Stream: Internet Engineering Task Force (IETF)
RFC: 9163
Category: Experimental
Published: June 2022
ISSN: 2070-1721
Author: E. Stark
Google

RFC 9163
Expect-CT Extension for HTTP

Abstract

This document defines a new HTTP header field named "Expect-CT", which allows web host operators to instruct user agents (UAs) to expect valid Signed Certificate Timestamps (SCTs) to be served on connections to these hosts. Expect-CT allows web host operators to discover misconfigurations in their Certificate Transparency (CT) deployments. Further, web host operators can use Expect-CT to ensure that if a UA that supports Expect-CT accepts a misissued certificate, that certificate will be discoverable in Certificate Transparency logs.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9163.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions

with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

This document defines a new HTTP header field ([RFC9110], Section 6.3) that enables UAs to identify web hosts that expect the presence of Signed Certificate Timestamps (SCTs) [RFC9162] in subsequent Transport Layer Security (TLS) [RFC8446] connections.

Web hosts that serve the Expect-CT header field are noted by the UA as "Known Expect-CT Hosts". The UA evaluates each connection to a Known Expect-CT Host for compliance with the UA's Certificate Transparency (CT) Policy. If the connection violates the CT Policy, the UA sends a report to a URI configured by the Expect-CT Host and/or fails the connection, depending on the configuration that the Expect-CT Host has chosen.

If misconfigured, Expect-CT can cause unwanted connection failures (for example, if a host deploys Expect-CT but then switches to a legitimate certificate that is not logged in Certificate Transparency logs or if a web host operator believes their certificate to conform to all UAs' CT policies but is mistaken). Web host operators are advised to deploy Expect-CT with precautions by using the reporting feature and gradually increasing the time interval during which the UA regards the host as a Known Expect-CT Host. These precautions can help web host operators gain confidence that their Expect-CT deployment is not causing unwanted connection failures.

Expect-CT is a trust-on-first-use (TOFU) mechanism. The first time a UA connects to a host, it lacks the information necessary to require SCTs for the connection. Thus, the UA will not be able to detect and thwart an attack on the UA's first connection to the host. Still, Expect-CT provides value by 1) allowing UAs to detect the use of unlogged certificates after the initial communication, and 2) allowing web hosts to be confident that UAs are only trusting publicly auditable certificates.

Expect-CT is similar to HTTP Strict Transport Security (HSTS) [RFC6797] and HTTP Public Key Pinning (HPKP) [RFC7469]. HSTS allows websites to declare themselves accessible only via secure connections, and HPKP allows websites to declare their cryptographic identifies. Similarly, Expect-CT allows websites to declare themselves accessible only via connections that are compliant with CT Policy.

This Expect-CT specification is compatible with [RFC6962] and [RFC9162], but not necessarily with future versions of Certificate Transparency. UAs will ignore Expect-CT header fields from web hosts that use future versions of Certificate Transparency, unless a future version of this document specifies how they should be processed.

1.1. Requirements Language

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

Terminology is defined in this section.

"Certificate Transparency Policy"
: A policy defined by the UA concerning the number, sources, and delivery mechanisms of Signed Certificate Timestamps that are associated with TLS connections. The policy defines the properties of a connection that must be met in order for the UA to consider it CT qualified.

"Certificate Transparency Qualified"
: Describes a TLS connection for which the UA has determined that a sufficient quantity and quality of Signed Certificate Timestamps have been provided.

"CT Qualified"
: An abbreviation for "Certificate Transparency Qualified".

"CT Policy"
: An abbreviation for "Certificate Transparency Policy".

"Effective Expect-CT Date"
: The time at which a UA observed a valid Expect-CT header field for a given host.

"Expect-CT Host"
: A conformant host implementing the HTTP server aspects of Expect-CT. This means that an Expect-CT Host returns the Expect-CT response header field in its HTTP response messages sent over secure transport. The term "host" is equivalent to "server" in this specification.

