Internet Engineering Task Force (IETF)
Request for Comments: 7469
Category: Standards Track
ISSN: 2070-1721

C. Evans
C. Palmer
R. Sleevi
Google, Inc.

April 2015

Public Key Pinning Extension for HTTP

Abstract

This document defines a new HTTP header that allows web host operators to instruct user agents to remember ("pin") the hosts' cryptographic identities over a period of time. During that time, user agents (UAs) will require that the host presents a certificate chain including at least one Subject Public Key Info structure whose fingerprint matches one of the pinned fingerprints for that host. By effectively reducing the number of trusted authorities who can authenticate the domain during the lifetime of the pin, pinning may reduce the incidence of man-in-the-middle attacks due to compromised Certification Authorities.

Status of This Memo

This is an Internet Standards Track document.

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). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7469.

Table of Contents

1. Introduction

This document defines a new HTTP header that enables UAs to determine which Subject Public Key Info (SPKI) structures will be present in a web host's certificate chain in future Transport Layer Security (TLS) [RFC5246] connections.

Deploying Public Key Pinning (PKP) safely will require operational and organizational maturity due to the risk that hosts may make themselves unavailable by pinning to a set of SPKIs that becomes invalid (see Section 4). With care, host operators can greatly reduce the risk of man-in-the-middle (MITM) attacks and other false-authentication problems for their users without incurring undue risk.

PKP is meant to be used together with HTTP Strict Transport Security (HSTS) [RFC6797], but it is possible to pin keys without requiring HSTS.

A Pin is a relationship between a hostname and a cryptographic identity (in this document, one or more of the public keys in a chain of X.509 certificates). Pin Validation is the process a UA performs to ensure that a host is in fact authenticated with its previously established Pin.

Key pinning is a trust-on-first-use (TOFU) mechanism. The first time a UA connects to a host, it lacks the information necessary to perform Pin Validation; UAs can only apply their normal cryptographic identity validation. (In this document, it is assumed that UAs apply X.509 certificate chain validation in accord with [RFC5280].)

The UA will not be able to detect and thwart a MITM attacking the UA's first connection to the host. (However, the requirement that the MITM provide an X.509 certificate chain that can pass the UA's validation requirements, without error, mitigates this risk somewhat.) Worse, such a MITM can inject its own PKP header into the HTTP stream, and pin the UA to its own keys. To avoid post facto detection, the attacker would have to be in a position to intercept all future requests to the host from that UA.

Thus, key pinning as described in this document is not a perfect defense against MITM attackers capable of passing certificate chain validation procedures -- nothing short of pre-shared keys can be. However, it provides significant value by allowing host operators to limit the number of certification authorities that can vouch for the host's identity, and allows UAs to detect in-process MITM attacks after the initial communication.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Server and Client Behavior

2.1. Response Header Field Syntax

The "Public-Key-Pins" and "Public-Key-Pins-Report-Only" header fields, also referred to within this specification as the PKP and PKP-RO header fields, respectively, are new response headers defined in this specification. They are used by a server to indicate that a UA should perform Pin Validation (Section 2.6) for the host emitting the response message, and to provide the necessary information for the UA to do so.

Figure 1 describes the syntax (Augmented Backus-Naur Form) of the header fields, using the grammar defined in [RFC5234] and the rules defined in Section 3.2 of [RFC7230]. The field values of both header fields conform to the same rules.

```
Public-Key-Directives = directive *( OWS ";" OWS directive )

directive             = directive-name [ "=" directive-value ]
directive-name        = token
directive-value       = token
                      / quoted-string
```

Figure 1: HPKP Header Syntax

Optional white space (OWS) is used as defined in Section 3.2.3 of [RFC7230]. token and quoted-string are used as defined in Section 3.2.6 of [RFC7230].

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. With the exception of pin-directives with the same pin-directive-name (see below), 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 do 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 PKP or PKP-RO header field otherwise satisfies the above requirements (1 through 5), the UA MUST process the directives it recognizes.

Additional directives extending the semantic functionality of the header fields can be defined in other specifications. The first such specification will need to define a registry for such directives. Such future directives will be ignored by UAs implementing only this specification, as well as by generally non-conforming UAs.

When a connection passes Pin Validation using the UA’s noted Pins for the host at the time, the host becomes a Known Pinned Host.

2.1.1. The Pin Directive

The pin directive specifies a way for web host operators to indicate a cryptographic identity that should be bound to a given web host. The syntax of a pin directive is as follows:

```
pin-directive       = pin-directive-name "=" pin-directive-value

pin-directive-name  = "pin-" token
pin-directive-value = quoted-string
```

Figure 2: Pin Directive Syntax

In the pin-directive, the token is the name of a cryptographic hash algorithm. The only algorithm allowed at this time is "sha256", i.e., the hash algorithm SHA256 [RFC6234]; additional algorithms may be allowed for use in this context in the future. The quoted-string is a sequence of base 64 digits: the base64-encoded SPKI Fingerprint [RFC4648] (see Section 2.4).

