Student Number: 180240438
Nicholas Hitch

Measuring Adoption of Security Mechanisms in the HTTPS Ecosystem

Supervisor: Simon Bell

Submitted as part of the requirements for the award of the MSc in Information Security at Royal Holloway, University of London.

Anti-Plagiarism Declaration

I declare that this assignment is all my own work and that I have acknowledged all quotations from published or unpublished work of other people. I also declare that I have read the statements on plagiarism in Section 1 of the Regulations Governing Examination and Assessment Offences, and in accordance with these regulations I submit this project report as my own work.

[illegible]

Nicholas Hitch

March 27, 2022

Acknowledgements

Thanks to my supervisor Simon, who has provided great advice throughout allowing the report to be what it is.

Thanks to my father, Roger, who helped with proof reading, which aided in making the report more grammatically correct as well as improving readability and the flow of the report.

Lastly thanks to all the previous research that has been cited which has allowed this study to build upon them.

Intentionally Blank

Executive Summary

This report is one of the required modules for the course `MSc in Information Security` at Royal Holloway, University of London. This is a longitudinal study for the adoption of several security mechanisms, including: `Transport Layer Security`, `Content Security Policy`, `STS Preloading` and `Security.txt`.

This is accomplished by scanning the top 1 million ranked domains each day for 16 months, from November 2020 through to the end of January 2022 and subsequent analysis of the data in order to determine the adoption of the chosen security mechanisms.

The analysis finds that the distribution of TLS versions is trending in the right direction with TLS 1.3 reaching a 64.4% proportion of TLS connections to websites in the top 1 million ranked domains as of January 2022.

There is still wide spread usage of the `unsafe-inline` and `unsafe-eval` keywords in Content Security Polices, especially with the `script-src` directive, which allows a great attack vector for cross site scripting attacks.

There are a relatively large proportion of domains that are STS preloaded but no longer meet the STS preload requirements, which puts them at risk of being removed from the STS preload list.

The security.txt mechanism currently has low utilisation, however the RFC is still in the draft stage and the number of domains using this mechanism is trending upward. Even with the low usage, the number of security.txt that have a Contact is encouraging.

This study also adds some novel contributions not found during the literature research phase which are:

- Analysis of the TLS versions that are supported by a website (rather than just the TLS version negotiated when making a HTTPS request) in section 5.3.
- Analysis of the newest Cross Origin Security headers in sections 5.10, 5.12 and 5.11.

It should be noted that scanning of websites commenced in November 2020, before the *"Project"* module was officially started in October 2021. This was due to the need to gather at least 12 months of data before the project submission deadline at the end of March 2022.

Acronyms

API	Application Programming Interface
CSS	Cascading Style Sheets
CORS	Cross-Origin-Request-Sharing
CSP	Content Security Policy
DOM	Document Object Model
FIPS	Federal Information Processing Standards
HTML	Hypertext Markup Language
HPKP	HTTP Public Key Pinning
HTTP	Hypertext Transfer Protocol
MIME	Multipurpose Internet Mail Extensions
URI	Uniform Resource Identifier
URL	Universal Resource Locator
SSL	Secure Socket Layer
TLS	Transport Layer Security
XSS	Cross-Site Scripting

Glossary

Application Programming Interface is an interface intended to be used by a computer program for the purpose of allowing two computer programs to communicate to one another in a structured format.

Cascading Style Sheets is a mechanism by which a web page can be styled by many different aspects such as fonts, colours and spaces.

Cipher Suite defines the implementation of cryptographic primitives and additional information referable via a unique identifier such as TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 for the use in establishment of SSL/TLS connections [1].

Cross Origin-Request-Sharing is a web browser mechanism for the purpose of allowing a resource to specify the origins allowed to access the resource via the use of HTTP Headers.

Cross Site Scripting A browser attack whereby a malicious script is inserted into a web page for the purpose of performing wanted actions on a user's online account and or gaining access to sensitive information (e.g. authentication cookies and session identifiers) usually via a web browser.

Content Security Policy is a HTTP protocol feature to restrict which resources can be fetched and or executed, such as for the purposes of obtaining content e.g. images, whilst on a specific web page in a web browser. The policy details can be specified in either a HTTP Header or in the head section of a web page via the meta tag.

Document Object Model is the web browsers internal representation of an html page as a result of the browser parsing the html [2].

Federal Information Processing Standards standards for federal computer systems which are developed by the National Institute of Standards and Technology (NIST)

Hypertext Markup Language a standardised language to create documents [3] for instructing a web browser how a web page should be rendered/visualised.

Hypertext Transfer Protocol a stateless application level protocol [4] for the transmission of data (e.g. HTML) typically between a website and a web browser.

Multi-purpose Internet Mail Extensions is used to announce the intended format of a resource (e.g document or file) [5].

Universal Resource Locator a specific type of URI that has a scheme (how to access) and a resource (where to access). The most basic form of a URI is as follows <`scheme`>:<`scheme-specific-part`>. An example URL is `https://www.example.com`

Secure Socket Layer a protocol to establish a secure communications channel mainly used by the HTTP protocol

Transport Layer Security a protocol, successor of the SSL protocol, to establish a secure communications channel mainly used by the HTTP protocol

List of Figures

List of Tables

Contents

Introduction

The world revolves around internet communication and websites are a big part of that. There have been many security improvements to the technology that powers the HTTPS ecosystem and as such, analysing how these are or are not being utilised is of great interest and value to the community.

Security is a battle ground with attackers ever changing their techniques and defenders doing their best to thwart the attackers attempts.

In 1994 SSL was proposed [6, 7] to make interactions with websites more secure. As time has moved on, more improved secure channel protocols, such as TLS 1.3 [8] being the most recent, have been introduced as well as enhancements to HTTP through security headers such as HTTP Strict Transport Security [9].

HTTPS is one of the critical security mechanisms and was introduced as HTTP does not have any protections against the following:

- Viewing the plaintext data being exchanged between the browser and website.
- Manipulating data being exchanged between the browser and website without detection.
- Entity impersonating a website without detection.

HTTPS overcomes these issues as it provides several protections including:

- Only the user browsing a website and the website itself are able to view the plaintext data being sent between them as it is encrypted in transit.
- If the encrypted data in transit is modified this can be detected.
- The authenticity of the identity of the website being visited can be verified using certificate authorities.

HTTPS is detailed more extensively in section 4.1.

This research aims to answer the question:

`What is the current adoption of security mechanisms in the HTTPS ecosystem ?`

Measuring the adoption of HTTPS security mechanisms is important for many reasons including:

- Allowing the community to gain insight in the their use.
- Identify trends (good, bad and unexpected) leading to initiatives to improve mechanism use.
- Identify the usage of obsolete and or mechanisms that should no longer be in use.
- Identify ineffective and or mis-configurations and how widespread they are.
- Attempt to infer why a mechanism has lower than expected/desired usage.

It is likely that the more complicated the security mechanism are to deploy and or implement, the less use it will see. This research intends to discover to what extent security mechanisms are currently deployed and attempt to determine possible reasons for large or small adoptions. Once the adoption has been measured it shall be compared to similar research to seek indications of adoption trends.

Chapter 2 details "Why Security Mechanisms Exist" as it is provides additional context when analysing the adoption of security mechanisms.

The usage of "Top 1 Million sites" lists are quite popular with researchers in measurement studies as shown by use in [10, 11, 12, 13, 14, 15, 16, 17, 18]. As these lists identify themselves as containing the "Top" sites, this provides a source of domains that are likely to be the most commonly used by users browsing the internet and thus a very relevant source for study. The lists containing 1 million domains is also of sufficient size to gain a broad insight into the websites on the internet.

The main contributions of this research are (1) Which TLS versions a website supports (section 5.3); (2) Continuation from previous studies on the adoption of security headers; (3) The adoption of `Cross Origin` security headers introduced relatively recently (sections 5.10, 5.11, 5.12).

Objectives and Scope

In order to conduct research to answer the question stated in the introduction in a constructive manner, one must set out a list of objectives to be completed:

1. *Identify security mechanisms of which to measure the adoption.*
2. *Analyse previous related measurement research for data capture methodology.*
3. *Scan services in the HTTPS ecosystem to obtain the required data to be able to measure adoption of the chosen security mechanisms.*
4. *Analyse the adoption of chosen security mechanisms.*
5. *Compare and contrast the analysis outcomes to related research.*

There are many services in the HTTPS ecosystem such as websites and API interfaces and as such the target(s) of measurement will be restricted otherwise the research could take a very long time if not endless. The research will be scoped by the following criteria:

1. *The measurements will be restricted to websites as they are interacted with by a large proportion of people today, thus providing useful analysis targets.*
2. *Scan websites no more than once a day for a period of at least 12 months.*
3. *Restrict the websites scanned to the top 1 million most popular.*
4. *Only metadata, i.e. headers and connection details, will be captured and not the body content of websites.*

The reasoning for the chosen scopes is discussed in section 6.1.1.

Document Structure

This report going forward is structured as follows:

Chapter 1 - Security Mechanisms Primer provides a breif background of relevant technologies related to the security mechanisms to be analysed. The chosen security mechanisms to be analysed are given a brief overview along with attacks they are designed to mitigate.

Chapter 2 - Why Security Mechanisms Exist details the need for security mechanisms, focusing on those mentioned in chapter 1, including the following attacks: *Cross Site Scripting (XSS)*, *Supply Chain Attacks*, *Clickjacking* and *Entity in the Middle Attacks*.

Chapter 3 - Data Acquisition discusses internet scanning, the issues to be aware of and how other similar research has captured their data. This part concludes with detailing how this project carried out the data capture which is analysed in chapter 5.

Chapter 4 - Security Mechanisms Overview goes into further detail about each chosen security mechanism as detailed in section 1.3 which are analysed in chapter 5. This is to allow the reader to gain further context for the analysis chapter.

Chapter 5 - Security Mechanism Adoption Analysis analyses the chosen security mechanisms described in section 1.3 from the data captured in chapter 3, over a 16 month period from November 2020 to February 2022.

Chapter 6 - Conclusions and Closing Remarks brings the report to an end concluding that apart from TLS the other chosen security mechanism, as detailed in section 1.3, that are currently not deprecated are of relatively low usage and work needs to be done to improve this.

Chapter 1

Security Mechanisms Primer

This chapter introduces the selected security mechanisms that are analysed in chapter 5 and technologies that they build upon.

1.1 HTTP

HTTP (Hypertext Transfer Protocol), standardised in 1996 [4], is a stateless protocol used for the transmission of data between a web browser and a website. When one enters an address into the address bar of a web browser or clicks a link on a web page, HTTP is used to send the request to and receive the response from a website.

1.1.1 HTTP Headers

HTTP Headers make up part of the additional meta-data that accompanies request and response payloads on the HTTP protocol.

Headers are key value pairs delimited by the colon `:` character [4]. The key of a header is case insensitive.

An example header could be `server: nginx` where `server` is the key and `nginx` is the value.

1.1.2 HTTP Methods

HTTP has a number of methods for the purpose of denoting the intended request action that the request initiator is requesting to perform on a resource.

The below are relevant methods for this project:

- GET – Requests a specified resource without altering the resource.
- HEAD – Identical to the GET method without the resource being returned (i.e. only metadata such as HTTP Headers)

1.1.3 HTTPS

The S in HTTPS signifies that the HTTP protocol communication shall be over a secure channel, provided by the SSL/TLS protocols [19]. A URL starting with `https://` e.g. `https://example.com` signifies it will be using a SSL/TLS secure channel.

SSL/TLS provides several protections including: confidentiality, data origin authentication and entity authentication. Entity authentication of a website is provided by the use of "certificates" which are issued by certificate authorities. A users' browser has a predefined list of certificate authorities that it trusts and a website certificate will only be deemed valid of it was issued by a certificate authority in the predefined list.

Slowly over time HTTPS has seen an ever increasing amount of adoption. There have been several incentives over the years to encourage the use of HTTPS and reduce the barrier to entry:

- Lets Encrypt provides free certificates, a prerequisite to enable HTTPS, for websites and is currently able to issue at least 200 million certificates a day as of February 2020 [20]
- Search engines will rank websites higher if they use HTTPS. Google uses HTTPS as a "very lightweight signal" [21]
- Google Chrome and Mozilla Firefox both started to show warnings in their browsers in 2017 when users typed in passwords on a site loaded over HTTP [22, 23]
- Governments and Community resources on why to use HTTPS [24, 25]

SSL/TLS is detailed in more depth in sections 1.3.1 and 4.2.

1.2 Same Origin Policy

Same-Origin Policy (SOP) is a critical base security mechanism of web browsers that restricts the origin of a resource (e.g. script) to only be able to communicate to resources from the same origin.

The concept of SOP started in 1996 with the release of Netscape Navigator 2.0 [26] and standardised by The Web Hypertext Application Technology Working Group (WHATWG) [27]

If the scheme (e.g. `https://`), port (e.g. 443) and host (e.g. images.example.com) are the same for two URLs they are deemed to be in the same origin. The combination of scheme, port and host can be referred to as the origin tuple or triplet.

A request is deemed to be "cross origin" when the origin triplets of the url of the request being made is different from that of the origin of the request.

The path of a URL after the origin triplet is NOT evaluated when origins are being compared, for example the origin triplet of `https://example.com/index.html` is `https://example.com`.

Table 1.1 shows several examples of same origin triplet violations of http requests when a user is currently viewing the url `https://shop.example.com/index.html` in their browser, which is of the origin `https://shop.example.com`.

1.2.1 Cross Origin Network Requests

A Cross origin request is one that is between two different origins, examples of which are shown as a violation in table 1.1.

Cross Origin network requests can be summarised into 3 different categories:

- Cross-Origin reads are mostly denied (to prevent attacks such as XSS which is discussed further in section 2.1.1).
- Cross-Origin embedding is mostly allowed (e.g. `<script src="..."></script>`)
- Cross-Origin writes are mostly allowed (e.g. form submissions, links and redirects)

URL	**Same Origin Triplet Violation**	**Reasoning**
https://shop.example.com/books.html	No	The difference is after the origin triplet
https://shop.example.com/music.html	No	The difference is after the origin triplet
https://blog.example.com/index.html	Yes	The host of the origin triplet, blog.example.com, is different.
https://shop.example.com:8000/forum.html	Yes	The port of the origin triplet is different (HTTPS default port is 443)
http://shop.example.com/index.html	Yes	The scheme of the origin triplet, HTTP, is different.

Table 1.1: Same Origin triplet example violations

1.2.2 Cross Origin Request Sharing

Cross Origin Request Sharing is a web browser mechanism for the purpose of allowing a resource to specify the restrictions to be able to access a resource via one or more of the HTTP Headers [28].

Cross-Origin-Request-Sharing (CORS) can be used to override the default SOP behaviour, however it must be used with caution to limit exposure to attacks.

A request is deemed to be "cross origin" when the origin triplets of the url of the request being made is different from that of the origin of the request. For example if a script on the web page `https://shop.example.com/index.com` which has the origin `https://shop.example.com` wants to make a request to `https://www.domain.com/login.html` which has the origin `https://www.domain.com` the two origins are different thus the request is deemed to be cross-origin.

Under certain circumstances a 'preflight', HTTP OPTIONS call, is made to the resource in question to obtain the criteria that must be met for the desired request to be allowed to be made.

1.3 Selected Security Mechanisms

The following security mechanisms are those that have been selected for their adoption to be analysed. They are briefly summarised here and detailed more in depth in chapter 4 and then analysed in chapter 5.

1.3.1 SSL/TLS

For the context of this research, SSL/TLS provides the secure channel used to send HTTP protocol traffic securely. This is more widely known as HTTPS.

SSL/TLS provides the following main security services:

- Confidentiality - the protection of data such that only those who have been given authorisation can access the data.
- Data Origin Authentication - establishes the integrity of the SSL/TLS channel data, i.e. to be able to verify the data was sent, and not altered in transit, by the corresponding party with whom the SSL/TLS channel was established with.
- Entity Authentication - the ability for an entity to prove their identity using mechanisms such as digital signatures.

This mechanism is detailed further in chapter 4.2 and analysed in section 5.3.

1.3.2 Security.txt

The “security.txt” mechanism is to aid in informing security researchers how to disclose security issues found on systems such as a website [29].

The specification specifies a number of locations where a “security.txt” file should be placed thus allowing security researchers to know where to look to find disclosure/reporting information should a site support the security.txt mechanism.

This mechanism is detailed further in section 4.3 and analysed in section 5.4.

1.3.3 Content Security Policy

The Content Security Policy (CSP) mechanism allows restrictions to be specified, in either a header or HTTP meta tag, in the form of a policy.

A policy can consist of a number of directives targeted at restricting the loading/executing of resources on a web page in a browser, such as only allowing scripts to be loaded from a specific list of domains.

The main attacks that are intended to be mitigated include Cross Site Scripting (XSS), data injection , packet sniffing and supply chain attacks.

This mechanism is detailed further in chapter 4.4 and analysed in section 5.5.

1.3.4 Strict Transport Security

The `Strict-Transport Security` response header (STS) is a mechanism that instructs the browser to load the website only over HTTPS and never HTTP [9]. This prevents an attacker trying to get the user to use HTTP to access the site so that the attacker could perform an entity in the middle attack (detailed further in section 2.1.4) allowing the attacker to impersonate the user or just capture sensitive information to name only a few possibilities.

There is an issue with this header such that a browser can only know to load a website only over HTTPS when the STS header is received. On the first visit to a website an attacker could intercept a HTTP request to the website and keep the connection HTTP and thus the user would not get the STS header. To overcome this, STS Preloading is required which is detailed next.

This mechanism is detailed further in chapter 4.5 and analysed in section 5.6.

1.3.5 STS Preloading

STS Preloading is a mechanism where a browser checks to see if the requested domain is in a predefined list and if it is, then the website must only be accessed over HTTPS.

The main reason for STS Preloading is to ensure the first time a user visits a website it is over HTTPS. Without preloading, should a user visit the site the first time over HTTP, an attacker could intercept the traffic and perform malicious actions such as the entity in the middle attack as described in section 2.1.4.

Google maintains a predefined list of websites where the website owners/maintainers have requested to be added [9].

This mechanism is detailed further in chapter 4.6 and analysed in section 5.7.

1.3.6 X Content Type Options

The `X-Content-Type-Options` (XCTO) response header is used to enforce that the "Multipurpose Internet Mail Extensions" (MIME) specified in the `Content-Type` header (e.g. `Content-Type: text/css`) matches that of the requested resource [30].

This mechanism is intended to mitigate against such attacks as *drive by downloads* and untrusted user content being treated as an executable.

This mechanism is detailed further in chapter 4.7 and analysed in section 5.8.

1.3.7 X Frame Options

The `X-Frame-Options` (XFO) response header is a mechanism that states if a browser should load the url in question in any of the following html elements: <`frame`>, <`iframe`>, <`embed`>, or <`object`>.

This mechanism is intended to mitigate against such attacks as clickjacking where an attacker invisibly embeds another site on top of theirs tricking users, via the use of enticing offers such as free electronics, into clicking a seemingly harmless link which actually triggers a function on the embedded site such as transferring money via one's banking website to the attacker.

This mechanism is detailed further in chapter 4.8 and analysed in section 5.9.

1.3.8 Cross Origin Embedder Policy

The `Cross-Origin-Embedder-Policy` (COEP) response header allows a website maintainer to require that any cross-origin resources are only allowed to be loaded if the resource is explicitly permitted to do so via a Cross Origin Resource Policy (CORP) or a Access Control Allow Origin CORS header or meta tag.

A driving factor for this header being introduced was that some developers may implement CORS in minimal way, such as setting the requesters origin in the Access-Control-Allow-Origin header [31]. This may have the undesired affect of allowing potentially sensitive data to be accessed unintentionally.

COEP has a lower risk than CORS of a similar situation form occurring as it is a lower privilege than CORS.

This mechanism is detailed further in chapter 4.9 and analysed in section 5.10.

1.3.9 Cross Origin Resource Policy

The `Cross-Origin-Resource-Policy` (CORP) response header is used to restrict which origins are allowed to read resource(s), such as scripts and images, protected by this mechanism.

A primary class of attacks that this mitigates against are speculative side-channel attacks. The Spectre attack [32] is one such attack which was publicly announced in 2018.

This mechanism is detailed further in chapter 4.10 and analysed in section 5.11.

1.3.10 Cross Origin Opener Policy

The `Cross-Origin-Opener-Policy` (COOP) response header provides a mechanism that prevents the sharing of browsing contexts of a top-level document with cross-origin documents. For example, preventing a popup window from communicating to the web page (document) that opened the popup via the use of process isolation [33].

This is also known as a process-isolation mechanism, preventing attackers from gaining access to a global object that created a popup window/tab for example.

This mechanism is detailed further in chapter 4.11 and analysed in section 5.12.

1.3.11 Public Key Pinning

The `Public-Key-Pins` (HPKP) response header is a mechanism that permits a cryptographic public key to be advertised that must be matched to the website certificate presented during the SSL/TLS connection establishment otherwise the website will be prevented from loading/rendering.

This header was intended to protect websites in the event a CA provider was compromised, such as DigiNotar in 2011 [34], and unauthorised certificates then being issued to an attacker for the purposes of impersonating websites. The HPKP mechanism would prevent the unauthorised certificate from being trusted by visitors that had previously visited the website i.e. preventing an entity in the middle attack.

This mechanism is detailed further in chapter 4.12 and analysed in section 5.13.

1.4 Summary

In this opening chapter, a brief overview of the core HTTP technology was presented leading to an overview of the selected security mechanisms (detailed in more depth in chapter 4) the adoption of which is to be analysed later in chapter 5. Next in chapter 2 "Why Security Mechanisms Exist" attacks related to the *selected security mechanisms*, as outlined in section 1.3, are detailed along with the security mechanisms that can be used to mitigate these attacks.

Chapter 2

Why Security Mechanisms Exist

This chapter details reasons for security mechanisms to exist and walks through several attack types with real world examples stating which of the chosen security mechanisms, detailed in section 1.3, are intended to mitigate the attacks.

2.1 The Need for Security Mechanisms

As communication technologies continually improve, it has allowed and attracted an ever-increasing amount of people to interact with services on the internet. Malicious actors and security researches are drawn to this large online user base interacting with online services such as those that:

- process payments [35]
- provide online banking [36]
- contain confidential information (e.g. medical related data) [37]
- provide communication channels (e.g. social networks and forums)

There are many entities that track trends and malicious activity, one of which is the non-profit organisation Open Web Application Security Project (OWASP) which works to advance the security of applications [38, 39].

OWASP maintain a Top Ten list, determined using a publicised methodology, which states the most important risks to web applications [39, 40]. In the "2021 Top Ten Report", injection attacks (of which Cross Site Scripting (XSS) detailed in section 2.1.1 is one variant) was ranked number three and of the 94% of the applications they tested, there was an average incident rate of 3% and 274k occurrences [41].

The "Global Data At Risk State of the Web" 2020 report [42] claims that on average, the top 1000 websites in the Alexa top one million website list, rely upon the 32 third party integrations. This gives a great opportunity for attackers to leverage the supply chain attack (detailed in section 2.1.2) to compromise websites that may be harder to compromise directly. One such example of this attack is the 2018 Magecart attack on Ticketmaster [35] to steal payment information via the compromise of third party resources.

Due to the susceptibility of data traversing the internet being intercepted and attacked, its integrity and confidentially needs to be protected. For browser traffic this is achieved by the use of HTTPS, which uses SSL/TLS secure channels, instead of the insure HTTP. Unfortunately not all websites use HTTPS and or redirect to the HTTPS version should it exist. Governments are encouraging the use of HTTPS [24] as well as the community itself [25]. The general advice is that if one is in control of a website, it should be using HTTPS regardless of the websites purpose or content.

The following attacks are ones that can be mitigated by one or more of the security mechanisms described section 1.3.

2.1.1 Cross Site Scripting (XSS)

Cross Site Scripting (XSS) attacks are part of the injection class of attacks where one or more malicious scripts, commonly written in JavaScript, are injected and subsequently executed on a web browser. There are four main variants of XSS attacks:

Attack Variants

Reflected

Stored XSS (also referred to as *Non-Persistent*, *Indirect* and *Type II*) occurs as a result of when a request is made to a web application, that contains malicious content, the response contains a malicious script that is executed by the web browser. The malicious script is reflected back to the user, hence the name of this variant [43].

Stored

Stored XSS (also referred to as *Persistent*, *Direct* and *Type I*) is where a malicious actor is able to store a malicious script on a website, by some means such as a forum post or comment section of a website such that the script will be executed, by the web application when a victim visits the page. The power of this variant is that the script is permanent, until when or even if the script is detected and removed. This variant has the potential to have many visitors to become victims depending on the amount of traffic to the compromised web page(s) [43].

DOM-Based

DOM-Based XSS (also referred to as *Type 0*) takes place entirely in the DOM. The Document Object Model (DOM) is the web browsers internal representation of an html page as a result of the browser parsing the html. The source (e.g. the parsed html), functionality and destination of the malicious script is all contained within the DOM [43, 44].

Mutation Based

Mutation Based XSS (also referred to as *mXSS*) is based on the use of the `innterHTML` functionality, often used to provide web application users to do custom styling, of a browser which allows direct manipulation of the HTML content bypassing the DOM entirely. The malicious actor crafts a "mutated" payload that once processed by the `innerHTML` functionality, allows the script to execute [45].

Historical Example

In October of 2005 it was discovered that when users visited an infected MySpace profile the phrase "but most of all, samy is my hero" was appended to the "hero" section of visiting users profiles. The infected profile contained a stored XSS payload which added itself to the visiting users profile and adding the aforementioned phrase. The attack was initiated by "Samy" infecting his own profile with the XSS payload. Over one million user profiles were reported to have been affected within 20 hours [46].

The attack is known as the "Samy Worm" as it propagated to other MySspace profiles just by visiting an infected profile.

Security Mechanism(s) Providing Mitigation

- Content Security Policy (CSP) – multiple directives

- Cross-Origin-Resource-Policy
- X-Frame-Options

2.1.2 Supply Chain

Supply chain attacks are when a malicious entity maliciously alters one or more third party resource(s) (such as a JavaScript library) that a web application uses, commonly hosted on a third party location such as a CDN. These malicious modifications are for several reasons including:

- Running a cryptominer [47] on browsers
- Capturing Payment Information
- Screen Scraping
- Ad injection

Historical Example

The 2018 Texthelp breach is one example of this type of attack where a malicious entity compromised a JavaScript library and modified it to include a cryptominer. A cryptominer uses computing resources to mine cryptocurrency such as (Bitcoin, Etherium and Monero). This single compromised JavaScript library was reportedly found to be used on 4,000 websites [48].

Security Mechanism(s) Providing Mitigation

- Content Security Policy (CSP) - sub resource integrity

2.1.3 Clickjacking

A clickjacking attack is one in which a user-initiated attack is hijacked to perform unwanted actions. A user is enticed to click on an element of a page, such as an image, however rather than an action or outcome the user expects to occur, one determined by the attacker takes place. To undertake the attack a malicious site renders a web page from the target website within an iframe. The malicious site uses styling to show only what the attacker desires of the target website [49].

Historical Example

The Twitter Attack, also referred to as the "Twitter Bomb", begins by a user seeing a twitter post that contains the text `Don't Click: http://tinyurl.com/amgzs6` [50] and clicking on the link. Visiting the link will present the user with a page seemingly only containing only a single button labelled `Don't Click`.

Unbeknownst to the user, the twitter home page is also rendered on the page but in an invisible iframe. The `Don't Click` button is directly underneath the invisible "update" button on the twitter home page, which posts a tweet. The iframe is configure to load the url `http://www.twitter.com/?status=Don't Click: http://tinyurl.com/amgzs6` [50] and twitters `?status=` url parameter feature will pre-load the users tweet box with the value from the `?status=` url parameter.

When the user clicks the `Don't Click` section of the page they are actually clicking the invisible `update` button of the twitter home page which posts a tweet with the text `Don't Click: http://tinyurl.com/amgzs6` [50] which helps to spread the message. There was no ill intent to this clickjacking attack and as such considered more of a prank.

Security Mechanism(s) Providing Mitigation

- Content Security Policy (CSP) – frame-ancestors directive
- X-Frame-Options

2.1.4 Entity in the Middle

Malicious actors are constantly on the lookout for ways to obtain sensitive data for a multitude of reasons including: intellectual property theft, stealing money and personal information.

One of the ways to help prevent this is to secure the communication traffic over the internet using HTTPS which relies on the SSL/TLS mechanism.

If a malicious actor is able to intercept and or passively monitor web traffic, the opportunity arises to perform what is known as an Entity in the Middle attack.