"Known Expect-CT Host"
: An Expect-CT Host that the UA has noted as such. See Section 2.3.2.1 for particulars.

"User Agent (UA)"
: For the purposes of this specification, a UA is an HTTP client application typically actively manipulated by a user [RFC9110].

"Unknown Expect-CT Host"
: An Expect-CT Host that the UA has not noted.

2. Server and Client Behavior

2.1. Response Header Field Syntax

The Expect-CT response header field is a new field defined in this specification. It is used by a server to indicate that UAs should evaluate connections to the host emitting the header field for CT compliance (Section 2.4).

Figure 1 describes the syntax (Augmented Backus-Naur Form) of the header field, using the grammar defined in [RFC5234] and the rules defined in Section 5 of [RFC9110]. The "#" ABNF extension is specified in Section 5.6.1 of [RFC9110].

```
Expect-CT           = 1#expect-ct-directive
expect-ct-directive = directive-name [ "=" directive-value ]
directive-name      = token
directive-value     = token / quoted-string
```

Figure 1: Syntax of the Expect-CT Header Field

The directives defined in this specification are described below. The overall requirements for directives are:

1. The order of appearance of directives is not significant.
2. A given directive **MUST NOT** appear more than once in a given header field. Directives are either optional or required, as stipulated in their definitions.
3. Directive names are case insensitive.
4. UAs **MUST** ignore any header fields containing directives, or other header field value data that does not conform to the syntax defined in this specification. In particular, UAs **MUST NOT** attempt to fix malformed header fields.
5. If a header field contains any directive(s) the UA does not recognize, the UA **MUST** ignore those directives.
6. If the Expect-CT header field otherwise satisfies the above requirements (1 through 5), and Expect-CT is not disabled for local policy reasons (as discussed in Section 2.4.1), the UA **MUST** process the directives it recognizes.

2.1.1. The report-uri Directive

The **OPTIONAL** `report-uri` directive indicates the URI to which the UA **SHOULD** report Expect-CT failures (Section 2.4). The UA POSTs the reports to the given URI as described in Section 3.

The `report-uri` directive is **REQUIRED** to have a directive value, for which the syntax is defined in Figure 2.

```
report-uri-value = (DQUOTE absolute-URI DQUOTE) / absolute-URI
```

Figure 2: Syntax of the report-uri Directive Value

The 'report-uri-value' **MUST** be quoted if it contains any character not allowed in 'token'.

`absolute-URI` is defined in Section 4.3 of [RFC3986].

UAs **MUST** ignore any `report-uri` that does not use the HTTPS scheme. UAs **MUST** check Expect-CT compliance when the host in the `report-uri` is a Known Expect-CT Host; similarly, UAs **MUST** apply HSTS [RFC6797] if the host in the `report-uri` is a Known HSTS Host.

UAs **SHOULD** make their best effort to report Expect-CT failures to the `report-uri`, but they may fail to report in exceptional conditions. For example, if connecting to the `report-uri` itself incurs an Expect-CT failure or other certificate validation failure, the UA **MUST** cancel the connection. Similarly, if Expect-CT Host A sets a `report-uri` referring to Expect-CT Host B, and if B sets a `report-uri` referring to A, and if both hosts fail to comply to the UA's CT Policy, the UA **SHOULD** detect and break the loop by failing to send reports to and about those hosts.

Note that the `report-uri` need not necessarily be in the same Internet domain or web origin as the host being reported about. Hosts are in fact encouraged to use a separate host as the `report-uri` so that CT failures on the Expect-CT Host do not prevent reports from being sent.

UAs **SHOULD** limit the rate at which they send reports. For example, it is unnecessary to send the same report to the same `report-uri` more than once in the same web-browsing session.