According to the processing rules of Section 2.1, the UA MUST ignore pin-directives with tokens naming hash algorithms it does not recognize. If the set of remaining effective pin-directives is empty, and if the host is a Known Pinned Host, the UA MUST cease to consider the host as a Known Pinned Host (the UA should fail open). The UA should indicate to users that the host is no longer a Known Pinned Host.

Note, per the processing rules of Section 2.1, the pin-directive-name is case insensitive.

2.1.2. The max-age Directive

The "max-age" directive specifies the number of seconds after the reception of the PKP header field during which the UA SHOULD regard the host (from whom the message was received) as a Known Pinned Host.

The "max-age" directive is REQUIRED to be present within a "Public-Key-Pins" header field. The "max-age" directive is meaningless within a "Public-Key-Pins-Report-Only" header field, and UAs MUST ignore it and not cache the header. See Section 2.3.3.

The max-age directive is REQUIRED to have a directive value, for which the syntax (after quoted-string unescaping, if necessary) is defined as:

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

Figure 3: max-age Value Syntax

delta-seconds is used as defined in [RFC7234], Section 1.2.1.

See Section 2.3.3 for limitations on the range of values for max-age.

2.1.3. The includeSubDomains Directive

The OPTIONAL includeSubDomains directive is a valueless directive that, if present (i.e., it is "asserted"), signals to the UA that the Pinning Policy applies to this Pinned Host as well as any subdomains of the host’s domain name.

2.1.4. The report-uri Directive

The OPTIONAL report-uri directive indicates the URI to which the UA SHOULD report Pin Validation failures (Section 2.6). The UA POSTs the reports to the given URI as described in Section 3.

When used in the PKP or PKP-RO headers, the presence of a report-uri directive indicates to the UA that in the event of Pin Validation failure it SHOULD POST a report to the report-uri. If the header is Public-Key-Pins, the UA should do this in addition to terminating the connection (as described in Section 2.6).

Hosts may set report-uris that use HTTP or HTTPS. If the scheme in the report-uri is one that uses TLS (e.g., HTTPS), UAs MUST perform Pinning Validation when the host in the report-uri is a Known Pinned Host; similarly, UAs MUST apply HSTS if the host in the report-uri is a Known HSTS Host.

Note that the report-uri need not necessarily be in the same Internet domain or web origin as the host being reported about.

UAs SHOULD make their best effort to report Pin Validation failures to the report-uri, but they may fail to report in exceptional conditions. For example, if connecting the report-uri itself incurs a Pinning Validation failure or other certificate validation failure, the UA MUST cancel the connection. Similarly, if Known Pinned Host A sets a report-uri referring to Known Pinned Host B, and if B sets a report-uri referring to A, and if both hosts fail Pin Validation, the UA SHOULD detect and break the loop by failing to send reports to and about those hosts.

In any case of report failure, the UA MAY attempt to re-send the report later.

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 per distinct set of declared Pins.

2.1.5. Examples

Figure 4 shows some example PKP and PKP-RO response header fields. (Lines are folded to fit.)

```
Public-Key-Pins: max-age=3000;
    pin-sha256="d6qzRu9zOECb90Uez27xWltNsj0e1Md7GkYYkVoZWmM=";
    pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g="

Public-Key-Pins: max-age=2592000;
    pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";
    pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ="

Public-Key-Pins: max-age=2592000;
    pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";
    pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ=";
    report-uri="http://example.com/pkp-report"

Public-Key-Pins-Report-Only: max-age=2592000;
    pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";
    pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ=";
    report-uri="https://other.example.net/pkp-report"
```

```
Public-Key-Pins:
    pin-sha256="d6qzRu9zOECb90Uez27xWltNsj0e1Md7GkYYkVoZWmM=";
    pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ=";
    max-age=259200

Public-Key-Pins:
    pin-sha256="d6qzRu9zOECb90Uez27xWltNsj0e1Md7GkYYkVoZWmM=";
    pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";
    pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ=";
    max-age=10000; includeSubDomains
```

Figure 4: HTTP Public Key Pinning (HPKP) Header Examples

2.2. Server Processing Model

This section describes the processing model that Pinned 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

When replying to an HTTP request that was conveyed over a secure transport, a Pinned Host SHOULD include in its response exactly one PKP header field, exactly one PKP-RO header field, or one of each. Each instance of either header field MUST satisfy the grammar specified in Section 2.1.

Establishing a given host as a Known Pinned 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) at least one valid PKP header field to the UA.

2. Through other mechanisms, such as a client-side preloaded Known Pinned Host List.

2.2.2. HTTP Request Type

Pinned Hosts SHOULD NOT include the PKP header field in HTTP responses conveyed over non-secure transport. UAs MUST ignore any PKP header received in an HTTP response 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 according to the scheme in Section 10 of [RFC6797].

