How can the internet presentation of the lint list for the rust-clippy project be improved?

Fridtjof Stoldt

May 4, 2021

Contents

List of Figures

List of Tables

Attachments

1 Introduction

Rust is a programming language that focuses on performance, security a reliability. The compiler is open source and dual-licensed under the Apache 2.0 and MIT license (Hoare and et al. 2019). Rust 1.0 the first stable version was announced in May 2015 (The Rust Core Team 2015). This release also marked the start of the *commitment to stability* which promises stability on future Rust stable releases (Turon and Matsakis 2014). This new commitment also introduced a 6-week release cycle as well as development channels for language users and early adapters (Turon and Matsakis 2014). The latest stable compiler version 1.51.0 has been released on 25 of March 2021 (The Rust Core Team 2021). Developers and teams within the project put high effort into open communication. This focussed is formalized in the official *Code of conduct* (The Rust Team 2021b). The language with its connected tools has attracted over 5900 individual contributes as of writing this (The Rust Team 2021a).

The Rust project consists out of several tools besides the compiler itself. These tools are seen as a vital part in automating parts of the development process and collaboration among teams. *Clippy* is the official linter for Rust and is being developed in the *rust-clippy* repository. The linter contains over 450 lints which span from complexity and style lints over to restriction lints which might be required by certificates (The Rust Clippy Developers 2021). Clippy is written in Rust itself and interfaces with the compiler directly. This direct connection enables the use of the existing lexer, parser and connected diagnostic tools and ensures that the project stays up to date with the latest compiler changes. Since 2018 Clippy is distributed as a component of the Rust installation itself (Lusby 2018).

1.1 Problem

Clippy maintains a website that contains documentation about all implemented lints. This list has the title *ALL the Clippy Lints* and will be referred to as *Clippy's lint list* or simply *lint list* in this paper. Diagnostic messages of the tool provide a suggestion and usually a small explanation with a reference to the website for a detailed lint documentation with examples. This makes Clippy's lint list the second point of contact for new users with the project itself. The lint list is also the only internet presentation of Clippy besides the GitHub repository inside the Rust organization.

Offering an online documentation gives a central point of reference that can be linked to and used in discussions. However, it also brings some responsibility when it comes to securely and functionality. The *Rust Infrastructure Team*, a team inside the project with members that organize and manage the entire infrastructure, has therefor defined some guidelines for static websites (The Rust Infrastructure Team 2020). Clippy's lint list is static and should therefor follow these rules. A small review of these requirements has shown that not all of them might be fulfilled when it comes to security. Not having them fulfilled might give of a bad impression for new users and reduce the search engine rating.

A secondary problem is the initial load time of the lint list which is noticeably slower than most other websites in the Rust ecosystem. This aspect also influences the user experience and search engine rating. However, this will not be evaluated as part of this paper due to the fact that there has been some recent discussions on the topic inside the community to change the display of content completely which would void all research on this topic.

1.2 Research question

The described problem in 1.1 leads to the following research question: *Does Clippy's lint list fulfill all requirements and if not how can this be improved?*

1.3 Goal

The primary goal of this paper is to review which requirements are currently not met and possibly find a solution to fulfill them. These solutions should ideally be simple to implement in the form of a pull request in the GitHub repository or as a suggestion how to change the settings of the hosting provider.

1.4 Approach

The start of this paper will provide some context about the Clippy's lint list and the current hosting provider. It will then collect the requirements defined for that static websites, like Clippy's lint list, inside the Rust ecosystem.

The next chapter will then measure the current fulfillment of these collected requirements to deduct which topics should be further investigated. The following section will analyze the measurements and explain the technical importance behind them as well as evaluate the importance for Clippy.

Based in this work the author will try to find or develop solutions for unfulfilled requirements. This section might include some practical test to see if certain changes have the desired effects.

The assignment will conclude with a summary of the investigated topics and suggestions for further work that can be done on the topic.

2 Clippy's lint list

Chapter 1 gives an introduction of Clippy and the lint list which is being maintained by the contributors of the rust-clippy project. This section will provide relevant background information about the website and the hosting provider. It will then summarize the relevant requirements and conclude with an overview which aspects will be further investigated.

2.1 Hosting

Clippy's lint list is a static website centered around an HTML file which displays a JSON document with lint documentation and metadata. It additionally references resources by other projects, but these two are the only once that are directly hosted as part of the project. The website is automatically updated and deployed with every merged pull request.

The rust-clippy project has selected *GitHub Pages* as a hosting provider. GitHub Pages provides a simple way to host project websites directly from the repository itself. GitHub additionally provides a project domain which is made up of the name of the organization or username and a path to the project (GitHub Docs 2021a). For Clippy this domain is `https://rust-lang.github.io/rust-clippy/`. The use of this hosting adds no additional cost if the user or organization has a paid product plan, like *GitHub Pro* or *GitHub Team* (GitHub Docs 2021b). The latter applies to the Rust Organization. GitHub Pages has soft limits when it comes to bandwidth usage, page site and amount of page updates per hour. The documentation also states that the hosting should not be used directly for commercial purposes or sensitive and personal data (GitHub Docs 2021a).

2.2 Requirements

The research question specified in 6 focuses on requirements that are put on the lint list as a static website that is provided as a part of the Rust community. This part of the paper outlines these requirements.

2.2.1 Website functionality

The website has *functional requirements* which describe direct functionality and behavior that the website should provide (Sommerville 2010, p. 83ff). An example is the implemented search and filter feature. These requirements will not be listed as part of this work as they are not necessarily needed to fulfil the technical requirements defined in the Rust development documentation. However, it is noteworthy that this functionality should not be impacted by suggestions in this paper. This paragraph will be referenced again if a suggested solution could impact them. A list of requested and implemented functionality can be retrieved from the rust-clippy issue tracker.