2.1.2. The enforce Directive

The **OPTIONAL** `enforce` directive is a valueless directive that, if present (i.e., it is "asserted"), signals to the UA that compliance to the CT Policy should be enforced (rather than report-only) and that the UA should refuse future connections that violate its CT Policy. When both the `enforce` directive and `report-uri` directive (as defined in Figure 2) are present, the configuration is referred to as an "enforce-and-report" configuration, signaling to the UA that both compliance to the CT Policy should be enforced and violations should be reported.

2.1.3. The max-age Directive

The `max-age` directive specifies the number of seconds after the reception of the Expect-CT header field during which the UA **SHOULD** regard the host from whom the message was received as a Known Expect-CT Host.

If a response contains an Expect-CT header field, then the response **MUST** contain an Expect-CT header field with a `max-age` directive. (A `max-age` directive need not appear in every Expect-CT header field in the response.) The `max-age` directive is **REQUIRED** to have a directive value, for which the syntax (after quoted-string unescaping, if necessary) is defined in Figure 3.

```
max-age-value = delta-seconds
delta-seconds = 1*DIGIT
```

Figure 3: Syntax of the max-age Directive Value

`delta-seconds` is used as defined in Section 1.3 of [RFC9111].

2.1.4. Examples

The following three examples demonstrate valid Expect-CT response header fields (where the second splits the directives into two field instances):

```
Expect-CT: max-age=86400, enforce

Expect-CT: max-age=86400,enforce
Expect-CT: report-uri="https://foo.example/report"

Expect-CT: max-age=86400,report-uri="https://foo.example/report"
```

Figure 4: Examples of Valid Expect-CT ResponseHeader Fields

2.2. Host Processing Model

This section describes the processing model that Expect-CT Hosts implement. The model has 2 parts: (1) the processing rules for HTTP request messages received over a secure transport (e.g., authenticated, non-anonymous TLS); and (2) the processing rules for HTTP request messages received over non-secure transports, such as TCP.

2.2.1. HTTP-over-Secure-Transport Request Type

An Expect-CT Host includes an Expect-CT header field in its response. The header field **MUST** satisfy the grammar specified in Section 2.1.

Establishing a given host as an Expect-CT Host, in the context of a given UA, is accomplished as follows:

1. Over the HTTP protocol running over secure transport, by correctly returning (per this specification) a valid Expect-CT header field to the UA.
2. Through other mechanisms such as a client-side preloaded Expect-CT Host list.

2.2.2. HTTP Request Type

Expect-CT Hosts **SHOULD NOT** include the Expect-CT header field in HTTP responses conveyed over non-secure transport.

2.3. User Agent Processing Model

The UA processing model relies on parsing domain names. Note that internationalized domain names **SHALL** be canonicalized by the UA according to the scheme in Section 10 of [RFC6797].

The UA stores Known Expect-CT Hosts and their associated Expect-CT directives. This data is collectively known as a host's "Expect-CT metadata".

2.3.1. Missing or Malformed Expect-CT Header Fields

If an HTTP response does not include an Expect-CT header field that conforms to the grammar specified in Section 2.1, then the UA **MUST NOT** update any Expect-CT metadata.

2.3.2. Expect-CT Header Field Processing

If the UA receives an HTTP response over a secure transport that includes an Expect-CT header field conforming to the grammar specified in Section 2.1, the UA **MUST** evaluate the connection on which the header field was received for compliance with the UA's CT Policy, and then process the Expect-CT header field as follows. UAs **MUST** ignore any Expect-CT header field received in an HTTP response conveyed over non-secure transport.

If the connection does not comply with the UA's CT Policy (i.e., the connection is not CT qualified), then the UA **MUST NOT** update any Expect-CT metadata. If the header field includes a `report-uri` directive, the UA **SHOULD** send a report to the specified `report-uri` (Section 2.3.3).