2.3.1. Public-Key-Pins Response Header Field Processing

If the UA receives, over a secure transport, an HTTP response that includes a PKP header field conforming to the grammar specified in Section 2.1, and there are no underlying secure transport errors or warnings (see Section 2.5), the UA MUST either:

- Note the host as a Known Pinned Host if it is not already so noted (see Section 2.3.3),

or,

- Update the UA's cached information for the Known Pinned Host if any of the max-age, includeSubDomains, or report-uri header field value directives convey information different from that already maintained by the UA.

The max-age value is essentially a "time to live" value relative to the time of the most recent observation of the PKP header field. If the max-age header field value token has a value of 0, the UA MUST remove its cached Pinning Policy information (including the includeSubDomains directive, if asserted) if the Pinned Host is Known, or, MUST NOT note this Pinned Host if it is not yet Known.

If a UA receives more than one PKP header field or more than one PKP-RO header field in an HTTP response message over secure transport, then the UA MUST process only the first PKP header field (if present) and only the first PKP-RO header field (if present).

If the UA receives the HTTP response over insecure transport, or if the PKP header is not a Valid Pinning Header (see Section 2.5), the UA MUST ignore any present PKP header field(s). Similarly, if the UA receives the HTTP response over insecure transport, the UA MUST ignore any present PKP-RO header field(s). The UA MUST ignore any PKP or PKP-RO header fields not conforming to the grammar specified in Section 2.1.

2.3.2. Interaction of Public-Key-Pins and Public-Key-Pins-Report-Only

A server MAY set both the "Public-Key-Pins" and "Public-Key-Pins-Report-Only" headers simultaneously. The headers do not interact with one another, but the UA MUST process the PKP header and SHOULD process both.

The headers are processed according to Section 2.3.1.

When the PKP-RO header is used with a report-uri, the UA SHOULD POST reports for Pin Validation failures to the indicated report-uri, although the UA MUST NOT enforce Pin Validation. That is, in the event of Pin Validation failure when the host has set the PKP-RO header, the UA performs Pin Validation to check whether or not it should POST a report, but not whether it should cause a connection failure.

Note: There is no purpose to using the PKP-RO header without the report-uri directive. User Agents MAY discard such headers without interpreting them further.

When the PKP header is used with a report-uri, the UA SHOULD POST reports for Pin Validation failures to the indicated report-uri, as well as enforce Pin Validation.

If a host sets the PKP-RO header, the UA SHOULD note the Pins and directives given in the PKP-RO header, ignoring any max-age directive. If the UA does note the Pins and directives in the PKP-RO header, it SHOULD evaluate the specified policy and SHOULD report any would-be Pin Validation failures that would occur if the report-only policy were enforced.

If a host sets both the PKP header and the PKP-RO header, the UA MUST note and enforce Pin Validation as specified by the PKP header, and SHOULD process the Pins and directives given in the PKP-RO header. If the UA does process the Pins and directives in the PKP-RO header, it SHOULD evaluate the specified policy and SHOULD report any would-be Pin Validation failures that would occur if the report-only policy were enforced.

2.3.3. Noting a Pinned Host - Storage Model

The Effective Pin Date of a Known Pinned Host is the time that the UA observed a Valid Pinning Header for the host. The Effective Expiration Date of a Known Pinned Host is the Effective Pin Date plus the max-age. A Known Pinned Host is "expired" if the Effective Expiration Date refers to a date in the past. The UA MUST ignore any expired Known Pinned Hosts in its cache.

For example, if a UA is beginning to perform Pin Validation for a Known Pinned Host and finds that the cached pinning information for the host indicates an Effective Expiration Date in the past, the UA MUST NOT continue with Pin Validation for the host, and MUST consider the host to no longer be a Known Pinned Host.

Known Pinned Hosts are identified only by domain names, and never IP addresses. If the substring matching the host production from the Request-URI (of the message to which the host responded) syntactically matches the IP-literal or IPv4address productions from Section 3.2.2 of [RFC3986], then the UA MUST NOT note this host as a Known Pinned Host.

Otherwise, if the substring does not congruently match an existing Known Pinned Host's domain name, per the matching procedure specified in Section 8.2 of [RFC6797], then the UA MUST add this host to the Known Pinned Host cache. The UA caches:

- the Pinned Host's domain name,
- the Effective Expiration Date, or enough information to calculate it (the Effective Pin Date and the value of the max-age directive),
- whether or not the includeSubDomains directive is asserted, and
- the value of the report-uri directive, if present.

If any other metadata from optional or future PKP header directives are present in the Valid Pinning Header, 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 Pins (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 a Pinned Host sets a max-age directive of 90 days in its Valid Pinning Header, 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 Pinned Host.) For UA implementation guidance on how to select a maximum max-age, see Section 4.1.

The UA MUST NOT modify any pinning metadata of any superdomain matched Known Pinned Host.

The UA MUST NOT cache information derived from a PKP-RO header. (PKP-RO headers are useful only at the time of receipt and processing.)