The Entity in the Middle attack is where an attacker is able to intercept and manipulate traffic between the victim and the target entity such as a website. This can lead to attacks such as victims unknowingly giving their credentials to attackers as well as attackers manipulating the response from the website before being sent back to the victim.

Over the years there have been several attacks against the SSL/TLS mechanism, compromising the confidentiality and or integrity of the information it is meant to secure.

Historical Examples

BEAST (2011)

The BEAST attack, announced in 2011, was for use against TLS 1.0. The attack exploited the TLS 1.0 implementation of the Cipher Block Chaining (CBC) encryption mode in order to be able to decrypt parts of TLS 1.0 data packets to reveal sensitive data such as website cookies that could be used to impersonate a user on a web application [1, 51]. Modification of padding, the primary exploitation vector, used in this attack is referred to by the term "padding oracle". The vulnerability was fixed in TLS 1.1.

Lucky 13 (2013)

The Lucky 13 attack is where a malicious attacker modifies, in transit, the padding (redundant data to make the data packet a certain size) of TLS data packets, that are using a CBC cipher suite, to analyse how the server reacts. Should the malicious actor be able to determine that the server has reacted to the modifications of padding, then this can lead to information leakage and plain text recovery [1, 52]. This attack is effective against TLS 1.0 - 1.3 and SSL 3.0 that use CBC mode (or any other mode that do not have padding oracle countermeasures.)

Poodle (2014)

The Poodle attack is similar to the Lucky 13 attack, where unprotected padding is exploited, but restricted to SSL 3.0. In order to force the use of SSL 3.0 the malicious attacker performed a downgrade attack which prevented a TLS handshake for any protocol version higher than SSL 3.0 to be established [1, 52]. The downgrade attack was possible as servers would fall back to SSL 3.0 should higher protocol versions fail.

Heartbleed (2014)

The 2014 Heartbleed attack, which gained attention in the mainstream media, exploited an implementation flaw in the length of a data bounds check. The flaw was in the rarely used but commonly enabled heartbeat protocol [53] of the TLS mechanism in the OpenSSL library (used by millions of servers).

To exploit the vulnerability, a malicious actor would send a `HeartbeatRequest` message, setting the payload length set to a length larger than the actual `HeartbeatRequest` message payload length [53]. This allowed the extraction of private memory on the target which could lead to the ex-filtration of sensitive data such as website private keys and cryptographic secrets.

The maintainers of the OpenSSL library released a fix for the vulnerability alongside the public announcement of the vulnerability [53].

FREAK (2015)

The 2015 FREAK attack shed light that the OpenSSL library would accept weak RSA encryption keys, from the use of export grade cipher suites, during a full-strength RSA TLS handshake. A malicious actor would force the client to use such a weak key by sending a `SeverKeyExchange` TLS protocol message to the client. If such a weak key was used, the malicious actor could capture the traffic, brute force the key within a matter of hours and decrypt the captured traffic. The removal of export grade cipher suites on the server stops this attack [1, 54].

Security Mechanism(s) Providing Mitigation

- TLS/SSL
- Strict Transport Security
- STS Preloading
- Public Key Pinning

Organisations should have policies in place to keep up to date with current state of TLS vulnerabilities and best practice recommendations to best mitigate risk to themselves and their users from malicious actors.

2.2 Summary

This chapter provided context for which attacks can be mitigated by the *selected security mechanisms*, as outlined in section 1.3. Real world examples of the attacks were used such that the reader can review them in more depth should that be of interest. Next in chapter 3 a critical review is performed of previous measurement studies leading to the measurement methodology used for this study.

Chapter 3

Data Acquisition

This chapter details the ethical considerations of measurement studies, critically analyses the related studies and details the methodology of this study to obtain the data in order to be able to analyse the adoption of the selected security mechanisms outlined in section 1.3.

3.1 Ethical Considerations

It is not feasible to request permission to conduct internet measurement scans from the desired targets. This means that researches should take it upon themselves to evaluate their methodology for ethical considerations in the absence of permission to conduct such research.

The ethical considerations as outlined in [34, 55, 56, 12] were considered for this research.

During the literature research, not all research that was of a similar nature contained ethical consideration statements. One would hope that ethical consideration was performed in the research, but should be outlined if only minimally.

The "BCS, The Chartered Institute For IT" code of conduct [57] outlines professional standards that should be met. This research was conducted with the same sentiment of these standards. Below are several of the most relevant excerpts taken from the document as presented below verbitem that represent to conduct of this research:

- *"NOT disclose or authorise to be disclosed, or use for personal gain or to benefit a third party, confidential information except with the permission of your Relevant*

Authority, or as required by Legislation."

- *"accept your personal duty to uphold the reputation of the profession and not take" any action which could bring the profession into disrepute.*
- *"only undertake to do work or provide a service that is within your professional competence"*
- *"have due regard for public health, privacy, security and wellbeing of others and the environment"*

3.1.1 Service Degradation

Using the "Tranco Top 1 Million Domains" list [58] which is a "most popular" type of domain list, similar to other lists as described in section 3.3.1, reduces the chance of service degradation as the more popular a domain is, the more resources it is likely has which reduces the impact of the measurement analysis.

For the purpose of obtaining the HTTP headers of a domain the "Home Page" of the domains are requested which are more likely to be cached than other pages.

To determine the TLS versions supported by a domain, other than that used on the HTTPS request to the home page, pure TLS connections are attempted which reduces the service impact as HTTP requests will add additional load to that of pure TLS connections. Additionally pure TLS connections are only attempted if an HTTPS connection was successful to a domain's home page.

Whilst obtaining the STS preload state of a domain, as detailed in section 3.4.5, it is first checked if the desired url has already been requested. During the processing of a scan, including redirects, it is quite possible that the information sought has already been obtained.

There is no retry logic for a request to a domain to additionally reduce service degradation of the target domain.

Randomisation is another potential tactic, however as the research uses a domain list rather than ip addresses. For randomisation to be effective the ip addresses would need to be pre resolved which is not being undertaken in this research.

3.1.2 Exploitation

The software used to conduct the research against domains does not attempt to:

- Send malformed requests
- Login
- Exploit vulnerabilities
- Access hidden/private paths/URLs

3.1.3 Information Disclosure

As this research obtains publicly available information and is not revealing vulnerabilities about any particular domain, there is not a concern of exposing information about domains that may reveal vulnerabilities.

3.1.4 Abuse Reports

As the domains being scanned have not given permission for research to be conducted upon them, the ip addresses that conduct the research have rDNS (reverse DNS) entries configured such that complaints/enquires can be submitted.

The cloud service providers for the servers that conduct the research have easy to use interfaces to allow responses to abuse reports should they be submitted.

The software that conducts the research was pre-equipped with a deny list for both ip addresses and domains. Both ip addresses and domains are required as an IP address can host more than one domain and a complaint may ask to block an IP address range.

There was a single abuse report, which came from the Cybersecurity and Infrastructure Security Agency (CISA) [59] USA government agency, which stated a request for "assistance in verifying possible malicious activity being hosted on a system registered to you". The reported domain and time of the request was checked against the scan logs and it was found that a scan request to the domain was at the time the abuse report stated. A response to the report was made stating that the request was made for the purpose of internet research.

3.2 Literature Review

Table 3.1 is a critical review of limitations identified from previous internet measurement studies found during the literature research and how this project aims to address them.

Table 3.1: Internet Measurement Studies Literature Review

Section	Limitation	Example(s)	How Addressed In This Project
Data Filtering	If data is filtered during data capture, it could result in unintended data being filtered	The researches in [10] filter headers during data capture, which if not done perfectly (e.g. spelling and case) could produce unintended results.	All HTTP headers, TLS certificates/connection information is captured unfiltered
www subdomain	If using only the base domains for HTTP(s) requests could produce inaccurate results as not all domains serve HTTP on the base domain. The www subdomain is the most common domain to serve HTTP requests besides the base domain.	Researchers [10, 34] do not state if the www subdomain was explicitly added during scanning.	The www subdomain is used if base domain does not have a DNS A record as shown in figure 3.3 / section 3.4.5.
HTTP method	Browsers use the HTTP GET method to obtain web pages. A website might behave differently if a non GET HTTP method (e.g. HEAD) was used. Some websites may not even allow the HTTP HEAD method.	Researchers [34] specifically use the HEAD HTTP method to reduce load on target domains.	The HTTP GET method (as detailed in the task flow stage 3 in section 3.4.5) is used which is what browsers use.

Continued on next page

Table 3.1 – *Continued from previous page*

Section	**Limitation**	**Example(s)**	**How Addressed In This Project**
Client Headers	Browsers send standard request headers on the requests to a websites home page. Including as many of these headers as possible when scanning domains, allows greater browser emulation. It is quite common when a non browser user agent header is detected a different response is sent from the website.	[13] uses 3 user-agents, [10, 34, 60, 18, 16, 11, 12, 15] use a single user agent. No other request headers are specified in the research.	Use chrome user agent along with other request headers as detailed in table 3.6.
SSL/TLS Verification	If SSL/TLS verification is done at capture time there is a possibility that the client/OS does not necessarily have access to all of the Root/Intermediate certificates that a browser would have.	None of the research [13, 10, 34, 60, 18, 61, 16, 11, 12, 62, 17, 15] processes SSL/TLS verification at the analysis stage. There could be many reasons for this such as it not being common knowledge of how browsers cache certificates and or assumed that not enough sites would be affected.	SSL/TLS certificate verification is performed at the analysis phase.
Monitoring	When a scanning session is conducted and not monitored, one or more sessions could fail without the researcher(s) being aware for an unknown amount of time.	It is possible that the outage mentioned in [18] could have been avoided or minimised by the use of effective monitoring.	Metrics of the status of scanning are created and alerted upon and investigated as detailed in section 3.4.6.

Continued on next page

Table 3.1 – *Continued from previous page*

Section	**Limitation**	**Example(s)**	**How Addressed In This Project**
Scanner Tool Verification	One cannot assume that the technology used to perform the scanning is doing what one expects or intends without due diligence being performed.	None of the research specifically calls out testing of the scanner with manual verification of results.	Successfully and unsuccessfully scanned domains were picked at random during the creation of the scanner, manually verifying results and correcting/updating scanner as required.
Scanner Tool Flow	The use of graphical flows allows greater accuracy for research replication and understanding.	Whilst most research gives a written overview of scanner methodology [10, 34, 13, 60, 18, 16, 11, 12, 17, 15], they vary in detail and do not have a graphical counterpart.	Figure 3.3 is a graphical representation with detailed descriptions to enhance the understanding of scanner methodology.

3.3 Data Acquisition

In order to be able to analyse HTTP Security Headers, TLS Versions, STS Preloading and security.txt adoption, or lack thereof, data on these desired metrics needs to be acquired. There are several overarching ways to acquire the desired data: Active, Passive and Third-Party data sets.

Active data collection is the act of establishing TLS connections and or making HTTP requests and storing the resultant connection/request/response data. This method requires the setup of infrastructure and software, which can involve the customisation and or development of entire software applications. Active scanning gives great versatility, however care needs to be taken to ensure the data is collected in a methodical and ideally a repeatable way, in order for the methodology to be able to be reproduced.

Passive data collection is capturing data as it flows through network infrastructure. The infrastructure required is likely to be less than that of active collection as it should only require the duplication of pre-existing network traffic to a device that is able to store the data. With any sort of data collection there is the always the ethical consideration to take into account, however with passive collection privacy is of higher concern than active data collection and needs to be treated as such.

Third Party data sets is the use of data collected by another entity that is using passive and or active collection. The only infrastructure that should be needed by the researcher would be that needed to analyse the data. There is great trust that the third party has conducted the data collection in a way that makes the analysis of it meaningful.

3.3.1 Scanning Targets

Active data collection, for the purpose of measuring security mechanism adoption, requires a list of entities to collect data from. Generating a list of domains to scan on ones own could be potentially quite an arduous task. Using resources that contain collections of domains is a much easier task, assuming the collection is of high relevance to the research that is to be conducted.

Pre-compiled Lists

Pre-compiled, often ranked by popularity, domain lists are made available by organisations that claim they are in a position to create such lists, such as: Alexa Top 1 Million sites [63], Cisco Umbrella [64], Majestic Million [65] and Tranco [58]. These lists are updated regularly, as often as once a day, which allows researchers to have up to date lists that are freely available.

The Alexa Top 1 Million site is a favourite among researchers as shown by use in [10, 11, 12, 13, 14, 15, 16, 17, 18]. These lists are primarily used as they are intended to reflect the most popular sites on the internet and thus are assumed to have a high amount of visitors making them great sources for research.

Zone Lists

Zone Lists, such as a country code top level domain (ccTLD) zone lists, which is a list of domains for a country, are another source of domains to scan. These are generally used when the researcher's intent is to gain insight into an entire country or continent such as the EU [34, 11, 16, 17].

The majority of zone lists are not openly available however ICANN has created the "Centralised Zone Data Service" (CZDS) [66] which allows one to register and request access to Zone Files.

The number of domains to scan using zone lists can be in the 10s or even 100s of millions of domains which would likely require more infrastructure to scan on a regular basis compared to the use of pre-compiled lists.

Search Engines

Utilising search engines can allow additional URLs for a domain to be acquired to allow for further insight how responses might change depending on which web page of a domain is requested [11].

IPv4 Address Space

Scanning the entire IPv4 Address space (i.e. all internet IPv4 addresses) is also a source of entities to scan [60]. The major difference is the use of an IP address rather than a domain which needs to take into account that servers may respond differently if an IP address is used as the website address rather than a domain name (e.g. `https://1.2.3.4` vs `https://example.com`).

The number of internet routable IP Addresses, i.e. ip addresses that can be access over the internet, is around 2^{32} total possible addresses minus approximately 288 million non route-able addresses which results in approximately 4 billion addresses.

Scanning the entire IP address range can lead to abuse reports, however these can often be revoked if the researcher works with the operator who raised the complaint to conduct the scan in a manner in which the operator deems acceptable.

3.3.2 Scanning

Scanner Technologies

There are quite a number of scanning technologies used in measurement research and as such there does not seem to be a proffered technology overall but what the researcher(s) are comfortable with. This does makes sense in regards to if a particular technology has worked for researcher in a previous study then using it in a future study, where applicable, as they are familiar with its capabilities giving more time for other activities.

Table 3.2 details a number of measurement technologies found to be used in previous measurement studies during the literature research.

Technology	Description	Use Case
PHP with Curl [10]	PHP is a web programming language. Curl provides a client library for communicating to web services such as HTTP web servers.	Make HTTP calls to websites and capture the response body and metadata.
Goscanner [34]	Tool for large scale TLS and SSH scans.	Obtain TLS connection information.
Bro [34, 60]	A network security monitor.	Passive network packet capture for TLS connection information.
PhantomJS [11, 16]	A headless scriptable browser.	A browser that can be easily used by JavaScript scripts for obtaining the body of a website.
Chrome [12]	A modern desktop website browser.	In headless mode, can be used to capture a page rendered as an end user would see it.
Scrapy [13]	A scriptable tool for extracting data from websites.	Extract HTTP headers from response to a request made to a website.
Java with Apache [14]	Java is a programming language, Apache provides a HTTP client for Java.	Make HTTP calls to websites and capture the response body and metadata.
Zmap / Zgrab [15, 17, 60]	Zmap is a single packet network scanner. ZGrab is an application layer scanner.	Zmap can be used to identify IP address with open ports. ZGrab could then be use to capture the body and metadata from HTTP requests to the open port(s).

Table 3.2: Technologies used in previous measurement studies

Domain Resolving

When a HTTP client makes a request to a URL (e.g. http://example.com) the client needs to lookup the IP address of the domain (e.g. example.com) in order to send the HTTP request.

Scanning a large number of domains can potentially put a strain on the DNS server the client is using to resolve the domain to an IP address. If there are a sufficiently large number of domains to be resolved at the same time, or over a very short period of time (e.g. a few seconds) it is possible that the DNS resolution will fail when there is an IP address that cannot be resolved. There are several ways around this such as pre determining the IP address for the domain [34], using more than DNS server and using retries.

Pre-computing DNS resolutions makes more sense when not following HTTP redirects such as for the TLS connection establishment in [34, 17].

Using the WWW subdomain

When using lists that contain base domains (e.g. example.com) it is not known if the website is actually hosted or available from the base domain. Websites are generally available from the base domain (e.g. example.com) and or its www subdomain (e.g. www.example.com).

To give a good chance of being able to obtain a HTTP response from a domain, it is common to try both the base domain and the www subdomain [11, 12, 14, 15]. Some researchers choose not to also try the www subdomain such as in [10, 34] (but do not necessarily state why they chose not to do so), which can lead to unnecessarily reduced data for analysis.

HTTP Method

In order to obtain HTTP Security Headers, one needs to make an HTTP Request. There are several types of HTTP Request, GET and HEAD being two of them.

HTTP HEAD requests result in only metadata being returned (this includes HTTP Headers) by the server. The main reason researchers choose to use HEAD HTTP Requests [34] is to save compute resources on the server (as the server does not have to generate the response payload such as a html web page) as well as reducing the amount of data being sent over the internet.

Whilst using HEAD requests does save server resources, which helps in the ethics of internet research of this kind, browsers perform GET requests which could potentially respond with different headers for the same URL. If the researcher is intending on analysing the response payload from a website, in the same way a browser would, then GET requests [11, 12] should be used.

Certain applications and programming languages, such as go [67], allow the use of GET request without requesting the actual payload from the server which leaves just the computing resource to generate the payload. If the home page is requested, this is more likely to be cached, as it will be one of the most visited pages which also helps to reduce the compute load on the target resource (e.g. website).

HTTP Client Headers

When browsers make HTTP requests to websites, they send several HTTP headers with the request. One of these headers is named `user-agent` and is often used to detect if the client making the request is a browser. When a website detects that a non-browser is making the request it can change the response which could skew the results of research such as analysing HTTP security header adoption as would be seen by a web browser.

For the purposes of internet research when attempting to gather data from responses to HTTP requests that would be sent to a browser, the use of a browser user-agent is typically used [10, 13, 14, 68, 18]. The declaration from researchers of the value for such headers is not always present [34] which could leave readers potentially questioning the reliability of the resultant analysis.

Redirects

When a HTTP request is made, the HTTP protocol has the ability to redirect to a different url. This is quite normal for websites to do this, and there are numerous reasons to do so, such as redirecting from the base domain e.g. (http://example.com) to the www subdomain e.g. (http://www.example.com) where the website is actually hosted. It is generally good practice to follow redirects as this is what browsers do, unless researchers wish to only analyse a response from a specific url such as the base domain, such as in [34].

Researchers need to be mindful however that redirects can lead to the same resultant domain for different initial domains (e.g. http://example.com and http://mydomain.com could both redirect to http://anotherdomain.com).

This phenomenon can potentially skew analysis results if the researchers do not actively take this into account such as duplicate resultant URLs being removed as was done in [68].

SSL/TLS Certificate Validation

When a TLS connection is being established a browser will try and verify the claimed identity of the website (e.g. example.com). If the server does not present all the necessary intermediate certificates, in order to be able to validate the websites claimed identity, the TLS connection will fail unless the browser already has the intermediate certificate available.

Intermediate certificates can be obtained from such means as: the website being visited, the certificate was obtained from a previously visited website, via Authority Information Access (AIA) [69] or from preloading [70]. For further details on PKI, including certificates and identity verification, please see [71, 72].

It is quite common to use non browser clients to conduct research. These clients are not guaranteed to use any of the above mentioned methods for obtaining intermediate certificates not provided by a website in order to validate the websites' claimed identity. To counteract this issue, the TLS/SSL validation can be disabled for the scanning of the website and done as an offline step which enables any missing certificates to be obtained at the researcher's leisure.

It is not evident that any of the papers in the literature research conducted for this project used such an approach, possibly assuming that the number of websites that would have this issue would be minimal enough not to affect the results of analysis conducted, or not aware of the issue itself.

Pre-Analysis Filtering

It is generally best to collect the rawest form of data for to give the greatest possible freedom and options when conducting any analysis. If filtering was carried out at the data acquisition phase this could restrict further analysis or potentially cause intended analysis not to be possible due to the way the data was filtered.

A potential example of this is where in [10] the researches decided to parse the headers of a HTTP response during the data acquisition phase. HTTP headers are caseless [4] and depending on how the filtering is done, some headers could have been missed, however if all the headers were captured this could have been a non issue during analysis.

3.3.3 Scanning Frequency

The number of times data gathered and how far apart they are, in terms of time, are important in terms of showing any potential trends and having enough data to confidently make statements from the analysis of such data.

Of the research sources that were reviewed for this project, those that conducted active scanning activities make at least two scans such as [10, 34, 11, 12] and some with daily scans for years [17].

The more data that is gathered in the same way, the higher value that analysis obtains, assuming that the methodology of the acquisition is sound.

3.3.4 Monitoring

Conducting long running data acquisition would greatly benefit from, at minimum, some basic metrics that show that a data acquisition has started, completed and if a critical issue was encountered. This would help to ensure that data acquisition was successful and to alert if not, so that it could be investigated and rectified to reduce any potential issues to the research being conducted.

The researchers in [18] stated that they had a "measurement outage" however they do not go into detail. It is possible that monitoring could have helped to reduce and or mitigate the outage.

3.3.5 Detailed Methodologies

The majority of research reviewed for this project (outlined in section 3.2) state the technologies used and the high level description of the methodology used to acquire the data later analysed, such as in [34, 11, 16].

Whilst this is generally enough for the reader to understand the high-level methodology, those who wish or are conducting similar or related research could benefit form a more detailed and formal method. There are several strategies presently available for researchers to utilise.

"Strategies for Sound Internet Measurement" [73] is one such paper with specific relevance to measurement research which is very relevant to the research for this report. The paper outlines the inherent difficulties and pitfalls one can encounter focusing on: *Precision*, *Meta-data*, *Accuracy*, *Misconception*, *Calibration*, *Data Volume* and *Reproduceability.*

3.4 Methodology

This section details the methodology this project used to obtain the data analysed in chapter 5.

3.4.1 Requirements

A scanner is to obtain the following information from a list of domains:

- If the domain redirects from `http://` to `https://`.
- If the domain redirects from `https://` to `http://`.
- The TLS version(s) the domain supports.
- The TLS version auto negotiated against the domain when making a HTTPS request.
- If the domain meets the STS preload requirements.
- The HTTP headers, including their values, in use by the base domain (or the www subdomain if the base domain does not have a `DNS A record` (IPv4 IP address)).
- The content of the security.txt of the domain.

3.4.2 High Level Design

This section covers the high-level design, represented in Figure 3.1, for the scanning of domains to provide the data required as stated in section 3.4.1.

Existing technologies were evaluated, however a tool that met all the data capture requirements was not identified. In particular obtaining if the STS Preload requirements were met was not found to be in a current tool.

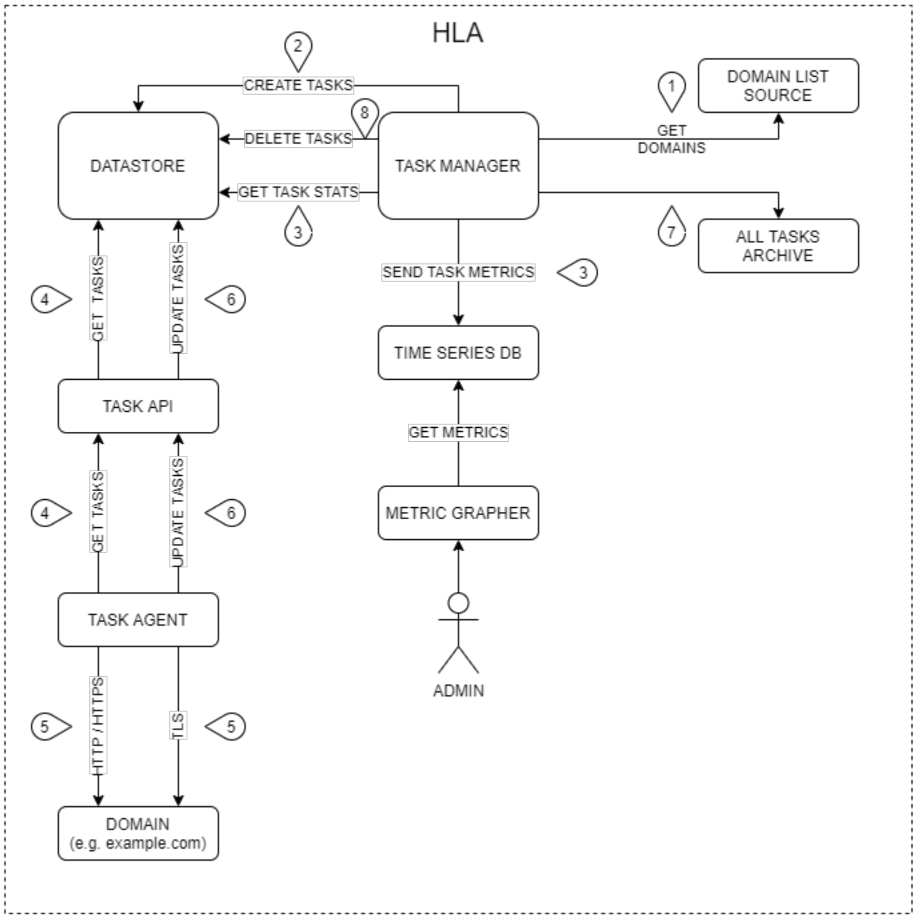


Figure 3.1: High Level Design

#	**Description**
1	A list of domains will be acquired to scan against.
2	For each of the domains in the domain list, a task manager will create a "task", to be stored in a database, representing a domain to be scanned.
3	Metrics on the state of tasks will be sent to a time series database.
4	A task agent will poll an API for a new batch of tasks.
5	A task agent will scan a domain as stated in the task being processed.
6	A task agent will send the result of a domain scan to an API to be stored in a database.
7	A task manager, once all tasks are completed, will extract all tasks from the database to an archive file format.
8	A task manager, once all tasks are completed and archived, will delete all the tasks from the database.

Table 3.3: Descriptions for Figure 3.1

3.4.3 Implementation

This section describes the chosen technologies and resources for the implementation, shown in Figure 3.2, of the design in section 3.4.2.

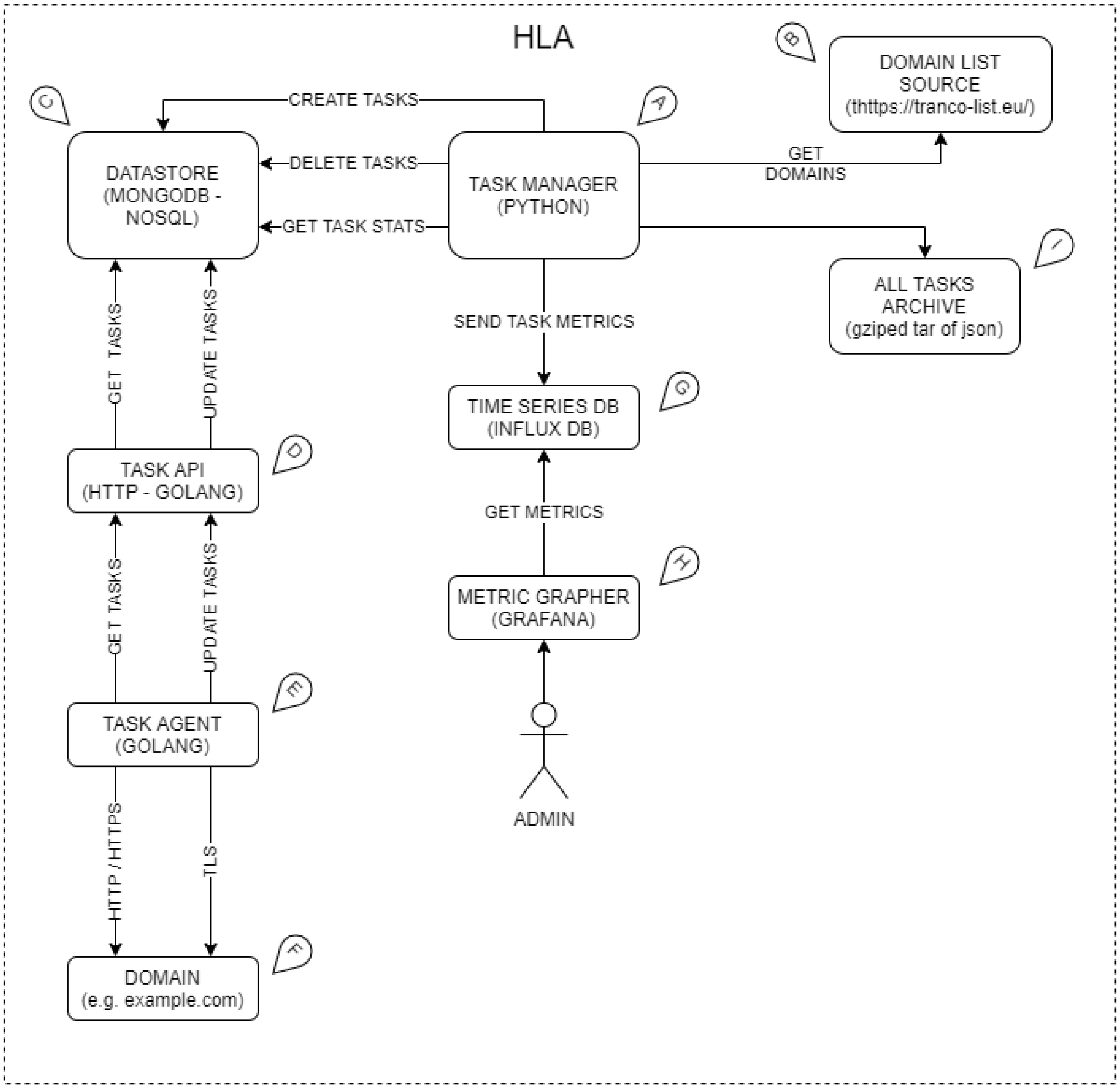


Figure 3.2: High Level Design - Implementation

[A] Task Manager

The Task Manager performs the following tasks:

- Obtains the list of domains from the “Tranco” domain list source (`https://tranco-list.eu/top-1m.csv.zip`).
- Create tasks directly in the Datastore (MongoDB).
- Creates an archive of the tasks via the Task API.
- Delete tasks from the MongoDB database.
- Identify and reset zombie tasks (tasks that have been marked as started but not complete after a certain amount of time).
- Generate task metrics and store them directly in the time series DB (Influx DB).