2.2.2 Requirements by the Rust Infrastructure Team

The Rust Infrastructure team has created a set of guidelines that static websites affiliated with the Rust project should fulfill to be hosted and managed by them. Clippy is an official Rust project and the website itself presents static content, the guidelines therefor apply to the website. The requirements are as follows (The Rust Infrastructure Team 2020):

- "The website must be managed by a Rust team, or be officially affiliated with the project."
  - This point excludes community projects due to finite resources of the infrastructure team.
- "The website's content and build tooling must be hosted on a GitHub repository in either the rust-lang[1] or rust-lang-nursery[2] organizations."
  - The teams wants to be able to rebuild the website at any time. They therefor require it to be hosted in a GitHub repository that is also managed by them.
- "The website must be built and deployed with a CI service."
- "The website must reach an A+ grade on the Mozilla Observatory[3]."
  - This requirement focuses on user security as it ensures that multiple security features are enabled for the website. The referenced tool analyzes security features than can toggled through header fields in the HTTP response by the hosting provider. The target grade indicates that the website is configured correctly.
- "The website must be hosted on platforms vetted by the infra team."
  - The documentation recommends the usage of *GitHub Pages* or *Amazon AWS* in combination with *CloudFront* as a *content delivery network*. Other providers can be suggested and requested as long as they are deemed to be secure and reliable.

[1] <https://github.com/rust-lang>

[2] <https://github.com/rust-lang-nursery>

[3] <https://observatory.mozilla.org/>

3 Fulfillment of requirements

The previous chapter has summarized the requirements that are put on Clippy's lint list and provided additional information and reasoning behind them. This section will investigate to which extent these are currently fulfilled. The goal is to identify key areas that could be improved.

3.1 Repository requirements

This first part evaluates the project and repository related requirements for the rust-clippy project. Clippy is as mentioned in 1 the official linter for the Rust language and deployed as part of the Rust installation. This makes Clippy an affiliated project to the Rust organization. The project repository is a part of the rust-lang organization an GitHub and therefor also satisfies the requirement of being managed by the Rust Infrastructure Team. Clippy therefor meets the project related requirements.

3.2 Hosting requirements

This section assesses the requirements connected to website hosting for Clippy. The project uses *continues integration* to deploy the lint list with every merged pull request (The Rust Clippy Developers 2020). This ensures that the documentation is always up to date with current development. The content is hosted on GitHub Pages, this is a vetted and even suggested website host by the Rust Infrastructure Team. Clippy's lint list therefor fulfills all requirements in relation to deployment and content hosting.

3.3 Mozilla Observatory rating

Mozilla Observatory is a collection of tools that can analyze a website to determine which available security measures have been utilized by it (King and et al. 2018b). The scan is focussed on opt-in security options that are set in the HTTP response header, these will then instruct the client to enforce them (The Rust Infrastructure Team 2020). The Rust development documentation links to a free online interface[4] for the Mozilla Observatory that is provided by the Mozilla Foundation free of charge. The requirements in 2.2 state that a website should archive the grade A+.

3.3.1 Scoring

The result of the analyses is summarized in a single score with a corresponding grade. The score is calculated using a baseline. Each checked criteria can add bonus points or subtracted a penalty. This implementation is used to give different weight to specific configurations. The significance of these modifiers are based on how important the analyzed aspect for security. Scores can range from a minimum of 0 to a maximum of 135, the score of 100 already indicates that the website is configured correctly a higher score can be archived by gaining bonus points. A score of 100 and above corresponds to the grade *A+* (King and et al. 2018a).

The observatory documentation notes that all websites are graded equally, this means certain graded configurations might be unimportant for the specific use case (King and et al. 2020).

[4] <https://observatory.mozilla.org/>

3.3.2 Measurement

Scanning Clippy's lint list results in an overall grade of *C* with a score of 55/100. It is to note that the analysis cut of the path to the lint list and graded the domain itself. The results are therefor for the URL `rust-lang.github.io` in general. The score was calculated using the baseline of 100 points and subtracting a penalty of 45 points. This sanction is the result of three failed tests that are shown in table 1.

No.	Score	Reason
1.	-20	HTTP Strict Transport Security (HSTS) header not implemented
2.	-20	X-Frame-Options (XFO) header not implemented
3.	-5	X-Content-Type-Options header not implemented

Table 1: Mozilla Observatory analysis penalties for *rust-lang.github.io* from 2021-04-24

The original scan output with all test results is included in attachment number 1.

3.4 Summary

The requirement evaluation has shown that Clippy's lint list fulfills all requirements except for archiving the A+ grade by the Mozilla Observatory rating engine. The actual grade is a C which is a result of three missing entries in the HTTP response header. The missing values are displayed in table 1.

4 Analysis of missing response headers

In 3 it was determined that Clippy's lint list currently misses three HTTP response header fields to fulfill all requirements that have been defined in 2.2. This chapter will inspect each of these fields individually by explaining the technical background, evaluating the relevance for Clippy's use case and then suggest what each option should optimally be set to.

The observatory scan focuses on HTTP header which are set by the server behind the domain. The scan was therefor conducted for the domain `rust-lang.github.io`. Clippy's lint list is indirectly included in this result as well as documentation from other repositories inside the rust-lang organization. Further investigation will continue to focus on the context of Clippy's lint list however changes to the server could therefor also indirectly improve other sites.

4.1 HTTP Strict-Transport-Security

`Strict-Transport-Security` is an optional HTTP header field that instructs the client to only use an encrypted connection for further requests. The instruction extends to all resources that are referenced by the requested result. It is therefor necessary that these resources hosts provide the option to download their resources over HTTPS (Hodges and et al. 2012, p. 6ff).

