Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 26 December 2025

D. Wing
Citrix

24 June 2025

Public Key Hash for Local Domains
draft-wing-settle-public-key-hash-01

Abstract

This specification eliminates security warnings when connecting to local domains using TLS. Servers use a unique, long hostname which encodes their public key that the client validates against the public key presented in the TLS handshake.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://danwing.github.io/public-key-hash/draft-wing-settle-public-key-hash.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-wing-settle-public-key-hash/.

Discussion of this document takes place on the SETTLE mailing list (mailto:settle@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/settle/. Subscribe at https://www.ietf.org/mailman/listinfo/settle/.

Source for this draft and an issue tracker can be found at https://github.com/danwing/public-key-hash.

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 https://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 26 December 2025.

Table of Contents

1. Introduction

Browsers are progressively requiring secure origins for new capabilities and features ([secure-context]). As secure origins are only obtainable, today, with a certificate signed by a Certification Authority trusted by the client, this leaves out devices and networks which cannot easily obtain such certificates. Such inability is due to network topology (e.g., firewall), lack of domain ownership, or

complicated procedures.

This draft discusses how a client can authenticate to HTTPS servers belonging to the local domain where the server name is a hash of the server's public key. By doing so, a secure origin can be established. This avoids the need for a certificate signed by a Certification Authority (CA) trusted by the client. This is a relaxed way of "doing HTTPS" for servers on the local domain.

2. Unique Host Names

Web browsers and other application clients store per-host state using the host name, including cached form data such as passwords, integrated and 3rd party password managers, cookies, and other data. When a name collision occurs (e.g., the same printer.local name on two different networks) the client cannot recognize a different host is being encountered. By creating a unique name, existing client software (browsers, password managers, client libraries) can continue storing origin-specific data for each of unique name.

A unique name is created by embedding the hash of the public key into the name itself. This achieves uniqueness and the encoding is also identifiable by the client to assist its validation of the server's public key (Section 4.1). Details on encoding the domain name are in Section 5.

3. Short Host Names

Unique host names containing encoded public keys are awkward for users. This section describes how short names can also be advertised by servers and securely validated by clients, so that the short name is presented to users while the unique name is used for the TLS connection.

A server already advertising its long name using [DNS-SD] can also advertise its short name. The client needs to validate they are the same server, prior to allowing the user to interact with the short name. The client can do this validation by making two connections: one connection to the long name and another to the short name and verify they both return the same public key and that both TLS handshakes finish successfully (proving the server has possession of the associated private key). Advertising both names enables incremental deployment within a local domain at the expense of user confusion (two names per device, like a human named both "Bob" and "Richard") and on-the-wire inefficiency.

NOTE: Also to be considered is including both the unique and short host name in the SubjectAltName field of the server's certificate. This avoids an additional advertisement but has worse incremental deployment characteristics -- legacy software won't notice the other name in the SubjectAltName.

The client need only look for matching short name and unique name within the same TLD domain name (that is, if a unique name is advertised with a ".local" domain, the client does not need to look for its accompanying short name within ".internal").

To avoid the problems described in Section 2, the TLS data connection always uses the long name. Thus, after the client has validated the short name as described above and a user attempts to connect to the short name (by typing or by some other user interaction), the client instead makes a connection to the unique name. This reduces the integration changes within the client, as clients already separate server-specific data based on the server's hostname (e.g., Cookie Store API, Credential Management API, Web Bluetooth, Storage API, Push API, Notifications API, WebTransport API).

4. Operation

4.1. Client Operation

When clients connect to such a local domain name or IP address (Section 6) using TLS they examine if the domain name starts with a registered hash identifier in the second label and if the rest of that label consists of an appropriate-length encoded hash. If those conditions apply, the client performs certificate validation as described below.