The task manager was developed in python as it:

- Is a relatively high-level programming language which aids in being easier to learn than a more low-level language.
- Has many libraries and resources freely available.
- There is a large global community generally willing to help each other.
- Being such a popular language, issues that one may come across may have already been encountered and solved already.
- Is cross platform, i.e. will run on any computer that has a python interpretor available to it.

[B] Tranco Domains Source List

As previously mentioned, similar research to this project have used pre-compiled lists, predominantly using the Alex Top Sites list [10, 11, 12, 13, 14, 15, 16, 17, 18].

The research conducted by the Tranco team [74], who provide the Tranco domain list [58], compared several of the “most popular domains” lists and identified that domains present in the list can vary wildly on a day to day basis and are susceptible to influence by malicious actors.

As a result of this paper and that the Tranco list uses other top site lists to influence their list, the Tranco list [58] was chosen as the source for the domains to scan for this project.

[C] Datastore

The database is intended to only be an ephemeral store as it will only store the tasks until they have all been completed.

The data that will be returned by the task agent will be of arbitrary length and structure. A relational database is not the optimal choice as it is based on having one or more "tables" that relate to one another with a fixed defined structure.

A NOSQL database is designed to store arbitrary length and a non pre-defined content structured data.

The MongoDB database technology was chosen as it:

- Is a NOSQL database.
- Is relatively mature technology implementation.
- Has a community version (i.e. free).
- Many programming languages have pre-existing support (libraries).
- Has native support for unstructured data.
- Has detailed online documentation.

[D] Task API

The Task API is meant purely as a common interface for the task agent such that the task database can be changed without the need to modify the task agent.

The HTTP protocol was chosen as the interface to the API as it is a very well known and a standard protocol to use for an API.

The intended clients of the API is the Task Agent (for running and updating the tasks) and the Task Manager (for creating an offline archive once all tasks have completed for later analysis).

The Task API can perform the following functions:

- Allow a client to retrieve one or more tasks via a HTTP GET request.
- Allow the following task database filtering via url arguments when a HTTP GET request is made:

  - Task Status (e.g completed).
  - Task Unique ID.
  - Sort by Task Unique ID.

- Allow a client to update one or more tasks via a HTTP POST request.

GO (also referred to as Golang) was chosen as the programming language to implement the API as it:

- Is a relatively low language which helps with improving performance.
- Is one of the easier lower-level languages to learn as it is one of the GO core principles to be able to pickup up the language quickly.
- Has the ability to run tasks concurrently (the more domains scanned concurrently the faster the scan of all domains can be completed).
- There are several community modules for GO to communicate with MongoDB (the chosen database for this project).
- Has numerous community modules for GO for the purpose of creating a HTTP API.

[E] Task Agent

The task agent obtains tasks from the Task API. A task contains the required information to allow the task agent to conduct scans against a domain.

Table 3.4 details the parameters of a Task Agent.

Parameter	Data Type	Value(s)	Description
max HTTP redirects	int	10 (Golang default)	The number of HTTP redirects allowed
max_concurrent_tasks	int	100	The maximum number of tasks that can be running concurrently
deny_domains	array of strings	Domains not to be scanned	domains not to be scanned
deny_ips	array of strings	IPv4 addresses not to be scanned	IPv4 addresses not to be scanned
dns_lookup_nameservers	array of strings	“open” recursive DNS servers	A list of recursive DNS servers

Table 3.4: Task Agent Parameters

Section 3.4.4 details the parameters of a task.
Section 3.4.5 details the high level flow for the process of performing a domain scanning Task.

Appendix A details TLS client capabilities of the Task Agent.

As with the Task API the Task Agent will use the programming language GO.

[F] Domain

This is the target domain of a task being performed by the Task Agent.

[G] Time Series Database

The Task Manager will create metrics about tasks and send them to the time series database. Time series databases are optimised for metrics that are time bound which is the type of metrics that is required to be stored.

InfluxDB was the time series database technology implementation chosen as it is is relativity modern, quick to get up and running and has a community version available (free).

[H] Metric Grapher

A metric grapher allows the user to create charts from metrics. In the case of this project a metric grapher was used to generate basic graphs about the state of tasks for monitoring purposes from the metrics stored in the InfluxDB time series database.

Grafana was chosen as it is free, has built in support for communicating with InfluxDB, has an HTTP interface and offers the ability to create basic alerts.

Alerts were configured in Grafana to fire under the following conditions:

- If there were zombie tasks (tasks that have been marked as started but not complete after a certain amount of time)..
- If the number of tasks being processed dropped below a certain value over a specific amount of time for a task agent, if there were tasks waiting to be processed.

[I] All Tasks Archive

Once all tasks have been processed, the Task Manager will use the Task API to create a compressed archive file containing a JSON structured representation for each task.

This compressed archive is transferred to an archive file server for later analysis.

3.4.4 Task Parameters

A task has up to two parameters as shown in table 3.5.

Name	Data Type	Mandatory	Description
fqdn	string	Yes	A domain from the domain list provided by tranco-list.eu
headers	string	No	A JSON string of one or more HTTP client headers and values

Table 3.5: Task Parameters

All tasks will be using the same request headers as shown in table 3.6. The header names and values were those present in the HTTP request from visiting the url `https://www.google.com` and were captured using chrome's browser developer tools.

Name	Value	Description
accept	text/html, application/xhtml+xml, application/xml; q=0.9,image/webp, image/apng, */*;q=0.8, application/signed-exchange; v=b3;q=0.9	The accepted content types e.g html page.
accept-encoding	gzip, deflate, br	The accepted content encoding(s) e.g. gzip
accept-language	en-US,en;q=0.9	The accepted languages e.g. en-US
sec-fetch-dest	document	Requested destination e.g. document
sec-fetch-mode	navigate	Differentiator for request type e.g. navigation or websocket
sec-fetch-site	none	If the request is coming from the same origin e.g. none (user-originated operation)
sec-fetch-user	?1	Sent for requests initiated by user activation
upgrade-insecure-requests	1	Signal to the server that an authenticated and encrypted response is preferred e.g. redirect to secure version of the site
user-agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.101 Safari/537.36	Identification of requests application e.g. browser identification

Table 3.6: Task Parameters - Headers

This was done to best represent the standard request headers a browser would send to a domain to have the best representation of response headers a user browsing the internet, would receive from a domain.

All the headers in table 3.6 where carefully checked as to their purpose and if they were suitable to be included in order to obtain the best response from domains for the most relevant data to analyse from the perspective of a generic user browsing the internet.

The `user-agent` header is generally considered one of the most critical headers to use for HTTP requests to be associated with coming from a browser, rather than a backend service application, as shown by almost all related research using this header [13, 10, 34, 60, 18, 16, 11, 12, 15].

The headers with the prefix `sec` are to enhance the premise that the request is coming from a user making the request manually in a browser. It could be argued that the `accept` headers are not needed however they should only give credit that the request is from a user rather than a service.

3.4.5 Task Flow

This section describes the high level task flow once a task has been obtained from the Task API and is now to be processed.

The following describes the stages that task processing goes through if the task parameter `fqdn` had a value of `example.com`. This is shown graphically in figure 3.3.

Stage 0 - Task Parameters

The task parameters are checked for validity in the following way:

- If the parameter name is not valid for the task, abort the task
- If a mandatory parameter is not present, abort the task

Stage 1 - Deny List

The `fqdn` parameter value `example.com` is checked to see if it is on the task agents deny list, and if present the task is aborted and no further attempts will be made to process the task.

Stage 2 - Resolve Domain

DNS lookups for Stage 2 uses a round robin methodology with the DNS servers specified in the Task Agent parameter dns_lookup_nameservers, to reduce the load on any one DNS resolver.

A DNS request to lookup the `A records` (IPv4 addresses) of the fqdn parameter `example.com` is made.

If IPv4 addresses are returned, `example.com` is now referred to as the resolved domain resulting in Stage 2 being complete and processing moves to Stage 3.

If no IPv4 addresses are returned, a further DNS request to lookup the `A records` (IPv4 addresses) of the `www` subdomain of the fqdn parameter i.e. `www.example.com`.

If the DNS request made to `www.example.com` returns no IPv4 addresses, the task is aborted and no further attempts will be made to process the task.

If the DNS request made to `www.example.com` returns IPv4 addresses, `www.example.com` is now referred to as the resolved domain resulting in Stage 2 being complete and processing moves to Stage 3.

NOTE: For the purpose of describing the task flow, for the further stages, the resolved domain shall from here on out be assumed to have been determined to be `www.example.com` for the `fqdn` task parameter of `example.com`.

Stage 3 - HTTP(S) Request

A HTTP GET request is made to the resolved domain `http://www.example.com`.

Following redirects, if the last url in the redirect chain starts with
`https://` (e.g. `https://www.example.com/index.php`) Stage 3 is deemed complete and processing moves to Stage 4.

If the previous request did not result in a `https://` url a HTTPS request is made to the resolvable domain `https://www.example.com`.

Following redirects, if the last url in the redirect chain starts with `https://` (e.g. `https://www.example.com/index.php`) Stage 3 is deemed complete and processing moves to Stage 4.

If the first request did result in a `http://` url (e.g. `http://www.example.com/index.php`), i.e. did not ultimately get redirected to a `https://` url, Stage 3 is deemed complete and processing moves to Stage 6 (skipping Stages 4 and 5 as they are only performed if a request ends in a HTTPS url).

If no `http://` or `https://` response is obtained from the resolved domain, Stage 3 is deemed to have failed and the task is aborted and no further attempts will be made to process the task.

Stage 4 - TLS Versions

TLS 1.0, 1.1, 1.2 and 1.3 connections are attempted against the domain of the last url in the redirect chain (e.g. domain `www.example.com` from last redirect url `http://www.example.com/index.php`) from Stage 3.

The TLS version used for the HTTPS connection in Stage 3 is skipped as it has already been attempted and was successful.

Stage 4 is deemed complete and processing moves to Stage 5.

Stage 5 - STS Preload

This stage gathers the required data, such that a determination can be made at a later stage, if the domain of last url in the redirect chain (e.g. `http://www.example.com/index.php`) from Stage 3 meets the STS Preload requirements.

The STS Preload requirements are detailed in section 4.6.

The following actions are performed on the base domain of the last url in the redirect chain from Step 3 and the results of which are recorded. The base domain of the url is determined by the use of the Golang library "Golang.org/x/net/publicsuffix". If any of the actions have already been performed in section 3, they are skipped as the results have already been captured:

- If the base domain (e.g. `example.com`) is running a HTTP server, on port 80, make a HTTP request to `example.com` i.e. `http://example.com`
- If the `www` subdomain has a DNS (A record) try and establish a TLS connection to the `www` subdomain `www.example.com`
- Make a HTTPS call to the base domain `https://example.com` and capture the HTTP headers

Stage 6 - security.txt

The path of the last url in the redirect chain (e.g. `http://www.example.com/index.php`) from Stage 3 is replaced with `.well-known/security.txt` i.e. `http://www.example.com/.well-known/security.txt`.

A call is made to this url and if the `Content-Type` header is present with the value `text/plain` the body of the response is captured.

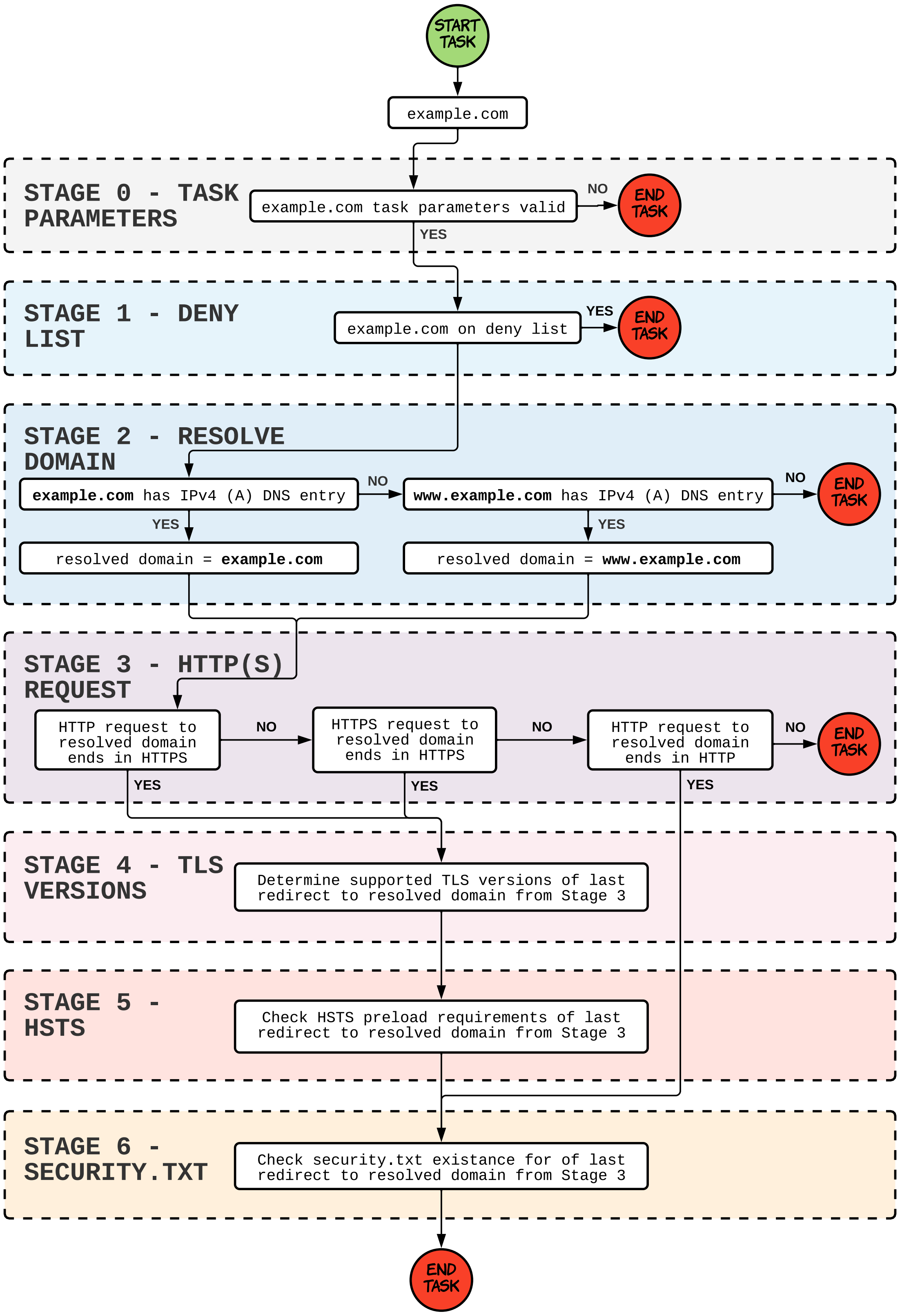

START TASK
example.com
STAGE 0 - TASK PARAMETERS
example.com task parameters valid
NO
END TASK
YES
STAGE 1 - DENY LIST
example.com on deny list
YES
END TASK
STAGE 2 - RESOLVE DOMAIN
example.com has IPv4 (A) DNS entry
NO
www.example.com has IPv4 (A) DNS entry
NO
END TASK
YES
YES
resolved domain = example.com
resolved domain = www.example.com
STAGE 3 - HTTP(S) REQUEST
HTTP request to resolved domain ends in HTTPS
NO
HTTPS request to resolved domain ends in HTTPS
NO
HTTP request to resolved domain ends in HTTP
NO
END TASK
YES
YES
YES
STAGE 4 - TLS VERSIONS
Determine supported TLS versions of last redirect to resolved domain from Stage 3
STAGE 5 - HSTS
Check HSTS preload requirements of last redirect to resolved domain from Stage 3
STAGE 6 - SECURITY.TXT
Check security.txt existance for of last redirect to resolved domain from Stage 3
END TASK

Figure 3.3: High Level Task Flow

3.4.6 Deployment

This section details how the implementation in section 3.4.3 was deployed.

The components as outlined in Section 3.4.3 were deployed using cloud infrastructure as detailed in Table 3.7.

Component(s)	**OS**	**# vCPU**	**RAM**	**SSD**	**Qty**
MongoDB	FreeBSD 12.2	2	2GB	25GB SSD	1
InfluxDB+Grafana	FreeBSD 12.2	1	1GB	25GB SSD	1
Task API	FreeBSD 12.2	1	1GB	25GB SSD	1
Task Agent	FreeBSD 13.0	1	512MB	60GB SSD	5

Table 3.7: Cloud Infrastructure Deployment

FreeBSD 13.0 was used for the task agents as it was the most recent version available at the time of deployment. There were no performance or security issues identified that required the effort to upgrade the components that remained on FreeBSD 12.2.

The number of Task Agents deployed was 5, such that the scanning could be conducted well within 24 hours to allow a reasonable amount of time to rectify issues should they occur. The average time for all 1 million domains to be scanned is approximately 11 hours.

Monitoring

A dashboard was created in Grafana to visualise the progress/status of scans. Over time several alerts were configured after issues were encountered as detailed in table 3.8. Figure 3.4 shows a screen shot of the dashboard representing a time window of 7 days.

There are several monitoring charts as detailed below:

- The Tasks Status By Agent chart displays the number of tasks of each task status' for a specific task agent. The "Agent" drop down box allows the user to select the task agent to be shown.
- The *In Progress Tasks* / *Completed Tasks* / *Ready* / *Error Cannot Retry* / *Pending Ready* charts show the number of tasks, per task agent, of the charts title task status.
- Ready Delta - This is to monitor and alert for task agents that do not continually take new tasks if there are pending tasks.
- The Tasks in Zombie State is to monitor and alert for tasks that are in the *In Progress* state but have not reached *Completed* or *Error Cannot Retry* status after a certain amount of time.

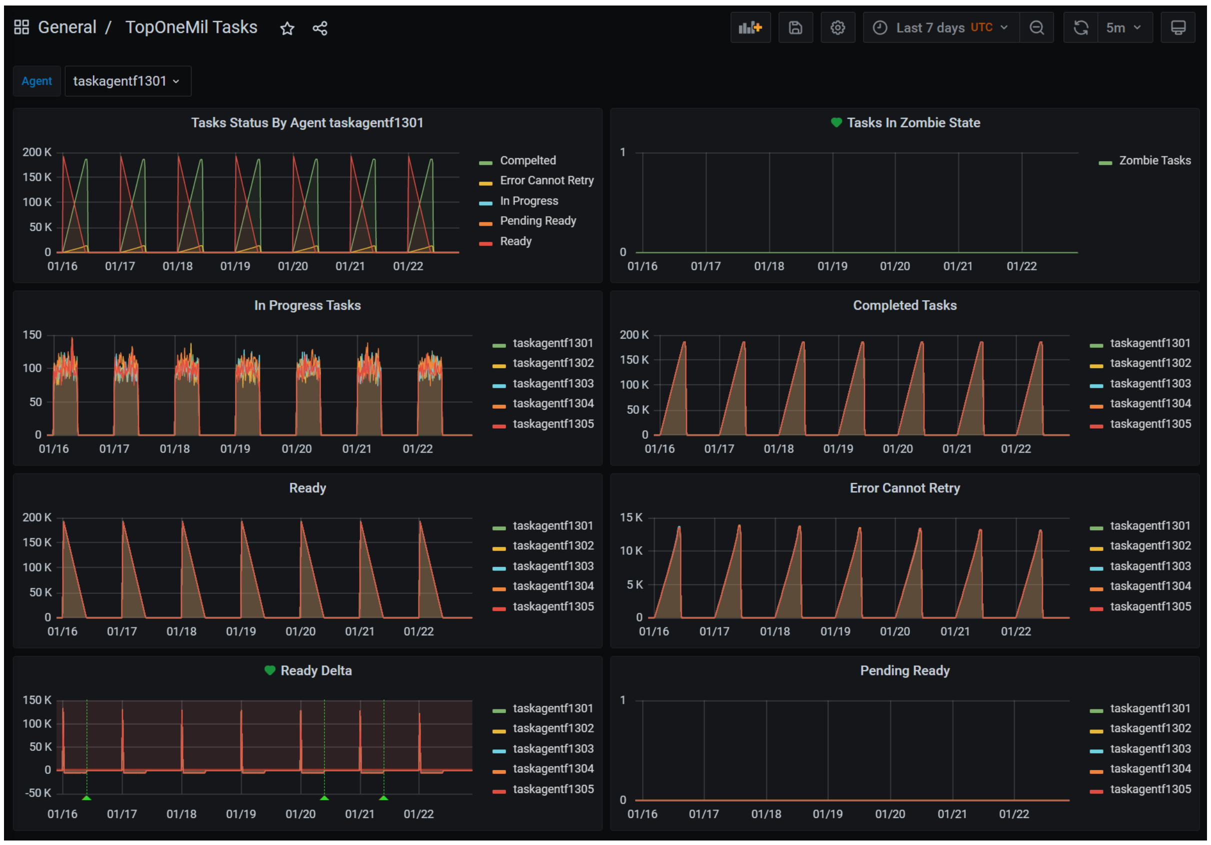


Figure 3.4: Grafana Dashboard

Automation

The Task Manager is scheduled to obtain the latest Tranco domain list and create tasks to scan each day at 0000 UTC. This will only occur if the previous day was completed successfully.

The Task Manager is scheduled to check if all the tasks have been completed between 1000 UTC to 2300 UTC. Once all tasks have been completed, they are collected to create the "All Tasks Archive" and then the tasks are deleted from the MonogDB datastore. Each "All Tasks Archive" is approximately 4GB in size.

A Task Agent continually look for new tasks to perform and performs up to 100 concurrent tasks.

Issues Encountered

Zombie Tasks

A zombie task is one where it is started but does not complete within a set amount of time. For this research the duration limit for a task deemed to be a zombie task is 120 minutes (2 hours). This occurred less than one task per 10 million tasks performed.

A scan outage for a duration of 27 days was caused by zombie tasks. A single task not completing will prevent the archive create process from starting. As a result changes were made such that when a low amount of zombie tasks were detected they were reset and re-attempted. All reset tasks completed successfully. If there were several zombie tasks detected this would raise alert and human intervention would be required to resolve the situation.

It was not looked into why zombie tasks occurred as the rate was so low it was not an efficient use of time and would not affect the research results as the tasks were successfully re-attempted.

Data Loss

An unexpected permanent loss of access to the storage location where the "All Tasks Archives" were stored resulted in the permanent loss of 17 days worth of scan archives. Since this incident, scan archives are stored in more than one location within several hours of a scan archive being created.

Programming

Golang has, as of November 2021, a bug [75] where when a non standard HTTP code is received, it will cause Golang to panic and the program will terminate. There was a single domain that caused this issue and it was put in the deny list to overcome this issue.

Development Work

An 8 day scan outage was the result of development tasks being created in the same database as the production tasks. The archive creation process will not initiate unless there are the correct amount of tasks present. A procedure was put in place to remove development tasks after use.

Missing Scans Overview

Table 3.8 summarises the extent of missing scan archives.

Start Date	Duration	Cause	Action Taken
24 Jan 2021	8 Days	Development tasks not removed after use which prevented scan archive creation	Added procedure to remove development tasks to mitigate further occurrences
16 March 2021	12 Days	Bug in archive creation process	Fixed bug and deployed updated scripts
31 July 2021	27 Days	One or more tasks did not complete preventing future scans to start	Enhanced alerting to detect long running tasks
17 September 2021	17 Days	Archive storage server permanently unavailable	Scan archives stored in multiple locations

Table 3.8: Missing Scan Archives

3.5 Summary

This chapter looked at previous research related to measurement studies and walked through the methodology that was used to obtain the required data for the selected security mechanism the adoption of which is to be analysed later in chapter 5. Next, in chapter 4 a more detailed look at the selected security mechanisms which were introduced in the opening chapter in section 1.3.

Chapter 4

Security Mechanism Overview

This chapter is a detailed overview of the selected security mechanisms from section 1.3 to said in providing context in understanding the analysis in chapter 5.

4.1 HTTPS

This mechanism is analysed in section 5.2.

The S in HTTPS signifies that the HTTP protocol communication shall be over a secure channel, provided by the SSL/TLS protocols [19]. A URL starting with `https://` e.g. `https://example.com` signifies it will be using a SSL/TLS secure channel.

It has become quite common for websites to automatically redirect a user to a HTTPS url if the user enters or follows a HTTP url e.g. the URL `http://example.com` redirecting to `https://example.com`. This helps to enable the use of the HTTP protocol over a secure channel without the user having to actively seek out the HTTPS URL of a website (which most users are unlikely to do).

4.2 TLS

This mechanism is analysed in section 5.3.

The SSL (Secure Socket Layer)/TLS (Transport Layer Security) protocols were designed to establish a secure communications channel between two entities e.g. a web browser and a website. Protocols such as HTTP can utilise SSL/TLS to secure its communication.

With each new version of SSL/TLS more security features were added in response to attacks against the protocols, weakness identified and the desire to make the protocols more secure.

The acronyms SSL and TLS are often used interchangeably to refer to the secure channel they provide, even though none of the SSL versions should be used any more due to the vulnerabilities and weaknesses of SSL 2.0 and SSL 3.0 [6].

4.2.1 Security Services

The SSL/TLS mechanism provides several security services, as of the release of TLS 1.3 the main categories are: Confidentiality, Data Origin Authentication, Entity Authentication and Perfect Forward Secrecy.

Confidentiality

Confidentiality is the protection of data such that only those who have been given authorisation can access the data.

Two entities should be able to establish a SSL/TLS connection between one another such that only these two entities can access the plain text data being sent over it. This should result in the prevention of any third parties that are able capture the traffic from being able to recover the plain text data being sent over the SSL/TLS channel [76].

Data origin authentication

The purpose of Data origin authentication is to establish the integrity of the SSL/TLS channel data such that any modifications can be detected. This is accomplished using message authentication codes (MAC) or authenticated encryption with associated data (AEAD) algorithms [1] which use keys derived from when the SSL/TLS connection was established.

Entity Authentication

Entity Authentication provides the ability for an entity to prove their identity using mechanisms such as digital signatures.

In SSL/TLS entity authentication is mandatory and used to show that during the establishment of a SSL/TLS channel the client can verify that it really is communicating directly with the website/server they intended to. This is accomplished via the use of PKI (Public Key Infrastructure) with X.509 certificates and root stores. For further details on PKI please see [71, 72].

It is optional for a client to prove its identity to the server using PKI during the establishment of the SSL/TLS channel.

Perfect forward secrecy

Perfect forward secrecy ensures that if the private key that is paired with the certificate of a server/website was compromised, any SSL/TLS traffic that was captured being sent to or from this server could not be decrypted with the private key being revealed. This is accomplished via the use of Ephemeral Diffie-Hellman key exchange [76].

Perfect forward secrecy is mandatory as of TLS 1.3 and as such RSA key exchange was removed in TLS 1.3 due to its reliance on the certificates' private key.

