Web Security
Internet-Draft
Intended status: Standards Track
Expires: October 31, 2014

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

April 29, 2014

Public Key Pinning Extension for HTTP
draft-ietf-websec-key-pinning-12

Abstract

This memo describes an extension to the HTTP protocol allowing web host operators to instruct user agents (UAs) to remember ("pin") the hosts' cryptographic identities for a given period of time. During that time, UAs will require that the host present 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 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.

Table of Contents

1. Introduction

We propose a new HTTP header to enable a web host to express to user agents (UAs) which Subject Public Key Info (SPKI) structure(s) UAs SHOULD expect to be present in the host's certificate chain in future connections using TLS (see [RFC5246]). We call this "public key pinning". At least one UA (Google Chrome) has experimented with the idea by shipping with a user-extensible embedded set of Pins. Although effective, this does not scale. This proposal addresses the scale problem.

Deploying public key pinning safely will require operational and organizational maturity due to the risk that hosts may make themselves unavailable by pinning to a SPKI that becomes invalid. (See Section 4.) We believe that, with care, host operators can greatly reduce the risk of MITM attacks and other false-authentication problems for their users without incurring undue risk.

We intend for hosts to use public key pinning together with HSTS ([RFC6797]), but is possible to pin keys without requiring HSTS.

This draft is being discussed on the WebSec Working Group mailing list, websec@ietf.org.

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 HTTP response header field (PKP header field) indicates to a UA that it should perform Pin Validation (Section 2.6) in regards to the host emitting the response message containing this header field, and provides the necessary information for the UA to do so.

Figure 1 describes the ABNF (Augmented Backus-Naur Form) syntax of the header field. It is based on the Generic Grammar defined in Section 2 of [RFC2616] (which includes a notion of "implied linear whitespace", also known as "implied *LWS").

```
Public-Key-Pins =
    "Public-Key-Pins" ":" [ directive ] *( ";" [ directive ] )
Public-Key-Pins-Report-Only =
    "Public-Key-Pins-Report-Only" ":" [ directive ] *( ";" [ directive ] )

directive        = simple-directive
                   / pin-directive

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

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

Figure 1: HPKP Header Syntax

token and quoted-string are used as defined in [RFC2616], Section 2.2.

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. All simple-directives MUST appear only once in a PKP header field. Directives are either optional or required, as stipulated in their definitions.

3. Directive names are case-insensitive.

4. UAs MUST ignore any PKP header fields containing directives, or other header field value data, that do not conform to the syntax defined in this specification.

5. If a PKP header field contains any directive(s) the UA does not recognize, the UA MUST ignore those directives.

6. If the PKP 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 PKP header field can be defined in other specifications, with a registry (having an IANA policy definition of IETF Review [RFC2616]) defined for them at such time. Such future directives will be ignored by UAs implementing only this specification, as well as by generally non-conforming UAs.

In the pin-directive, the token is the name of a cryptographic hash algorithm, and MUST be "sha256". (In the future, additional hash algorithms MAY be registered and used.) The quoted-string is a sequence of base 64 digits: the base 64-encoded SPKI Fingerprint ([RFC4648]). See Section 2.4.

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 connection passed Pin Validation with the UA's existing noted pins for the Host (i.e. the Host is a Known Pinned Host), the UA MUST cease to consider the Host as a Known Pinned Host. (I.e. the UA should fail open.) The UA SHOULD indicate to users that the Host is no longer a Known Pinned Host.

2.1.1. The max-age Directive

The REQUIRED "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 delta-seconds production is specified in [RFC2616].

The syntax of the max-age directive's REQUIRED value (after quoted-string unescaping, if necessary) is defined as:

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

Figure 2: max-age Value Syntax

delta-seconds is used as defined in [RFC2616], Section 3.3.2.

NOTE: A max-age value of zero (i.e., "max-age=0") signals the UA to cease regarding the host as a Known Pinned Host, including the includeSubDomains directive (if asserted for that Known Pinned Host). See Section 2.3.1.

2.1.2. The includeSubDomains Directive

The OPTIONAL "includeSubDomains" directive is a valueless directive which, 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.

2.1.3. 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 Public-Key-Pins or Public-Key-Pins-Report-Only header, 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, HTTPS, or other schemes. If the scheme in the report-uri is 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 Known Pinned Host.

UAs SHOULD make their best effort to report Pin Validation failures to the report-uri, but 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 (and MAY attempt to re-send the report later). 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.

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.

UAs MUST NOT send a report if the Host is not already a Known Pinned Host. (I.e., the UA's first connection to a Host fails Pin Validation.) The reason for this is so that a potential active network attacker cannot learn about a UA's certificate validation and Pin Validation procedures and state.

2.1.4. Examples

Figure 3 shows some example response header fields using the Pins extension (folded for clarity).

```
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="http://example.com/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 3: HPKP Header Examples

2.2. Server Processing Model

This section describes the processing model that Pinned Hosts implement. The model comprises two facets: the processing rules for HTTP request messages received over a secure transport (e.g. TLS [RFC5246]); and 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 that MUST satisfy the grammar specified above in Section 2.1.