This header protects the user from *passive network attacks* where an attacker eavesdrop on the exchanged data. This can be used to collect personal information, passwords or browsing habits. A connection that is not encrypted is also vulnerable to *active network attack*. With this an attacker can impersonate the actual site or deliver a modified version all together. An encrypted connection on the other hand can be used to request a certificate and validate that the content is delivered from the expected source.

An additional advantage of this header is that it prevents accidental use of unencrypted connections by developers (Hodges and et al. 2012, p. 6ff).

4.1.1 Importance for Clippy

Clippy's lint list only displays publicly available information about lints in an easy accessible and searchable way. A passive network attack could therefor not collect any secret or personal information about the user. Except the fact that they visited the domain at all. However, this would still be possible with the header as the connected IP is not effected by it. The biggest thread could actually be an active network attack that injects a donation button into the website as several developers have expressed interest to donate to the Rust Foundation in general. This button would then forward the user to another page of the attacker to donate. However, the chance of this is probably negotiable due to the low traffic that Clippy's lint list actually receives. Such an attack would therefor be targeted towards a specific user.

With all of this being said it has to be noted that all references to the website already include `https` at the start and a user has to deliberately enter the domain with http in front. Most browsers will then still recommend using the encrypted connection or at least add a *not encrypted* notice next to the URL. All of this results in a very low risk. The header should still be set if the hosting provider provides a simple setting for this. Also, due to the fact that the targeted A+ rating would require this field.

4.1.2 Configuration

The header can take up to three arguments that configure which domains are included in this instruction and a duration for how long an encrypted connection should be forced (Hodges and et al. 2012, p. 14ff). Both Mozilla and the Rust development documentation recommends defining a duration of two years in the header field. This is equivalent to the value `"max-age=63072000"` (King 2018, citerust-forge.static-websites). This is therefor also the recommended value for Clippy's lint list.

4.2 X-Frame-Options

The `X-Frame-Options` header was initially accepted by some browsers as an opt-in security measure to prevent clickjacking. In 2013 the header was formally specified by the *Internet Engineering Task Force* (*IETF*) in RFC7014 (Ross and Gondrom 2013, p. 3).

HTML supports frame elements which allow a website to embed an external website into the users view. This can be used to add a complimentary view that is externally hosted or provides additional information. The parent document can for security reasons not directly interact with the framed contend. Clickhijacking describes an attack where a frame is used to make a user unknowingly interact with framed content trough clicks as these will be accepted by the frame as interactions. This interaction can then be used to trigger some behavior or gain access in some other way. Browsers have added support for the `X-Frame-Options` header which enables content provider to deny the display of content in frames (Ross and Gondrom 2013, p. 3ff). Therefor preventing clickhijacking all together.

4.2.1 Importance for Clippy

Clickhijacking is used to make a victim interacts with a different website to use the privileges or data that the user has saved on that site. Clippy's lint list provides the same data to everyone and the only user specific data is the selected color theme. An attacker has therefor nothing to gain with this attack. Adding the header would actually reduce flexibility from external users to embed the lint list in their own interface, even if the project at this point does not know of a website doing so.

However, Clippy's lint list is just one site that's hosted under the domain, it should be investigated if other sites contain sensitive data that would require the header. This paper will still look into setting the header as it is required so receive the grade A+ by Mozilla Observatory.

4.2.2 Configuration

The option can be set to three mutually exclusive values (Ross and Gondrom 2013, p. 4):

- *DENY*: Indicates that the content should not be displayed in any frame.
- *SAMEORIGIN*: Allows the display of the content inside a frame as long as it originated from the same origin as the frame.
- *ALLOW-FROM*: This prohibits the display of the content except for the origins that are defined after the "ALLOW-FROM" value.

The Rust development documentation provides an example configuration that uses `DENY` (The Rust Infrastructure Team 2020). `DENY` is the most restrictive setting but can easily be adapted to allow framing if requested. The author for this reason suggest the initial value of `DENY` as well.

4.3 X-Content-Type-Options

In 2008 the `X-Content-Type-Options` HTTP header was initially implemented by Microsoft in Internet Explorer 8 to prevent attacks that abuse *MIME-sniffing* (Lawrence 2008b). HTTP response header include a `content-type` field that indicate the type of content that is being delivered, these types are called *MIME types*. Most browsers have a mechanic called *MIME-sniffing* to determine what MIME type the received resource is in. This functionality is used for backwards compatibility with for legacy servers that serve all content with the `text/plain` content type. MIME-Sniffing can determine that received data is in a different data type than specified and display it in the newly determined way. This would for instance render an HTML document that is delivered with the `text/plain` content type if the text contains HTML elements (Lawrence 2008a).

The feature has however introduced some security concerns for content hosts. Attackers could create content, like images, that contain HTML text with scrips. The sniffing functionality could then falsely determine during the inspection that the received resource is an HTML document and execute the contained script instead of showing an image (Lawrence 2008a). This led to the introduction of the `X-Content-Type-Options` field that can be used to disable sniffing and therefor enforce the use of the specified content type (Lawrence 2008b).

4.3.1 Importance for Clippy's lint list

This field can actually be of high importance to the project. Clippy, like all Rust projects, has a review policy that only allows the merge of changes if they have been reviewed by a project member. This type of attack especially focuses on hiding the malicious code inside other resources, like an image. This could therefor also easily be overlooked during the review process. Additionally due to the fact that the project maintainers mainly focus on Rust and not the website.