2.3.4. HTTP-Equiv <Meta> Element Attribute

UAs MUST NOT heed http-equiv="Public-Key-Pins" or http-equiv="Public-Key-Pins-Report-Only" attribute settings on <meta> elements [W3C.REC-html401-19991224] in received content.

2.4. Semantics of Pins

An SPKI Fingerprint is defined as the output of a known cryptographic hash algorithm whose input is the DER-encoded ASN.1 representation of the Subject Public Key Info (SPKI) of an X.509 certificate. A Pin is defined as the combination of the known algorithm identifier and the SPKI Fingerprint computed using that algorithm.

The SPKI Fingerprint is encoded in base 64 for use in an HTTP header [RFC4648].

In this version of the specification, the known cryptographic hash algorithm is SHA-256, identified as "sha256" [RFC6234]. (Future specifications may add new algorithms and deprecate old ones.) UAs MUST ignore Pins for which they do not recognize the algorithm identifier. UAs MUST continue to process the rest of a PKP response header field and note Pins for algorithms they do recognize.

Figure 5 reproduces the definition of the SubjectPublicKeyInfo structure in [RFC5280].

```
SubjectPublicKeyInfo  ::=  SEQUENCE  {
    algorithm            AlgorithmIdentifier,
    subjectPublicKey     BIT STRING  }

AlgorithmIdentifier  ::=  SEQUENCE  {
    algorithm            OBJECT IDENTIFIER,
    parameters           ANY DEFINED BY algorithm OPTIONAL  }
```

Figure 5: SPKI Definition

If the certificate’s Subject Public Key Info is incomplete when taken in isolation, such as when holding a DSA key without domain parameters, a public key pin cannot be formed.

We pin public keys, rather than entire certificates, to enable operators to generate new certificates containing old public keys (see [why-pin-key]).

See Appendix A for an example non-normative program that generates SPKI Fingerprints from certificates.

2.5. Noting Pins

Upon receipt of the PKP response header field, the UA notes the host as a Known Pinned Host, storing the Pins and their associated directives in non-volatile storage (for example, along with the HSTS metadata). The Pins and their associated directives are collectively known as Pinning Metadata.

The UA MUST note the Pins for a Host if and only if all three of the following conditions hold:

- It received the PKP response header field over an error-free TLS connection. If the host is a Pinned Host, this includes the validation added in Section 2.6.
- The TLS connection was authenticated with a certificate chain containing at least one of the SPKI structures indicated by at least one of the given SPKI Fingerprints (see Section 2.6).
- The given set of Pins contains at least one Pin that does NOT refer to an SPKI in the certificate chain. (That is, the host must set a Backup Pin; see Section 4.3.)

If the PKP response header field does not meet all three of these criteria, the UA MUST NOT note the host as a Pinned Host. A PKP response header field that meets all these criteria is known as a Valid Pinning Header.

Whenever a UA receives a Valid Pinning Header, it MUST set its Pinning Metadata to the exact Pins, Effective Expiration Date (computed from max-age), and (if any) report-uri given in the most recently received Valid Pinning Header.

For forward compatibility, the UA MUST ignore any unrecognized PKP and PKP-RO header directives, while still processing those directives it does recognize. Section 2.1 specifies the directives max-age, Pins, includeSubDomains, and report-uri, but future specifications and implementations might use additional directives.

Upon receipt of a PKP-RO response header field, the UA SHOULD evaluate the policy expressed in the field, and SHOULD generate and send a report (see Section 3). However, failure to validate the Pins in the field MUST have no effect on the validity or non-validity of the policy expressed in the PKP field or in previously noted Pins for the Known Pinned Host.

The UA need not note any Pins or other policy expressed in the PKP-RO response header field, except for the purpose of determining that it has already sent a report for a given policy. UAs SHOULD make a best effort not to inundate report-uris with redundant reports.

2.6. Validating Pinned Connections

When a UA connects to a Pinned Host using a TLS connection, if the TLS connection has errors, the UA MUST terminate the connection without allowing the user to proceed anyway. (This behavior is the same as that required by [RFC6797].)

If the connection has no errors, then the UA will determine whether to apply a new, additional correctness check: Pin Validation. A UA SHOULD perform Pin Validation whenever connecting to a Known Pinned Host, as soon as possible (e.g., immediately after receiving the Server Certificate message). It is acceptable to allow Pin Validation to be disabled for some Hosts according to local policy. For example, a UA may disable Pin Validation for Pinned 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).

To perform Pin Validation, the UA will compute the SPKI Fingerprints for each certificate in the Pinned Host's validated certificate chain, using each supported hash algorithm for each certificate. (As described in Section 2.4, certificates whose SPKI cannot be taken in isolation cannot be pinned.) The UA MUST ignore superfluous certificates in the chain that do not form part of the validating chain. The UA will then check that the set of these SPKI Fingerprints intersects the set of SPKI Fingerprints in that Pinned Host's Pinning Metadata. If there is set intersection, the UA continues with the connection as normal. Otherwise, the UA MUST treat this Pin Validation failure as a non-recoverable error. Any procedure that matches the results of this Pin Validation procedure is considered equivalent.