Establishing a given host as a Known Pinned Host, in the context of a given UA, MAY be accomplished over the HTTP protocol, which is in

turn running over secure transport, by correctly returning (per this specification) at least one valid PKP header field to the UA. Other mechanisms, such as a client-side pre-loaded Known Pinned Host list MAY also be used.

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

This section describes the HTTP Public Key Pinning processing model for UAs.

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:

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

or,

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

- o 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.

- o 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 in an HTTP response message over secure transport, then the UA MUST process only the first such header field.

Otherwise:

- 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).

- The UA MUST ignore any PKP 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 Public-Key-Pins header and SHOULD process both.

The Public-Key-Pins header is processed as according to Section 2.3.1.

When the Public-Key-Pins-Report-Only 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 Public-Key-Pins-Report-Only header, the UA performs Pin Validation only to check whether or not it should POST a report, but not for causing connection failure.

Note: There is no purpose to using the Public-Key-Pins-Report-Only header without the report-uri directive. User Agents MAY discard such headers without interpretting them further.

If a Host sets the Public-Key-Pins-Report-Only header, the UA SHOULD note the Pins and directives given in the Public-Key-Pins-Report-Only header as specified by the max-age directive. If the UA does note the Pins and directives in the Public-Key-Pins-Report-Only 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 Public-Key-Pins header and the Public-Key-Pins-Report-Only header, the UA MUST note and enforce Pin Validation as specified by the Public-Key-Pins header, and SHOULD note the Pins and directives given in the Public-Key-Pins-Report-Only header. If the UA does note the Pins and directives in the Public-Key-Pins-

Report-Only 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 all expired Known Pinned Hosts from its cache if, at any time, an expired Known Pinned Host exists in the cache.

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 a Known Pinned Host's domain name, per the matching procedure specified in Section 8.2 of [RFC6797], then the UA MUST note this host as a Known Pinned Host, caching the Pinned Host's domain name and noting along with it the Effective Expiration Date (or enough information to calculate it, i.e. the Effective Pin Date and the value of the max-age directive), whether or not the includeSubDomains directive is asserted, the value of the report-uri directive (if present). If any other metadata from optional or future PKP header directives is present in the Valid Pinning Header, the UA MAY note them if it understands them, and need not note them if it does not understand them.

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 5184000 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.

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 SubjectPublicKeyInfo (SPKI) field 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. (See [RFC4648].)

In this version of the specification, the known cryptographic hash algorithm is SHA-256, identified as "sha256" ([RFC4634]). (Future versions of this specification 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; UAs MUST recognize "sha256".

Figure 4 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 4: SPKI Definition

If the SubjectPublicKeyInfo of a certificate 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 SubjectPublicKeyInfo fields in certificates.

2.5. Noting Pins

Upon receipt of the Public-Key-Pins response header field, the UA notes the host as a 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 observe these conditions when noting a Host:

- The UA MUST note the Pins if and only if it received the Public-Key-Pins 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 UA MUST note the Pins if and only if 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 UA MUST note the Pins if and only if 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 Public-Key-Pins response header field does not meet all three of these criteria, the UA MUST NOT note the host as a Pinned Host. A Public-Key-Pins response header field that meets all these critera 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, 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 Public-Key-Pins 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.

2.6. Validating Pinned Connections

When a UA connects to a Pinned Host, 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, but MAY allow Pin Validation to be disabled for 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.

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. (For the purposes of Pin Validation, the UA MUST ignore certificates whose SPKI cannot be taken in isolation, and 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 Failure as a non-recoverable error. Any procedure that matches the results of this Pin Validation procedure is considered equivalent.

Note that, although the UA has previously received Pins at the HTTP layer, it can and MUST perform Pin Validation at the TLS layer, before beginning an HTTP conversation over the TLS channel. The TLS layer thus evaluates TLS connections with pinning information the UA received previously, regardless of mechanism: statically preloaded, via HTTP header, or some other means (possibly in the TLS layer itself).

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.

UAs that support additional sources of pinning information MUST use the most recently observed pinning information when performing Pin Validation for a host. The most recently observed pinning information is determined based upon the most recent Effective Pin Date, as described in Section 2.3.3. The Effective Pin Date of built-in pin lists is UA implementation-defined.

If the result of noting a Valid Pinning Header is to disable pinning for the host, such as through supplying a max-age directive with a value of 0, UAs MUST allow this new information to override any other

pinning data. That is, a host must be able to un-pin itself, even in the presence of built-in Pins.

Example: A UA may ship with a pre-configured list of Pins that are collected from past observations of Valid Pinning Headers supplied by hosts. In such a solution, the pre-configured list should track when the Valid Pinning Header was last observed, in order to permit site operators to later update the value by supplying a new Valid Pinning Header. Updates to such a pre-configured list should not update the Effective Pin Dates for each host unless the list vendor has actually observed a more recent header. This is to prevent situations where updating the Effective Pin Date on a pre-configured list of Pins may effectively extend the max-age beyond the site operator's stated policy.