This header requires that the `content-type` header is set correctly for content that is being delivered by the host. GitHub Pages doesn't support the manual specification of the content type. It instead uses an open source database to determine the correct MIME type based on the file extension (GitHub Docs 2021a). Clippy's lint list is composed out of a `.html` and a `.json` file which both are delivered with the correct content type as can be seen in attachment 2 and 3. The security measure option can therefor be enabled without side effects.

4.3.2 Configuration

The `X-Content-Type-Options` response header can only be set to `nosniff` which disables the sniffing feature altogether. This option is also supported by all major browsers (Bengtsson and et al. 2021). This will therefor also be the suggested value to the field.

4.4 Summary

This chapter reviewed the three missing header fields that are required to gain the grade A+ by Mozilla Observatory. It was chosen a suggested value for each filed that will be used for further reference in this paper. These values were chosen based on the technical background and importance for Clippy. The results are summarizes in table 2.

HTTP header field	Value	Reference
`Strict-Transport-Security`	`max-age=63072000`	See 4.1.2
`X-Frame-Options`	`DENY`	See 4.2.2
`X-Content-Type-Options`	`nosniff`	See 4.3.2

Table 2: Determined values for the investigated HTTP header fields

5 Setting HTTP header fields

Chapter 3 has determined that three HTTP response headers would need to be set to fulfill all requirements from 2.2. Section 4 has analyzed each field and suggested a value for each of them. The results are displayed in table 2. This chapter will now investigate how these values can be set for Clippy's lint list.

5.1 GitHub Pages configuration

The GitHub Pages documentation does not contain any information if and how these HTTP headers can be set. There has been requests to support user defined HTTP headers in several places by the GitHub community. All of them have concluded that this is currently not possible (trante and et al. 2013, Laukenstein and Balter 2017, yawnoc and et al. 2021).

Searching in the documentation for the header functionality reveals that GitHub Pages provides an option called *Enforce HTTPS*. This option can be enabled for each hosted site, under the condition that the original `github.io` domain is used (GitHub Docs 2021c). Putting this setting to the test under a personal fork of the rust-clippy project reveals that the effect is limited. Requesting the project domain over HTTP results in a *301 Moved Permanently* responds that forwards the browser to the same domain using HTTPS. However, the `Strict-Transport-Security` header which could enforce this behavior by the client is not set. The response for the test page is included in attachment 4. This forward message only works for the root project URL, other resources and direct HTML pages can still be loaded without an encrypted connection. Clippy uses paths to display version specific documentation. This setting is therefor not helpful in enforcing HTTPS security for Clippy' lint list.

The GitHub Pages documentation currently does not contain any information regarding the other header options.

5.2 HTML meta tag

A discussion on the topic of setting HTTP header fields in GitHub Pages included suggestions to use a meta tag inside the main HTML file header (trante and et al. 2013). The meta tag is part of the living HTML standard defined by the *Web Hypertext Application Technology Working Group* (*WHATWG*). It can be used to add supplementary information for the client. The tag can contain a `http-equiv` attribute with a linked `content` attribute that can define values that would usually be set in the HTTP response header. The standard currently defines a set of fields that can be set with the meta attribute. The missing fields defined in table 2 are not listed in the living standard (WHATWG 2021). However, clients can still deviate from this standard or support additional functionality that has not yet been specified.

Putting the meta tag to the test reveals that both Firefox and Chromium accept values for `Strict-Transport-Securit` and `X-Content-Type-Options`. Assigning a value to `X-Frame-Options` produces a warning in the Chromium console with the message that this option is not supported in the meta tag and should be set as an HTTP response header. After setting the meta tags both browsers take care to enforce HTTPS connections for all requested resources. Accessing an HTTP connection produces in both cases an error message indicating that mixed content is not allowed and the request has been blocked.

The meta tag can therefor be used to define `Strict-Transport-Security` and `X-Content-Type-Options` fields for individual websites and with that increase security. The `max-time` information defined in the `Strict-Transport-Security` field also ensures that future accesses to the website will use HTTPS. This solution still has four drawbacks:

1. The header to enforce HTTPS is only set during the loading of the page. An attacker could therefor still modify the page and remove the tag if they catch the initial request where HTTPS is not yet enforced.
2. The meta tag to set these headers is not yet fully specified and can still change. The fact that Chromium and Firefox both accept these headers is an additional functionality.
3. The `X-Frame-Options` header can not be set via a meta tag. This header is as discussed in 4.2.1 the least important for now but still relevant when it comes to the Mozilla Observatory rating.
4. These meta tags are defined in HTML files, it will therefor not increase the Mozilla Observatory scoring, and they have to be added to each project in each HTML file to ensure that they are enforces in the project.

5.3 Content Delivery Network

The Rust Infrastructure Team has noticed that some hosting providers have limitations when it comes to available configuration. The team has therefor set up a *CloudFront* account for rust projects (The Rust Infrastructure Team 2020). CloudFront is a *content delivery network* (*CDN*) provided by Amazon. It can be used to deliver content on a global scale by acting as an intermediary agent. Each content request is wired over the network, the network then saves the data in caches to speedup future access (Amazon Web Services 2021). Using a content delivery network enables the definition of custom behavior. This can be used to define additional HTTP headers (The Rust Infrastructure Team 2020).

CloudFront is already used to provide the website for the Rust project *rustup* at rustup.rs[5]. That website archives the highest grade of A+ when evaluated with Mozilla Observatory. The rating is included in attachment number 5. However, the distribution over CloudFront requires the use of a domain as the direct access to GitHub Pages can not be intercepted via a CDN.