A UA that has previously noted a host as a Known Pinned Host MUST perform Pin Validation when setting up the TLS session, before beginning an HTTP conversation over the TLS channel.

UAs send validation failure reports only when Pin Validation is actually in effect. Pin Validation might not be in effect, e.g., because the user has elected to disable it, or because a presented certificate chain chains up to a user-defined trust anchor. In such cases, UAs SHOULD NOT send reports.

2.7. Interactions with Preloaded Pin Lists

UAs MAY choose to implement additional sources of pinning information, such as through built-in lists of pinning information. Such UAs should allow users to override such additional sources, including disabling them from consideration.

The effective policy for a Known Pinned Host that has both built-in Pins and Pins from previously observed PKP header response fields is implementation-defined.

2.8. Pinning Self-Signed End Entities

If UAs accept hosts that authenticate themselves with self-signed end entity certificates, they MAY also allow hosts to pin the public keys in such certificates. The usability and security implications of this practice are outside the scope of this specification.

3. Reporting Pin Validation Failure

When a Known Pinned Host has set the report-uri directive, the UA SHOULD report Pin Validation failures to the indicated URI. The UA does this by POSTing a JSON [RFC7159] message to the URI; the JSON message takes this form:

```
{
  "date-time": date-time,
  "hostname": hostname,
  "port": port,
  "effective-expiration-date": expiration-date,
  "include-subdomains": include-subdomains,
  "noted-hostname": noted-hostname,
  "served-certificate-chain": [
    pem1, ... pemN
  ],
  "validated-certificate-chain": [
    pem1, ... pemN
  ],
  "known-pins": [
    known-pin1, ... known-pinN
  ]
}
```

Figure 6: JSON Report Format

Whitespace outside of quoted strings is not significant. The key/value pairs may appear in any order, but each MUST appear only once.

The date-time indicates the time the UA observed the Pin Validation failure. It is provided as a string formatted according to Section 5.6, "Internet Date/Time Format", of [RFC3339].

The hostname is the hostname to which the UA made the original request that failed Pin Validation. It is provided as a string.

The port is the port to which the UA made the original request that failed Pin Validation. It is provided as an integer.

The effective-expiration-date is the Effective Expiration Date for the noted Pins. It is provided as a string formatted according to Section 5.6, "Internet Date/Time Format", of [RFC3339].

include-subdomains indicates whether or not the UA has noted the includeSubDomains directive for the Known Pinned Host. It is provided as one of the JSON identifiers "true" or "false".

noted-hostname indicates the hostname that the UA noted when it noted the Known Pinned Host. This field allows operators to understand why Pin Validation was performed for, e.g., foo.example.com when the noted Known Pinned Host was example.com with includeSubDomains set.

The served-certificate-chain is the certificate chain, as served by the Known Pinned Host during TLS session setup. It is provided as an array of strings; each string pem1, ... pemN is the Privacy-Enhanced Mail (PEM) representation of each X.509 certificate as described in [RFC7468].

The validated-certificate-chain is the certificate chain, as constructed by the UA during certificate chain verification. (This may differ from the served-certificate-chain.) It is provided as an array of strings; each string pem1, ... pemN is the PEM representation of each X.509 certificate as described in [RFC7468]. UAs that build certificate chains in more than one way during the validation process SHOULD send the last chain built. In this way, they can avoid keeping too much state during the validation process.

The known-pins are the Pins that the UA has noted for the Known Pinned Host. They are provided as an array of strings with the syntax:

```
known-pin = token "=" quoted-string
```

Figure 7: Known Pin Syntax

As in Section 2.4, the token refers to the algorithm name, and the quoted-string refers to the base64 encoding of the SPKI Fingerprint. When formulating the JSON POST body, the UA MUST either use single-quoted JSON strings or use double-quoted JSON strings and backslash-escape the embedded double quotes in the quoted-string part of the known-pin.