SSL/TLS Versions

SSL 2.0 [1994]

Netscape Communications started developing a protocol, named Secure Socket Layer (SSL), for securing HTTP communications in 1993 [6], due to the fact that when HTTP was first introduced it did not provide a means by which to secure its communications [6]. Netscape Communications introduced SSL 2.0 in 1994 [6, 7].

SSL 3.0 [1995]

SSL 3.0 was introduced 1995 [1, 6] and was a re-write of the SSL protocol, rather than additions/modifications, as many vulnerabilities and weaknesses had been identified in SSL 2.0 [1, 77].

SSL 3.0 was eventually standardised as RFC6101 [78].

TLS 1.0 [1999]

In 1996 the IETF Transport Layer Security (TLS) Working Group was established [6, 79] which produced TLS 1.0 as RFC4436 in 1999 [80].

TLS 1.0 was based on SSL 3.0 and introduced some enhancements and modifications [81]. Due to the changes made in TLS 1.0, it was allowed to be used by the US government as it gained FIPS approval [1].

TLS 1.1 [2006]
TLS 1.1 was released as RFC4346 in 2006 [82] which included a number of changes from TLS 1.0 some of which are:

- CBC (block mode) has to use explicit IVs which addressed the predictable IV weakness [1].
- bad_record_macalert required to be used in the response when there are padding problems to protect against padding attacks [1].
- The addition of TLS extensions [1] as described in RFC3546 [83]

TSL 1.2 [2008]
TLS 1.2 was released as RFC5246 in 2008 [84] which included a number of changes from TLS 1.1 some of which are:

- IDEA and DES cipher suites were removed
- Authenticated encryption was added
- The extension signature_algorithms was added

TLS 1.3 [2018]
TLS 1.3 was released as RFC8446 in 2018 [8] which is currently the latest published, as an RFC, TLS version as of 2022 which included a number of changes from TLS 1.2 some of which are:

- Removal of RSA for key exchange
- Removal of MD5, SHA-224 and DSA for Signature Algorithms
- Enforcement of Perfect Forward Secrecy

4.3 Security.txt

This mechanism is analysed in section 5.4.

The security.txt [29] is a text file for the primary purpose of providing a security researcher the required information on how to report a security issues they have identified on such systems as websites.

For websites, a file named "security.txt" **must** be placed in the path `/.well-known/security.txt` e.g. `https://example.com/.well-known/security.txt`.

For legacy reasons it is also permitted to store a security.txt at the top level path e.g. `https://example.com/security.txt` or redirect to the `/.well-known/security.txt` path.

A response to a request of the security.txt file must respond with a `Content-Type` header with the value of `text/plain`, and served only over HTTPS.

The RFC is currently in the draft stage and seeing updates as recently as May 2021.

An example security.txt file is presented below:

```
Contact: mailto:security@example.com
Expires: 2022-12-31T00:00:00.000Z
Encryption: https://example.com/pgp-key.txt
Acknowledgments: https://example.com/hall-of-fame.txt
Preferred-Languages: en, es, ru
```

4.4 Content Security Policy

This mechanism is analysed in section 5.5.

The Content-Security-Policy (CSP) is a mechanism of a web browser that allows the maintainer of a website to control, via the use of a policy, the resources that are allowed to be retrieved and or loaded by the browser. CSP is an evolution of the *Same-Origin Policy (SOP)* browser mechanism detailed in section 1.2.

The `Same Origin Policy` mechanism is rather restrictive and this is where Content Security Policy gives more control to website maintainers and developers.

4.4.1 Development

CSP is developed and maintained by The World Wide Web Consortium (W3C) [85]. CSP Level 1 initial draft was published in 2012 [85] and finalised in 2015 [86]. CSP Level 2 had its first draft in 2014 [87] and was finalised in 2016 [88] which included the following changes:

- base-uri, child-src, form-action, frame-ancestors, plugin-types directives added
- frame-src depreciated
- Inline scripts / stylesheets allowed via nonces
- New fields added to violation reports

CSP Level 3 first public draft introduced in 2016 [89] with the most recent draft in 2021 [90] and is currently in the working draft state which included the following changes:

- Spec re-written
- frame-src undepreciated (uses child-src if not present)
- manifest-src, worker-src directives added
- 'strict-dynamic' source expression added
- child-src substantially changed
- deprecation of the report-uri directive in favour of a more recent directive report-to

4.4.2 Policy Delivery

The CSP policy can be delivered either by the use of one of two HTTP headers (Content-Security-Policy and Content-Security-Policy-Report-Only) or in the `<meta>` element in the html body of a web page.

CSP Directive

A CSP Policy consists of one or more directives which are bound to a specific type of resource. For example, the script-src directive controls the restrictions, or lack thereof, for the location(s) scripts can be obtained, executed or loaded from. A CSP Directive can have one or more values.

Should a directive be absent from a CSP Policy it is deemed to have no values i.e. "open". Configuring the default-src directive in a CSP policy cause most directives that are not present in a csp policy to inherit the values of specified for the default-src directive.

Table 4.1 details the directives from CSP Level 1 until CSP Level 3 (29 June 2021). CSP Level 3 is still a working draft and as such until CSP Level 3 is finalised the data within table 4.1 may become inaccurate should there be any changes before CSP Level 3 is finalised.

Directive	Restrictive Context Sources	CSP Level
default-src	Fall-back for interactions that are covered by CSP but not specified in other directives	1
script-src	script sources: e.g. <`script`> elements, inline script blocks and XSLT stylesheets	1
object-src	Restricts the source of plugin requests/content	1
style-src	Style sources: e.g. <`style`> elements and <`link rel='stylesheet'> elements`	1
connect-src	The URLs being called from scripts e.g. XMLHttpRequest requests	1
img-src	Image sources: e.g. <`img`> elements	1
media-src	Restricts the source of video/audio and related metadata	1
frame-src	Restrict the source of loading resources into child browsing contexts (e.g. <`iframe`> elements)	1
font-src	Restricts the source of font resources	1
connect-src	Restricts source of script interface requests	1
report-uri	Destination URI for sending CSP violates reports Deprecated as of CSP Level 3 in favour of `report-to`	1
base-uri	Restrict URIs for a document's base element	1.1
child-src	web worker sources and child browsing contexts (e.g. <`iframe`> elements)	1.1
form-action	Restrict URIs for Form action target	1.1
frame-ancestors	Restrict source of embedded child browsing contexts (e.g. <`iframe`> elements)	1.1
plugin-types	Restricts the media type of plugins. No longer present as of `W3C Working Draft, 10 March 2021` [91]	1.1
sandbox	Apply sandbox policy to a resource	1.1
manifest	Restricts source of application manifests	3
prefetch-src	Restricts source of pre-fetched/pre-rendered resources	3
script-src-elem	Restricts source of <script> elements only	3
script-src-attr	Restricts source of inline scripts (e.g. onclick) (not <script> elements)	3
style-src-elem	Restricts source of <style> and <link> elements containing `rel="stylesheet"` only	3
style-src-attr	Restricts source of inline style	3
worker-src-attr	Restricts source of resources loaded as Worker, SharedWorker, or ServiceWorker	3
navigate-to	Restricts source of any navigations	3
report-to	Reporting group for sending CSP violates reports	3

Table 4.1: Summary of CSP directives

4.4.3 Directives defined in other standards

There are several CSP directives defined in other standards.

upgrade-insecure-requests

Converts any `http://` request made by the browser to use `https://` i.e. all `http://` requests will be forced to use `https://`.

block-all-mixed-content

Does not allow the browser to make any `http://` requests for assets if the page was loaded over `https://`. This directive is processed after the upgrade-insecure-requests is processed, should it be present.

4.4.4 Source Lists

Below are the types of sources for values of CSP directives.

Keyword

The values below are keywords that can be used as values for directives:

- `none` – does not allow the use of resources that apply to the directive in question.
- `self` – restricts the resources to the domain of the web page and does not include subdomains.
- `unsafe-inline` – Permits the resource to be inline, .e.g. a scripts code resides in a <`script`> element rather that in a separate file.
- `unsafe-eval` – Permits the use of mechanisms, such as `eval()`, which transform plain text into executable code.
- `unsafe-allow-redirects` – Used with the `navigate-to` directive. Not currently supported in browsers.
- `unsafe-hashes` – Allows the use of inline scripts if the script matches a hash in the `script-src` directive.

Both `unsafe-inline` and `unsafe-eval` are deemed to be dangerous as inline resources and the use of mechanisms such as `eval()` are attack vectors for malicious actors, hence the use of the prefix `unsafe`.

Host

One or more “hosts” can be specified as values for a directive. Wildcards are permitted via the use of the star (*) character. The following are all valid examples of host values:

- domain.com
- *.domain.com
- https://*.domain.com:81
- domain.com/script.js

Scheme

Scheme values, referring the to protocol via which to access resources, may be used in directives. For example:

- http:
- https:
- data:

Nonce

A nonce can be used to allow inline resources to be allowed without the need to use the dangerous `unsafe-inline` keyword. The nonce would need to be included in both the CSP Policy directive and the inline resource.

For example if an in-line script was required, only scripts from the same origin are to be allowed, and the nonce was to be `MyRandomNonce`, then the CSP Policy would be:

```
Content-Security-Policy:  script-src ’self’ ’nonce-MyRandomNonce’
```

The inline script would also have the nonce:

```
<script nonce="MyRandomNonce">...</script>
```

Digest

A hash (digest) value can be used for scripts, the script-src directive, and non-script elements, such as the style-src directive. An example digest for the script-src directive:

```
Content-Security-Policy:  script-src 'sha256-MyHashDigest='
```

If a digest value is present, each script element will be hashed and the resultant hash compared to the hash value in the directive and if a match occurs the script is deemed to be allowed.

The current hashing functions specified in CSP Level 3 are `sha256`, `sha368` and `sha512`.

4.4.5 Reporting Directives

report-uri

Upon a CSP Policy violation and a value is present for report-uri, the browser will send a report to the location specified for report-uri. The report is in the form of JSON document(s) and sent using the HTTP POST method.

As of the most recent CSP Level 3 working Draft (29 June 2021), report-uri is deprecated in favour of `report-to`.

An example report-uri configuration can be seen below:

```
Content-Security-Policy:  ...; report-uri https://report.example.com;
```

report-to

Upon a CSP Policy violation and a value is present for report-to and the group is defined in the header `Report-To`, the browser will send a report to the location specified in the group configuration. The report is in the form of JSON document(s) and sent using the HTTP POST method.

An example report-to configuration can be seen below:

```
Content-Security-Policy:  ...; report-to <group name>
```

4.4.6 CSP Headers

There are a number of CSP headers, shown below, that have been created leading up to the current `Content-Security-Policy` and `Content-Security-Policy-Report=Only` headers.

- Content-Security-Policy
- Content-Security-Policy-Report-Only
- X-Content-Security-Policy (Used by and now deprecated by Firefox browser)
- X-Webkit-CSP (Introduced and now deprecated by the chromium project since 2013 [92])

4.4.7 Content-Security-Policy-Report-Only

The header `Content-Security-Policy-Report-Only` behaves the same as `Content-Security-Policy` with the exception that violations are only reported and not enforced. One of the main uses of this is when webmasters and developers are creating a CSP Policy and do not want to impact the end user experience until the policy is configured as desired.

Example basic CSP policy report only header: `Content-Security-Policy-Report-Only: default-src 'self'; script-src 'self' cdn.example.com; style-src 'self'`

4.5 Strict Transport Security

This mechanism is analysed in section 5.6.

The `Strict-Transport Security` response header (STS) is a mechanism that instructs the browser to load the website only over HTTPS and never HTTP [9].

4.5.1 Directives

max-age
The duration, in seconds, the browser will honour the Strict-Transport Security request to only load the page over HTTPS.

includeSubDomains

Instructs the browser to also load any subdomains of the site only over HTTPS.

preload

Must be present if the site is to be preloaded via Google's STS preload service.

Example STS header configuration: `Strict-Transport-Security: max-age 31536000; includeSubDomains; preload`.

4.6 STS Preload

This mechanism is analysed in section 5.7.

STS Preloading is a mechanism where a browser checks to see if the requested site is in a predefined list and if it is then the website must only be accessed over HTTPS.

Google maintains a predefined list, launched in 2010 [15], of websites where the website owners/maintainers have requested to be added and required an email to be sent to a google engineer to be included in the list [93]. This service later gained a web portal in mid-2014 [15] for the request to be included in the preload list.

The Firefox browser generates its own preload list. Each domain from the preload lost maintained by Google is checked and if it meets all the preload requirements it is included in the Firefox preload list.

The following criteria must be met in order for a domain to qualify to be included in the preload list:

- The website has a valid certificate trusted by browsers.
- If the domain (e.g. `example.com`) is running a HTTP server, on port 80, it must redirect to the HTTPS on the same domain.

  e.g.: if `http://example.com` is accessible, it must redirect to `https://example.com`
- All subdomains should be served over HTTPS, however only the `www` subdomain (e.g. `www.example.com`) is checked if present in DNS.

- A STS header (`Strict-Transport-Security`) is present on the base domain (e.g. `example.com`) for HTTPS requests with the following configuration:
  - `max-age` directive must be no less than 31536000 seconds (i.e. 1 year).
  - `includeSubDomains` directive must be present.
  - `preload` directive must be present.
  - Should the site redirect away when accessed over HTTPS, the site must have the STS header.
  - Example STS header preload configuration: `Strict-Transport-Security: max-age 31536000; includeSubDomains; preload`.

If a domain in the preload list stops meeting the above criteria it runs the risk of being removed from the preload list.

4.7 X Content Type Options

This mechanism is analysed in section 5.8.

The `X-Content-Type-Options` header (XCTO) is used to enforce that the "Multipurpose Internet Mail Extensions" (MIME) specified in the `Content-Type` header (e.g. `Content-Type: text/css`) matches that of the requested resource [30].

4.7.1 Directives

nosniff

Denies requests under either of the two below conditions:

- A request has a destination of type script-like (one of `audioworklet`, `paintworklet`, `script`, `serviceworker`, `sharedworker`, or `worker`) and the `MIME` type (from the `Content-Type` header) is not a JavaScript `MIME` type.
- A request has a destination of type style and the MIME type (from the `Content-Type` header) is not `text/css`.

Example XCTO Header: `X-Content-Type-Options: nosniff`

4.8 X Frame Options

This mechanism is analysed in section 5.9.

The `X-Frame-Options` (XFO) response header is a mechanism that states whether or not a browser should load the url in question in any of the following html elements: <`frame`>, <`iframe`>, <`embed`>, or <`object`>.

4.8.1 Directives

DENY

The page must not be rendered in any of the following html elements: <`frame`>, <`iframe`>, <`embed`>, or <`object`>.

SAMEORIGIN

The page is allowed to be rendered in any of the following html elements: <`frame`>, <`iframe`>, <`embed`>, or <`object`>, if the origin of the page that contains the element is that of the page to be rendered in the element.

ALLOW-FROM <uri>

Behaves the same as the above SAMEORIGIN directive but only for <`frame`> elements. This directive is now deprecated and should not to be used.

Example XFO Header: `X-Frame-Options: DENY`

4.9 Cross Origin Embedder Policy

This mechanism is analysed in section 5.10.

The `Cross-Origin-Embedder-Policy` (COEP) header provides a mechanism that does not allow any cross-origin resources from being loaded unless explicitly allowed (via the use of Cross-Origin Resource Sharing or a Cross-Origin-Resource-Policy).

4.9.1 Directives

unsafe-none

Disables the mechanism allowing resources to be loaded without explicit permission being given. This is the default behaviour.

require-corp

Enables this mechanism only allowing resources being loaded from the same origin or from other origins that explicitly give permissions to do so (via the use of Cross-Origin Resource Sharing or a Cross-Origin-Resource-Policy).

Example COEP Header: `Cross-Origin-Embedder-Policy: require-corp`

4.10 Cross Origin Resource Policy

This mechanism is analysed in section 5.11.

The `Cross-Origin-Resource-Policy` (CORP) header provides a mechanism that restricts which origins are permitted to load a resource [94].

4.10.1 Directives

same-site

If the request initiator's origin and the request destination's origins are of the same site then load the resource. For example, if the request initiator's origin is `https://account.example.com/index.html` and the request destination is `https://signin.example.com/script.js`, the resource can be loaded as both are of the same site `example.com`.

same-origin

If the request initiator's origin and the request destination's origin are the same then load the resource. For example, if the request initiator's origin is `https://blog.example.com` and the request destination is `https://blog.example.com`, the resource can be loaded as both are of the same origin `https://blog.example.com`.

cross-origin

The resource is allowed to be loaded regardless of the sites/origins involved.

Example COEP Header: `Cross-Origin-Resource-Policy:  same-origin`

Example CORP Header: `Cross-Origin-Embedder-Policy:  require-corp`

4.11 Cross Origin Opener Policy

This mechanism is analysed in section 5.12.

The `Cross-Origin-Opener-Policy` (COOP) header provides a mechanism that prevents the sharing of browsing contexts of a top-level document with cross-origin documents. For example, preventing a popup window from communicating to the web page (document) that opened the popup via the use of process isolation [33].

Directives

unsafe-none

Disables the mechanism unless the document (e.g. popup) uses this header with either of the two directives: `same-origin` and `same-origin-allow-popups`. This is the default behaviour.

same-origin-allow-popups

Allows the communication of a new window or tab (e.g. popup) to the originating page (document) if a COOP header is not present, or has the directive `unsafe-none`, on the new window or tab.

same-origin

Does not allow communication (isolates) from new windows or tab to the originating page (document) unless they are from the same origin. If not from the same origin they are placed in a separate browsing contexts.

Example COOP Header: `Cross-Origin-Opener-Policy:  same-origin`

4.12 Public Key Pins

This mechanism is analysed in section 5.13.

The `Public-Key-Pins` (HPKP) response header is a mechanism that permits a cryptographic public key to be advertised that must be matched to the website certificate presented during the SSL/TLS connection establishment otherwise the website will be prevented from loading/rendering.

There have been several issues with HPKP such as HPKP Suicide [95, 96] or Ransom PKP [96] which are forms of Hostile Pinning [97]

This header is now deprecated by most modern web browsers.

4.12.1 Public-Key-Pins-Report-Only

The `Public-Key-Pins-Report-Only` header does not enforce the pinning rather only sends violation reports.

4.13 Summary

This chapter gave a in depth overview of the selected security mechanisms which were introduced in the opening chapter in section 1.3. This should allow the reader to have a better understanding and context of these security mechanisms for the next chapter "Security Mechanism Adoption Analysis" which analyses and discusses the adoption of the mechanisms.

Chapter 5

Security Mechanism Adoption Analysis

This chapter takes the data captured as described in chapter 3 and analyses the adoption of the selected security mechanisms (introduced in section 1.3 and detailed in chapter 4).

5.1 Scans Overview

5.1.1 Impact To Available Data For Analysis

There are several gaps in the available scanning archives summarised in table 3.8, detailed in section 3.4.6 and listed below:

- 24 Jan 2021 for 8 days
- 16 March 2021 for 12 days
- 31 July 2021 for 27 days
- 17 September 2021 for 17 days

In the following sections these date ranges will be visible on certain charts. As the scanning has been for an extended period and any single time period is no longer than 27 days, it is believed it will not have a material impact on any trends identified.

5.1.2 Analysis Methodology

Each scan archive (which consists of a single days scan of 1 million domains as described in section 3.4.6) was processed, grouping related data together, such as all the security.txt files, to produce a JSON file to allow for easier analysis of the data.

A python application was created to parse these JSON files to produce further JSON files that were be used for creating charts.

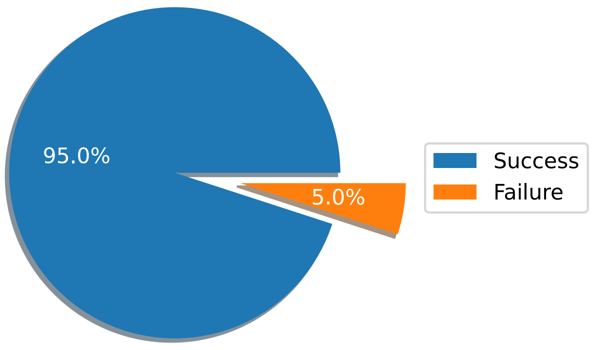
A further python application was created to plot charts using the matplotlib charting library [98].

Each analysis section details how the data was processed.

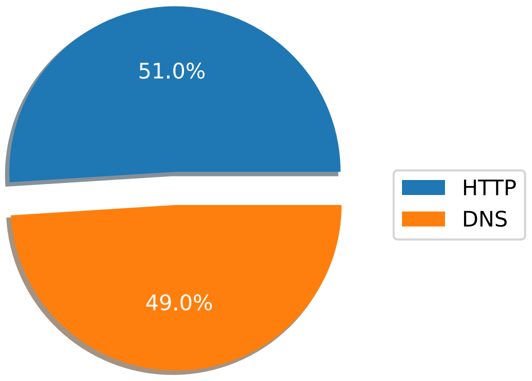
5.1.3 Domains Unable To Be Scanned

As with other research, not all of the domains in the list of domains to be scanned were able to have a HTTP or HTTPS connection made to them.

On average the scanner was able to obtain a HTTP and or a HTTPS request 95% of the time from the Tranco list of 1 million domains, shown in figure 5.1.



(a) Average percentage of domain scan final status



(b) Average percentage of domain scan failure cause

Figure 5.1: Failed Tasks

Of the 5% of domains that the scanner was not able to obtain a HTTP or a HTTPS response, 51% were due to differing errors, such as timeouts, in the establishment of a HTTP/HTTPS connection (stage 3 of the task flow) and the transmission of data across such a connection. The remaining 49% was due to not receiving one or more IP addresses from the DNS lookup request (stage 2 of the task flow).

5.2 HTTP(S) Redirection

Background for this mechanism is detailed in section 4.1.

5.2.1 Analysis

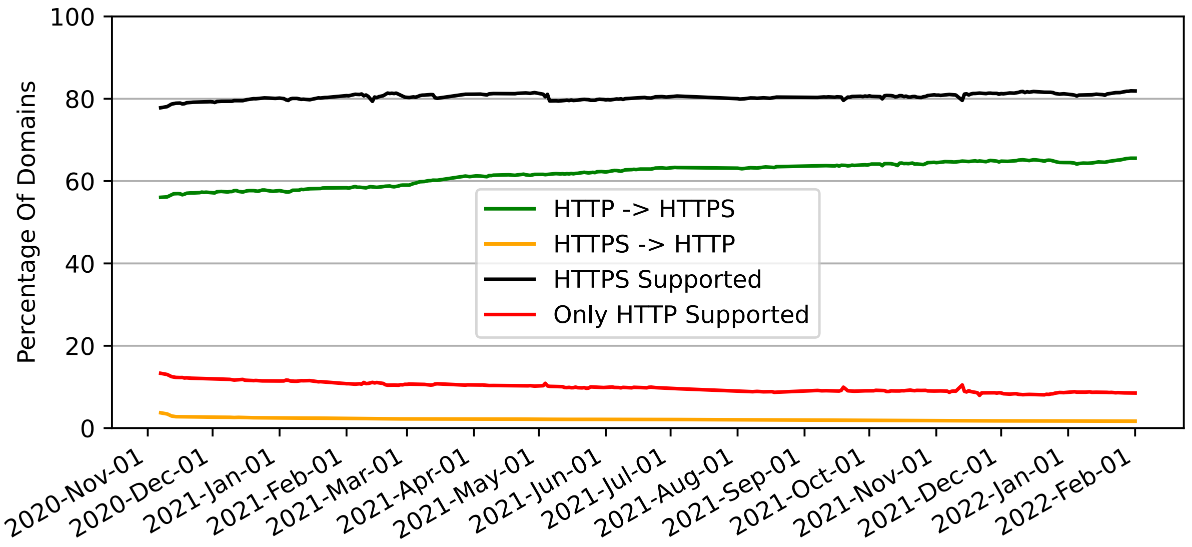


Figure 5.2: HTTP(S) Redirections for Top 1 Million Domains

The research by `Buchanan et al` [10] found that in August 2015 only 6.7% of sites when accessed over HTTP redirected to HTTPS and increased to 24.78% in May 2017. This study finds that in November 2020 the number of sites redirecting from HTTP to HTTPS further increased to 56% and continued this trend to reach 64.5% in January 2022 as shown in table 5.1 and figure 5.2.

Redirection	**Buchanan et al** [10] August 2015	**Buchanan et al** [10] May 2017	November 2020	January 2022
HTTP > HTTP**S**	6.7%	24.78%	56%	64.5%
HTTP**S** > HTTP	NA	NA	3.5%	1.7%

Table 5.1: HTTP(S) Redirection

This upward trend of HTTP to HTTPS redirection is slowing. However it is hoped that more sites continue to implement redirection such that future research does not show a plateau before nearly all sites in the top 1 Million redirect to HTTPS.

To some surprise there are a number of sites that redirect from HTTPS to HTTP, 3.5% in November 2020 and down to 1.7% in January 2022. It is assumed that this is done on purpose, and that as the trend is on the decline that the reasons for implementing HTTPS to HTTP redirection are being overcome or are not relevant any more.

There are a number of sites that only support HTTP and do not support HTTPS, ∼16% in November 2020 and down to ∼9% in January 2022 again trending in the right direction.

5.3 TLS

Background for this mechanism is detailed in section 4.2.

5.3.1 Purpose Overview

TLS is used to protect the data in transit between a users' browser and the website they are interacting with.

Changes to the data being sent across the TLS secure channel during its journey over the network (e.g the internet) are detected. The identity of the website is verified before the secure channel is established. The data being sent across the TLS secure channel is also encrypted. All of these properties of TLS are intnded to prevent entity in the middle attacks.

There have been several vulnerabilities found in the SSL/TLS protocols, some of which are detailed earlier in section 2.1.

5.3.2 Analysis

Negotiated TLS Versions

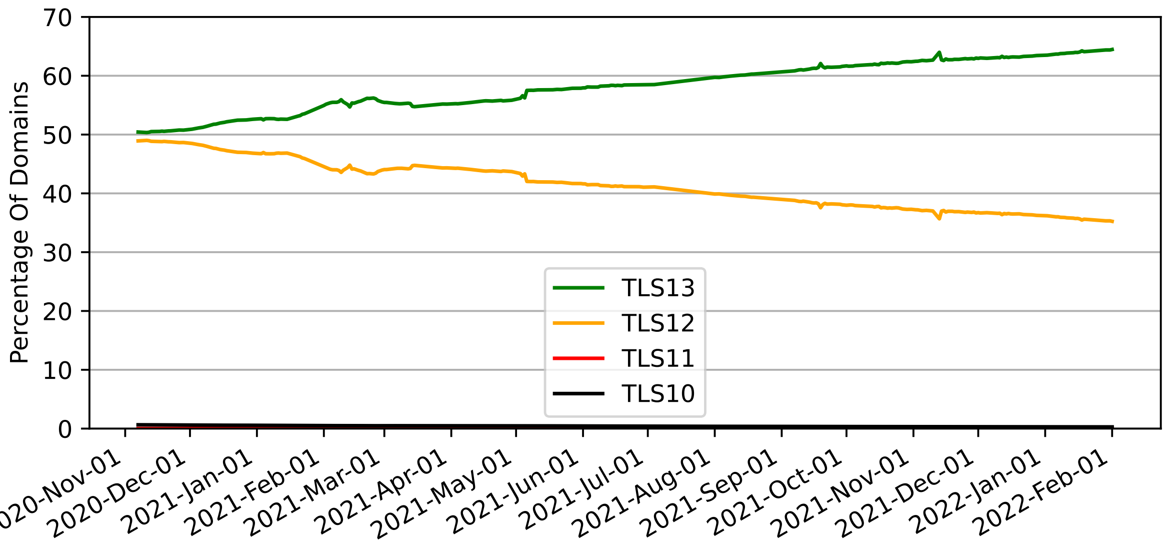


Figure 5.3: TLS version negotiated for Top 1 Million Domains that support HTTPS

Figure 5.3 shows the negotiated TLS version of websites that ended with HTTPS URLs after HTTP redirection was completed. The scanner supported the following TLS versions when attempting to connect to a website with a HTTPS url:

- TLS 1.0
- TLS 1.1
- TLS 1.2
- TLS 1.3

No version of the legacy SSL technology is supported by the built in HTTP client in Go (the programming language used to create the scanner used to gather the data for analysis). The lack of support of SSL is not of concern as the use of SSLv3 (the last version of SSL that was widely used) currently is essentially 0% as shown by the research by *Kotzias et al* [60] which showed that analysing passively collected SSL/TLS usage, SSLv3 usage was 0.01% of all TLS/SSL connections they had for analysis in February 2018.