Using CloudFront, a different hosting or content provider would, as seen in the rustup website example, improve the website rating to a possible grade of A+ if configured correctly. A valid configuration is also provided in the Rust development documentation (The Rust Infrastructure Team 2020). This is therefor a valid solution. However, using an additional new service like CloudFront would also add some additional complexity and use up resources of the Rust project in general. These are two disadvantages that have to be put into consideration when deciding for or against the usage. Additionally due to the fact that GitHub Pages is still a recommended content host even with its shortcomings (The Rust Infrastructure Team 2020).

6 Conclusion

This paper succeeded in reviewing the research question defined in . It determined that Clippy's lint list currently does not fulfill all requirements. The paper was then able to find a possible solution for these missing aspects.

6.1 Summary

Section 1 introduces the rust-clippy project and the problem that same requirements put on the website might not be met currently. This led to the following research question: *Does Clippy's lint list fulfill all requirements and if not how can this be improved?*

To answer this question the paper first introduced Clippy's lint list in section 2 and then summarized the technical requirements that the lint list of the rust-clippy project should fulfill.

The assignment then continued in section 3 with an evaluation to which extent the requirements are currently fulfilled. The results reveal that the user security is impacted by the three following missing HTTP headers:

- Strict-Transport-Security
- X-Frame-Options
- X-Content-Type-Options

[5] <https://rustup.rs/>

The following section 4 analyzed the security concern of these missing headers by explaining their behavior and reasoning behind them. This explanation was based on the formal specification and most relevant documentation. The paper then discussed the importance of Clippy's lint list and suggested an initial value that the specific field should be set to. The results of this evaluation have been summarized in table 2.

The 5 chapter then investigated how these values can be in the selected content host. This includes an investigation of which settings are provided by the used hosting provider GitHub Pages. Following an evaluating of the HTML meta tag and the use of a content delivery network to set these headers.

6.2 Solution

Chapter 5 determines that GitHub Pages configurations are not sufficient to meet the set requirements. In 5.2 it is found that the HTML meta tag can be used to set two of the required headers in Firefox and Chromium. This solution has the drawbacks that it is not a universal solution and only HTML document specific. The last investigated solution is the use of a CDN which is able to add the headers under the condition of using a custom domain. The last solution adds complexity to the project and uses additional resources.

The paper therefor concludes with two possible actions that can be taken. It now has to be evaluated which one of these should be taken if any. This evaluation extends over the scope of this paper.

6.3 Steps moving forward

Section 5 concludes that the missing headers can be set with the use of a content delivery network. The next step is now to investigate if the added complexity and additional use of resources worth it as the website is currently operating to no additional cost to the project. It also has to be taken into consideration that GitHub Pages is still a recommended hosting provider by the Rust Infrastructure Team.

If the decision is made to continue the use and deployment using GitHub Pages then it might be worth to investigate the HTML meta tag a bit more. The use of the meta tag can address the two main security concerns in regard of the `Strict-Transport-Security` and `X-Content-Type-Options` response header fields. Before using this feature it should to be investigated if this functionality is supported by all major browsers as it is not part of the living standard of HTML.

References

Amazon Web Services (2021). *Amazon CloudFront*. URL: https://aws.amazon.com/cloudfront/ (visited on 2021-05-01).

Bengtsson, Peter and et al. (2021-03). *X-Content-Type-Options*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options (visited on 2021-04-26).

GitHub Docs (2021a). *About GitHub Pages*. URL: https://docs.github.com/en/pages/getting-started-with-github-pages/about-github-pages (visited on 2021-05-02).

– (2021b). *GitHub's products*. URL: https://docs.github.com/en/github/getting-started-with-github/githubs-products (visited on 2021-05-02).

– (2021c). *Securing your GitHub Pages site with HTTPS*. URL: https://docs.github.com/en/pages/getting-started-with-github-pages/securing-your-github-pages-site-with-https (visited on 2021-04-30).

Hoare, Graydon and et al. (2019-01). *COPYRIGHT*. URL: https://github.com/rust-lang/rust/blob/master/COPYRIGHT (visited on 2021-04-20).

Hodges, Jeff and et al. (2012-11). *RFC6797: HTTP Strict Transport Security (HSTS)*. Tech. rep. Internet Engineering Task Force (IETF). URL: https://tools.ietf.org/html/rfc6797 (visited on 2021-04-25).

King, April (2018-07). *Web Security Cheat Sheet*. URL: https://infosec.mozilla.org/guidelines/web_security (visited on 2021-05-03).

King, April and et al. (2018a-01). *HTTP Observatory Scoring Methodology*. URL: https://github.com/mozilla/http-observatory/blob/fa38ab4/httpobs/docs/scoring.md (visited on 2021-04-24).

– (2018b-03). *Mozilla HTTP Observatory*. URL: https://github.com/mozilla/http-observatory/blob/1bb1566/README.md (visited on 2021-04-24).

– (2020-10). *Frequently Asked Questions*. URL: https://observatory.mozilla.org/faq/ (visited on 2021-04-24).

Laukenstein, Binyamin and Ben Balter (2017-02). *[Github Pages] Modify headers, respect _config.yml webrick headers*. URL: https://github.com/github/pages-gem/issues/415 (visited on 2021-04-30).

Lawrence, Eric (2008a-02). *IE8 Security Part V: Comprehensive Protection*. URL: https://docs.microsoft.com/en-us/archive/blogs/ie/ie8-security-part-v-comprehensive-protection (visited on 2021-04-29).

– (2008b-02). *IE8 Security Part VI: Beta 2 Update*. URL: https://docs.microsoft.com/en-us/archive/blogs/ie/ie8-security-part-vi-beta-2-update (visited on 2021-04-29).

Lusby, Jane (2018-07). *Add clippy to the tools list #1461*. URL: https://github.com/rust-lang/rustup/pull/1461 (visited on 2021-04-17).