If the connection complies with the UA's CT Policy (i.e., the connection is CT qualified), then the UA **MUST** either:

- Note the host as a Known Expect-CT Host if it is not already so noted (see Section 2.3.2.1) or
- Update the UA's cached information for the Known Expect-CT Host if the `enforce`, `max-age`, or `report-uri` header field value directives convey information different from that already maintained by the UA. If the `max-age` directive has a value of 0, the UA **MUST** remove its cached Expect-CT information if the host was previously noted as a Known Expect-CT Host and **MUST NOT** note this host as a Known Expect-CT Host if it is not already noted.

If a UA receives an Expect-CT header field over a CT-compliant connection that uses a version of Certificate Transparency other than [RFC6962] or [RFC9162], the UA **MUST** ignore the Expect-CT header field and clear any Expect-CT metadata associated with the host.

2.3.2.1. Noting Expect-CT

Upon receipt of the Expect-CT response header field over an error-free TLS connection (with X.509 certificate chain validation as described in [RFC5280], as well as the validation described in Section 2.4 of this document), the UA **MUST** note the host as a Known Expect-CT Host, storing the host's domain name and its associated Expect-CT directives in non-volatile storage.

To note a host as a Known Expect-CT Host, the UA **MUST** set its Expect-CT metadata in its Known Expect-CT Host cache (as specified in Section 2.3.2.2), using the metadata given in the most recently received valid Expect-CT header field.

For forward compatibility, the UA **MUST** ignore any unrecognized Expect-CT header field directives while still processing those directives it does recognize. Section 2.1 specifies the directives `enforce`, `max-age`, and `report-uri`, but future specifications and implementations might use additional directives.

2.3.2.2. Storage Model

If the substring matching the host production from the Request-URI (of the message to which the host responded) does not exactly match an existing Known Expect-CT Host's domain name, per the matching procedure for a Congruent Match specified in Section 8.2 of [RFC6797], then the UA **MUST** add this host to the Known Expect-CT Host cache. The UA caches:

- the Expect-CT Host's domain name.
- whether the `enforce` directive is present.
- the Effective Expiration Date, which is the Effective Expect-CT Date plus the value of the `max-age` directive. Alternatively, the UA **MAY** cache enough information to calculate the Effective Expiration Date. The Effective Expiration Date is calculated from when the UA observed the Expect-CT header field and is independent of when the response was generated.
- the value of the `report-uri` directive, if present.

If any other metadata from optional or future Expect-CT header directives are present in the Expect-CT header field, and the UA understands them, the UA **MAY** note them as well.

UAs **MAY** set an upper limit on the value of `max-age` so that UAs that have noted erroneous Expect-CT Hosts (whether by accident or due to attack) have some chance of recovering over time. If the server sets a `max-age` greater than the UA's upper limit, the UA may behave as if the server set the `max-age` to the UA's upper limit. For example, if the UA caps `max-age` at 5,184,000 seconds (60 days), and an Expect-CT Host sets a `max-age directive` of 90 days in its Expect-CT header field, the UA may behave as if the `max-age` were effectively 60 days. (One way to achieve this behavior is for the UA to simply store a value of 60 days instead of the 90-day value provided by the Expect-CT Host.)

2.3.3. Reporting

If the UA receives, over a secure transport, an HTTP response that includes an Expect-CT header field with a `report-uri` directive, and the connection does not comply with the UA's CT Policy (i.e., the connection is not CT qualified), and the UA has not already sent an Expect-CT report for this connection, then the UA **SHOULD** send a report to the specified `report-uri` as specified in Section 3.

2.4. Evaluating Expect-CT Connections for CT Compliance

When a UA sets up a TLS connection, the UA determines whether the host is a Known Expect-CT Host according to its Known Expect-CT Host cache. An Expect-CT Host is "expired" if the Effective Expiration Date refers to a date in the past. The UA **MUST** ignore any expired Expect-CT Hosts in its cache and not treat such hosts as Known Expect-CT Hosts.

