedding Prevention** - The CSP uses frame-ancestors (or X-Frame-Options) to prevent resource embedding attacks.

Directives

Directive	Description
`base-uri`	Defines a set of allowed URLs that can be used in the src attribute of an HTML base tag.
`connect-src`	Applies to XMLHttpRequest (AJAX), WebSocket, fetch(), `<a ping>` or EventSource. If not allowed, the browser emulates a 400 HTTP status code.
`default-src`	Defines the default policy for fetching resources, such as JavaScript, Images, CSS, Fonts, AJAX requests, Frames, and HTML5 Media. Not all directives fall back to `default-src`.
`font-src`	Defines valid sources of font resources (loaded via @font-face).
`form-action`	Restricts URLs that can be used as the target of form submissions. Satisfies the form-action-objective.
`frame-ancestors`	Prevents resource embedding attacks. Satisfies the resource-embedding-objective.

`frame-src`	Defines valid sources for loading frames. In CSP Level 2, this was deprecated in favor of the `child-src` directive. In CSP Level 3, it has been un-deprecated. If not present, it will continue to defer to `child-src`.
`img-src`	Defines valid sources of images.
`manifest-src`	Restricts the URLs that application manifests can be loaded.
`media-src`	Defines valid sources of audio and video, e.g., HTML5 `<audio>` and `<video>` elements.
`object-src`	Prevents fetching and executing plugin resources embedded using `<object>`, `<embed>`, or `<applet>` tags. The most common example is Flash. This directive satisfies the plugin-types-objective.
`plugin-types` *[deprecated]*	Restricts the set of plugins that can be embedded into a document by limiting the types of resources that can be loaded. Since this is deprecated, we recommend using `object-src: "none"` to address this.
`report-to`	Instructs the user agent to store reporting endpoints for an origin. Satisfies the reporting-objective.
`report-uri` *[deprecated]*	A pointer that locates where violations of CSP are logged. Satisfies the reporting-objective.
`script-src`	Defines valid sources of JavaScript.
`script-src-attr`	Specifies valid sources for JavaScript inline event handlers. This includes only inline script event handlers like onclick, but not URLs loaded directly into `<script>` elements.
`script-src-elem`	Specifies valid sources for JavaScript `<script>` elements, but not inline script event handlers, like onclick.
`style-src`	Defines valid sources of stylesheets or CSS.
`style-src-attr`	Specifies valid sources for inline styles applied to individual DOM elements.
`style-src-elem`	Specifies valid sources for stylesheets `<style>` and `<link>` elements with `rel="stylesheet"`.
`worker-src`	Restricts the URLs that may be loaded as a Worker, SharedWorker, or ServiceWorker.

Considerations

- CSP in HTML is considered only if:
  - The HTML head element is an ancestor.
  - The policy is inside a meta element, like so:

    ```
    http-equiv=="Content-Security-Policy"
    ```

> ⓘ This is case sensitive.

    - The element has a content attribute.
    - The directive is known.
- CSP defined in meta tags don't support the following directives:
    - `report-uri`
    - `frame-ancestors`
    - `sandbox`

Checks

- The syntax of CSP directives is checked for correctness of the data, duplicate directives, and if optional/known directives are added.
- Multiple headers are allowed.
- Either the `X-Frame-Options` or `frame-ancestors` directive should be present and correctly set.
- Reporting should be present and there should be a directive for logging errors.
- There should already be an annotation indicating that the required header is missing. This is indicated if the CSP record is report only, which is a type of CSP record that has the specific Content-Security-Policy-Report-Only header instead of Content-Security-Policy.

Proper Content-Security-Policy (CSP) Implementation

Required for HTTP/1.1 and HTTP/1.0

A properly configured Content-Security-Policy (CSP) can help prevent cross-site scripting (XSS) attacks by restricting the origins of JavaScript, CSS, and other potentially dangerous resources.