As shown in figure 5.3, November 2020 appears to show that it was just before this time that TLS 1.3 started to become the dominant TLS version negotiated, replacing TLS 1.2 as the dominant version. This assumption is enhanced by the research of `Holz et al` [17] which showed that in November 2019 from the analysis of passively collected TLS connection details, ∼79% of connections were using TLS 1.2 with a downward trend and ∼19% of connections using TLS 1.3 with an upward trend.

TLS Version	**Kotzias et al** [60] July 2013	**Kotzias et al** [60] August 2014	**Holz et al** [17] November 2019	January 2022
TLS 1.0	∼73%	∼48%	<1%	<1%
TLS 1.1	∼17%	<1%	<1%	<1%
TLS 1.2	∼1%	∼48%	∼79%	35.2%
TLS 1.3	NA	NA	∼19%	64.4%

Table 5.2: Negotiated TLS Versions

Supported TLS Versions

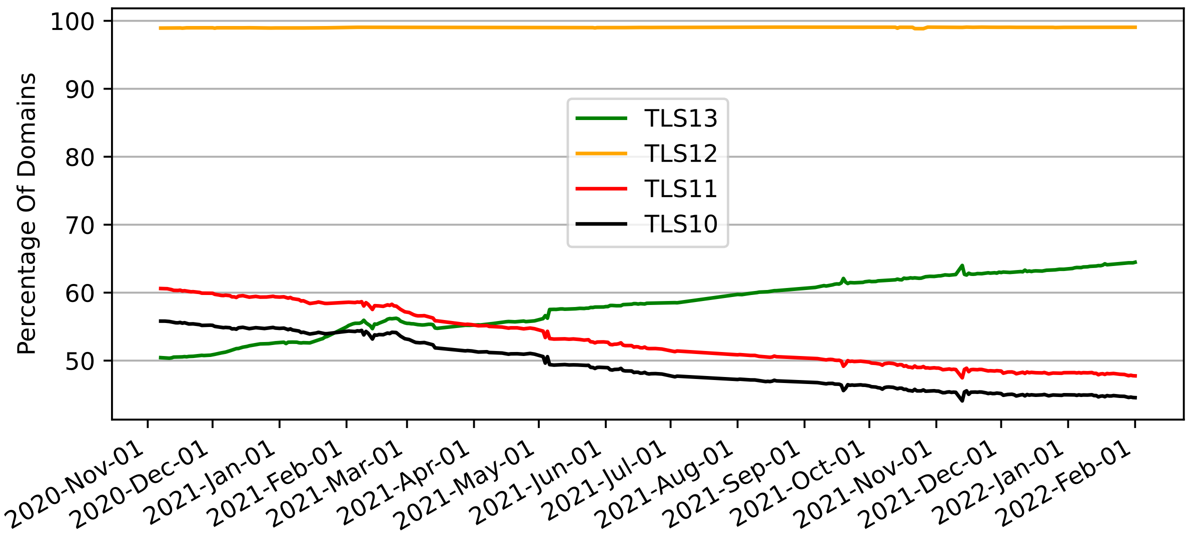


Figure 5.4: TLS Versions Supported

For each website that did support HTTPS, it was also checked to see which of the 4 TLS versions it supported. Figure 5.4 shows how many domains supported the 4 TLS versions. The measurement of supported TLS versions was not observed in the research consulted for this project, indicating that it could be uncommon and as such this report may be one of only a handful that has conducted these measurements.

TLS 1.2 is supported by over 99% of domains. The reason for this is likely due to TLS 1.2 being the most recent TLS versions supported by some legacy devices and is likely to remain like this for many years to come. As one might expect, TLS 1.3 is on an upward trend going from 50% in November 2020 and reaching 64% in January 2022.

Both TLS 1.0 and TLS 1.1 look to follow the same downward trend with 44% and 48% in January 2022 respectively. As they follow the same trend it is likely due to when libraries and or applications are upgraded, rather than manual configuration changes to remove support for TLS 1.0 and TLS 1.1.

For the 0.2% of websites that negotiate with TLS 1.0 or TLS 1.1 they did not support any version higher than that they negotiated with. There should not be a significant reason that TLS 1.2 and TLS 1.3 cannot be supported whilst also supporting TLS 1.0 and or TLS

# Domains	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
9	●	●		
23	●			●
32		●		●
69	●	●		●
75	●		●	●
1579	●			●
2305	●			
4888				●
12898		●	●	
15954		●	●	●
137501	●	●	●	
201151	●	●	●	●
261762			●	●

Table 5.3: Supported TLS Version matrix for 02 Jan 2022

1.1 for use by legacy devices.

It is possible that websites are running old/outdated versions of applications, supporting only TLS 1.0 and or TLS 1.1, serving these websites and the maintainers are unaware or the updating of these older systems are of low priority. A reverse proxy would at least allow the public endpoint of such websites to use TLS 1.2 and TLS 1.3.

Table 5.3 is a matrix for the different combinations of TLS versions supported by domains on 02 Jan 2022. The highest concentration of domains is for supporting both TLS 1.2 and TLS 1.3 which is quite encouraging.

5.3.3 Summary

The trend for negotiated TLS versions is going in the right direction, with TLS 1.3 being the dominant version currently, as this research began with an upward trend and as yet does not show sings of tapering off. TLS negotiations that result in either TLS 1.0 or TLS 1.1 is less than 1% of all TLS negotiations.

The use of TLS 1.1 and TLS 1.0 were declining rapidly at the same rate at the end of 2020, however the trend is now starting to slow down which likely means TLS 1.1 and TLS 1.0 will still be in use s for some time.

5.4 Security.txt

Background for this mechanism is detailed in section 4.3.

5.4.1 Purpose Overview

The “security.txt” mechanism is to aid in informing security researchers how to disclose security issues found on systems such as a domain hosting a HTTP(S) sever [29].

The primary emphasis of using a security.txt is that its location is known. Without knowing where to find the information required to report vulnerabilities, security researches may not be able to reach the right team in the organisation able to take action on the report for some time or even at all.

Security researches can spend quite a long time trying numerous channels such as Linkedin, contact pages on websites and guessing security email addresses. If private messaging attempts fail, some researchers resort to seeking assistance via twitter, however this also draws attention to there being a vulnerability on the organisation in the tweet which could allow malicious actors to discover and exploit the vulnerability before it is fixed.

5.4.2 Scanning

As the RFC Draft stated that “*For web-based services, organizations MUST place the ”security.txt” file under the ”/.well-known/” path*” [29] e.g. `https://example.com/.well-known/security.txt` the scanner was initially configured to only try and obtain the security.txt from this location. The statement regarding an exception for legacy devices was missed when creating the scanner. The legacy exception was added in the 8th revision of the RFC dated November 19, 2019 [29].

At the end of November 2021 the research paper *Who you gonna call? an empirical evaluation of website security.txt deployment* [18] was discovered and found that 18% of security.txt were found only at the top level path of the domain e.g. `https://example.com/security.txt`.

As a result of this information, in December 2021 the scanner was reconfigured to attempt to capture at both paths. Even thought this would only allow a short time of data analysis it would allow, if only at least minimally, a comparison of results from this research to the research from [18].

5.4.3 Parser

A parser was created in python that processed security.txt request responses that had HTTP 2xx or 3xx status codes, split each security.txt on new line terminations and proceeded to process each line individually. Lines that started with a hash "#" (%x23) were skipped as these are comments as defined in the draft RFC.

If the first line was deemed to represent a non security.txt file, such as a html or php file, the entire security.txt file was skipped.

All non comment lines that were not able to have a field name determined were not processed any further and checked once all security.txt files had been processed. If any of the apparent non RFC format adhering lines were found to have been skipped, when they should not have been, the parser logic was analysed and corrected as needed until there we only non RFC con-formant lines skipped.

As the scanner followed redirects, the parser kept track of the resultant domain of any such redirects and skipped processing a security.txt if the resultant domain had already had a security.txt processed.

If after processing a security.txt file no fields were detected, the security.txt was deemed to be invalid.

Contact values and expires date-times had multiple specially crafted regular expressions constructed in order to accurately categorise the field values, mainly due to many variants of the following:

- email obfuscation formats
- phone number formats
- date-time formats

Even though `Acknowledgments` (American spelling) is the spelling used in the RFC `Acknowledgements` (British spelling - has an extra e), the parser considered it to be `Acknowledgments` for the purposes of charting as the they are both valid spellings of the same word.

If a security.txt file was deemed to be con-formant it was hashed using the SHA256 hashing algorithm to determine how many unique security.txt files were present and how many domains shared the same security.txt file.

During the process of creating the parser it was noticed that some security.txt files had a dynamic value set for the Expires field (visiting the file manually on different occasions would show the expires time to be a constant time from the time of request). As this would affect the detection of unique security.txt, all Expires field lines were removed before hashing took place.

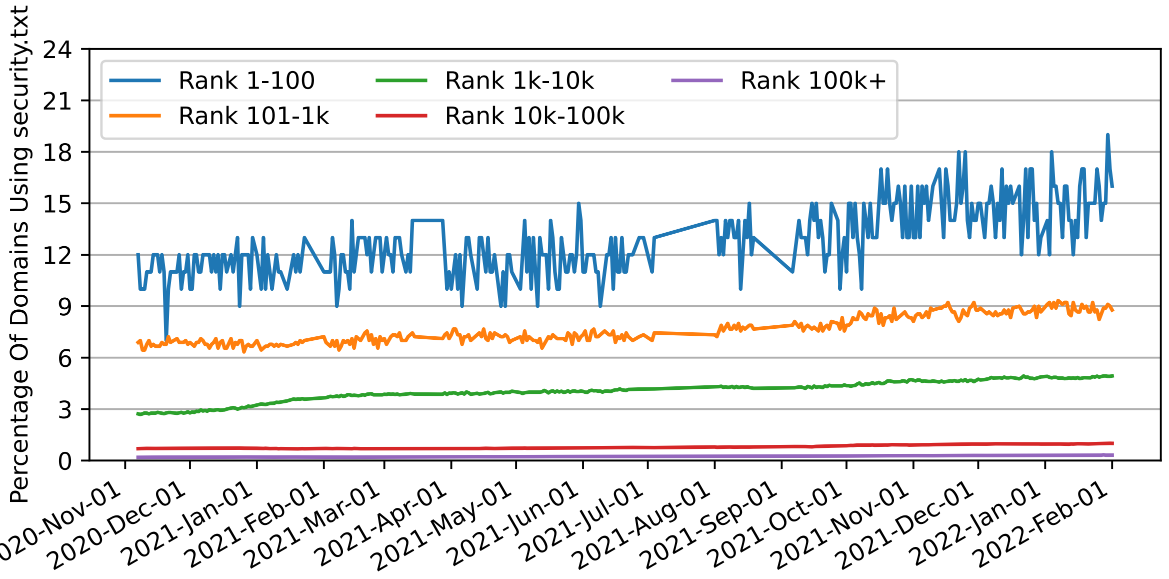


Figure 5.5: security.txt (only /.well-known path) by rank

5.4.4 Analysis

Distribution by Rank

Figure 5.5 shows the percentage of domains that the scanner was able to identify having a valid security.txt at the path `/.well-known/security.txt` e.g. `https://example.com/.well-known/security.txt`. The line chart is split by rank groups using the rank provided in the domains list file from tranco-list.eu for the day the scan was conducted.

Figure 5.5 closely matches, with a uniformly overall reduced percentage however, that of figure 1 in [18]. It is not clear from [18] whether the ranking always starts from 1 or from the previous smaller rank.

The ranks ranges in Figure 5.5 in this report are as stated in the legend, domains from rank 1 to rank 100, rank 101 to 1000 and so on. The last rank is from 100001 to 1 million. Rank 1-100 from figure 5.5 should be comparative to that of the Top 100 of figure 1 in [18] while taking into account the potential 18% discrepancy (where security.txt are only found in the top level path e.g. `https://example.com/security.txt`).

The uniformly overall percentage difference is roughly the 18% of security.txt files that are

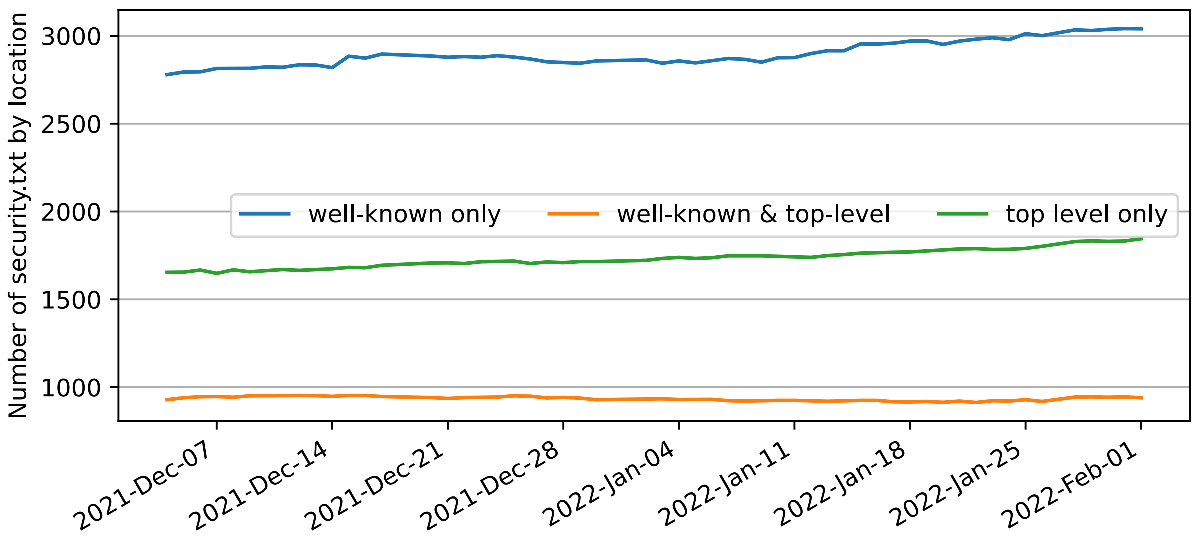


Figure 5.6: security.txt file locations

potentially missing from this reports figure 5.5 due to the finding in [18] that 18% of sites only serve a security.txt at their top level path.

Distribution by Path

As noted in the scanning section above, the scanner was initially configured to only scan for security.txt at the `/.well-known` location. Figure 5.6 shows (from December 2021) onwards, that there are a significant about of security.txt only hosted at the top level path of the domain e.g.
`https://example.com/security.txt`.

Unique Domains vs Unique Security.txt

Figure 5.7 shows the number of unique security.txt files across all domains scanned and the number of unique domains that had a security.txt file (separated by whether they are valid or not) that were found in the `/.well-known` location. The number of invalid security.txt files are a small percentage of all security.txt files and the majority of them are html documents unrelated to the security.txt mechanism.

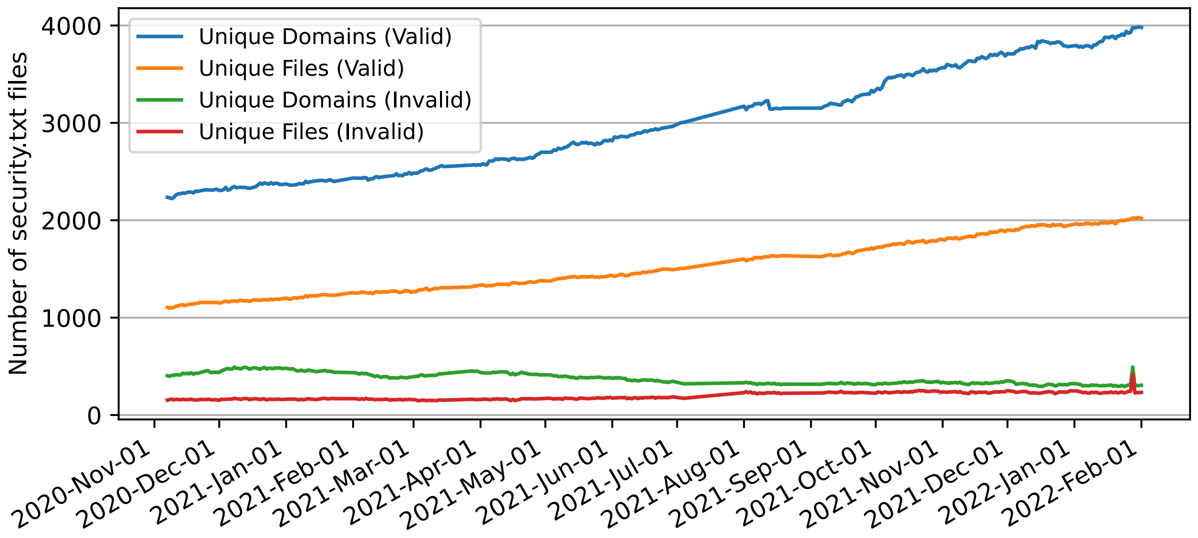


Figure 5.7: security.txt (only /.well-known path) Unique Domains vs Unique Files

The number of unique security.txt files is roughly half of all security.txt files found on unique domains. A reason for this could be that one organisation is responsible for multiple domains and uses a common security.txt for most and or all domains under their control.

Due to the large difference between the number of unique security.txt and number of unique domains further investigation was conducted to identify any large groupings of domains that used the same security.txt. From the 02 January 2022 scan, Google has the highest number of domains (198) with tumblr having (189) coming in a close second. The domains were grouped by the hash of the security.txt file. As one might expect the domains associated with Google predominantly started with `google.` e.g. `google.com`. The domains associated with tumblr had "*Please report abusive content (including spam, privacy violations, etc) at https://www.tumblr.com/abuse*" in their security.txt file and did not appear to have a particular naming format. However the Hackerone link revealed that an organisation named "Automatic" manages the Hackerone representation.

Whilst this is nice to see that organisations look to be using unified configurations this does give another potential method by which to identify/fingerprint sites that are controlled by large organisations. This gives the opportunity to find sites with potentially weaker security (inherited from an acquisition for example) in order to laterally move to the parent organisations' infrastructure.

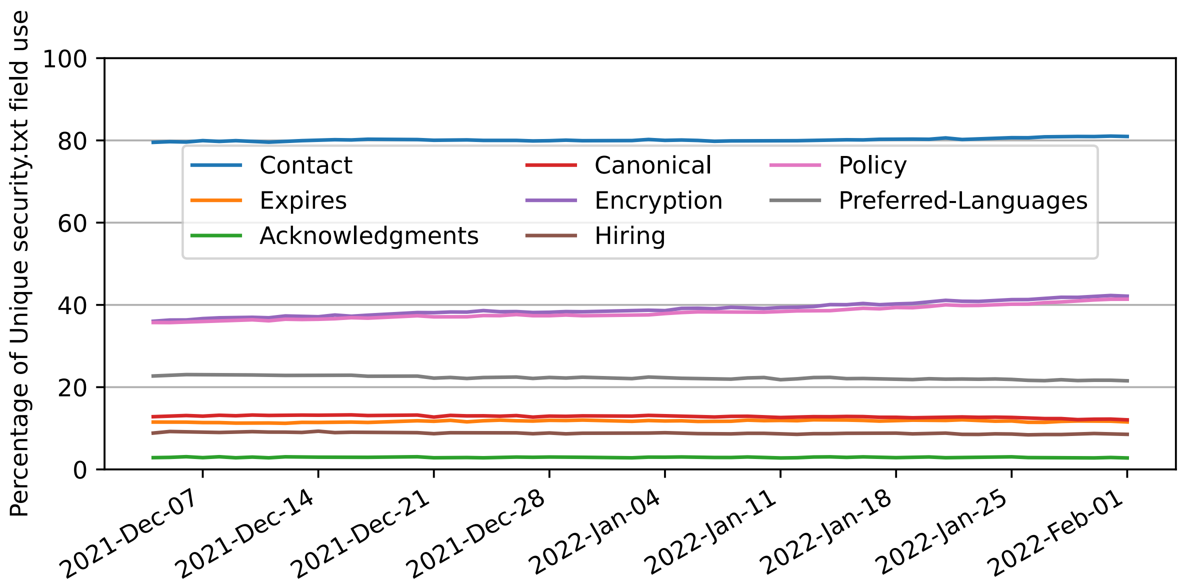


Figure 5.8: unique security.txt (top level path) by field use

Fields

A low amount of security.txt files (17%) contained digital signatures which allow the security researcher to verify the legitimacy of the security.txt. It should be noted that the digital signature should be allowed to be verified using out of band methods (i.e. not a link in the security.txt itself) such as DNSSEC.

Figure 5.9 shows the field use of unique security.txt files. If a field is present more than once in a security.txt it is only counted once for the purposes of only showing if a field is present in a security.txt.

Figure 5.8 shows the field use distribution from December 2021 onwards for security.txt files found in the top level path location. Compared to the distributions in figure 5.9 they do not differ to such an extreme to discount the analysis from herein out which is only using the security.txt files found in the `/.well.known` location due to larger number of scans for analysis.

There are 8 defined fields in the RFC, 2 of which are mandatory (Contact and Expires shown in bold):

- Acknowledgments - links to acknowledge security researches for their efforts in re-

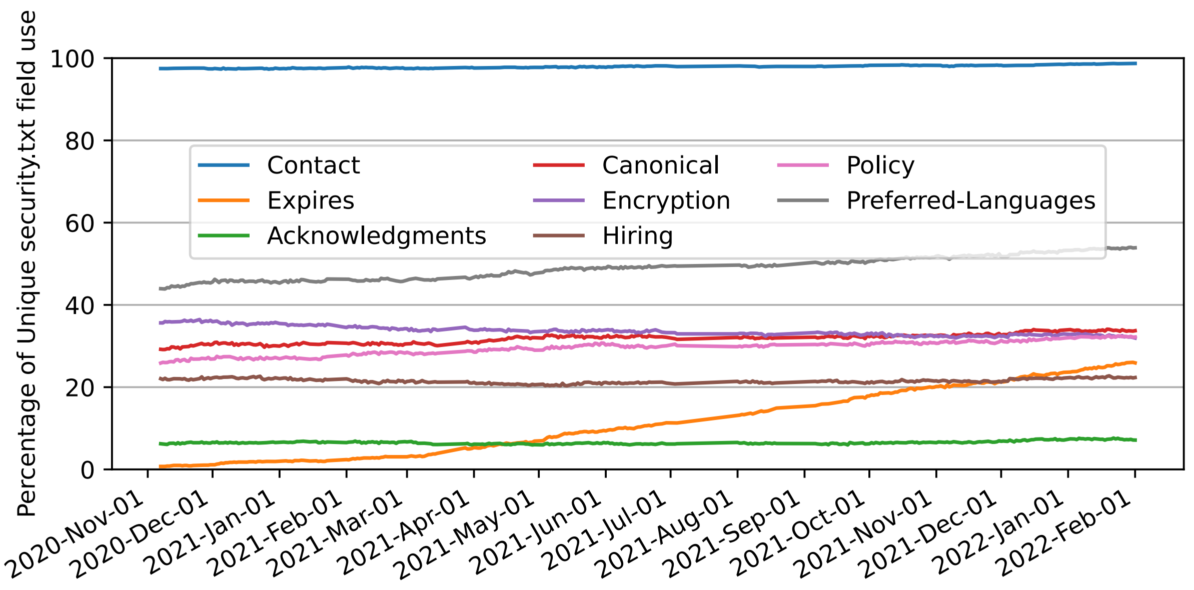


Figure 5.9: unique security.txt (/.well-known path) by field use

porting security vulnerabilities.

- Canonical - link to a canonical security.txt location.
- **Contact** - A method to use for reporting security vulnerabilities (e.g. email, phone and website URLs).
- Encryption - States or links on how to encrypt communication.
- **Expires** - The date-time after which that data present in the security.txt should not be used.
- Hiring - Link to a page detailing security related positions.
- Policy - Link to the vulnerability disclosure.
- Preferred-Languages - One or more languages, in order of preference, that the security report maybe be submitted in.

Contact

Contact is a mandatory field and as such one would expect to see near 100% usage and this is the case as can be seen in figure 5.9 which shows only that only 1.5-2% of security.txt files do not have a `Contact` field present.

Contact Type	**Average Presence**
Email Address	88%
HTTP Link	22%
Phone Number	1.6%

Table 5.4: Presence of contact types from unique security.txts that contain a contact type

Table 5.4 shows the average presence of the contact types detected in unique security.txt files. Even as the number of unique security.txt files increased over time as shown in figure 5.7, the presence percentage deviation was less than 1% for all contact types.

Email Username	**Average Usage**
security	48%
support	2.2%
info	2.2%
admin	1.7%
webmaster	1.7%

Table 5.5: Top 5 email usernames for security.txt email contacts

The most prevalent email username used was `security`, as shown in table 5.5 which is of no real surprise due to the purpose of the security.txt.

There were several email addresses that were made obscure (obfuscated) which required multiple dedicated regular expressions to be created in order for the parser to detect them as email addresses. Some examples of email obfuscation observed are:

- security [at] example [dot] com
- security [@] example <do$t$ > com
- security [a t] example [d o t] com
- security[@]example.com
- security(<remove me...>)at(<remove me...>)example_dot_com
- the email address ”security” on the domain ”example.com”

One reason why email addresses have been obfuscated is not wanting them to be identified on automated scraping/crawling. This could be a result of an organisational policy or the person(s) responsible assumed that a person would be looking at the security.txt file and as such there was not much of a down side.

Contact Count	**Average Presence**
1	82.5%
2	13.2%
3	2.5%
4	0.4%
5	0.05%
6	0.1%

Table 5.6: Number of contacts present in a unique security.txt

Of the security.txt files that had one or more contacts, the majority contained only 1 contact as shown in table 5.6. Interestingly there were double the amount of security.txt files that contained 6 contacts than those that had 5 contacts.

Of the 1.5-2% of security.txt files that did contain one or more fields but did not contain a `Contact` field, several did contain other non RFC contact fields including:

- openbugbounty field - openbugbounty is a not for profit organisation for the reporting of security vulnerabilities.
- email
- mailto
- contact us at

Expires

`Expires` is a mandatory field and has a rather underwhelming presence starting in November 2020 with 0.7% (the least used field), going on to reach 4% in mid March 2021. From March 2021 there is a marked increase in the rate at which `Expires` is included in security.txt files seen in figure 5.9 reaching 24% presence at the end of December 2021 becoming the 3rd least used field.

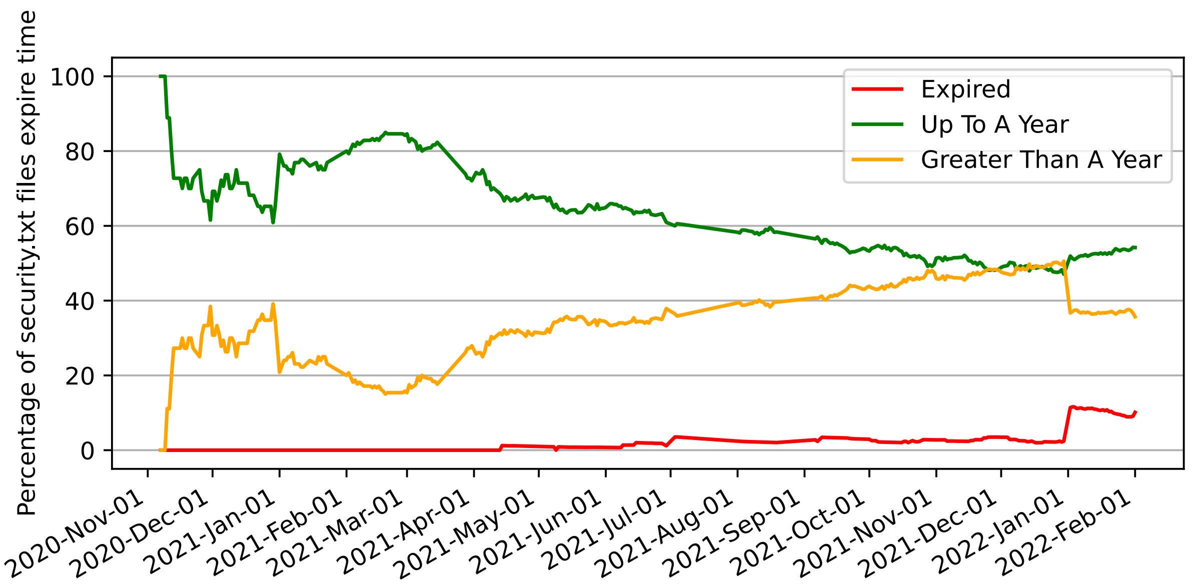


Figure 5.10: Percentage of security.txt files (that use expires) by time until expires from scan date

If this trend continues, `Expires` will likely get close to 100% use such as is with the `Contact` field.