When a UA connects to a Known Expect-CT Host using a TLS connection, if the TLS connection has no errors, then the UA will apply an additional correctness check: compliance with a CT Policy. A UA should evaluate compliance with its CT Policy whenever connecting to a Known Expect-CT Host. However, the check can be skipped for local policy reasons (as discussed in Section 2.4.1) or in the event that other checks cause the UA to terminate the connection before CT compliance is evaluated. For example, a Public Key Pinning failure [RFC7469] could cause the UA

to terminate the connection before CT compliance is checked. Similarly, if the UA terminates the connection due to an Expect-CT failure, this could cause the UA to skip subsequent correctness checks. When the CT compliance check is skipped or bypassed, Expect-CT reports (Section 3) will not be sent.

When CT compliance is evaluated for a Known Expect-CT Host, the UA **MUST** evaluate compliance when setting up the TLS session, before beginning an HTTP conversation over the TLS channel.

If a connection to a Known Expect-CT Host violates the UA's CT Policy (i.e., the connection is not CT qualified), and if the Known Expect-CT Host's Expect-CT metadata indicates an `enforce` configuration, the UA **MUST** treat the CT compliance failure as an error. The UA **MAY** allow the user to bypass the error unless connection errors should have no user recourse due to other policies in effect (such as HSTS, as described in Section 12.1 of [RFC6797]).

If a connection to a Known Expect-CT Host violates the UA's CT Policy, and if the Known Expect-CT Host's Expect-CT metadata includes a `report-uri`, the UA **SHOULD** send an Expect-CT report to that `report-uri` (Section 3).

2.4.1. Skipping CT Compliance Checks

It is acceptable for a UA to skip CT compliance checks for some hosts according to local policy. For example, a UA **MAY** disable CT compliance checks for hosts whose validated certificate chain terminates at a user-defined trust anchor rather than a trust anchor built in to the UA (or underlying platform).

If the UA does not evaluate CT compliance, e.g., because the user has elected to disable it, or because a presented certificate chain chains up to a user-defined trust anchor, UAs **SHOULD NOT** send Expect-CT reports.

3. Reporting Expect-CT Failure

When the UA attempts to connect to a Known Expect-CT Host and the connection is not CT qualified, the UA **SHOULD** report Expect-CT failures to the `report-uri`, if any, in the Known Expect-CT Host's Expect-CT metadata.

When the UA receives an Expect-CT response header field over a connection that is not CT qualified, if the UA has not already sent an Expect-CT report for this connection, then the UA **SHOULD** report Expect-CT failures to the configured `report-uri`, if any.

3.1. Generating a Violation Report

To generate a violation `report object`, the UA constructs a JSON [RFC8259] object with the following keys and values:

"date-time"
: The value for this key indicates the UTC time that the UA observed the CT compliance failure. The value is a string formatted according to Section 5.6 of [RFC3339], "Internet Date/Time Format".

"hostname"
: The value is the hostname to which the UA made the original request that failed the CT compliance check. The value is provided as a string.

"port"
: The value is the port to which the UA made the original request that failed the CT compliance check. The value is provided as an integer.

"scheme"
: (optional) The value is the scheme with which the UA made the original request that failed the CT compliance check. The value is provided as a string. This key is optional and is assumed to be "https" if not present.

"effective-expiration-date"
: The value indicates the Effective Expiration Date (see Section 2.3.2.2) for the Expect-CT Host that failed the CT compliance check, in UTC. The value is provided as a string formatted according to Section 5.6 of [RFC3339], "Internet Date/Time Format".

"served-certificate-chain"
: The value is the certificate chain as served by the Expect-CT Host during TLS session setup. The value is provided as an array of strings, which **MUST** appear in the order that the certificates were served; each string in the array is the Privacy-Enhanced Mail (PEM) representation of each X.509 certificate as described in [RFC7468].