- See the W3: Content-Security-Policy Level 2 and Mozilla: Content-Security-Policy documentation for more information about this directive.
- Goals of CSP.
- The overall Web Application Headers letter grade is computed from the weight of all required directives and the optional directives that are not present. See the assessed CSP directives.
- We check for the presence of mixed content within headers; If `upgrade-insecure-requests` is present, there's no penalty on the presence of any HTTP links within an HTTPS page.

A good CSP should:

- Implement directives that set valid source restrictions from where the client can load frames and scripts as well as limit where the client can submit form data.
- Restrict plugins and specify a valid resource for reporting policy violations.
- Not contain "unsafe" keywords or include wildcards that are ineffective for restricting sources.

Example Policy

- Script source restrictions are inherited by the default-src value, which does not contain an "unsafe" keyword or a broad wildcard source.
- Frame embedding is restricted.
- Plugins are blocked.
- A form-action directive is in place.
- A valid reporting endpoint is provided.

```
default-src 'self'; object-src 'none'; form-action 'none'; report-uri
/example-reporting-endpoint;
```

Either X-Frame-Options should be good or it should include frame-ancestors.

Good X-Frame-Options:

```
X-Frame-Options: sameorigin
```

Includes frame-ancestors:

```
default-src 'self'; object-src 'none'; form-action 'none'; frame-ancestors
'self';  report-uri /example-reporting-endpoint;
```

Messages

Unsafe Keywords and Sources

The policy allows resources to be fetched from unsafe sources and can facilitate XSS by allowing code to be included directly in the document.

Message	Description	Remediation Instructions
"Unsafe-eval" is insecure	Unsafe-eval is vulnerable to XSS attacks.	Remove the "unsafe-eval" keyword from your CSP.
"Unsafe-inline" is insecure	Unsafe-inline is vulnerable to XSS attacks.	Remove the "unsafe-inline" keyword from your CSP.
Insecure redirect	This security policy allows a redirect that is not secure.	Remove any use of unsafe-redirect in your CSP.
"Blob" source is insecure	The "blob" source may allow the loading of unsafe resources.	Remove "blob:" from the source list in your CSP.
"Data" source is insecure	The "data" source may allow the loading of unsafe resources.	Remove "data:" from the source list in your CSP.
"Filesystem" source is insecure	The "filesystem" source may allow the loading of unsafe resources.	Remove "filesystem:" from the source list in your CSP.
Potentially insecure policy	This CSP has issues that possibly makes it insecure.	Remove any instances of "unsafe-" directives and "blob, data, filesystem" sources. Ensure your CSP directives are correctly configured. Learn more at W3C CSP.

Unspecified or Invalid Sources

The policy fails to specify directive sources or incorporates a wildcarded source.

Message	Description	Remediation Instructions
Asterisks in the source are insecure	This source allows potentially unsafe resources to be loaded from anywhere.	Remove any instances of the asterisk character (*) that are by itself from your CSP.
Source is too broad	This source is too broad to properly prevent attacks.	Use specific sources, such as https://www.example.com. Remove generalizations, such as http:, https:, https://*.com.
Conflicting source expressions	The "none" source expression, which represents a lack of URLs, is listed along with other URLs.	Choose either "none" or specify source URLs, but do not use both in your CSP.

Default-src inherited	This directive was not explicitly specified, so the default-src will be used instead.	Set specific policies for your directives. Otherwise, the default-src directive will be used instead.
Invalid host source	The host source is invalid or improperly formatted.	Ensure your host source is properly formatted in your CSP. Learn more at W3C CSP.
Invalid source expression	This source can not be included in a CSP.	Use only valid source expressions in your CSP. Learn more at W3C CSP.
Missing default-src	The default-src directive is not set.	Use the default-src directive in your CSP, as specified in W3C CSP.
Missing source list	There is no source list in this CSP.	Add components to the source list for your CSP. Learn more at W3C CSP.

Invalid Directives

The policy contains directives that are invalid or circumvent other directives. The policy is likely not operating as intended.