Figure 8 shows an example of a Pin Validation failure report. (PEM strings are shown on multiple lines for readability.)

```
{
  "date-time": "2014-04-06T13:00:50Z",
  "hostname": "www.example.com",
  "port": 443,
  "effective-expiration-date": "2014-05-01T12:40:50Z"
  "include-subdomains": false,
  "served-certificate-chain": [
    "-----BEGIN CERTIFICATE-----\n
    MIIEBDCCAuygAwIBAgIDAjppMA0GCSqGSIb3DQEBBQUAMEIxCzAJBgNVBAYTAlVT\n
    ...
    HFa9llF7b1cq26KqltyMdMKVvvBulRP/F/A8rLIQjcxz++iPAsbw+zOzlTvjwsto\n
    WHPbqCRiOwY1nQ2pM714A5AuTHhdUDqB1O6gyHA43LL5Z/qHQF1hwFGPa4NrzQU6\n
    yuGnBXj8ytqU0CwIPX4WecigUCAkVDNx\n
    -----END CERTIFICATE-----",
    ...
  ],
  "validated-certificate-chain": [
    "-----BEGIN CERTIFICATE-----\n
    MIIEBDCCAuygAwIBAgIDAjppMA0GCSqGSIb3DQEBBQUAMEIxCzAJBgNVBAYTAlVT\n
    ...
    HFa9llF7b1cq26KqltyMdMKVvvBulRP/F/A8rLIQjcxz++iPAsbw+zOzlTvjwsto\n
    WHPbqCRiOwY1nQ2pM714A5AuTHhdUDqB1O6gyHA43LL5Z/qHQF1hwFGPa4NrzQU6\n
    yuGnBXj8ytqU0CwIPX4WecigUCAkVDNx\n
    -----END CERTIFICATE-----",
    ...
  ],
  "known-pins": [
    'pin-sha256="d6qzRu9zOECb90Uez27xWltNsj0e1Md7GkYYkVoZWmM="',
    "pin-sha256=\"E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=\""
  ]
}
```

Figure 8: Pin Validation Failure Report Example

4. Security Considerations

Pinning public keys helps hosts strongly assert their cryptographic identity even in the face of issuer error, malfeasance, or compromise. But, there is some risk that a host operator could lose (or lose control of) their host's private key (such as by operator error or host compromise). If the operator had pinned only the key of the host's end-entity certificate, the operator would not be able to serve their web site or application in a way that UAs would trust for the duration of their pin's max-age. (Recall that UAs MUST close the connection to a host upon Pin Failure.)

Therefore, there is a necessary trade-off between two competing goods: pin specificity and maximal reduction of the scope of issuers on the one hand; and flexibility and resilience of the host's cryptographic identity on the other hand. One way to resolve this trade-off is to compromise by pinning to the key(s) of the issuer(s) of the host's end-entity certificate(s). Often, a valid certificate chain will have at least two certificates above the end-entity certificate: the intermediate issuer and the trust anchor. Operators can pin any one or more of the public keys in this chain, and indeed MUST pin to issuers not in the chain (as, for example, a Backup Pin). Pinning to an intermediate issuer, or even to a trust anchor or root, still significantly reduces the number of issuers who can issue end-entity certificates for the Known Pinned Host, while still giving that host flexibility to change keys without a disruption of service.

4.1. Maximum max-age

As mentioned in Section 2.3.3, UAs MAY cap the max-age value at some upper limit. There is a security trade-off in that low maximum values provide a narrow window of protection for users who visit the Known Pinned Host only infrequently, while high maximum values might result in a UA's inability to successfully perform Pin Validation for a Known Pinned Host if the UA's noted Pins and the host's true Pins diverge.

Such divergence could occur for several reasons, including: UA error; host operator error; network attack; or a Known Pinned Host that intentionally migrates all pinned keys, combined with a UA that has noted true Pins with a high max-age value and has not had a chance to observe the new true Pins for the host. (This last example underscores the importance for host operators to phase in new keys gradually and to set the max-age value in accordance with their planned key migration schedule.)

There is probably no ideal upper limit to the max-age directive that would satisfy all use cases. However, a value on the order of 60 days (5,184,000 seconds) may be considered a balance between the two competing security concerns.

4.2. Using includeSubDomains Safely

It may happen that Pinned Hosts whose hostnames share a parent domain use different Valid Pinning Headers. If a host whose hostname is a parent domain for another host sets the includeSubDomains directive, the two hosts' Pins may conflict with each other. For example, consider two Known Pinned Hosts, example.com and subdomain.example.com. Assume example.com sets a Valid Pinning Header such as this:

```
Public-Key-Pins: max-age=12000; pin-sha256="ABC...";
    pin-sha256="DEF..."; includeSubDomains
```

Figure 9: example.com Valid Pinning Header

Assume subdomain.example.com sets a Valid Pinning Header such as this:

```
Public-Key-Pins: pin-sha256="GHI..."; pin-sha256="JKL..."
```

Figure 10: subdomain.example.com Valid Pinning Header

Assume a UA that has not previously noted any Pins for either of these hosts. If the UA first contacts subdomain.example.com, it will note the Pins in the Valid Pinning Header, and perform Pin Validation as normal on subsequent connections. If the UA then contacts example.com, again it will note the Pins and perform Pin Validation on future connections.

However, if the UA happened to visit example.com before subdomain.example.com, the UA would, due to example.com's use of the includeSubDomains directive, attempt to perform Pin Validation for subdomain.example.com using the SPKI hashes ABC... and DEF..., which are not valid for the certificate chains subdomain.example.com (which uses certificates with SPKIs GHI... and JLK...). Thus, depending on the order in which the UA observes the Valid Pinning Headers for hosts example.com and subdomain.example.com, Pin Validation might or might not fail for subdomain.example.com, even if the certificate chain the UA receives for subdomain.example.com is perfectly valid.