"validated-certificate-chain"
: The value is the certificate chain as constructed by the UA during certificate chain verification. (This may differ from the value of the "served-certificate-chain" key.) The value is provided as an array of strings, which **MUST** appear in the order matching the chain that the UA validated; each string in the array is the PEM representation of each X.509 certificate as described in [RFC7468]. The first certificate in the chain represents the end-entity certificate being verified. UAs that build certificate chains in more than one way during the validation process **SHOULD** send the last chain built.

"scts"
: The value represents the SCTs (if any) that the UA received for the Expect-CT Host and their validation statuses. The value is provided as an array of JSON objects. The SCTs may appear in any order. Each JSON object in the array has the following keys:

  - A "version" key, with an integer value. The UA **MUST** set this value to 1 if the SCT is in the format defined in Section 3.2 of [RFC6962] or 2 if it is in the format defined in Section 4.5 of [RFC9162].
  - The "status" key, with a string value that the UA **MUST** set to one of the following values: "unknown" (indicating that the UA does not have or does not trust the public key of the log

from which the SCT was issued); "valid" (indicating that the UA successfully validated the SCT as described in Section 5.2 of [RFC6962] or Section 8.1.3 of [RFC9162]); or "invalid" (indicating that the SCT validation failed because of a bad signature or an invalid timestamp).

- The "source" key, with a string value that indicates from where the UA obtained the SCT, as defined in Section 3 of [RFC6962] and Section 6 of [RFC9162]. The UA **MUST** set the value to one of the following: "tls-extension", "ocsp", or "embedded". These correspond to the three methods of delivering SCTs in the TLS handshake that are described in Section 3.3 of [RFC6962].
- The "serialized_sct" key, with a string value. If the value of the "version" key is 1, the UA **MUST** set this value to the base64-encoded [RFC4648] serialized `SignedCertificateTimestamp` structure from Section 3.2 of [RFC6962]. The base64 encoding is defined in Section 4 of [RFC4648]. If the value of the "version" key is 2, the UA **MUST** set this value to the base64-encoded [RFC4648] serialized `TransItem` structure representing the SCT, as defined in Section 4.5 of [RFC9162].

"failure-mode"
: The value indicates whether the Expect-CT report was triggered by an Expect-CT policy in enforce or report-only mode. The value is provided as a string. The UA **MUST** set this value to "enforce" if the Expect-CT metadata indicates an `enforce` configuration, and "report-only" otherwise.

"test-report"
: (optional) The value is set to true if the report is being sent by a testing client to verify that the report server behaves correctly. The value is provided as a boolean and **MUST** be set to true if the report serves to test the server's behavior and can be discarded.

3.2. Sending a Violation Report

The UA **SHOULD** report Expect-CT failures for Known Expect-CT Hosts: that is, when a connection to a Known Expect-CT Host does not comply with the UA's CT Policy and the host's Expect-CT metadata contains a `report-uri`.

Additionally, the UA **SHOULD** report Expect-CT failures for hosts for which it does not have any stored Expect-CT metadata; that is, when the UA connects to a host and receives an Expect-CT header field that contains the `report-uri` directive, the UA **SHOULD** report an Expect-CT failure if the connection does not comply with the UA's CT Policy.

The steps to report an Expect-CT failure are as follows.

1. Prepare a JSON object `report object` with the single key "expect-ct-report", whose value is the result of generating a violation `report object` as described in Section 3.1.
2. Let `report body` be the JSON stringification of `report object`.
3. Let `report-uri` be the value of the `report-uri` directive in the Expect-CT header field.
4. Send an HTTP POST request to `report-uri` with a `Content-Type` header field of `application/expect-ct-report+json` and an entity body consisting of `report body`.

The UA **MAY** perform other operations as part of sending the HTTP POST request, such as sending a Cross-Origin Resource Sharing (CORS) preflight as part of [FETCH].

Future versions of this specification may need to modify or extend the Expect-CT report format. They may do so by defining a new top-level key to contain the report, replacing the "expect-ct-report" key. Section 3.3 defines how report servers should handle report formats that they do not support.