Message	Description	Remediation Instructions
Directive is not allowed	This is not a valid directive for this HTTP header.	Ensure your directives use approved expressions and they do not contain spelling errors.
Directive used multiple times	This directive can only be used once.	Remove duplicate directives from your policy.
Empty policy	This security policy has no sources, rendering it ineffective.	Make sure that your source-list is not empty and refers to complete URLs or IP addresses, as described in Policy Delivery: Content-Security-Policy Header Field (section 3.1).
Header overwritten	Another header has overwritten this CSP, rendering it invalid.	Check your CSP headers and ensure that your headers do not conflict with each other.
Header set more than once	This header cannot be set more than once.	Remove duplicate headers and duplicate header definitions from your HTTP headers.

Proper Expires Implementation

Required for HTTP/1.0

The Expires header is specified in RFC-7234 (section 5.3). The date format for the Expires header is defined in RFC-7231 (section 7.1.1.1). The date format strictly specifies GMT as the time zone.

Expired headers must either have a value that's an integer less than or equal to zero or contain a valid date in the format specified by RFC-7231. A date more than one year in the future will incur a slight penalty.

Message	Description	Remediation Instructions
Expires date is invalid	The Expires date is invalid or improperly formatted.	Ensure the field contains a valid date in the format specified by RFC-7231. For example, “Sun, 06 Nov 1994 08:49:37 GMT.”
Expires date is too far in the future	The Expires date should not be more than a year in the future.	Change the date to be no more than one year ahead of the current day.

Proper HTTP Strict-Transport-Security (HSTS) Implementation

Required for both HTTP/1.1 and HTTP/1.0

HTTP Strict-Transport-Security (HSTS) enforces the use of HTTP over TLS/SSL. Properly using this header can help prevent man-in-the-middle attacks (MITM). This header is defined in RFC-6797.

Requirements

- The `max-age` parameter is required.
- The `includeSubDomains` and `preload` parameters are optional. If present, we check to see if they are legitimate and that there are no associated values (i.e., syntax parsing).
- An HSTS header on an HTTP response is ignored.

Messages

Message	Description	Remediation Instructions
Directive used multiple times	This directive can only be used once.	Remove duplicate directives from your policy.
"includeSubDomains" is misspelled	The "includeSubDomains" phrase is misspelled.	Ensure the capitalization for "includeSubDomains" is exact.
Invalid max-age	This field is not an integer, is too long, or contains a syntax error.	The Max-age must be an integer between -2 ^ 31 and 2 ^ 31 - 1 and cannot contain any other characters aside from numbers.
Max-age is not set	Max-age is a required directive used to help prevent man-in-the-middle (MITM) attacks.	Use the max-age directive in your HTTP Strict-Transport-Security header, as specified in RFC-6797 (section 6.1.1). Ensure it contains only a positive number. To avoid the "max-age is too small" warning, set the max-age to at least 86400.
Max-age is too small	The Max-age should be set to at least 86400.	Change the max-age directive in your HTTP Strict-Transport-Security header to be greater than or equal to 86400.
"Preload" is misspelled	The word "preload" is misspelled.	Ensure "preload" is spelled correctly.

Proper Location Implementation

Optional for both HTTP/1.0 and HTTP/1.1

The Location header is used to redirect the visitor.

Message	Description	Remediation Instructions
HTTPS redirect to HTTP.	HTTPS URI is redirecting to HTTP URI.	Avoid downgrading user connections from secure to insecure.

Reference
RFC-2616 (Section 14.30)

Proper Set-Cookie Implementation

Optional for HTTPS and HTTP

`Set-Cookie` provides information for setting cookies. The `Set-Cookie` header is defined in RFC-6265.