Ross, David and Tobias Gondrom (2013-10). *RFC7034: HTTP Header Field X-Frame-Options*. Tech. rep. Internet Engineering Task Force (IETF). URL: https://tools.ietf.org/html/rfc7034 (visited on 2021-04-25).

Sommerville, Ian (2010). *Software Engineering*. 9th edition. 501 Boylston Street, Suite 900, Boston, Massachusetts 02116: Pearson Education, Inc. ISBN: 978-0-13-703515-1.

The Rust Clippy Developers (2020-10). *deploy.yml*. URL: https://github.com/rust-lang/rust-clippy/blob/master/.github/workflows/deploy.yml (visited on 2021-04-17).

The Rust Clippy Developers (2021-03). *Clippy*. URL: https://github.com/rust-lang/rust-clippy/blob/7fcd1/README.md (visited on 2021-04-17).

The Rust Core Team (2015-05). *Announcing Rust 1.0*. URL: https://blog.rust-lang.org/2015/05/15/Rust-1.0.html (visited on 2021-04-17).

– (2021-03). *Announcing Rust 1.51.0*. URL: https://blog.rust-lang.org/2021/03/25/Rust-1.51.0.html (visited on 2021-04-17).

The Rust Infrastructure Team (2020-03). *Rust Infrastructure hosting for static websites*. URL: https://forge.rust-lang.org/infra/guidelines/static-websites.html (visited on 2021-04-24).

The Rust Team (2021a-04). *All-time Contributors*. URL: https://thanks.rust-lang.org/rust/all-time/ (visited on 2021-04-20).

– (2021b). *Code of conduct*. URL: https://www.rust-lang.org/policies/code-of-conduct (visited on 2021-04-20).

trante and et al. (2013-02). *Github pages, HTTP headers*. Last time updated on 2019-06-18. URL: https://stackoverflow.com/questions/14798589/github-pages-http-headers (visited on 2021-04-30).

Turon, Aaron and Niko Matsakis (2014-10). *Stability as a Deliverable*. URL: https://blog.rust-lang.org/2014/10/30/Stability.html (visited on 2021-04-20).

WHATWG (2021-04). *HTML Living Standard*. Tech. rep. Web Hypertext Application Technology Working Group (WHATWG). URL: https://html.spec.whatwg.org/multipage/semantics.html (visited on 2021-05-01).

yawnoc and et al. (2021-04). [*Feature request*] *Set HTTP header to opt out of FLoC in GitHub Pages*. URL: https://github.community/t/feature-request-set-http-header-to-opt-out-of-floc-in-github-pages/174978 (visited on 2021-04-30).

7 Attachments

```
{
    "content-security-policy": {
        "expectation": "csp-implemented-with-no-unsafe",
        "name": "content-security-policy",
        "output": {
            "data": {
                "connect-src": [
                    "'self'"
                ],
                "default-src": [
                    "'none'"
                ],
                "img-src": [
                    "data:"
                ],
                "style-src": [
                    "'unsafe-inline'"
                ]
            },
            "http": true,
            "meta": true,
            "policy": {
                "antiClickjacking": false,
                "defaultNone": true,
                "insecureBaseUri": true,
                "insecureFormAction": true,
                "insecureSchemeActive": false,
                "insecureSchemePassive": false,
                "strictDynamic": false,
                "unsafeEval": false,
                "unsafeInline": false,
                "unsafeInlineStyle": true,
                "unsafeObjects": false
            }
        },
        "pass": true,
        "result": "csp-implemented-with-unsafe-inline-in-style-src-only",
        "score_description": "Content Security Policy (CSP) implemented with unsafe sources inside style-src. This includes 'unsafe-inline', data: or overly broad sources such as https:.",
        "score_modifier": 0
    },
    "contribute": {
        "expectation": "contribute-json-only-required-on-mozilla-properties",
        "name": "contribute",
        "output": {
            "data": null
        },
        "pass": true,
        "result": "contribute-json-only-required-on-mozilla-properties",
        "score_description": "Contribute.json isn't required on websites that don't belong to Mozilla",
        "score_modifier": 0
    },
    "cookies": {
        "expectation": "cookies-secure-with-httponly-sessions",
        "name": "cookies",
        "output": {
            "data": null,
            "sameSite": null
        },
        "pass": true,
        "result": "cookies-not-found",
        "score_description": "No cookies detected",
        "score_modifier": 0
    },
    "cross-origin-resource-sharing": {
        "expectation": "cross-origin-resource-sharing-not-implemented",
        "name": "cross-origin-resource-sharing",
        "output": {
            "data": {
                "acao": "*",
                "clientaccesspolicy": null,
                "crossdomain": null
            }
        },
        "pass": true,
        "result": "cross-origin-resource-sharing-implemented-with-public-access",
        "score_description": "Public content is visible via cross-origin resource sharing (CORS) Access-Control-Allow-Origin header",
        "score_modifier": 0
    },
    "public-key-pinning": {
        "expectation": "hpkp-not-implemented",
            "name": "public-key-pinning",
            "output": {
                "data": null,
                "includeSubDomains": false,
                "max-age": null,
                "numPins": null,
                "preloaded": false
            },
            "pass": true,
            "result": "hpkp-not-implemented",
            "score_description": "HTTP Public Key Pinning (HPKP) header not implemented",
            "score_modifier": 0
        },
        "redirection": {
            "expectation": "redirection-to-https",
            "name": "redirection",
            "output": {
                "destination": null,
                "redirects": true,
                "route": [
                    "http://rust-lang.github.io/",
                    "https://rust-lang.github.io/"
                ],
                "status_code": null
            },
            "pass": true,
            "result": "redirection-to-https",
            "score_description": "Initial redirection is to HTTPS on same host, final destination is HTTPS",
            "score_modifier": 0
        },
        "referrer-policy": {
            "expectation": "referrer-policy-private",
            "name": "referrer-policy",
            "output": {
                "data": null,
                "http": false,
                "meta": false
            },
            "pass": true,
            "result": "referrer-policy-not-implemented",
            "score_description": "Referrer-Policy header not implemented",
            "score_modifier": 0
        },
        "strict-transport-security": {
            "expectation": "hsts-implemented-max-age-at-least-six-months",
            "name": "strict-transport-security",
            "output": {
                "data": null,
                "includeSubDomains": false,
                "max-age": null,
                "preload": false,
                "preloaded": false
            },
            "pass": false,
            "result": "hsts-not-implemented",
            "score_description": "HTTP Strict Transport Security (HSTS) header not implemented",
            "score_modifier": -20
        },
        "subresource-integrity": {
            "expectation": "sri-implemented-and-external-scripts-loaded-securely",
            "name": "subresource-integrity",
            "output": {
                "data": {}
            },
            "pass": true,
            "result": "sri-not-implemented-but-no-scripts-loaded",
            "score_description": "Subresource Integrity (SRI) is not needed since site contains no script tags",
            "score_modifier": 0
        },
        "x-content-type-options": {
            "expectation": "x-content-type-options-nosniff",
            "name": "x-content-type-options",
            "output": {
                "data": null
            },
            "pass": false,
            "result": "x-content-type-options-not-implemented",
            "score_description": "X-Content-Type-Options header not implemented",
            "score_modifier": -5
        },
        "x-frame-options": {
            "expectation": "x-frame-options-sameorigin-or-deny",
            "name": "x-frame-options",
        "output": {
            "data": null
        },
        "pass": false,
        "result": "x-frame-options-not-implemented",
        "score_description": "X-Frame-Options (XFO) header not implemented",
        "score_modifier": -20
    },
    "x-xss-protection": {
        "expectation": "x-xss-protection-1-mode-block",
        "name": "x-xss-protection",
        "output": {
            "data": null
        },
        "pass": true,
        "result": "x-xss-protection-not-needed-due-to-csp",
        "score_description": "X-XSS-Protection header not needed due to strong Content Security Policy (CSP) header",
        "score_modifier": 0
    }
}
```