Thus, Pinned Host operators must use the includeSubDomains directive with care. For example, they may choose to use overlapping pin sets for hosts under a parent domain that uses includeSubDomains, or to

not use the includeSubDomains directive in their effective-second-level domains, or to simply use the same pin set for all hosts under a given parent domain.

4.3. Backup Pins

The primary way to cope with the risk of inadvertent Pin Validation failure is to keep a Backup Pin. A Backup Pin is a fingerprint for the public key of a secondary, not-yet-deployed key pair. The operator keeps the backup key pair offline, and sets a pin for it in the PKP header. Then, in case the operator loses control of their primary private key, they can deploy the backup key pair. UAs, who have had the backup key pair pinned (when it was set in previous Valid Pinning Headers), can connect to the host without error.

Because having a backup key pair is so important to recovery, UAs MUST require that hosts set a Backup Pin (see Section 2.5). The down side of keeping a not-yet-deployed key pair is that, if an attacker gains control of the private key, she will be able to perform a MITM attack without being discovered. Operators must take care to avoid leaking the key such as keeping it offline.

4.4. Interactions With Cookie Scoping

HTTP cookies [RFC6265] set by a Known Pinned Host can be stolen by a network attacker who can forge web and DNS responses so as to cause a client to send the cookies to a phony subdomain of the host. To prevent this, hosts SHOULD set the "secure" attribute and precisely scope the "domain" attribute on all security-sensitive cookies, such as session cookies. These settings tell the browser that the cookie should only be sent back to the specific host(s) (and not, e.g., all subdomains of a given domain), and should only be sent over HTTPS (not HTTP).

4.5. Hostile Pinning

An attacker who is able to obtain a valid certificate for a domain, either through misissuance by a Certification Authority or through other means, such as being the prior owner of a given domain, may attempt to perform ’hostile’ pinning. In this scenario, the attacker provides a Valid Pinning Header that pins to a set of SPKIs of the attacker’s choice. If a UA has not previously noted pins for that host, it may note the attacker’s pins, preventing access to the legitimate site.

This attack is mitigated through several means. Most prominently, the attack can only persist for the maximum max-age (see Section 4.1). Web host operators can reduce the opportunity for

attack by working to preload the host's pins within the UA. Operators may further detect such misissuance through other means, such as certificate transparency ([RFC6962]).

5. Privacy Considerations

Hosts can use HSTS or HPKP as a "super-cookie", by setting distinct policies for a number of subdomains. For example, assume example.com wishes to track distinct UAs without explicitly setting a cookie, or that a previously set cookie is deleted from the UA's cookie store. Here are two attack scenarios.

- example.com can use report-uri and the ability to pin arbitrary identifiers to distinguish UAs.

  1. example.com sets a Valid Pinning Header in its response to requests. The header asserts the includeSubDomains directive and specifies a report-uri directive as well. Pages served by the host also include references to subresource https://bad.example.com/foo.png.

  2. The Valid Pinning Header includes a "pin" that is not really the hash of an SPKI but is instead an arbitrary distinguishing string sent only in response to a particular request. For each request, the host creates a new, distinct distinguishing string and sets it as if it were a pin.

  3. The certificate chain served by bad.example.com does not pass Pin Validation given the pin set the host asserted in step (1). The HPKP-conforming UA attempts to report the Pin Validation failure to the specified report-uri, including the certificate chain it observed and the SPKI hashes it expected to see. Among the SPKI hashes is the distinguishing string in step (2).

- Different site operators/origins can optionally collaborate by setting the report-uri to be in an origin they share administrative control of. UAs MAY, therefore, refuse to send reports outside of the origin that set the PKP or PKP-RO header.

- example.com can use server name indication (SNI; [RFC3546]) and subdomains to distinguish UAs.

  1. example.com sets a Valid Pinning Header in its response to requests. The header asserts the includeSubDomains directive.

2. On a subsequent page view, the host responds with a page including the subresource https://0.fingerprint.example.com/foo.png, and the server responds using a certificate chain that does not pass Pin Validation for the pin-set defined in the Valid Pinning Header in step (1). The HPKP-conforming UA will close the connection, never completing the request to 0.fingerprint.example.com. The host may thus note that this particular UA had noted the (good) Pins for that subdomain.

3. example.com can distinguish 2^N UAs by serving Valid Pinning Headers from an arbitrary number N distinct subdomains. For any given subdomain n.fingerprint.example.com, the host may deliver a Valid Pinning Header to one UA, but not deliver it to a different UA. The server may then change the configuration for n.fingerprint.example.com. If the UA fails to connect, it was in the set of UAs that were pinned, which can be distinguished from the UAs that were not pinned, as they will succeed in connecting. The host may repeat this for a sufficient number of subdomains necessary to distinguish individual UAs.