- For HTTPS connections (if present) to be graded **GOOD**, `Set-Cookie` should be secure.
- For HTTP connections:
    - No headers are graded unless `Set-Cookie` is defined and the finding grade will default to **NEUTRAL**.
    - The secure directive is not required when `Set-Cookie` is present.
- The HttpOnly and Secure attributes for the following cookies are not evaluated since they are not user-configurable and do not contain sensitive information:
    - AWS ELB
    - Cloudflare __cfuid
    - AppDynamics ADRUM

If `Set-Cookie` is not properly implemented, you may get one the following finding details:

Message	Description	Remediation Instructions
Empty cookie value	The cookie value is empty.	The "cookie-value" field cannot be empty. Ensure it is a valid cookie ID, enclosed in double quotes, and contains only valid ASCII characters.
Invalid character	There is an invalid character in the cookie value.	Make sure that in your Set-Cookie header, the cookie-value attribute contains only US-ASCII characters (excluding CTLs), whitespace, commas, semicolons, and backslashes. See Set-Cookie syntax.
Invalid cookie pair	The cookie name-value pair is invalid.	Ensure the cookie-name, cookie-value, and cookie-pair attributes are used correctly and have correct values in your Set-Cookie header. See Set-Cookie syntax for additional details.
Invalid domain	This field does not contain a valid domain.	Ensure the domain-value attribute in your Set-Cookie header has a value that refers to an existing domain, is spelled correctly, and if it is an IP address, that it is complete.
Invalid expires value	The Expires date is invalid or is improperly formatted.	Ensure the field contains a valid date in the format specified by RFC-7231. For example, "Sun, 06 Nov 1994 08:49:37 GMT."
Invalid max-age	This field is not an integer, is too long, or contains a syntax error.	The Max-age must be an integer between -2 ^ 31 and 2 ^ 31 - 1 and cannot contain any other characters aside from numbers.

Invalid path	The path setting does not contain a valid path.	Ensure the path-av attribute has a value that refers to an actual and existing forward path (yourdomain.com/path).
No cookie pair	No cookie pair found.	Ensure the “cookie pair” attribute in your Set-Cookie header exists and is used in the following manner: cookie-pair = cookie-name "=" cookie-value. See Set-Cookie syntax for additional details.
No Set-Cookie	For HTTP connections, no headers are graded unless Set-Cookie is defined.	Please review all header requirements.
No Set-Cookie found	For HTTP connections, no headers are graded unless Set-Cookie is defined.	Please review all header requirements. Define your set-cookie header to be graded as “GOOD” and enable grading for all other headers.
Repeated ID	Two or more cookies are using the same ID.	Ensure the first “name = value;” pair in your Set-Cookie header is not using a duplicated setting.
Secure is not set	The secure directive is not set.	Ensure the secure value in your Set-Cookie header is being used in your directive.

Proper WWW-Authenticate Implementation

Optional for both HTTP/1.0 and HTTP/1.1

The WWW-Authenticate header indicates the authentication scheme.

Responses

Response	Description	Header
401	If the header isn't used, there will be no record to assess. We grade any request for credentials that uses the WWW-Authenticate header as “BAD.”	● HTTP 1.0 (non-HTTPS) ● HTTP 1.1 (non-HTTPS)

Messages

Message	Description	Remediation
Authentication over HTTP.	Requiring authentication over HTTP.	Only use auth forms on HTTPS resources.

Reference

RFC-7235 (Section 4.1)

Proper X-Content-Type-Options Implementation

Required for both HTTP/1.1 and HTTP/1.0

Multipurpose Internet Mail Extensions (MIME) sniffing (also known as “content sniffing”) can occur when a website allows users to upload data to the server. This is important if a user uploads an HTML page when the web server expects a different content type. When the web server sends the data back to a browser, the browser may interpret it as a web page even though the web server intended it to be (say) a CSV.

See the Fetch Living Standard for details on the processing performed by browsers.

Recommendations:

- Concatenate all the X-Content-Type-Options headers together.
- Set the X-Content-Type-Options header to “nosniff.”

Message	Description	Remediation Instructions
Must be “nosniff”	The first field should contain a “nosniff” value.	Set the value for X-Content-Type-Options to be “nosniff.”

Proper X-Frame-Options (Frame-Options) Implementation

Required for HTTP/1.0

Properly setting X-Frame-Options helps prevent clickjacking attacks by not allowing the browser to render this page in a frame. The X-Frame-Options header is defined in RFC-7034. The only valid options for this header are `DENY` and `SAMEORIGIN`. Though `ALLOW-FROM` is ignored by modern browsers, it does not currently negatively impact the Web Application Headers grade.

