Web Security
Internet-Draft
Intended status: Standards Track
Expires: April 19, 2013

C. Evans
C. Palmer
Google, Inc.

October 16, 2012

Public Key Pinning Extension for HTTP
draft-ietf-websec-key-pinning-03

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 or more of the pinned fingerprints for that host. By effectively reducing the scope 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 and other authentication errors and attacks.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 19, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

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 MUST expect to be present in the host's certificate chain in future connections using TLS (see [rfc-5246]). We call this "public key pinning". At least one user agent (Google Chrome) has experimented with 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 3.) 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 (as defined in [hsts-draft], 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 [rfc-2119].

2. Server and Client Behavior

2.1. Response Header Field Syntax

To set a pin, hosts use a new HTTP header field, Public-Key-Pins, in their HTTP responses. Figure 1 describes the syntax of the header field.

```
Public-Key-Pins = "Public-Key-Pins" ":" LWS directives

directives      = max-age LWS ";" LWS pins
                  / pins LWS ";" LWS max-age

max-age         = "max-age" LWS "=" LWS delta-seconds

pins            = pin
                  / pin LWS ";" LWS pins

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

Figure 1

In the pin rule, the token is the name of a cryptographic hash algorithm, and MUST be either "sha1" or "sha256". (Future versions of this specification may change the hash functions.) The quoted-string is a sequence of base64 digits: a base64-encoded hash. See Section 2.2.

Figure 2 shows some example response header fields using the pins extension (folded for clarity).

```
Public-Key-Pins: max-age=500;
    pin-sha1="4n972HfV354KP560yw4uqe/baXc=";
    pin-sha1="IvGeLsbqzPxdI0b0wuj2xVTdXgc="

Public-Key-Pins: max-age=31536000;
    pin-sha1="4n972HfV354KP560yw4uqe/baXc=";
    pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ="

Public-Key-Pins: pin-sha1="4n972HfV354KP560yw4uqe/baXc=";
    pin-sha1="qvTGHdzF6KLavt4PO0gs2a6pQ00=";
    pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ=";
    max-age=2592000
```

Figure 2

2.2. Semantics of Pins

The fingerprint is the SHA-1 or SHA-256 hash of the DER-encoded ASN.1 representation of the SubjectPublicKeyInfo (SPKI) field of the X.509 certificate. Figure 3 reproduces the definition of the SubjectPublicKeyInfo structure in [rfc-5280].

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

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

Figure 3

The SPKI hash is then encoded in base-64 for use in an HTTP header. (See [rfc-4648].)

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 public key fingerprints from SubjectPublicKeyInfo fields in certificates.

2.3. 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 max-age in non-volatile storage (for example, along with the HSTS metadata). The pins and their associated max-age 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. The UAs MUST ignore Public-Key-Pins response header fields received on HTTP (non-HTTPS) connections.

- 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 fingerprints. (See Section 2.4.)

- 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 3.1.)

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, and MUST discard any previously set Pinning Metadata for that host in its non-volatile store. Public-Key-Pins response header fields that meet all these critera are known as Valid Pinning Headers.

Whenever a UA receives a Valid Pinning Header, it MUST set its Pinning Metadata to the exact pins and max-age given in the most recently received Valid Pinning Header.

2.3.1. max-age

max-age specifies the number of seconds, after the reception of the Public-Key-Pins HTTP Response Header, during which the UA regards the host as a Pinned Host (up to a maximum of 30 days; see Section 2.5). The delta-seconds production is specified in [rfc-2616].

Note that by setting a low or 0 value for max-age, hosts effectively instruct UAs to cease regarding them as Pinned Hosts.

2.4. 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 [hsts-draft].)

If the connection has no errors, the UA will then apply a new, additional correctness check: Pin Validation. To perform Pin Validation, the UA will compute the fingerprints of the SPKI structures in each certificate in the host’s validated certificate chain. (For the purposes of Pin Validation, the UA MUST ignore certificates who SPKI cannot be taken in isolation and superfluous certificates in the chain that do not form part of the validating chain.) The UA will then check that the set of these fingerprints intersects the set of fingerprints in that 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.

Note that, although the UA has previously received public key 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, such as specified in [tack-draft]).

2.5. Pin Validity Times

In harmony with section 5.3.4 "Create and activate pins" of [tack-draft], clients MUST enforce a maximum age for pins that is no longer than the least of (a) 30 days (30 * 24 * 60 * 60 seconds) after the most recent time that the client noted the pin; (b) the amount of time the pin has been noted; or (c) the most recent time the pin was noted + max-age:

```
active_period_end = MIN(current + current - initial,
                        time_pin_noted + max-age,
                        current + 30 days)
```

Figure 4

2.6. Interactions With Preloaded Pin Lists

UAs MAY choose to implement built-in public key pins, alongside any built-in HSTS opt-in list. UAs MUST allow users to override a built-in pin list, including turning it off.

UAs MUST use the newest information -- built-in or set via Valid Pinning Header -- when performing Pin Validation for the host.

2.7. 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. Security Considerations

Pinning public keys helps hosts assert their cryptographic identity, but there is some risk that a host operator could lose or lose control of their host's private key. In this case, 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.)

3.1. Backup Pins

The primary way to cope with the risk of inadvertant 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 2.3.)

4. IANA Considerations

This document has no actions for IANA.

5. 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.

6. Acknowledgements

Thanks to Tobias Gondrom, Jeff Hodges, Adam Langley, Nicolas Lidzborski, SM, James Manger, and Yoav Nir for suggestions and edits that clarified the text. Thanks to Trevor Perrin for suggesting a mechanism to affirmatively break pins ([pin-break-codes]). Adam Langley provided the SPKI fingerprint generation code.

7. What's Changed

Removed the section on pin break codes and verifiers, in favor the of most-recently-received policy (Section 2.3).

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.

8. References

8.1. Normative References

[hsts-draft]
Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", October 2011, <http://tools.ietf.org/html/draft-ietf-websec-strict-transport-sec-03>.

[rfc-2119]
Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.

[rfc-2616]
Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", June 1999, <http://www.ietf.org/rfc/rfc2616.txt>.

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

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

[rfc-5280]
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", May 2008, <http://www.ietf.org/rfc/rfc5280.txt>.

8.2. Informative References

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

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

[tack-draft]
Perrin, T. and M. Marlinspike, "Trust Assertions for Certificate Keys", May 2012, <http://tools.ietf.org/html/draft-perrin-tls-tack>.

Appendix A. Fingerprint Generation

This Go program generates public key fingerprints, suitable for use in pinning, from PEM-encoded certificates. It is non-normative.

```
package main

import (
        "io/ioutil"
        "os"
        "crypto/sha1"
        "crypto/x509"
        "encoding/base64"
        "encoding/pem"
        "fmt"
)

func main() {
        if len(os.Args) < 2 {
                fmt.Printf("Usage: %s PEM-filename\n", os.Args[0])
                os.Exit(1)
        }
        pemBytes, err := ioutil.ReadFile(os.Args[1])
        if err != nil {
                panic(err.String())
        }
        block, _ := pem.Decode(pemBytes)
        if block == nil {
                panic("No PEM structure found")
        }
        derBytes := block.Bytes
        certs, err := x509.ParseCertificates(derBytes)
        if err != nil {
                panic(err.String())
        }
        cert := certs[0]
        h := sha1.New()
        h.Write(cert.RawSubjectPublicKeyInfo)
        digest := h.Sum()

        fmt.Printf("Hex: %x\nBase64: %s\n", digest,
                base64.StdEncoding.EncodeToString(digest))
}
```

Figure 5

Appendix B. Deployment Guidance

This section is non-normative guidance which 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.

- 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 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