There are several possible reasons that domains are not including the expires including:

- the administrative burden of regularly reviewing the security.txt.
- if a static date-time is used and it is left to expire, security researches may decide to seek conformation other than using the information in the contact field to verify the correct means of reporting a security vulnerability, thus delaying such reporting. If no expires is included then it cannot expire.

Figure 5.10 shows that there are a minimal amount of security.txt files that have an expires value in the past (ranging from 0% to ∼3%). The number of security.txt files that expire in less than a year is trending down and those that are to expire greater than an year is increasing. This looks to indicate that expires values are being changed to last more than one year against the recommendation of the RFC, which is to have an expire value less than a year to avoid staleness.

Encryption

The RFC states that keys should not be included in security.txt files. Despite this, ~0.7% of unique files do include keys. This looks to indicate either that the majority are conforming to the RFC recommendation and or do not wish to include a key in the security.txt.

Type	**Presence**
HTTPS	~93.5%
HTTP	~2%
OpenPGP4 Fingerprint Reader	~3.0%

Table 5.7: Percentage of encryption field value type of unique security.txt

Table 5.7 shows three types of encryption reference: HTTPS, HTTP and OpenPGP4 Fingerprint Reader (openpgp4fpr). The RFC states that the `Encryption` field must begin with `https://`, however despite this OpenPGP4 Fingerprint Reader has a presence of ~3.0%. A reason for this could be that it was more convenient to use an OpenPGP4 Fingerprint Reader, however this might hinder the use of it if security researchers are not familiar with this scheme type. Should a website operator want to use OpenPGP4 Fingerprint Reader scheme, it is best placed in a separate web page such that instructions can be included on how to use such a scheme. 2% of values use the scheme `http://` which is quite surprising given the purpose of a security.txt and being present in the `Encryption` field.

Preferred-Languages

As one may expect there were many different languages (over 50) represented in the `Preferred-Languages` field. Table 5.8 shows the presence of the top 5 languages for unique security.txt files.

Language	**Presence**
English	99%
German	13%
French	11%
Czech	13%
Dutch	4%

Table 5.8: Percentage of preferred-languages field value type of unique security.txt files

English is the most prominent language which makes sense as that is also the most common language used for business. If a researcher did not speak English, or any of the other languages listed, they may be very well know someone who does in order to report an issue.

Other RFC Fields

The remaining fields defined in the RFC that have not been analysed in this section are of less interest and as such only their relative presence in a security.txt was analysed and displayed in table 5.9 and figure 5.9.

Field	**Presence**
Acknowledgements	∼7%
Canonical	30%-33%
Hiring	∼21%
Policy	28%-31%

Table 5.9: Average field presence of remaining RFC fields in unique security.txt files

Non Standard Fields

There are a number of fields, many present in only a single security.txt, that appear in security.txt but are not defined in the RFC. Table 5.10 details the top 5 most common of these non standard fields.

Field	**Presence**	**Inferred Use**
Signature	∼4.5%	Link to a digital signature file
Openbugbounty	∼3.5%	Link to openbugbounty.org organisation page
Permission	∼1.5%	Permission to test vulnerabilities
Disclosure	∼1%	Type of public disclosure permitted
Sitemap	∼1%	HTTPS link to an XML file of the websites page structure

Table 5.10: Average field presence of remaining RFC fields in unique security.txt files

All of the `Permission` values appear to be set to `none` which would seem to indicate that permission is not given to perform testing for vulnerabilities against the site.

The `Disclosure` values were seen to be a HTTPS link, `Full`, `Partial` or `None` which would seem to indicate that if a security vulnerability were to be submitted the security researcher would act in good faith and adhere to this value in regards to the public disclosure of the security vulnerability found.

5.4.5 Summary

The use of security.txt is quite minimal with 0.5%, as of January 2022, of domains scanned having a valid security.txt file. This is not all to surprising as the RFC is still in the draft stage.

Approximately half of all the valid security.txt files seen were unique, meaning that a large proportion of domains share the same security.txt file content.

Almost all security.txt files contained one or more contact which is the primary purpose of the security,txt mechanism to provide a contact point for security researchers to have an easy way to report vulnerabilities.

As previously mentioned, one of the paths a security.txt file can be hosted at, as specified in the draft RFC, was not utilised by the scanner for the majority of the scans conducted. In any future studies, all paths specified in the latest version of the RFC should be utilised.

5.5 Content Security Policy

Background for this mechanism is detailed in section 4.4.

5.5.1 Purpose Overview

CSP mainly provides mitigations for web applications against XSS Cross Site Scripting (XSS), data injection attacks and packet sniffing attacks. There are several basic policy settings that can be utilised for protection against such attacks.

Restrict locations resources can be loaded from

Using the script-src directive to only allow resources to be loaded from trusted sources is a great first step to prevent attackers from loading scripts. The additional use of sub resource integrity [99] can be used to prevent supply chain attacks where the attacker takes control of a trusted source script.

Remove the need to use unsafe directives

Both `unsafe-inline` and `unsafe-eval` should be avoided especially with the `script-src` and `style-src` directive to prevent attackers from injecting malicious script code into a website. Inline scripting is one of the biggest attack vectors for attackers trying to run scripts in the browser. An alternative to removing inline scripting is the use of the `Digest` keyword or the `Nonce` keyword to allow only specified inline scripts to run.

Avoid using the data scheme

Relatively tiny amounts of information can be used with the `data` schema and can be utilised for the purpose of triggering the download of malicious resources by attackers. A malicious script could be converted and placed into a `data` schema, injecting it into a page, leading to it being executed.

Keep up to date with CSP Bypass Techniques

CSP policy maintainers should keep up to date with CSP bypass techniques such as those described in "*Complex Security Policy? A Longitudinal Analysis of Deployed Content Security Policies*" [61].

Category	**Description**
Allow All	The value `*`
Keyword	Any keyword value e.g. `self`
Scheme	Any scheme value e.g. `blob:`
Nonce	Any nonce value e.g. `nonce-MyRandomNonce`
Digest	Any digest value e.g. `sha256-MyHashDigest=`
Host	Any host value e.g. `*.example.com`

Table 5.11: CSP Directive Categories

5.5.2 Parser

A parser was created in python which processed the value for a content security policy header. The parser checked all of the headers below of a scan result for the presence of a CSP header.

- Content-Security-Policy
- Content-Security-Policy-Report-Only
- X-Content-Security-Policy
- X-Webkit-CSP

The CSP text was split on the ; (semi-colon) character which denotes that a directive configuration has ended. Each directive configuration was split on the space character which is the separator between directive configuration values.

A number of regular expressions were created to place each directive into one of the following categories as described in table 5.11.

As the scanner followed redirects, the parser kept track of the resultant domain of any such redirects and skipped processing a CSP if the resultant domain had already had a CSP processed.

5.5.3 Analysis

Distribution by Rank

Figure 5.12 shows the percentage of domains that the scanner was able to identify having a CSP header. The line chart is split by rank groups using the rank provided in the domains list file from tranco-list.eu for the day the scan was conducted.

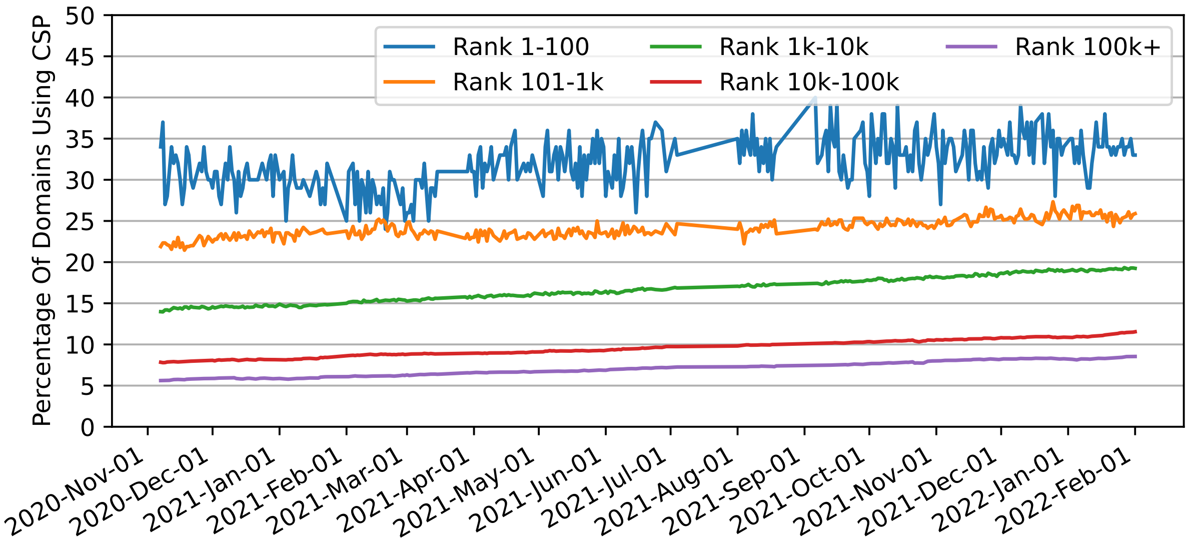


Figure 5.11: CSP headers use by rank for unique domains

Each rank group shows a steady rise in the use of a CSP headers, with domains in the Rank of 1-100 group seeing the highest relative percentage use, which is an encouraging sign. There is a long way to go, however, to have the majority of sites using a CSP policy.

In the research by `Weissbacher et al` [100] in March 2014 showed only 2% of the sites ranked in the top 100 use CSP increasing to 7% in the research by `Ying et al` [14] in June 2015 and as of January 2022 36% use a CSP policy which is a marked increase.

HTTP Header	**Weissbacher et al** [100] March 2014	**Ying et al** [14] June 2015	Jan 2022
Content-Security-Policy	815	1241	83208
Content-Security-Policy-Report-Only	35	180	15280
X-Content-Security-Policy	N/A	111	1808
X-Webkit-CSP	N/A	194	728

Table 5.12: Historical CSP Header Use

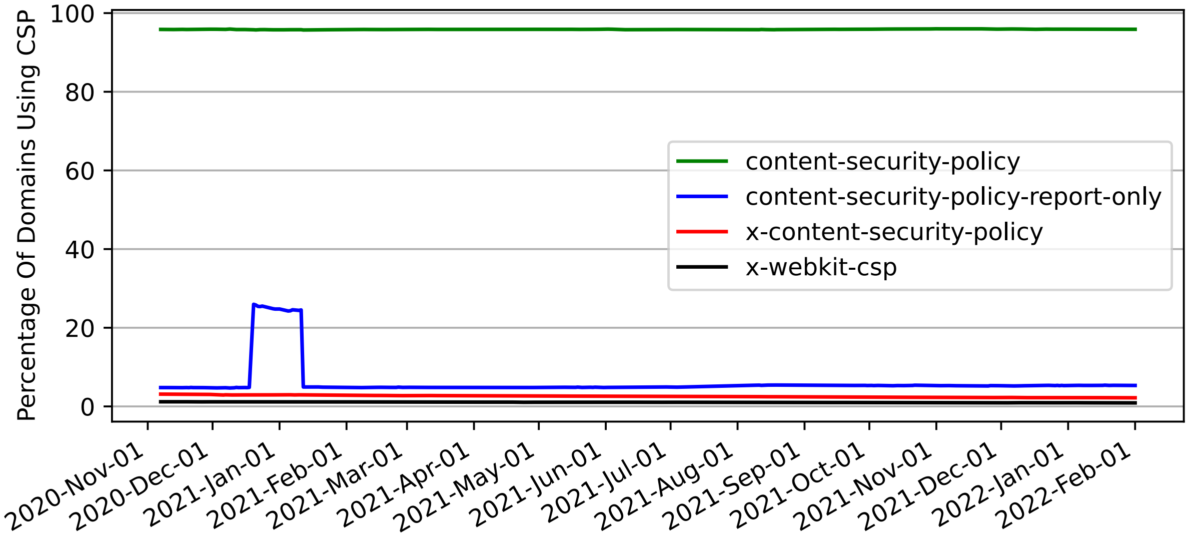


Figure 5.12: CSP headers use by name

Distribution by Header

Even though the `X-WebKit-CSP` and `X-Content-Security-Policy` CSP headers are now deprecated, there is still an increase in use from the `Ying et al` study [14] in June 2015 as shown in table 5.12. The current (Jan 2022) overall percentage of use for these two headers, however, is less that 4% of all the CSP polices from all the CSP headers combined.

Of the 4% of domains that use the `X-WebKit-CSP` and `X-Content-Security-Policy` headers, only 13% of these only use either of these two headers i.e. 87% of domains that use `X-WebKit-CSP` and `X-Content-Security-Policy` also use the `Content-Security-Policy`.

The presumed reason that domains use both deprecated and current CSP headers is backwards comparability for older browsers. Other possible reasons could be that legacy configurations that have been added to rather than removing the deprecated headers, or tooling in use automatically configures the deprecated headers.

There is a spike in use from 20 December 2020 to 11 January 2021 for the `Content-Security-Policy-Report-Only` header. The policy looks to be exclusively `worker-src 'none'; report-uri /csp-report` and all the `server` header values were *cloudflare*. Cloudflare provides many services for websites such as web increased performance via caching.

The CSP policy will send a report if a Worker, SharedWorker, or ServiceWorker script is attempted to be run. There does not seem to be any correlation between the websites other than using Cloudflare services. The short period the policy was deployed for, indicates that Cloudflare was doing some testing and or a beta program of some kind.

As of January 2022 33% of domains that are using the `Content-Security-Policy-Report-Only` header have not configured a `report-uri` or `report-to` directive, which results in reports not being sent upon policy violations. Developers might consider they do not as yet need to configure a reporting directive as they are still in the development phase and are using browser developer tools to identify policy violations.

Directives

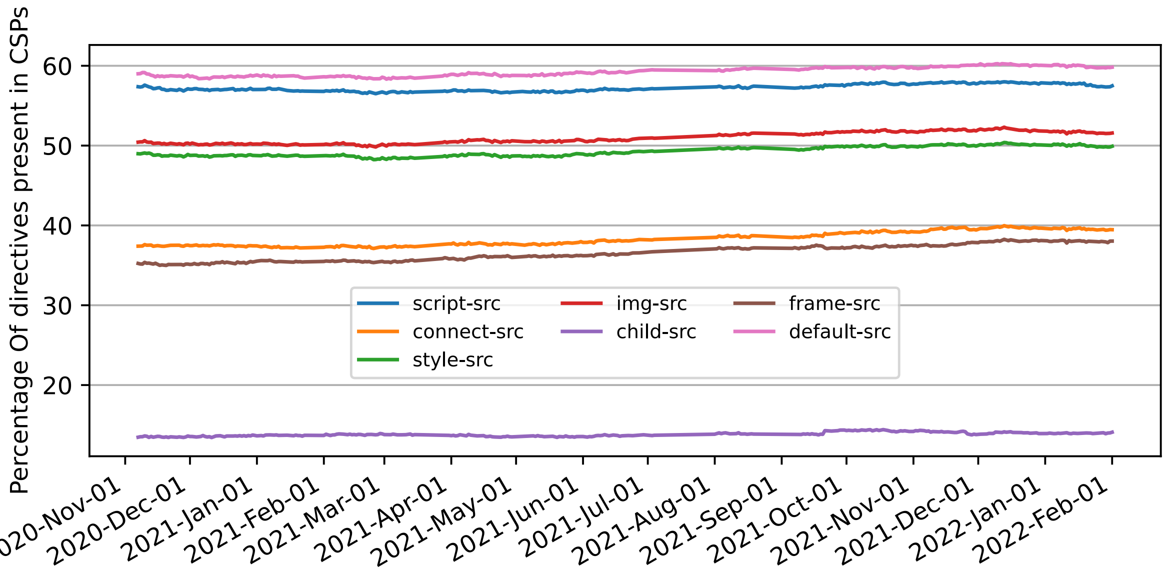


Figure 5.13: Percentage of directives present in the `Content-Security-Policy`

Figure 5.13 shows the percentage of directives present in the polices from the `Content-Security-Policy` header. The reason to single out `Content-Security-Policy` is that it is the only non deprecated header that is available for enforcing a CSP Policy as `Content-Security-Policy-Report-Only` merely reports on CSP violations.

default-src

Keyword	**Ying et al** [14] June 2015	Jan 2022
`*`	14.10%	4.40%
`none`	2.74%	9.3%
`self`	42.55%	72.76%
`unsafe-inline`	5.80%	37.56%
`unsafe-eval`	6.69%	31.14%
`unsafe-hashes`	N/A	8.38%

Table 5.13: Percentage of keyword use in default-src directive for unique CSP Policies

The distribution use of keywords has changed quite significantly since the study by Ying et al [14] in June 2015. It should be noted that there are also significantly more domains utilising CSP as well as the understanding of CSP will have increased which could very well have had an effect of keyword distribution change seen.

It is good to see that keyword * (essentially providing no protections) proportionally has reduced significantly from 14% to 4%. Unfortunately the proportional use of both unsafe-inline and unsafe-eval has over doubled.

The self keyword (restricting resources to the origin of the web page) sees a decent proportional increase from 43% to 73%. unsafe-hashes does not see that much use which is not surprising as it is a relatively new (introduced in 2018) keyword.

script-src

Keyword	**Ying et al** [14] June 2015	Jan 2022
*	14.34%	6.99%
none	0.00%	0.12%
self	51.09%	83.70%
unsafe-inline	58.66%	86.83%
unsafe-eval	62.61%	79.17%
unsafe-hashes	N/A	1.41%

Table 5.14: Percentage of keyword use in script-src directive for unique CSP Policies

It can be seen from table 5.13 that self has had a marked increase in proportional use from Ying et al [14] in June 2015 for the script-src directive. The use of keywords unsafe-inline and unsafe-eval have unfortunately significantly increased to over 87% and 79% respectively.

It is quite disappointing that the use of unsafe-inline and unsafe-eval is still so prevalent as setting these keywords allow a big attack vector for malicious actors. This indicates that website maintainers are struggling to be able to create CSP policies that are more restrictive but also do not impact the users experience, some may have even given up on the prospect.

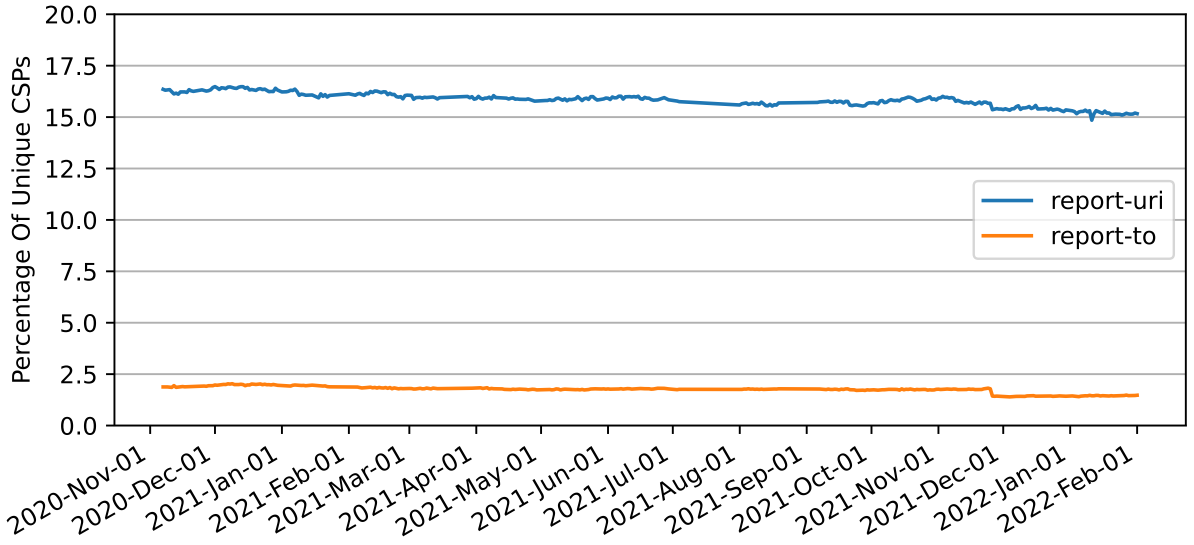


Figure 5.14: Percentage of reporting directives present in the `Content-Security-Policy`

Reporting Directives

The reporting directives are a great way for policy developers to gain feedback on the polices, via the violation reports, being written and deployed for issues such as missing sources that need to be allowed, spelling mistakes in URLs and other flaws with a configuration to name a few.

There are online services dedicated to receiving and processing the CSP policy violation reports which greatly aids in getting value from the reports and reduces the barrier to entry which can lead to more use of reporting directives that are configured to send violation reports.

The reports can also serve as a possible detection mechanism for malicious entities trying to or have already exploit a website.

The `report-uri` reporting directive, part of the CSP specification that was released in 2012, is in use by 15% of domains using the `Content-Security-Policy` in January 2022 and the `report-to` directive is in use by 1.5% of domains. The `report-uri` reporting directive is deprecated as of CSP 3, which is yet to be published (it is still in a working draft stage) which would point as to the reason `report-uri` is dominant over the `report-to` directive.

Figure 5.14 shows that the overall percentage use of the reporting directive is declining, however the number of domains using a CSP policy is increasing which indicates that figure 5.14 is showing as more domains start using a CSP policy they are not using a reporting directive or potentially domains are removing the reporting directives or even a combination of both.

5.5.4 Summary

The ability to utilise a CSP policy has been available since 2012. It is now 10 years on and in January 2022 only 83 thousand domains (9%) in the top 1 million ranked websites are using a CSP policy to protect their sites. This could indicate that website developers are finding it difficult to implement a CSP policy.

An alarming amount of `script-src` directive configurations are still using the `unsafe` prefixed keywords (86% that allow inline scripts) which allows one of the biggest attack vectors for websites that allow attacks such as cross site scripting attacks.

Websites are becoming more and more feature rich and being developed at an every increasing rate for many reasons such as staying competitive and demands from users. Libraries are often used along with numerous scripts created by the website developers which makes it very difficult or seemingly impossible to maintain a CSP policy that is both effective in protecting users as well as not impacting user experience (e.g. not blocking scripts that have changed and or added to add functionality to the website).

For a greater adoption of CSP, the frameworks and tooling used to create websites need to incorporate the facility to generate CSP polices as this would allow the ”system” to know which scripts and external sources are needed thus already knowing the information required to create a policy.

5.6 Strict Transport Security

Background for this mechanism was detailed in section 4.5.

5.6.1 Purpose Overview

The STS header prevents an attacker trying to get the user to use HTTP to access the site so that the attacker could perform an entity in the middle attack (described in section 2.1.4), allowing the attacker to impersonate the user or just capture sensitive information to name only a few possibilities.

There is an issue however, in that a browser can only know to load over HTTPS when the header is received. On the first visit to a website an attacker could intercept a HTTP request to the website and keep the connection HTTP and thus the user would not get the STS header. To overcome this, preloading is required which was discussed in the section 4.6.

5.6.2 Analysis

Figure 5.15 shows the usage of the STS header and its directives `max-age`, `includeSubDomains` and `preload`. It is difficult to see, however, the usage of the STS Header as directive `max-age` is present in almost every STS Header.

The usage of the directive `includeSubDomains` is consistently 40% of all directives used. Modern websites and applications can have a vast amount of sub-domains and as such will take time to make them all HTTPS if not already. This can be a major factor in the relatively low use of this directive.

There are many more sites that have the `preload` directive set (21.5%) than are in Google's preload list (3.5% - shown in figure 5.15 as “Preloaded Domains”). This could be websites preparing to be added to the preload list but as yet are not ready to meet the preload requirements. Currently there is no known use for using the `preload` directive other than being included in Google's preload list.

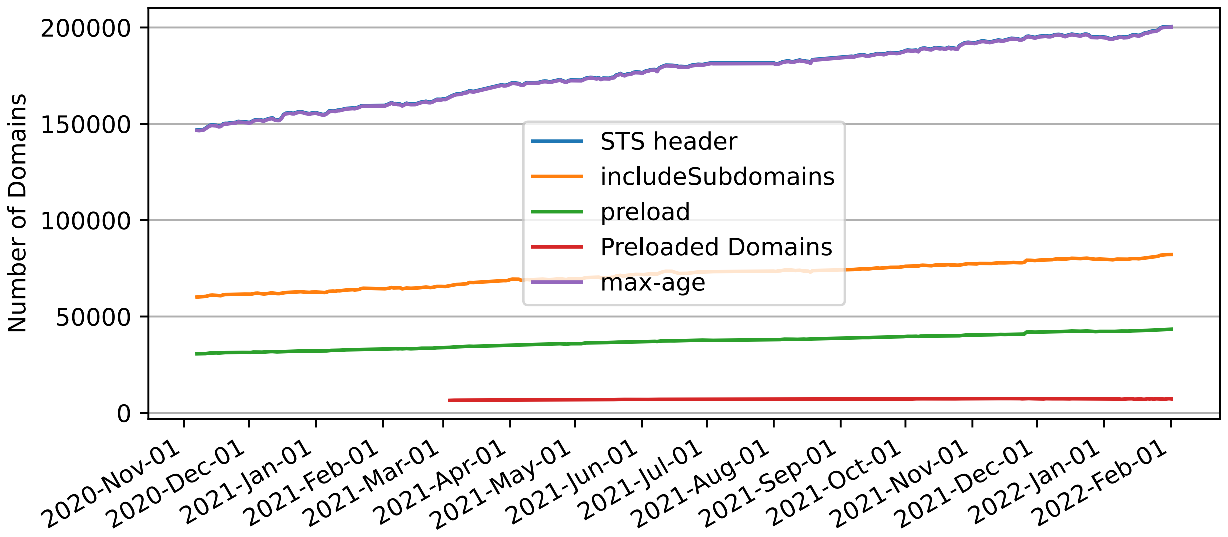


Figure 5.15: Number of Domains using the STS header and its directives

Weissbacher et al [100] March 2014	**Ying et al** [14] June 2015	**Buchanan et al** [10] May 2017	January 2022
3,005 (∼ 0.3%)	9,795 (∼ 0.9%)	45,527 (5.4%)	200493 (21.1%)

Table 5.15: Historical STS Header Use

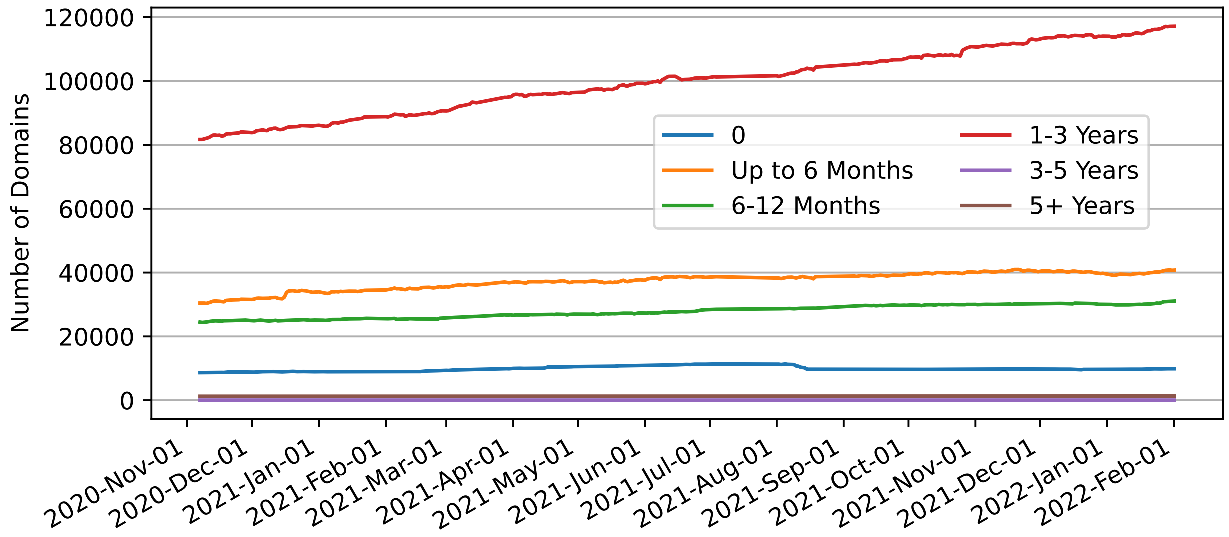


Figure 5.16: Duration groupings of the `max-age` directive

Figure 5.16 shows groups of the duration of the `max-age` directive. The most common duration is 1-3 years which is not unsurprising as the RFC states a value of 1 year. The upward trend is in the 1-3 years bracket which indicates that as sites add a STS header they are doing so with at least a year for the `max-age` directive.

This shows confidence that website maintainers are intending to keep their websites HTTPS for the foreseeable future.

As of January 2022 5% of the `max-age` directive values are set to a value of 0, down from 6% in November 2020. This essentially is "opting out" of the STS mechanism itself.