Attachment 1: Mozilla Observatory output for `rust-lang.github.io` from 2021-04-24

```
HTTP/1.1 200 OK
Date: Fri, 30 Apr 2021 19:08:24 GMT
Via: 1.1 varnish
Cache-Control: max-age=600
Expires: Fri, 30 Apr 2021 19:18:25 GMT
Age: 0
X-Served-By: cache-fra19138-FRA
X-Cache: MISS
X-Cache-Hits: 0
X-Timer: S1619809705.954395,VS0,VE90
Vary: Accept-Encoding
X-Fastly-Request-ID: 941e117097d3830dfbc28eda40096862241b1bcc
Server: GitHub.com
Content-Type: text/html; charset=utf-8
permissions-policy: interest-cohort=()
Last-Modified: Mon, 26 Apr 2021 21:40:11 GMT
Access-Control-Allow-Origin: *
ETag: W/"6087333b-4a05"
Content-Encoding: gzip
x-proxy-cache: MISS
X-GitHub-Request-Id: CB52:4B29:333A3A:3A24C8:608C3FC5
Content-Length: 4722
Accept-Ranges: bytes
```

Attachment 2: The HTTP response header for `https://rust-lang.github.io/rust-clippy/master/index.html` from 2021-04-30

```
HTTP/1.1 200 OK
Date: Fri, 30 Apr 2021 19:08:25 GMT
Via: 1.1 varnish
Cache-Control: max-age=600
Expires: Fri, 30 Apr 2021 19:18:25 GMT
Age: 0
X-Served-By: cache-fra19138-FRA
X-Cache: MISS
X-Cache-Hits: 0
X-Timer: S1619809705.316870,VS0,VE94
Vary: Accept-Encoding
X-Fastly-Request-ID: 30deedad3daffa933d4f55f7174a69c2fd13ffa3
Server: GitHub.com
Content-Type: application/json; charset=utf-8
permissions-policy: interest-cohort=()
Last-Modified: Mon, 26 Apr 2021 21:40:11 GMT
Access-Control-Allow-Origin: *
ETag: W/"6087333b-4ec5f"
Content-Encoding: gzip
x-proxy-cache: MISS
X-GitHub-Request-Id: 98EA:29DE:BE987C:C3D1A8:608C3FC5
Content-Length: 80925
Accept-Ranges: bytes
```

Attachment 3: The HTTP response header for `https://rust-lang.github.io/rust-clippy/master/lints.json` from 2021-04-30

```
HTTP/1.1 301 Moved Permanently
Server: GitHub.com
Content−Type: text/html
permissions−policy: interest−cohort=()
Location: https://xfrednet.github.io/rust−clippy/
X−GitHub−Request−Id: 3620:3A01:104182F:110536D:608C3D70
Content−Length: 162
Accept−Ranges: bytes
Date: Fri, 30 Apr 2021 17:25:04 GMT
Via: 1.1 varnish
Age: 0
X−Served−By: cache−fra19120−FRA
X−Cache: MISS
X−Cache−Hits: 0
X−Timer: S1619803505.769013,VS0,VE87
Vary: Accept−Encoding
X−Fastly−Request−ID: b837829c5922d053b087ff1e129d92f5b470a120
```