- Conforming implementations (as well as implementations conforming to [RFC6797]) must store state about which domains have set policies, hence which domains the UA has contacted. Because these policies cause remotely detectable behaviors, it is advisable that UAs have a way for privacy-sensitive users to clear current Pins for Pinned Hosts and that UAs allow users to query the current state of Pinned Hosts. In addition, note that because pinning a host implies a degree of persistent state, an attacker with physical access to a device may be able to recover information about hosts a user has visited, even if the user has cleared other parts of the UA’s state.

- Pin reports, as noted in Section 3, contains information about the certificate chain that has failed pin validation. In some cases, such as 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.

6. IANA Considerations

IANA has registered the response headers described in this document under "Permanent Message Header Field Names" in the "Message Headers" registry [message-headers] with the following parameters:

- Header Field Names: Public-Key-Pins and Public-Key-Pins-Report-Only
- Protocol: http
- Status: standard
- Reference: RFC 7469

7. Usability Considerations

When pinning works to detect impostor Pinned Hosts, users will experience denial of service. It is advisable for UAs to explain the reason why, i.e., that it was impossible to verify the confirmed cryptographic identity of the host.

It is advisable that UAs have a way for users to clear current Pins for Pinned Hosts and that UAs allow users to query the current state of Pinned Hosts.

8. References

8.1. Normative References

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

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002, <http://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, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

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

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[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, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011, <http://www.rfc-editor.org/info/rfc6234>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, April 2011, <http://www.rfc-editor.org/info/rfc6265>.

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

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, June 2014, <http://www.rfc-editor.org/info/rfc7234>.

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

[W3C.REC-html401-19991224] Raggett, D., Hors, A., and I. Jacobs, "HTML 4.01 Specification", World Wide Web Consortium Recommendation REC-html401-19991224, December 1999, <http://www.w3.org/TR/1999/REC-html401-19991224>.

[message-headers]
IANA, "Message Headers",
<http://www.iana.org/assignments/message-headers/>.

8.2. Informative References

[RFC3546] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 3546, June 2003, <http://www.rfc-editor.org/info/rfc3546>.

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

[TACK] Marlinspike, M., "Trust Assertions for Certificate Keys", Work in Progress, draft-perrin-tls-tack-02, January 2013.

[why-pin-key]
Langley, A., "Public Key Pinning", Imperial Violet: Adam Langley’s Weblog, May 2011, <https://www.imperialviolet.org/2011/05/04/pinning.html>.

Appendix A. Fingerprint Generation

This Portable Operating System Interface (POSIX) shell program generates SPKI Fingerprints, suitable for use in pinning, from PEM-encoded certificates. It is non-normative.

```
openssl x509 -noout -in certificate.pem -pubkey | \
    openssl asn1parse -noout -inform pem -out public.key
openssl dgst -sha256 -binary public.key | openssl enc -base64
```

Figure 11: Example SPKI Fingerprint Generation Code

Appendix B. Deployment Guidance

This section is non-normative guidance that may smooth the adoption of public key pinning.

- Operators should get the backup public key signed by a different (root and/or intermediary) CA than their primary certificate, and store the backup key pair safely offline. The semantics of an SPKI Fingerprint do not require the issuance of a certificate to construct a valid Pin. However, in many deployment scenarios, in order to make a Backup Pin operational, the server operator will need to have a certificate to deploy TLS on the host. Failure to obtain a certificate through prior arrangement will leave clients that recognize the site as a Known Pinned Host unable to successfully perform Pin Validation until such a time as the operator can obtain a new certificate from their desired certificate issuer.

- It is most economical to have the backup certificate signed by a completely different signature chain than the live certificate, to maximize recoverability in the event of compromise of either the root or intermediary signer.

- Operators should periodically exercise their Backup Pin plan -- an untested backup is no backup at all.

- Operators should start small. Operators should first deploy public key pinning by using the report-only mode together with a report-uri directive that points to a reliable report collection endpoint. When moving out of report-only mode, operators should start by setting a max-age of minutes or a few hours and gradually increase max-age as they gain confidence in their operational capability.

Acknowledgements

Thanks to Tobias Gondrom, Jeff Hodges, Paul Hoffman, Ivan Krstic, Adam Langley, Barry Leiba, Nicolas Lidzborski, SM, James Manger, Yoav Nir, Trevor Perrin, Eric Rescorla, Pete Resnick, Tom Ritter, and Yan Zhu for suggestions and edits that clarified the text.

TACK [TACK] is a fruitful source of alternative design considerations.

Authors' Addresses

Chris Evans
Google, Inc.
1600 Amphitheatre Pkwy
Mountain View, CA 94043
United States

EMail: cevans@google.com

Chris Palmer
Google, Inc.
1600 Amphitheatre Pkwy
Mountain View, CA 94043
United States

EMail: palmer@google.com

Ryan Sleevi
Google, Inc.
1600 Amphitheatre Pkwy
Mountain View, CA 94043
United States

EMail: sleevi@google.com