In table 5.15 it can be seen that there has been a near 5 fold increase in the use of the STS header from May 2017.

5.6.3 Summary

As of January 2022 21.1% of the top 1 million domains are using the STS header with an upward trend. This is quite promising to see and it is hoped the trend continues.

The most common value for the `max-age` directive is 1-3 years and a number of domains are using a value of 0 which means they are now actively opting out of the STS mechanism perhaps due to some endpoints not available over https as yet.

5.7 STS Preload

Background for this mechanism was detailed in section 4.6.

5.7.1 Purpose Overview

The main reason for STS Preloading is to ensure the first time a user visits a website it is over HTTPS. Without preloading, should a user visit the site the first time over HTTP, an attacker could intercept the traffic and perform malicious actions such as the entity in the middle attack as described in section 2.1.4.

5.7.2 Parser

The scanner was not initially configured to capture all the required information to be able to evaluate the conformance to the preload requirements. In March 2021 the base redirect and www subdomain check were added to the scanner.

The parser was configured to only analyse domains that were specifically added to the preload list. This is due to entire TLDs (e.g. .app, bank and .dev) being configured in the STS preload list and thus domains of these TLDs were not necessarily set up to meet the preload list criteria.

In the preload commit (Tue Nov 09 22:54:06 2021) there are 34 TLDs configured. This means that any site that is of one of these preload lists is preloaded (i.e. the domain itself is not in the preload list, only the TLD)

All of the STS preload lists were obtained from chromium.googlesource.com and the most recent list preceding the time of a scan was used to identify if a domain was preloaded or not.

A random selection of domains that the parser identified as both meeting the preload criteria and not meeting the preload criteria were checked on the https://STSpreload.org/ website for verification that the parser was working as intended.

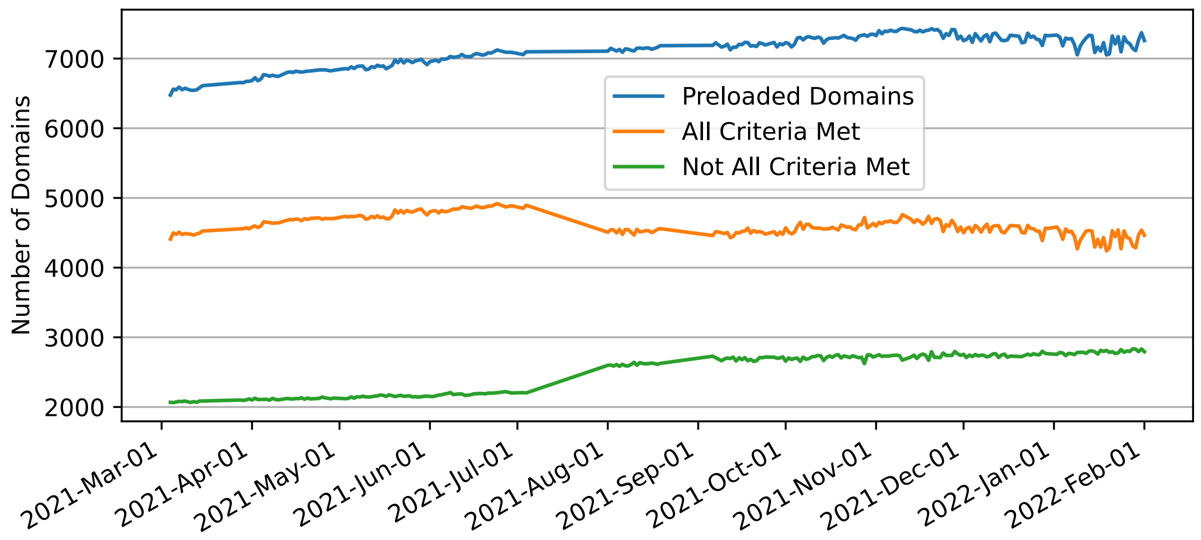


Figure 5.17: STS Preload Overview

5.7.3 Analysis

Preload Criteria Conformity

Figure 5.17 shows an overall summary of the current state of top 1 million domains that are in the preload list managed by Google. The number of preloaded domains is quite a low percentage. As of January 2022 only 0.72% of the top 1 million domains were preloaded.

The most recent STS preload list is from February 2022, as of January 2022, which contains 157192 (up from 1258 in 2010 [15]) entries which would seem to indicate that most of the preloaded domains are not in the top 1 million most popular domains.

The low amount of sites that are preloaded in the top 1 million most popular domains could be due to that the majority of sites being maintained by individuals with an interest in security and thus not likely to be a very popular domains compared to all the site on the internet.

As of January 2022 38% of domains preloaded do not meet the preload requirements anymore. This is rather a high percentage and these domains are at risk of being removed as per the message displayed on STSpreload.org when checking a domain that no longer meets the preload requirements: *"Status: <domain> is currently preloaded, but no longer meets the requirements. It may be at risk of removal."*.

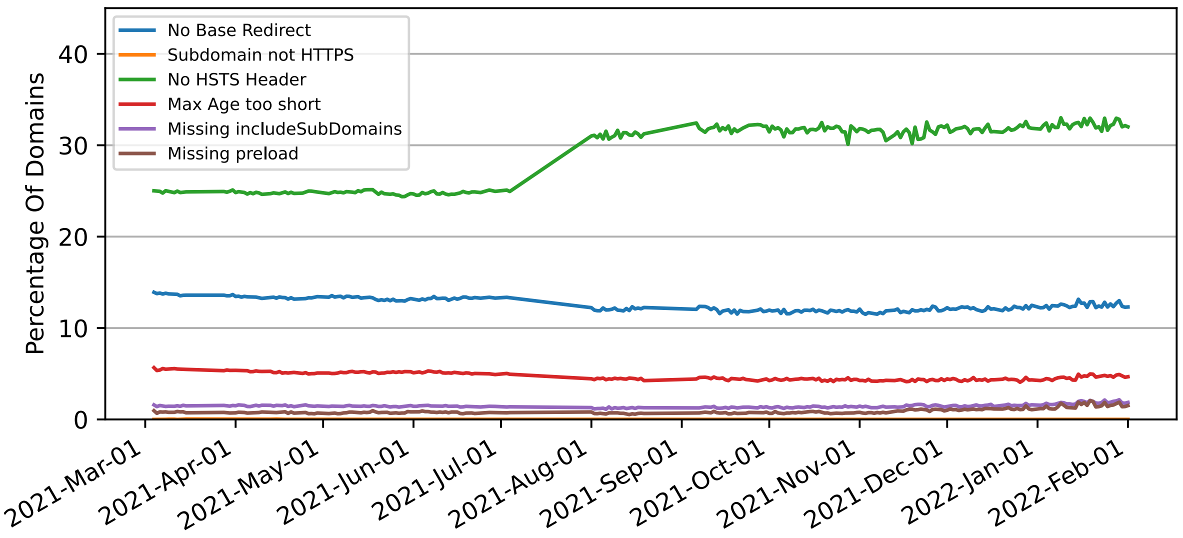


Figure 5.18: Criteria not met for preloaded domains

Domains in the preload list that no longer meet the criteria and were in the preload list before the requirements were formally documented and the web portal rolled out in 2014, are not subject for removal from the preload list however.

Any data/publications detailing actual removal of domains due to them no longer meeting the requirements was unable to be identified.

Criteria Not Met

Figure 5.18 shows the percentage of each criteria not met for domains that are preloaded from those that were scanned. The most prevalent criteria not met was the lack of presence of the STS header followed by not redirecting from HTTP to HTTPS on the base domain.

No Base Redirect

If a website is running a HTTP server on its base domain on port 80 it must redirect to HTTPS on the same domain. For example the base domain of `https://blog.example.com` is `example.com`. If a HTTP connection was able to be made to `http://example.com` it must then redirect to `https://example.com`.

As of January 2022 12.6% of preloaded domains no longer redirect correctly from the base domain. A reason for this could be that the redirect was changed from redirecting to the base domain over HTTPS e.g. `https://example.com` but to the main url of the website such as `https://wwww.example.com` to save a redirect to the desired main website url for the end user.

Subdomain not HTTPS

The `includeSubdomains` parameter must be included on the STS header, meaning that all subdomains of the base domain will be forced to load over HTTPS. Only the www subdomain is checked to see if it has a DNS entry and subsequently checked if the domain supports HTTPS on the www subdomain e.g. `https://www.example.com`.

As of January 2022 less than 0.1% of preloaded domains no longer use HTTPS on the www subdomain. This is not at all unsurprising, as it is unlikely that the www subdomain would have HTTPS support removed after it has been preloaded since it would be a primary subdomain to check for issues with HTTPS during the stage of making a domain preload compliant.

STS Header

- *No STS Header*

By far the most prevalent reason that a domain no longer met the preload criteria was a missing STS header. Many websites have numerous subdomains and it can be hard to keep track of all that are in use. Those managing the security aspect may not be aware of all the subdomains and by preloading the base domain could have caused issues if one or more subdomains were configured with only HTTP or invalid HTTPS configurations. This would lead to the request to have the base domain removed from the preload list.

- *Max Age too short*

4.2% of preloaded domains that presented a STS header, have a `max-age` directive value of less than one year (31,536,000 seconds). This also happens to be the value given in the STS RFC [9].

For 02 January 2022 the most common duration under a year was 180 days (6 months). There were also 3 domains that presented a max-age of 31104000 which is only 5 days below the required 365 days. 20 domains used a value of 0, essentially “opting out” of both the preloading and STS mechanism itself.

- *Missing preload*

1.6% of preloaded domains did not provide the directive `preload` in the STS header. As this directive is only used for the purpose of preloading, one might assume that this was done on purpose indicating that preloading was not longer desired.

- *Missing includeSubdomains*

1.8% of preloaded domains do not include the `includeSubdomains` directive and 0.8% are missing both the `includeSubdomains` directive and the `preload` directive.

5.7.4 Summary

The usage of STS Preloading is as yet not all that common with 0.72% of the top 1 million domains currently utilising it.

Of the domains that do use STS Preloading, 38% no longer meet the requirements which could result in their removal from the list thus removing the extra security layer the STS Preloading mechanism provides.

As of January 2022, not having a STS header was 30% of all reasons that a domain no longer met the preload requirement meaning they have opted out of both the STS header protection and STS Preloading.

5.8 X Content Type Options

Background for this mechanism was detailed in section 4.7.

5.8.1 Purpose Overview

This mechanism is intended to mitigate against such attacks as drive by downloads and untrusted user content being treated as an executable.

5.8.2 Analysis

It should be noted that in figure 5.19 representation for `XTCO header` cannot be seen. This is due to the fact that `nosniff` was the only value for this header and has an identical representation, thus it is completely overshadowing `XTCO header`.

Figure 5.19 shows the usage of the XCTO header and its single directive `nosniff`. As there is only one directive for XCTO, the number of domains using it and those using the directive `nosniff` are identical.

There is a somewhat linear increase in usage over the measurement period (November 2020 - January 2022) which is encouraging to see.

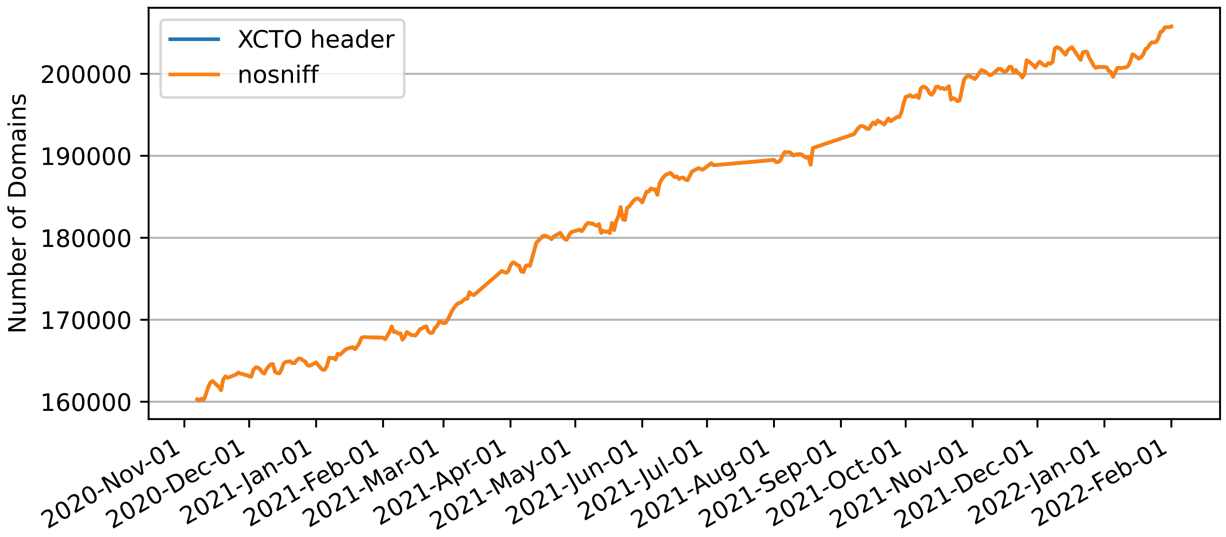


Figure 5.19: Number of Domains using the XCTO header and its directives

Weissbacher et al [100] March 2014	**Ying et al** [14] June 2015	**Buchanan et al** [10] May 2017	Jan 2022
44,954 (∼ 4.5%)	46,405 (∼ 4.6%)	89,053 (10%)	205,679 (21.5%)

Table 5.16: Historical XCTO Header Use

In table 5.16 it can be seen that there has been a near 5 fold increase in the use of the XTCO header from 2015 and just over a 2 fold increase since May 2017. As of January 2022 the XCTO header is in use by only 21% of scanned domains. Whilst the trend is on a linear increase trend, more awareness and guidance is needed to get a faster adoption of XCTO and the other security headers analysed in this report.

5.9 X Frame Options

Background for this mechanism was detailed in section 4.8.

5.9.1 Purpose Overview

The XFO mechanism intended to mitigate against such attacks as clickjacking where an attacker invisibly embeds another site on top of theirs tricking users, via the use of enticing offers such as free electronics, into clicking a seemingly harmless link which actually triggers a function on the embedded site such as transferring money via one's banking website to the attacker.

5.9.2 Analysis

Figure 5.20 shows the usage of the XFO header and its directives `deny`, `sameorigin` and the deprecated `allow-from`. The `sameorigin` directive has a near constant usage of 84% which is great to see and not all that unsurprising as it is not all that common for the need to embed foreign origins into ones website or the need for foreign origins to embed one's website.

That being said, with only a 21% usage of the XFO header as of January 2022, many more websites should be able to use the header with the `sameorigin`, or the more secure `deny`, directive without affecting the user experience and in turn protect the users from attacks such as clickjacking.

The `deny` directive has a fairly constant presence in domains with 14.3% use in January 2022. `deny` gives the most protection by not allowing the website to be embedded in any of the following html elements: <`frame`>, <`iframe`>, <`embed`>, or <`object`>.

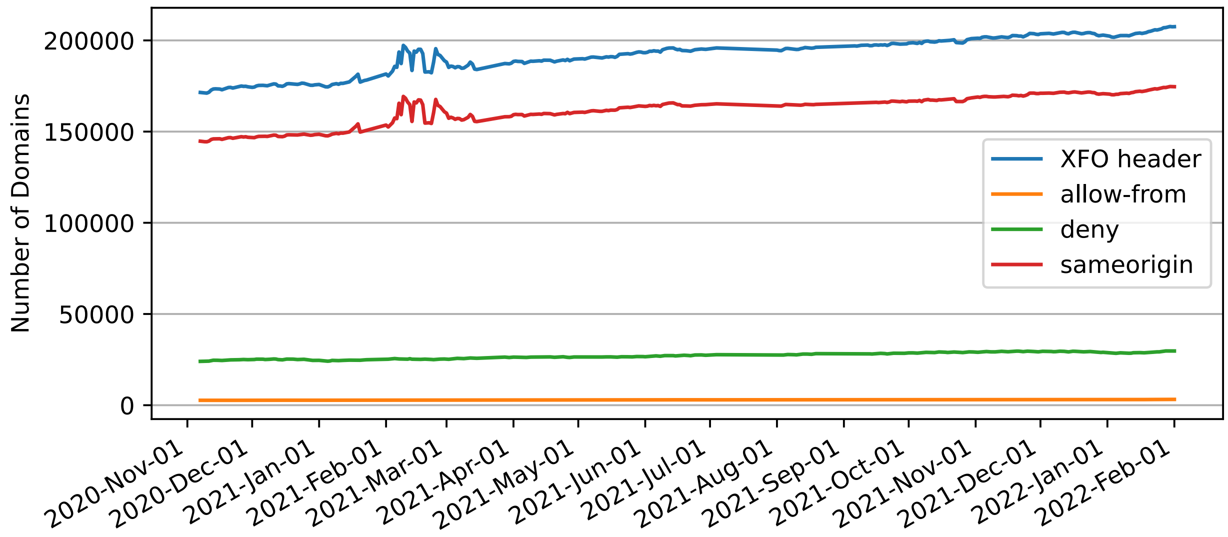


Figure 5.20: Number of Domains using the XFO header and its directives

Weissbacher et al [100] March 2014	**Ying et al** [14] June 2015	**Buchanan et al** [10] May 2017	Jan 2022
25,282 (∼ 2.5%)	40,848 (∼ 4.0%)	93,601 (11.1%)	207,522 (21.8%)

Table 5.17: Historical XFO Header Use

The deprecated directive `allow-from` has a very low but constant usage of 1.5%. The low usage reflects its deprecation status and is likely present due to configurations that have either not been changed in a while or not removed when configuration changes have been made.

5.10 Cross Origin Embedder Policy

Background for this mechanism was detailed in section 4.9.

5.10.1 Purpose Overview

The COEP mechanisms is to allow a website maintainer to enforce cross-origin resources that must be explicitly allowed to be loaded via the use of the CORP header or CORS otherwise the resources will be blocked from being loaded in the browsers.

A driving factor for this header being introduced was that some developers may implement CORS in minimal way, such as setting the requesters origin in the Access-Control-Allow-Origin header [31]. This may have the undesired affect of allowing potentially sensitive data to be accessed unintentionally.

5.10.2 Analysis

Figure 5.21 shows the usage of the COEP header and its directives `unsafe-none` and `require-corp`. The usage of the COEP header is rather low and as of January 2022 only 353 domains in the top 1 million most popular use it and of those only 15% of them use the `require-corp` directive (the only directive that imposes restrictions).

There is a sharp rise in its use in March 2021 with the directive of `unsafe-none` which indicated that the header was added perhaps for the purposes of an audit or similar that suggested its use as `unsafe-none` is the default behaviour (i.e. the same as not specifying the header at all).

Alternatively the sharp rise could be due to preparation of changing the headers directive to `require-corp` at a later date. However the directive appears to have remained set to `unsafe-none` for 9 months now.

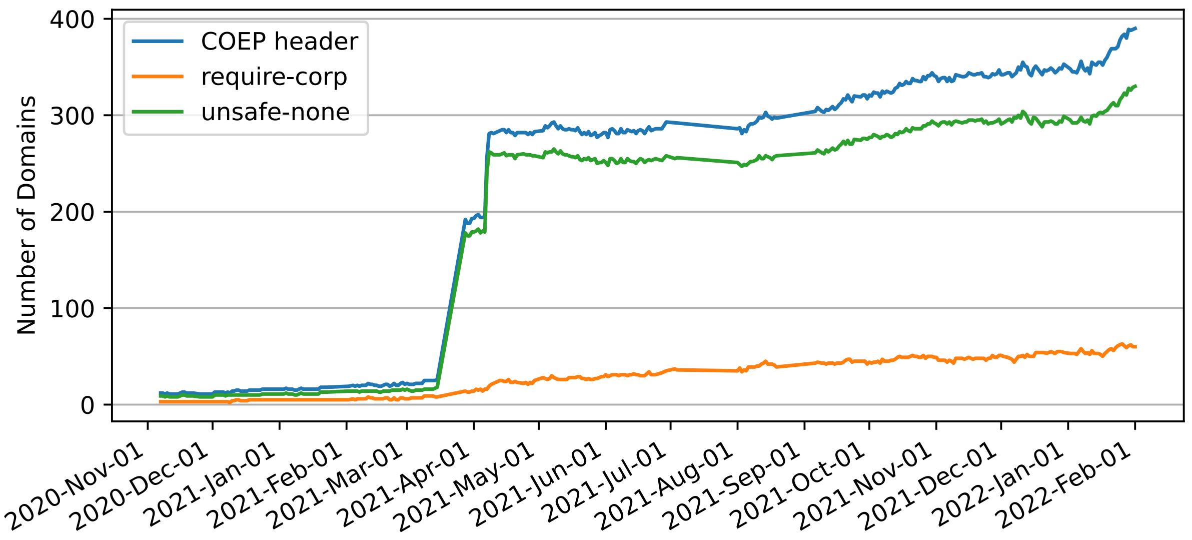


Figure 5.21: Number of Domains using the COEP header and its directives

The use of the `require-corp` directive is very low as of January 2022.

The COEP header was introduced recently enough that similar analysis of header usage was not seen in research consulted thus no comparison is able to be made.

5.11 Cross Origin Resource Policy

Background for this mechanism was detailed in section 4.10.

5.11.1 Purpose Overview

A primary class of attacks that this mitigates against are speculative side-channel attacks. The Spectre attack [32] is one such attack which was publicly announced in 2018.

5.11.2 Analysis

Figure 5.22 shows the usage of the CORP header and its directives `same-site`, `same-origin` and `cross-origin`. The usage of the CORP header almost double that of the COEP header in January 2022, but is still rather low.

The most common directive used is `same-origin`, 63% of all directives (387 domains) in January 2022, which the most restrictive of the available directives which is great to see.

The use of `cross-origin`, being the least restrictive directive, allows resources to be loaded from any source is the second most used with 157 domains and 25% of all directives used in January 2022. This would indicate the there are one or more resources that that are not from the same site as the website being visited or that website maintainers have added the header with the least restrictive directive or a combination of both.

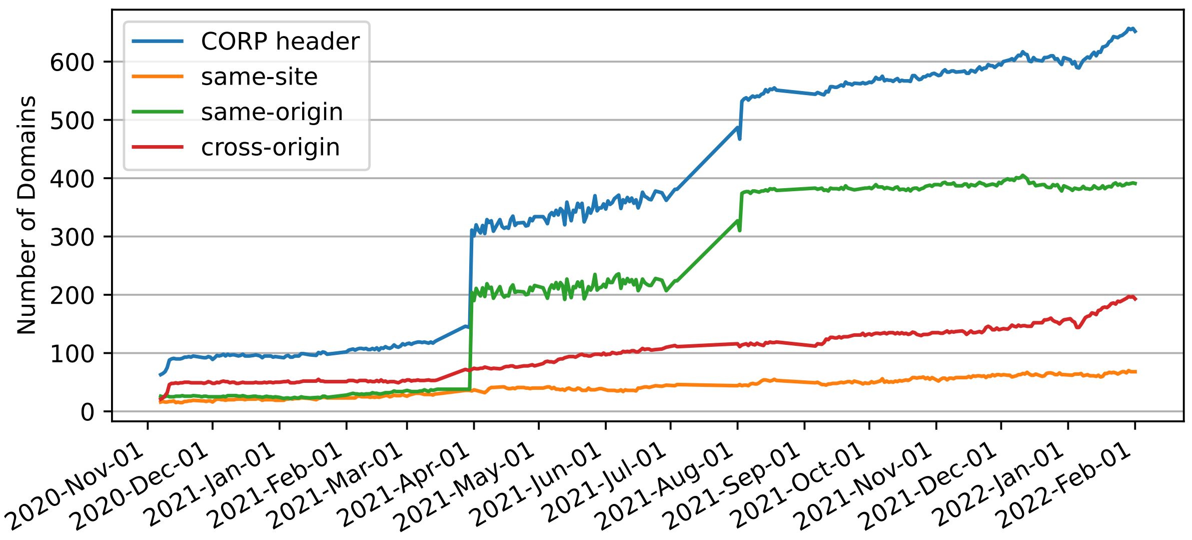


Figure 5.22: Number of Domains using the CORP header and its directives

Lastly `same-site` which allows resources to be loaded from any site whos origin matches the origin of the current website being visited is in use by 63 sites.

There are a couple of spikes. One in April 2021 and another in July 2021 for the use of `same-origin` directive.

The CORP header was introduced recently enough that similar analysis of header usage was not seen in research consulted, thus no comparison is able to be made.

5.12 Cross Origin Opener Policy

Background for this mechanism was detailed in section 4.11.

5.12.1 Purpose Overview

The COOP mechanism is to prevent a new window or tab spawned from a website from communicating back to a website. If an attacker is able to spawn this new window or tab that is of a site they control, COOP with the directive `same-origin` will prevent the attacker from communicating back to the website that spawned the new window or tab. This stops the attacker from trying to control the website or exfiltrating data from it.

5.12.2 Analysis

Figure 5.23 shows the usage of the COOP header and its directives `unsafe-none`, `same-origin-allow-popups` and `same-origin`. The usage of the COOP header is almost double that of the CORP header and quadruple that of the COEP header in January 2022, but is still rather low.

The most common directive used by far is `same-origin`, with 67% of all directives (1013 domains) in January 2022. It is the most restrictive of the available directives which is great to see. It is of no real surprise that this is the case and it is relatively safe to assume that most websites would want isolation between windows or tabs as most functionality required would be within the same tab/window.

The `same-origin-allow-popups` directive has the second most usage with 289 domains and 19% of all directives used in January 2022. Lastly the `unsafe-none` the least used with 194 domains and 13% of all directives used in January 2022.

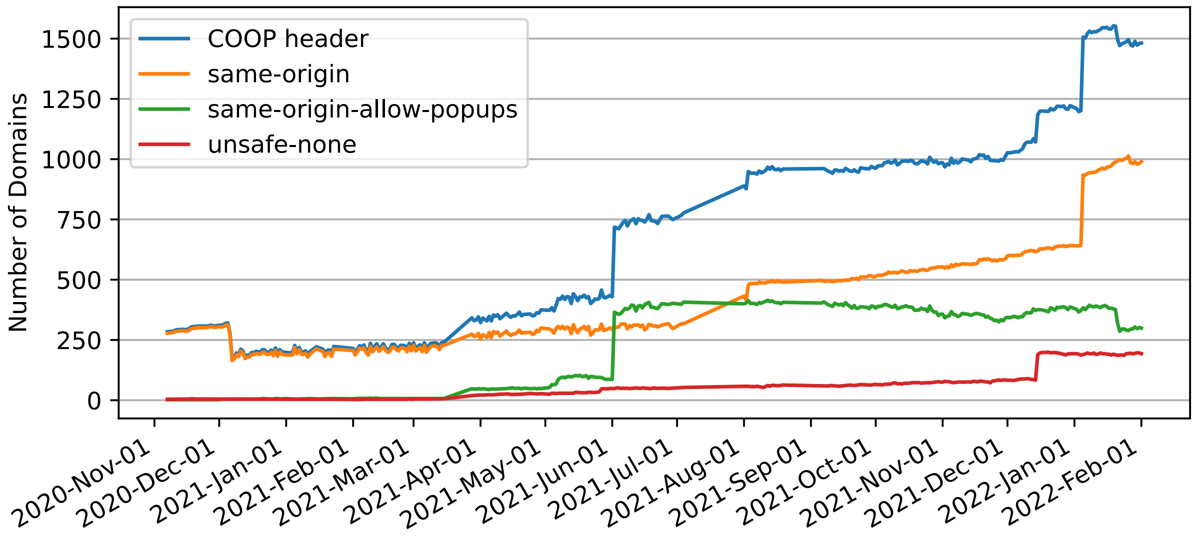


Figure 5.23: Number of Domains using the COOP header and its directives

The COOP header was introduced recently enough that similar analysis of header usage was not seen in research consulted, thus no comparison is able to be made.

5.13 Public Key Pins

Background for this mechanism was detailed in section 4.12.

5.13.1 Purpose Overview

This header was intended to protect websites in the event a CA provider was compromised, such as DigiNotar in 2011 [34], and unauthorised certificates being issued to an attacker for the purposes of impersonating websites. The HPKP mechanism would prevent the unauthorised certificate from being trusted by visitors that had previously visited the website i.e. preventing an entity in the middle attack.

5.13.2 Analysis

Figure 5.24 shows the usage of the HPKP header. As the header is deprecated by most modern browsers only the domains that used the header were analysed.

From November 2020 to mid March 2021 there was a steady decline, however since then usage has slightly fluctuated and the use in January 2022 is that of Mid March 2021. This indicates that there are a number of domains that have as yet to deprecate the HPKP header.