Upon receipt of the server's certificate, the client validates validates the certificate ([RFC9525], [RFC5280], and Section 4.3.4 of [RFC9110] if using HTTPS). When performing such a connection to a local domain, the client might avoid warning about a self-signed certificate because the Certification Authority (CA) signature will certainly not be signed by a trusted CA. Rather, a more subtle indication might be warranted for TLS connections to a local domain, perhaps only the first time or perhaps each time. The client parses the returned certificate and extracts the public key and compares its hash with the hash contained in the hostname. If they match, the TLS session continues. If they do not match, the client might warn the user about the certificate (as is common today) or simply abort the TLS connection.

Authorizing which servers are allowed on the local network is discussed in Section 7.2.

4.2. Server Operation

A server running on a local network (see Section 2) uses a unique host name that includes a hash of its public key. This unique name is encoded as described in Section 5. Existing servers might be configurable with such a hostname, without software changes.

Oftentimes, servers operating on a local network already advertise their presence using [DNS-SD] and should continue doing so, advertising their unique name that includes their public key hash and optionally also a shorter nickname (Section 3).

5. Unique Host Name Encoding Details

The general format is hostname, a period, a digit indicating the hash algorithm, and then the hash of the server's public key as shown in Figure 1. The binary hash output is base32 encoded (Section 6 of [RFC4648]) without trailing "=" padding. Currently only SHA256 hash is defined with the value "0" (Section 8). While base32 encoding is specified as uppercase, implementations should treat uppercase, lowercase, and mixed case the same. (Base32 encoding was chosen instead of base64 or base64url encoding to avoid their illegal hostname characters and their case preservation requirement.)

```
friendly-name = 1*63(ALPHA / DIGIT / "-")

hash-algorithm = DIGIT   ; 0=SHA256

base32-digits = "2" / "3" / "4" / "5" / "6" / "7"

hash = 1*62(/ ALPHA / base32-digits )
     ; 62+1 octet limit from RFC1035

encoded-hostname = friendly-name "."
                   hash-algorithm
                   hash
```

Figure 1: ABNF of hostname encoding

An example encoding is shown in Appendix A.

6. Identifying Servers as Local

This section defines the domain names and IP addresses considered "local".

6.1. Local Domain Names

The following domain name suffixes are considered "local":

* ".local" (from [mDNS])
* ".home-arpa" (from [Homenet])
* ".internal" (from [I-D.davies-internal-tld])
* both ".localhost" and "localhost" (Section 6.3 of [RFC6761])

6.2. Local IP Addresses

Additionally, if any host resolves to a local IP address and connection is made to that address, those are also considered "local":

* 10/8, 172.16/12, and 192.168/16 (from [RFC1918])
* 169.254/16 and fe80::/10 (from [RFC3927] and [RFC4291])
* fc00::/7 (from [RFC4193])
* 127/8 and ::1/128 (from Section 3.2.1.3 of [RFC1122] and [RFC4291])

7. Security Considerations

7.1. Validating Hostname Authenticity

By associating a server's public key with its origin (defined as the scheme, hostname, and port per [RFC6454]), a client can perform a TLS handshake with the server to ensure the server possesses the private key associated with that key hash. This allows the client to validate the authenticity of a hostname advertised on the local network (such as advertised via mDNS).

7.2. Authorizing Servers on Local Domain

A client may also want to defend against rogue servers installed on the local domain. This requires legitimate servers be enrolled with a trust anchor system such as a local domain Certification Authority (e.g., [I-D.sweet-iot-acme]) or other system (e.g., [EST]) and that enrollment verified by the client.

7.3. Public Key Hash

Because the server's public key is encoded into its hostname, changing the public key would also change its hostname -- thus, its identity as known by the client changes, disrupting configuration of that server on the client (e.g., printer configuration, SMB share configuration, password manager). As such, an identity change is extremely disruptive, it needs to be avoided. This means the public/private key pair on a server needs to stay static. The tradeoff is a stolen private key allows an attacker to intercept traffic to that server for a long time, without requiring the attacker to retain access to the server (to steal its changed private key). This is not ideal but superior to plain text communication or conditioning users to accept self-signed certificate warnings for servers on the local domain.

8. IANA Considerations

New registry for hash type, 0=SHA256. Extensions via IETF Action.