3.3. Receiving a Violation Report

Upon receiving an Expect-CT violation report, the report server **MUST** respond with a 2xx (Successful) status code if it can parse the request body as valid JSON, the report conforms to the format described in Section 3.1, and it recognizes the scheme, hostname, and port in the "scheme", "hostname", and "port" fields of the report. If the `report body` cannot be parsed or does not conform to the format described in Section 3.1, or the report server does not expect to receive reports for the scheme, hostname, or port in the report, then the report server **MUST** respond with a 400 Bad Request status code.

As described in Section 3.2, future versions of this specification may define new report formats that are sent with a different top-level key. If the report server does not recognize the report format, the report server **MUST** respond with a 501 Not Implemented status code.

If the report's "test-report" key is set to true, the server **MAY** discard the report without further processing but **MUST** still return a 2xx (Successful) status code. If the "test-report" key is absent or set to false, the server **SHOULD** store the report for processing and analysis by the owner of the Expect-CT Host.

4. Usability Considerations

When the UA detects a Known Expect-CT Host in violation of the UA's CT Policy, end users will experience denials of service. It is advisable for UAs to explain to users why they cannot access the Expect-CT Host, e.g., in a user interface that explains that the host's certificate cannot be validated.

5. Authoring Considerations

Expect-CT could be specified as a TLS extension or X.509 certificate extension instead of an HTTP response header field. Using an HTTP header field as the mechanism for Expect-CT introduces a layering mismatch; for example, the software that terminates TLS and validates Certificate Transparency information might know nothing about HTTP. Nevertheless, an HTTP header field was chosen primarily for ease of deployment. In practice, deploying new certificate extensions requires certificate authorities to support them, and new TLS extensions require server software updates, including possibly to servers outside of the site owner's direct control (such as in the case of a third-party Content Delivery Network (CDN)). Ease of deployment is a high priority for Expect-CT because it is intended as a temporary transition mechanism for user agents that are transitioning to universal Certificate Transparency requirements.

6. Privacy Considerations

Expect-CT can be used to infer what Certificate Transparency Policy a UA is using by attempting to retrieve specially configured websites that pass one user agent's policies but not another's. Note that this consideration is true of UAs that enforce CT policies without Expect-CT as well.

Additionally, reports submitted to the `report-uri` could reveal information to a third party about which web page is being accessed and by which IP address, by using individual `report-uri` values for individually tracked pages. This information could be leaked even if client-side scripting were disabled.

Implementations store state about Known Expect-CT Hosts and, hence, which domains the UA has contacted. Implementations may choose to not store this state subject to local policy (e.g., in the private browsing mode of a web browser).

Violation reports, as noted in Section 3, contain information about the certificate chain that has violated the CT Policy. In some cases, such as an organization-wide compromise of the end-to-end security of TLS, this may include information about the interception tools and design used by the organization that the organization would otherwise prefer not be disclosed.

Because Expect-CT causes remotely detectable behavior, it's advisable that UAs offer a way for privacy-sensitive end users to clear currently noted Expect-CT Hosts and allow users to query the current state of Known Expect-CT Hosts.

7. Security Considerations

7.1. Hostile Header Attacks

When UAs support the Expect-CT header field, it becomes a potential vector for hostile header attacks against site owners. If a site owner uses a certificate issued by a certificate authority that does not embed SCTs nor serve SCTs via the Online Certificate Status Protocol (OCSP) or TLS extension, a malicious server operator or attacker could temporarily reconfigure the host to comply with the UA's CT Policy and add the Expect-CT header field in enforcing mode with a long `max-age`. Implementing user agents would note this as an Expect-CT Host (see Section 2.3.2.1). After having done this, the configuration could then be reverted to not comply with the CT Policy, prompting failures. Note that this scenario would require the attacker to have substantial control over the infrastructure in question, being able to obtain different certificates, change server software, or act as a man in the middle in connections.