Table 5.18 shows that May 2017 was potentially the peak use of the PKP header and as of January 2022 there are only 526 sites using the header.

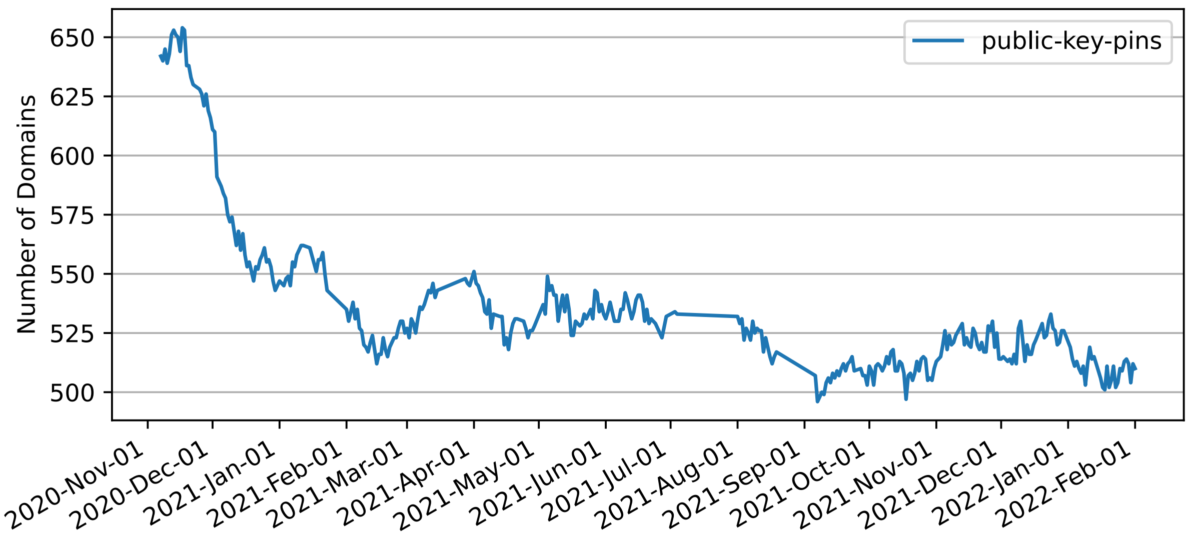


Figure 5.24: Number of Domains using the HPKP header

Weissbacher et al [100] March 2014	**Ying et al** [14] June 2015	**Buchanan et al** [10] May 2017	Jan 2022
NA	118 (∼ 0.01%)	6,624 (0.7%)	526 (0.05%)

Table 5.18: Historical PKP Header Use

Chapter 6

Discussion and Closing Remarks

6.1 Discussion

6.1.1 Objectives Scope Justification

As a refresher the scope for the objectives for this study are:

1. *The measurements will be restricted to websites as they are interacted with by a large proportion of people today, thus providing useful analysis targets.*
2. *Scan websites no more than once a day for a period of at least 12 months.*
3. *Restrict the websites scanned to the top 1 million most popular.*
4. *Only metadata, i.e. headers and connection details, will be captured and not the body content of websites.*

The reason for scanning against domains that are amongst the most popular was to analyse domains that had a high number of visitors. It would seem prudent to perform analysis on domains that had the highest amount off traffic as that should have the most number of users affected by security mechanism adoption on domains.

Scans were conducted at a maximum of once a day as it is unlikely any significant changes that were intended to persist would remain much longer than a single day and is also appropriate as this study is performing analysis over a large time period (16 months). Scanning once a day could be deemed as quite high, however an added benefit is if there were issues with some scans there should be enough remaining scans to allow effective analysis to occur. There were such issues with this research as detailed in section 5.1.1, which has occurred for other research, for example in [18].

The target domains for each scan was limited to the Tranco top 1 million list as it would have been impractical to generate a source list manually as there was limited time and resources which was better spent focussing on the scanning itself. There being 1 million domains in the Tanco list was deemed to be off sufficient amount and quality and as previously mentioned in section 3.3.1 many other studies have used such lists.

The use of a richer set of domains could have had an effect of the outcome of the analysis, however this is true of all measurement studies. Scanning a larger set of domains would provide a greater insight into security mechanism adoption however this could start to provide diminishing returns unless a significant amount of domains are used. A significantly larger set of domains might show sub trends within specific industries and or countries. Using a set of domains in an extreme range, such as in the 100's of millions or all of the domains registered for one or more TLD's (top level domains e.g. .com or .co.uk), would seem to be of some use if the desire was to use more than the top 1 million websites.

A secondary reason to use only 1 million domains was to reduce the resources required for them all to be scanned once a day as there is a real world cost to running the infrastructure required which was ∼$51 a month which equates to $816 for a 16 month period. This is not an insignificant amount of money for a personal budget.

With any measurement study one needs to set a limit to the amount of data to be analysed. The choice to restrict the data collection to only metadata was to limit the scope such that there was a reasonable chance of being able to sufficiently analyse the data in the limited time made available to complete this study. This looks to have been a wise choice as it has taken the majority of time available to complete this study.

6.1.2 Scanning

As detailed in section 5.1.1 there were several time periods of missing scan data not available for analysis. This essentially comes down to effective monitoring in order to reduce the impact. It is not possible to make a system 100% reliable, however extra vigilance such as frequent manual inspection could have reduce the impact to missing data.

Creating an application is easy, however, creating a stable system with effective monitoring takes a lot of effort and generally requires teams of people working in shifts to be very effective.

6.1.3 Analysis

A large proportion of time spent on the analysis phase was constructing the tooling to process the data from the domain scans. Great care had to be taken to verify the output of this tooling and on several occasions when analysing the charts generated, inaccuracies in the data representation could be seen and had to be corrected.

The analysis conducted was not merely just charting chosen data over the time period, significant deviations from trends warranted investigation. One such deviation, detailed in section 5.5.3, was a spike in `content-security-policy-report-only` header use for a short time period where cloudflare appeared to be conducting some sort of trial.

Unless one conducts a measurement study it is not particularly realised how much time it takes to collect and process the data into a usable form and how easy it is to misrepresent and or inaccurately represent the collected data. The study "Strategies for Sound Internet Measurement" [73] goes into great detail into what one needs to be aware of for measurement studies in order to obtain a more accurate outcome.

6.1.4 Future Work

There is a large scope for future studies due to the ever changing HTTPS ecosystem and its complexity. A number of possible areas for further work are detailed below that came to light during this study.

Legacy TLS Usage

For the websites the only support the legacy TLS versions, TLS 1.0 and TLS 1.1, it could be investigated into why this is the situation and leading to what could be done to support TLS 1.2 and TLS 1.3.

STS Preloaded sites that no longer meet the preload criteria

There are rather a large proportion of websites that are currently STS preloaded but no longer meet the criteria. It could be investigated why this is the case and thus could lead to enhanced documentation in how to better prepare for and keep conformity for preloading and or enhanced organisational procedures to prevent unintentionally no longer meeting one or more of the preload criteria.

Content Security Policy - Unsafe keyword usage

The use of the `unsafe` prefixed keywords especially for the `script-src` directive is relatively high currently. If a study was able to uncover the reasons for this it could help both the website operators trying to implement secure policies as well as the body that maintains the CSP standard in order to work together to best protect users with CSP policies.

Continued Adoption Studies

There is always a need for adoption studies as it gives insight into how security mechanisms are being used, which influences the evolution of such mechanisms, provides guidance for which mechanisms should be looked into more deeply and a historical record for longitudinal research.

6.2 Conclusions

The overarching goal of this study was to answer the question:

`What is the current adoption of security mechanisms in the HTTPS ecosystem ?`

This was achieved by scanning the top 1 million ranked websites (as determined by Tranco [58]) each day for 16 months and analysing the results.

Throughout the study the number of websites supporting HTTPS remained at a near constant of ∼80% indicating that a plateau of websites supporting HTTPS has been reached. For these remaining sites it may very likely need a simple, potentially, free service to get these sites supporting HTTPS. There are a number of these services available, however it may seem too much of a barrier as the operators may not see HTTPS as a necessary feature of their site unfortunately.

With TLS 1.3 climbing from 50% in November 2020 to 64% in January 2022 (as shown in figure 5.3), of websites scanned that support HTTPS, there a is strong support for the latest TLS versions and this trend looks as thou it will continue to rise providing more security for users browsing the internet. Also encouraging to see is the continued decline of the support of TLS 1.0 and TLS 1.1 (as shown in figure 5.4) which could indicate the website operators are increasingly comfortable to stop supporting these legacy versions possibly as there are a minimal amount of devices that require them still in use.

The security.txt mechanisms is still in its infancy as it is still in the draft RFC stage which shows in its adoption which is 0.5% (as stated in section 5.4.5) of websites scanned as of January 2022. The goal of this mechanism is admirable and its relevance looks quite promising as the majority of security.txt files found have a contact field defined, which is the primary piece of information that security researches are looking for in order to report a security vulnerability. It is hoped that more tutorials and setup procedures for website hosting, prompts for the information for a security.txt to be deployed such that a wider adoption can be made.

A Content Security Policy is a critical mechanism in assisting to mitigate cross site scripting attacks, however only 9% of domains scanned are using a CSP policy as of January 2022. The use of `usafe-inline` and `unsafe-eval` keywords are still very prevalent, especially for the `script-src` directive (as shown in table 5.14) which negates one of the strongest protections against XSS attacks. This is likely due to several contributing factors including the complexity of websites and the amount of work required to refactor website code in order not to have the need for these directives. This finding is quite disappointing and an in depth study should be conducted to uncover the definitive primary reasons why this is still the case.

As previously stated there are several time periods during the data capture phase where there is missing data, which is not all that uncommon for this type of study. The reasons for the missing data has been identified (as detailed in section 3.4.6), along with measures intended to prevent further occurrences which will hopefully allow future studies to learn from them. The largest gap is 27 days and even though this is almost a month of data, it seems not to have been an issue for analysis and the identification of longitudinal trends. The main issue is with the identification of short live spikes/fluctuations, such as that seen by the use of the CSP report only header.

Another critique of this study is of the choice not to retry HTTP(S) or TLS connections that failed in order to reduce the load on the target domains. If retries had been utilised it is quite possible that some of these failed HTTP(S) or TLS connections would have been successful, however this is likely not to be significant enough to have changed the analysis findings.

To conclude, apart from TLS the other security mechanisms see a rather low usage within the top 1 million ranked websites. More work should be done to improve their usage (such as better tutorials, prompts during setup wizzards, default configs and best practice guidance) as well as finding out why the usage is as low as it is.

Intentionally Blank

Bibliography

[1] I. Ristic, *Bulletproof SSL and TLS.* Feisty Duck, Jul. 2017.

[2] Apple, Google, Mozilla, and Microsoft, "DOM standard," *URL https://dom.spec.whatwg.org/; Accessed 04 November 2021*.

[3] T. Berners-Lee and D. Connolly, "RFC 1866: Hypertext markup language - 2.0, august 1995," *URL https://datatracker.ietf.org/doc/html/rfc1866; Accessed 24 October 2021*, 1995.

[4] T. Berners-Lee, R. Fielding, and H. Frystyk, "RFC1945: Hypertext transfer protocol - HTTP/1.0, may 1996," *URL https://datatracker.ietf.org/doc/html/rfc1945; Accessed 24 October 2021*, 1996.

[5] N. Freed, J. Klensin, and T. Hansen, "RFC6838: Media type specifications and registration procedures," *URL https://datatracker.ietf.org/doc/html/rfc6838; Accessed 02 November 2021*, 2013.

[6] R. Oppliger, *SSL and TLS: Theory and Practice, Second Edition.* Artech House, Mar. 2016.

[7] C.-H. j. Wu and J. David Irwin, *Introduction to Computer Networks and Cybersecurity.* CRC Press, Apr. 2016.

[8] E. Rescorla, "RFC8446: The transport layer security (TLS) protocol version 1.3, august 2018," *URL https://datatracker.ietf.org/doc/html/rfc8446; Accessed 24 October 2021*, 2018.

[9] J. Hodges, C. Jackson, and A. Bart, "RFC6797: HTTP strict transport security (HSTS), november 2012," *URL https://datatracker.ietf.org/doc/html/rfc6797; Accessed 24 October 2021*, no. rfc6797, 2012.

[10] W. J. Buchanan, S. Helme, and A. Woodward, "Analysis of the adoption of security headers in HTTP," *IET Inf. Secur.*, vol. 12, no. 2, pp. 118–126, Mar. 2018.

[11] P. Chen, L. Desmet, C. Huygens, and W. Joosen, "Longitudinal study of the use of client-side security mechanisms on the european web," in *Proceedings of the 25th International Conference Companion on World Wide Web*, ser. WWW '16 Companion. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, Apr. 2016, pp. 457–462.

[12] D. Kumar, Z. Ma, Z. Durumeric, A. Mirian, J. Mason, J. A. Halderman, and M. Bailey, "Security challenges in an increasingly tangled web," in *Proceedings of the 26th International Conference on World Wide Web*, ser. WWW '17. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, Apr. 2017, pp. 677–684.

[13] K. Patil and Vishwakarma Institute of Information Technology, India, "An insecure wild web: A large-scale study of effectiveness of web security mechanisms," *ICTACT j. commun. technol.*, vol. 08, no. 01, pp. 1466–1471, Mar. 2017.

[14] M. Ying and S. Q. Li, "CSP adoption: current status and future prospects," *Secur. Commun. Netw.*, vol. 9, no. 17, pp. 4557–4573, Nov. 2016.

[15] K. Michael and B. Joseph, "Upgrading https in mid-air: An empirical study of strict transport security and key pinning," in *NDSS Symposium*, 2015.

[16] T. van Goethem, P. Chen, N. Nikiforakis, L. Desmet, and W. Joosen, "Large-Scale security analysis of the web: Challenges and findings," in *Trust and Trustworthy Computing*. Springer International Publishing, 2014, pp. 110–126.

[17] R. Holz, J. Hiller, J. Amann, A. Razaghpanah, T. Jost, N. Vallina-Rodriguez, and O. Hohlfeld, "Tracking the deployment of tls 1.3 on the web: A story of experimentation and centralization," *Comput. Commun. Rev.*, vol. 50, no. 3, pp. 3–15, Jul. 2020.

[18] T. Poteat and F. Li, "Who you gonna call? an empirical evaluation of website security.txt deployment," in *Proceedings of the 21st ACM Internet Measurement Conference*, ser. IMC '21. New York, NY, USA: Association for Computing Machinery, Nov. 2021, pp. 526–532.

[19] E. Rescorla, “RFC2818: HTTP over TLS, may 2000,” *URL https://datatracker.ietf.org/doc/html/rfc2818; Accessed 24 October 2021*, 2000.

[20] “Preparing to issue 200 million certificates in 24 hours,” https://letsencrypt.org/2021/02/10/200m-certs-24hrs.html, accessed: 2021-12-22.

[21] “HTTPS as a ranking signal,” https://developers.google.com/search/blog/2014/08/https-as-ranking-signal, accessed: 2021-12-22.

[22] T. Vyas, “Communicating the dangers of non-secure HTTP,” https://blog.mozilla.org/security/2017/01/20/communicating-the-dangers-of-non-secure-http/, Jan. 2017, accessed: 2021-12-22.

[23] Google, “Moving towards a more secure web,” https://security.googleblog.com/2016/09/moving-towards-more-secure-web.html, accessed: 2021-12-22.

[24] “Serve websites over HTTPS (always),” https://www.ncsc.gov.uk/blog-post/serve-websites-over-https-always, accessed: 2021-12-16.

[25] “Does my site need HTTPS?” https://doesmysiteneedhttps.com/, accessed: 2021-12-16.

[26] S. Preston, *Learn HTML5 and JavaScript for iOS*. Apress, 2012.

[27] Multiple, “Same origin policy,” *URL https://html.spec.whatwg.org/multipage/browsers.html#cross-origin-objects; Accessed 01 November 2021*, 1996.

[28] Apple, Google, Mozilla, and Microsoft, “CORS protocol,” *URL https://fetch.spec.whatwg.org/#cors-protocol; Accessed 03 November 2021*, 2006.

[29] E. Foudil and Y. Shafranovich, “RFC-DRAFT: A file format to aid in security vulnerability disclosure,” *URL: https://datatracker.ietf.org/doc/html/draft-foudil-securitytxt-12; Accessed 03 November 2021*, 2021.

[30] Apple, Google, Mozilla, and Microsoft, “X-Content-Type-Options header,” *URL https://fetch.spec.whatwg.org/#x-content-type-options-header; Accessed 02 November 2021.*

[31] “Cross-origin embedder policy,” https://wicg.github.io/cross-origin-embedder-policy/, accessed: 2022-2-7.

[32] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019, pp. 1–19.

[33] Apple, Google, Mozilla, and Microsoft, "Cross-origin opener policies," *URL: https://html.spec.whatwg.org/multipage/origin.html#the-headers; Accessed 03 November 2021.*

[34] J. Amann, O. Gasser, Q. Scheitle, L. Brent, G. Carle, and R. Holz, "Mission accomplished? HTTPS security after diginotar," in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC '17. New York, NY, USA: Association for Computing Machinery, Nov. 2017, pp. 325–340.

[35] J. Herman and Y. Klijnsma, "Magecart: The state of a growing threat," *Computer Fraud & Security*, vol. 2019, no. 10, p. 4, Oct. 2019.

[36] A. Gezer, G. Warner, C. Wilson, and P. Shrestha, "A flow-based approach for trickbot banking trojan detection," *Computers & Security*, vol. 84, pp. 179–192, Jul. 2019.

[37] D. M. Mrdjenovich, "Saving the electronic person from digital assault: The case for more robust protections over our electronic medical records," *Duquesne Law Rev.*, vol. 58, p. 146, 2020.

[38] "OWASP foundation," https://owasp.org/about/, accessed: 2021-11-4.

[39] D. Kellezi, C. Boegelund, and W. Meng, "Securing open banking with Model-View-Controller architecture and OWASP," *Proc. Int. Wirel. Commun. Mob. Comput. Conf.*, vol. 2021, Sep. 2021.

[40] "OWASP top 10:2021," https://owasp.org/Top10/, accessed: 2021-11-4.

[41] "OWASP top ten 2021 - injection attacks," https://owasp.org/Top10/A03_2021-Injection/, accessed: 2021-11-4.

[42] Tala Security, "Global data at risk state of the web report," *https://go.talasecurity.io/hubfs/Content/White%20Papers%20and%20Reports/_Global%20Data%20* Jul. 2020, accessed: 2021-12-15.

[43] G. E. Rodríguez, J. G. Torres, P. Flores, and D. E. Benavides, “Cross-site scripting (XSS) attacks and mitigation: A survey,” *Computer Networks*, vol. 166, p. 106960, Jan. 2020.

[44] A. Klein, “DOM based cross site scripting or XSS of the third kind,” *Web Application Security Consortium, Articles*, vol. 4, pp. 365–372, 2005.

[45] M. Heiderich, J. Schwenk, T. Frosch, J. Magazinius, and E. Z. Yang, “mXSS attacks: attacking well-secured web-applications by using innerHTML mutations,” in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, ser. CCS ’13. New York, NY, USA: Association for Computing Machinery, Nov. 2013, pp. 777–788.

[46] C.-H. Lee, S. Tenneti, and D. Y. Eun, “Transient dynamics of epidemic spreading and its mitigation on large networks,” in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc ’19. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 191–200.

[47] E. Tekiner, A. Acar, A. S. Uluagac, E. Kirda, and A. A. Selcuk, “In-Browser cryptomining for good: An untold story,” in *2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, Aug. 2021, pp. 20–29.

[48] A. Billman, “Cryptojacking: Abusing computational power for profit,” Ph.D. dissertation, 2018.

[49] K. S. Jamwal, “Clickjacking attack: Hijacking user’s click,” *International Journal of Advanced Networking and Applications*, vol. 10, no. 1, pp. 3735–3740, Aug. 2018.

[50] N. B. Jani and B. B. Panchal, “A critical review of scriptless timing attacks and web browser privacy,” *IJSRD - International Journal for Scientific Research & Development*, vol. 3, no. 1, pp. 46–48, 2015.

[51] O. Levillain, B. Gourdin, and H. Debar, “TLS record protocol: Security analysis and defense-in-depth countermeasures for HTTPS,” in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS ’15. New York, NY, USA: Association for Computing Machinery, Apr. 2015, pp. 225–236.

[52] N. J. Al Fardan and K. G. Paterson, "Lucky thirteen: Breaking the TLS and DTLS record protocols," in *2013 IEEE Symposium on Security and Privacy*, May 2013, pp. 526–540.

[53] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey, and J. A. Halderman, "The matter of heartbleed," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, ser. IMC '14. New York, NY, USA: Association for Computing Machinery, Nov. 2014, pp. 475–488.

[54] B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, A. Pironti, P.-Y. Strub, and J. K. Zinzindohoue, "A messy state of the union: Taming the composite state machines of TLS," 2015.

[55] C. Partridge and M. Allman, "Ethical considerations in network measurement papers," *Commun. ACM*, vol. 59, no. 10, pp. 58–64, Sep. 2016.

[56] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by Internet-Wide scanning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, Oct. 2015, pp. 542–553.

[57] BCS, "CODE OF CONDUCT FOR BCS MEMBERS," Jun. 2011.

[58] "Tranco," https://tranco-list.eu/, accessed: 2021-11-17.

[59] "Cisa: Report incidents, phishing, malware, or vulnerabilities," https://us-cert.cisa.gov/report, accessed: 2021-12-1.

[60] P. Kotzias, A. Razaghpanah, J. Amann, K. G. Paterson, N. Vallina-Rodriguez, and J. Caballero, "Coming of age: A longitudinal study of TLS deployment," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18. New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 415–428.

[61] S. Roth, T. Barron, S. Calzavara, N. Nikiforakis, and others, "Complex security policy? a longitudinal analysis of deployed content security policies," *Proceedings of the 27th*, 2020.

[62] S. Calzavara, A. Rabitti, and M. Bugliesi, "Semantics-Based analysis of content security policy deployment," *ACM Trans. Web*, vol. 12, no. 2, pp. 1–36, Jan. 2018.

[63] "Alexa - top sites," https://www.alexa.com/topsites, accessed: 2021-11-17.

[64] "Cisco popularity list," http://s3-us-west-1.amazonaws.com/umbrella-static/index.html, accessed: 2021-11-17.

[65] "Majestic million - majestic," https://majestic.com/reports/majestic-million, accessed: 2021-11-17.

[66] "ICANN centralised zone data service," https://czds.icann.org/home, accessed: 2021-11-17.

[67] "Go HTTP client GET method," https://pkg.go.dev/net/http, accessed: 2021-11-17.

[68] A. Lavrenovs and F. J. R. Melón, "HTTP security headers analysis of top one million websites," in *2018 10th International Conference on Cyber Conflict (CyCon)*, May 2018, pp. 345–370.

[69] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, and Housley, R: Polk, W, "RFC5280: Internet x.509 public key infrastructure certificate and certificate revocation list (CRL) profile," 2008.

[70] D. Keeler, "Preloading intermediate CA certificates into firefox," https://blog.mozilla.org/security/2020/11/13/preloading-intermediate-ca-certificates-into-firefox, Nov. 2020, accessed: 2021-11-18.

[71] J. Clark and P. C. van Oorschot, "SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements," in *2013 IEEE Symposium on Security and Privacy*, May 2013, pp. 511–525.

[72] R. Holz, L. Braun, N. Kammenhuber, and G. Carle, "The SSL landscape: a thorough analysis of the x.509 PKI using active and passive measurements," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '11. New York, NY, USA: Association for Computing Machinery, Nov. 2011, pp. 427–444.

[73] V. Paxson, "Strategies for sound internet measurement," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, ser. IMC '04. New York, NY, USA: Association for Computing Machinery, Oct. 2004, pp. 263–271.

[74] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A Research-Oriented top sites ranking hardened against manipulation," Jun. 2018.

[75] "http: http2pipe.closeWithErrorCrashed · issue #43965 · golang/go," https://github.com/golang/go/issues/43965, accessed: 2021-12-9.

[76] K. M. Martin, *Everyday Cryptography: Fundamental Principles and Applications*. Oxford University Press, 2017.

[77] D. Wagner and B. Schneier, "Analysis of the SSL 3.0 protocol," in *The Second USENIX Workshop on Electronic Commerce Proceedings*, vol. 1, 1996, pp. 29–40.

[78] A. Freier, P. Karlton, and P. Kocher, "RFC6101: The secure sockets layer (SSL) protocol version 3.0, november 1996," *URL https://datatracker.ietf.org/doc/html/rfc6101; Accessed 24 October 2021*, 2011.

[79] S. Farrell, "Why didn't we spot that? [practical security]," *IEEE Internet Computing*, vol. 14, no. 1, p. 85, Jan. 2010.

[80] T. Dierks and C. Allen, "RFC2246: The TLS protocol version 1.0, january 1999," *URL https://datatracker.ietf.org/doc/html/rfc2246; Accessed 24 October 2021*, 1999.

[81] E. Rescorla, *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley, 2001.

[82] T. Dierks and E. Rescorla, "RFC4346: The transport layer security (TLS) protocol version 1.1, april 2006," *URL https://datatracker.ietf.org/doc/html/rfc4346; Accessed 24 October 2021*, 2006.

[83] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright, "RFC3546: Transport layer security (TLS) extensions," *URL https://datatracker.ietf.org/doc/html/rfc3546; Accessed 30 October 2021*, 2003.

[84] T. Dierks and E. Rescorla, "RFC5246: The transport layer security (TLS) protocol version 1.2, august 2008," *URL https://datatracker.ietf.org/doc/html/rfc5246; Accessed 24 October 2021*, 2008.

[85] A. Barth and B. Sterne, "Content security policy 1.0, 2012," *URL https://www.w3.org/TR/2012/CR-CSP-20121115/; Accessed 01 November 2021*, 2012.

[86] A. Barth and B. Sterne, "Content security policy 1.0, 2015," *URL https://www.w3.org/TR/2015/NOTE-CSP1-20150219/; Accessed 01 November 2021*, 2015.

[87] M. West, A. Barth, D. Veditz, and B. Sterne, "Content security policy level 2, 2014," *URL https://www.w3.org/TR/2014/WD-CSP2-20140703/; Accessed 01 November 2021*, 2014.

[88] M. West, A. Barth, D. Veditz, and B. Sterne, "Content security policy level 2, 2016," *URL https://www.w3.org/TR/2016/REC-CSP2-20161215; Accessed 01 November 2021*, 2016.

[89] M. West, "Content security policy level 3; 2016," *URL https://www.w3.org/TR/2016/WD-CSP3-20160126/; Accessed 01 November 2021*, 2016.

[90] M. West, "Content security policy level 3, 2021," *URL https://www.w3.org/TR/2021/WD-CSP3-20210629/; Accessed 01 November 2021*, 2021.

[91] "Remove plugin-types," accessed: 2022-01-29.

[92] Google, "Rich notifications in chrome," https://blog.chromium.org/2013/05/, accessed: 2022-2-1.

[93] A. Langley, "Strict transport security," https://www.imperialviolet.org/2010/01/26/sts.html, accessed: 2022-2-12.

[94] Apple, Google, Mozilla, and Microsoft, "Cross-Origin-Resource-Policy header," *URL https://fetch.spec.whatwg.org/#cross-origin-resource-policy-header; Accessed 02 November 2021.*

[95] P. Chen, "Empirical study on the use of client-side web security mechanisms," Ph.D. dissertation, ARENBERG DOCTORAL SCHOOL, Sep. 2018.

[96] L. Chuat, C. Krähenbühl, P. Mittal, and A. Perrig, "F-PKI: Enabling innovation and trust flexibility in the HTTPS Public-Key infrastructure," Aug. 2021.

[97] C. Evans and Palmer, C: Sleevi, R, "RFC7469: Public key pinning extension for HTTP," *URL: https://datatracker.ietf.org/doc/html/rfc7469, Accessed 28 November 2021*, 2018.

[98] "Matplotlib — visualization with python," https://matplotlib.org/, accessed: 2022-1-23.

[99] w3c, "Subresource integrity," https://w3c.github.io/webappsec-subresource-integrity/, accessed: 2022-3-27.

[100] M. Weissbacher, T. Lauinger, and W. Robertson, "Why is CSP failing? trends and challenges in CSP adoption," in *Research in Attacks, Intrusions and Defenses*. Springer International Publishing, 2014, pp. 212–233.

Appendices

A Task Agent TLS Client

The below list contains the TLS 1.0 - TLS 1.2 cipher suites that the Task Agent supported:

- TLS_RSA_WITH_RC4_128_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_RC4_128_SHA
- TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256

The below list contains the TLS 1.3 cipher suites that the Task Agent supported:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256