Example: A UA may ship with a pre-configured list of Pins that are collected through out-of-band means, such as direct contact with the site operator. In such a solution, the site operator accepts responsibility for keeping the configured Valid Pinning Header in sync with the vendor's list, allowing the UA vendor to have each update to the list be treated as as an update of the Effective Pin Date.

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 ([RFC4627]) 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,
  "served-certificate-chain": [
    pem1, ... pemN
  ],
  "validated-certificate-chain": [
    pem1, ... pemN
  ],
  "known-pins": [
    known-pin1, ... known-pinN
  ]
}
```

Figure 5: JSON Report Format

Whitespace outside of quoted strings is not significant. The key/value pairs may appear in any order, but each SHOULD 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 either as a string or 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.

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 PEM representation of each X.509 certificate as described in [I-D.josefsson-pkix-textual].

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 [I-D.josefsson-pkix-textual]. For UAs that build certificate chains in more than one way during the validation process, they 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 6: Known Pin Syntax

As in Section 2.4, the token refers to the algorithm name, and the quoted-string refers to the base 64 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 \-escape the embedded double quotes in the quoted-string part of the known-pin.

Figure 7 shows an example of a Pin Validation failure report. (PEM strings are shown on multiple lines for readability in this document.)

```
{
  "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 7: 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 could 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 potentially 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 (5184000 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 8: 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 9: 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 conections. 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 first 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 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 Public-Key-Pins 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.)

5. Privacy Considerations

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. A forensic attacker might find this information useful, even if the user has cleared other parts of the UA's state.

More importantly, 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 if 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 (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).

- example.com can use SNI 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, giving some UAs Valid Pinning Headers for some, but not all subdomains (causing subsequent requests for n.fingerprint.example.com to fail), and giving some UAs no Valid Pinning Header for other subdomains (causing subsequent requests for m.fingerprint.example.com to succeed).

6. IANA Considerations

IANA is requested to register the header described in this document in the "Message Headers" registry, with the following parameters:

- Header Field Names should be "Public-Key-Pins" and "Public-Key-Pins-Report-Only".

- Protocol should be "http"

- Status should be "standard"

- Reference should be this document

7. Usability Considerations

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

UAs MUST have a way for users to clear current Pins for Pinned Hosts. UAs SHOULD have a way for users to query the current state of Pinned Hosts.

8. Acknowledgements

Thanks to Tobias Gondrom, Jeff Hodges, Paul Hoffman, Ivan Krstic, Adam Langley, Nicolas Lidzborski, SM, James Manger, Yoav Nir, Eric Rescorla, and Tom Ritter for suggestions and edits that clarified the text. Thanks to Trevor Perrin for suggesting a mechanism to affirmatively break Pins ([pin-break-codes]).

9. What's Changed

[RFC EDITOR: PLEASE REMOVE THIS SECTION]

Removed the strict directive.

Removed the requirement that the server set the Valid Pinning Header on every response.

Added normative references for SHA, JSON, and base-64.

Added the Privacy Considerations section.

Changed non-normative pin generation code from Go to POSIX shell script using openssl.

Changed max-max-age from SHOULD to MAY, and used the example of 60 days instead of 30.

Removed the section "Pin Validity Times", which was intended to be in harmony with [I-D.perrin-tls-tack]. Now using max-age purely as specified in [RFC6797].

Added new directives: includeSubDomains, report-uri and strict.

Added a new variant of the PKP Header: Public-Key-Pins-Report-Only.

Removed the section on pin break codes and verifiers, in favor the of most-recently-received policy (Section 2.5).

Now using a new header field, Public-Key-Pins, separate from HSTS. This allows hosts to use pinning separately from Strict Transport Security.

Explicitly requiring that UAs perform Pin Validation before the HTTP conversation begins.

Backup Pins are now required.

Separated normative from non-normative material. Removed tangential and out-of-scope non-normative discussion.

10. References

10.1. Normative References

[I-D.josefsson-pkix-textual]
Josefsson, S. and S. Leonard, "Text Encodings of PKIX and CMS Structures", draft-josefsson-pkix-textual-02 (work in progress), October 2013.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC4634] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, July 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[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.

[RFC6797] Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", RFC 6797, November 2012.

[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>.

10.2. Informative References

[I-D.perrin-tls-tack]
Marlinspike, M., "Trust Assertions for Certificate Keys", draft-perrin-tls-tack-02 (work in progress), January 2013.

[pin-break-codes]
Perrin, T., "Self-Asserted Key Pinning", September 2011, <http://trevp.net/SAKP/>.

[why-pin-key]
Langley, A., "Public Key Pinning", May 2011, <http://www.imperialviolet.org/2011/05/04/pinning.html>.

Appendix A. Fingerprint Generation

This 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 | base64
```

Figure 10: Example SPKI Fingerprint Generation Code

Appendix B. Deployment Guidance

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

- o 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 either root or intermediary signer compromise.

- 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.

Authors' Addresses

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

Email: cevans@google.com

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

Email: palmer@google.com

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

Email: sleevi@google.com