Site operators can mitigate this situation by one of the following: reconfiguring their web server to transmit SCTs using the TLS extension defined in Section 6.5 of [RFC9162]; obtaining a certificate from an alternative certificate authority that provides SCTs by one of the other methods; or by waiting for the user agent's persisted notation of this as an Expect-CT Host to reach its `max-age`. User agents may choose to implement mechanisms for users to cure this situation, as noted in Section 4.

7.2. Maximum max-age

There is a security trade-off in that low maximum values provide a narrow window of protection for users that visit the Known Expect-CT Host only infrequently, while high maximum values might result in a denial of service to a UA in the event of a hostile header attack or simply an error on the part of the site owner.

There is probably no ideal maximum for the `max-age` directive. Since Expect-CT is primarily a policy-expansion and investigation technology rather than an end-user protection, a value on the order of 30 days (2,592,000 seconds) may be considered a balance between these competing security concerns.

7.3. Amplification Attacks

Another kind of hostile header attack uses the `report-uri` mechanism on many hosts not currently exposing SCTs as a method to cause a denial of service to the host receiving the reports. If some highly trafficked websites emitted a non-enforcing Expect-CT header field with a `report-uri`, implementing UAs' reports could flood the reporting host. It is noted in Section 2.1.1 that UAs should limit the rate at which they emit reports, but an attacker may alter the Expect-CT header fields to induce UAs to submit different reports to different URIs to still cause the same effect.

8. IANA Considerations

8.1. Header Field Registry

This document registers the "Expect-CT" header field in the "Hypertext Transfer Protocol (HTTP) Field Name Registry" registry located at <https://www.iana.org/assignments/http-fields>.

Header field name: Expect-CT

Applicable protocol: http

Status: permanent

Author/Change controller: IETF

Specification document(s): This document

Related information: (empty)

8.2. Media Types Registry

This document registers the `application/expect-ct-report+json` media type (which uses the suffix established in [RFC6839]) for Expect-CT violation reports in the "Media Types" registry as follows.

Type name: application

Subtype name: expect-ct-report+json

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: binary

Security considerations: See Section 7

Interoperability considerations: n/a

Published specification: This document

Applications that use this media type: UAs that implement Certificate Transparency compliance checks and reporting

Additional information:

Deprecated alias names for this type: n/a

Magic number(s): n/a

File extension(s): n/a

Macintosh file type code(s): n/a

Person & email address to contact for further information:
Emily Stark (estark@google.com)

Intended usage: COMMON

Restrictions on usage: none

Author: Emily Stark (estark@google.com)

Change controller: IETF

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6797] Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", RFC 6797, DOI 10.17487/RFC6797, November 2012, <https://www.rfc-editor.org/info/rfc6797>.

[RFC6839] Hansen, T. and A. Melnikov, "Additional Media Type Structured Syntax Suffixes", RFC 6839, DOI 10.17487/RFC6839, January 2013, <https://www.rfc-editor.org/info/rfc6839>.

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, DOI 10.17487/RFC6962, June 2013, <https://www.rfc-editor.org/info/rfc6962>.

[RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <https://www.rfc-editor.org/info/rfc7468>.

[RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <https://www.rfc-editor.org/info/rfc7469>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9111] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Caching", STD 98, RFC 9111, DOI 10.17487/RFC9111, June 2022, <https://www.rfc-editor.org/info/rfc9111>.

[RFC9162] Laurie, B., Messeri, E., and R. Stradling, "Certificate Transparency Version 2.0", RFC 9162, DOI 10.17487/RFC9162, December 2021, <https://www.rfc-editor.org/info/rfc9162>.

9.2. Informative References

[FETCH] WHATWG, "Fetch - Living Standard", <https://fetch.spec.whatwg.org>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Author's Address

Emily Stark
Google
Email: estark@google.com