Attachment 4: The HTTP response for the test page with *enfore HTTPS* enabled from 2021-04-30

```
{
    "content−security−policy": {
        "expectation": "csp−implemented−with−no−unsafe",
        "name": "content−security−policy",
        "output": {
            "data": {
                "default−src": [
                    "'none'"
                ],
                "font−src": [
                    "'self'"
                ],
                "img−src": [
                    "'self'",
                    "https://www.rust−lang.org"
                ],
                "script−src": [
                    "'self'"
                ],
                "style−src": [
                    "'self'"
                ]
            },
            "http": true,
            "meta": false,
            "policy": {
                "antiClickjacking": false,
                "defaultNone": true,
                "insecureBaseUri": true,
                "insecureFormAction": true,
                "insecureSchemeActive": false,
                "insecureSchemePassive": false,
                "strictDynamic": false,
                "unsafeEval": false,
                "unsafeInline": false,
                "unsafeInlineStyle": false,
                "unsafeObjects": false
            }
        },
        "pass": true,
        "result": "csp−implemented−with−no−unsafe−default−src−none",
        "score_description": "Content Security Policy (CSP) implemented with default−src 'none' and no 'unsafe'",
        "score_modifier": 10
    },
    "contribute": {
        "expectation": "contribute−json−only−required−on−mozilla−properties",
        "name": "contribute",
        "output": {
            "data": null
        },
        "pass": true,
        "result": "contribute−json−only−required−on−mozilla−properties",
        "score_description": "Contribute.json isn't required on websites that don't belong to Mozilla",
        "score_modifier": 0
    },
    "cookies": {
        "expectation": "cookies-secure-with-httponly-sessions",
        "name": "cookies",
        "output": {
            "data": null,
            "sameSite": null
        },
        "pass": true,
        "result": "cookies-not-found",
        "score_description": "No cookies detected",
        "score_modifier": 0
    },
    "cross-origin-resource-sharing": {
        "expectation": "cross-origin-resource-sharing-not-implemented",
        "name": "cross-origin-resource-sharing",
        "output": {
            "data": {
                "acao": null,
                "clientaccesspolicy": null,
                "crossdomain": null
            }
        },
        "pass": true,
        "result": "cross-origin-resource-sharing-not-implemented",
        "score_description": "Content is not visible via cross-origin resource sharing (CORS) files or headers",
        "score_modifier": 0
    },
    "public-key-pinning": {
        "expectation": "hpkp-not-implemented",
        "name": "public-key-pinning",
        "output": {
            "data": null,
            "includeSubDomains": false,
            "max-age": null,
            "numPins": null,
            "preloaded": false
        },
        "pass": true,
        "result": "hpkp-not-implemented",
        "score_description": "HTTP Public Key Pinning (HPKP) header not implemented",
        "score_modifier": 0
    },
    "redirection": {
        "expectation": "redirection-to-https",
        "name": "redirection",
        "output": {
            "destination": "https://rustup.rs/",
            "redirects": true,
            "route": [
                "http://rustup.rs/",
                "https://rustup.rs/"
            ],
            "status_code": 200
        },
        "pass": true,
        "result": "redirection-to-https",
        "score_description": "Initial redirection is to HTTPS on same host, final destination is HTTPS",
        "score_modifier": 0
    },
    "referrer-policy": {
        "expectation": "referrer-policy-private",
        "name": "referrer-policy",
        "output": {
            "data": "no-referrer, strict-origin-when-cross-origin",
            "http": true,
            "meta": false
        },
        "pass": true,
        "result": "referrer-policy-private",
        "score_description": "Referrer-Policy header set to \"no-referrer\", \"same-origin\", \"strict-origin\" or \"strict-origin-when-cross-origin\"",
        "score_modifier": 5
    },
    "strict-transport-security": {
        "expectation": "hsts-implemented-max-age-at-least-six-months",
        "name": "strict-transport-security",
        "output": {
            "data": "max-age=63072000; includeSubDomains",
            "includeSubDomains": true,
            "max-age": 63072000,
            "preload": false,
            "preloaded": false
        },
        "pass": true,
        "result": "hsts-implemented-max-age-at-least-six-months",
        "score_description": "HTTP Strict Transport Security (HSTS) header set to a minimum of six months (15768000)",
        "score_modifier": 0
    },
    "subresource-integrity": {
        "expectation": "sri-implemented-and-external-scripts-loaded-securely",
        "name": "subresource-integrity",
        "output": {
            "data": {}
        },
        "pass": true,
        "result": "sri-not-implemented-but-all-scripts-loaded-from-secure-origin",
        "score_description": "Subresource Integrity (SRI) not implemented, but all scripts are loaded from a similar origin",
        "score_modifier": 0
    },
    "x-content-type-options": {
        "expectation": "x-content-type-options-nosniff",
        "name": "x-content-type-options",
        "output": {
            "data": "nosniff"
        },
        "pass": true,
        "result": "x-content-type-options-nosniff",
        "score_description": "X-Content-Type-Options header set to \"nosniff\"",
        "score_modifier": 0
    },
    "x-frame-options": {
        "expectation": "x-frame-options-sameorigin-or-deny",
        "name": "x-frame-options",
        "output": {
            "data": "DENY"
        },
        "pass": true,
        "result": "x-frame-options-sameorigin-or-deny",
        "score_description": "X-Frame-Options (XFO) header set to SAMEORIGIN or DENY",
        "score_modifier": 0
    },
    "x-xss-protection": {
        "expectation": "x-xss-protection-1-mode-block",
        "name": "x-xss-protection",
        "output": {
            "data": "1; mode=block"
        },
        "pass": true,
        "result": "x-xss-protection-enabled-mode-block",
        "score_description": "X-XSS-Protection header set to \"1; mode=block\"",
        "score_modifier": 0
    }
}
```

Attachment 5: Mozilla Observatory output for `rustup.rs` from 2021-05-01