Message	Description	Remediation Instructions
Too many directives	Browsers only support one X-Frame-Options header and one value within that header.	Ensure the X-Frame-Options header contains either only the DENY or the SAMEORIGIN option.

Proper X-XSS-Protection Implementation

Optional for both HTTP/1.0 and HTTP/1.1

Setting X-XSS-Protection to “FIELD” helps to prevent common cross-site scripting attacks by filtering and blocking suspected malicious scripts. For the first directive, “0” disables XSS protection on the client side and “1” enables XSS protection. “mode = block” prevents the browser from loading pages potentially compromised by XSS. The report directive can either be a path or a URL.

For more information about this directive, see:

- OWASP, “Cross Site Scripting Prevention Cheat Sheet”
- Veracode Guidelines for Setting Security Headers

Message	Description	Remediation
Incompatible setting	This setting is incompatible with earlier settings in this header.	Ensure that your settings do not conflict with each other, as specified in W3C Access Control for CSR.
Must be “block”	The mode must be set to “block,” as it is the only accepted value.	If X-XSS-Protection is enabled, the mode must be set to “block” and cannot be set to anything else.
Must be 0 or 1	The first directive must be either “0 or “1,” as these are the only values that enable or disable the header.	The first directive in the X-XSS-Protection header must be 0 or 1, cannot be any other number or contain text.
Report must be last	The Report directive must come last, otherwise a client might ignore this value.	Ensure your X-XSS-Protection header directives are ordered correctly and that “Report” is the last directive. Read more at Guidelines for Setting Security Headers.

Data Collection Methods Overview

To ensure we provide the most relevant and comprehensive ratings on cyber security performance; we are committed to continuously expanding the data quality, breadth and innovation used in Security Ratings. We will continue to add breadth to our data sources and risk vectors to continuously expand the visibility of a company's performance.

We do this by owning proprietary data streams and working closely with partners around the globe to ensure access to multiple and diverse data feeds. We do rigorous analysis on the quality, origin, and confidence of all collected data. Because of the breadth, we can cross-correlate and improve confidence based on multiple observation points and methods.

In addition, we can provide historical data going back 1 year, giving organizations a long-term view of security performance across the enterprise.

Collection Methods

Our data sources are carefully curated for their detective breadth and technical reliability due to the daily volume of processed data. Once the legitimacy of a source is qualified, our inclusion criteria is straightforward and the event is included in your organization's rating.

Method	Description
Sinkhole	This is a technique that intercepts botnet traffic on its way to a command and control server (C&C or C2 server). By intercepting the botnet traffic, the sinkhole can get infection information and details on its origin.
Spam Trap	A threat intelligence group planted email addresses, which no one should ever use or know about, to trap Spammers. These spam traps are placed where Spammers look for address lists. The collected address lists receive malicious messages. These events are attributed to your company's rating based on the sender IP/domain information contained in the message.
Email Header Analysis	Analysis has been done on a message or set of messages that hit a spam trap. The header is determined to have shared properties with similar messages that indicate malicious intent. Some spambots are detected and characterized via email header analysis alone, which is not evidence of an outreach to a C&C server, but they do fit specific patterns that allow us to identify them with specific names (i.e. Lethic).

Mail Server Connection Analysis	Spam type: Impossible HELO An Impossible HELO is a characteristic of a spambot detected during the establishment of a session with an external mail server. Upon the completion of a Transmission Control Protocol (TCP) handshake, a legitimate email announces itself as coming from a valid domain. It will proceed to send email from the valid domain. For Impossible HELOs, the handshake is completed, but the email client announced itself as coming from an external mail server domain that's known to be "impossible." **Example:** These events are detected by a network of sensors that are deployed across the Internet through thousands of Mail Transfer Agents (MTA). These sensors are programmed with the known HELO strings of major mail providers and messages that are sent from infected clients with HELO strings that are known to be impossible. This is a simple check that is virtually impossible to fail if mail servers are configured correctly.