9. References

9.1. Normative References

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

9.2. Informative References

[DNS-SD] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/rfc/rfc6763>.

[EST] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <https://www.rfc-editor.org/rfc/rfc7030>.

[Homenet] Pfister, P. and T. Lemon, "Special-Use Domain 'home.arpa.'", RFC 8375, DOI 10.17487/RFC8375, May 2018, <https://www.rfc-editor.org/rfc/rfc8375>.

[I-D.davies-internal-tld]
Davies, K., McConachie, A., and W. Kumari, "A Top-level Domain for Private Use", Work in Progress, Internet-Draft, draft-davies-internal-tld-03, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-davies-internal-tld-03>.

[I-D.sweet-iot-acme]
Sweet, M., "ACME-Based Provisioning of IoT Devices", Work in Progress, Internet-Draft, draft-sweet-iot-acme-07, 7 February 2025, <https://datatracker.ietf.org/doc/html/draft-sweet-iot-acme-07>.

[mDNS] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/rfc/rfc6762>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/rfc/rfc1122>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <https://www.rfc-editor.org/rfc/rfc1918>.

[RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, DOI 10.17487/RFC3927, May 2005, <https://www.rfc-editor.org/rfc/rfc3927>.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <https://www.rfc-editor.org/rfc/rfc4193>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/rfc/rfc4291>.

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

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <https://www.rfc-editor.org/rfc/rfc6454>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <https://www.rfc-editor.org/rfc/rfc6761>.

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

[RFC9525] Saint-Andre, P. and R. Salz, "Service Identity in TLS", RFC 9525, DOI 10.17487/RFC9525, November 2023, <https://www.rfc-editor.org/rfc/rfc9525>.

[secure-context] W3C, "Web Platform Design Principles", June 2024, <https://w3ctag.github.io/design-principles/#secure-context>.

Appendix A. Example Encoding

Server with private key in PEM format is:

```
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCOkTU8TOKW/IZ6
whhyhg+1I4u0jm019z4SVUKAwKyOtFPtoEvRGPdKYUoLaPZmVyN/VTYlubVn7dE8
IYpibwkDhs38DKmyo0vdWjUaopQOaygtLB+PZ12l/XaSCE1cWsl45ShUvktcVR/D
DKwv7DWEIZrRTLKy6M+8Ne4x++kKXmwbSj8WsNQ4kU4uFhS+JXHXToZvhvoQLkTs
486XXPn4SJPLyTN62b6KHTLM1vb7RY4i4U7N6hS1UWe7bIxZNZ0vnf4vZ7A6SR7W
nM31DoaW5XCBH7CL56MSdn7dmuksRimfNmnsEmvBXZmuQMHnUZghBLMHPC9xmHhT
8q3pSY5jAgMBAAECggEANiAY1vDVob7zi01/HJObCQkatAzSl4drUGiAHAOKP49k
wbV2s0bIM7vl8ZkC2u3AM0p1iTMNFQzrv+l38VD4WhdmwodIMeLfHYVu3dLVZPf3
w9aZkMcMfcVRq7VtMV/iV3ygqDOqxr4mldWM1ZDW7HgZn9Z/jX7nxyuuZ9mcquuH
Brl8pcUba7666jcz+F9NNjXTPCwfm7ihCPkTeYr1NflQGTR5PJ+D5dywb53iulm1
ZTk2zBXJMujbIyTL0p+MqdEKXci7oQJqf7bQsxsO2ZUD24CmzYldsE6vmYUFxJpw
ZbYzO/a/Mv0mXQhcUTWKkJkU78QT2Us7SuSL+IPGSQKBgQDC5iRKtlYulUgxV9gu
TmX30R0W7R0nnsEjolNAqUwcIoUMHk8ODXEsp7jVOSFMJhHRMXL+VKYiBsiIV7vk
GlTbLRP34HgK54auRF6PTxBfNAkF+FQxl2mzWxj7wi5mg0g+tCJTLereUXULz8+r
h5Vqp4BCjcoumlyY0xlLtbr9/wKBgQC7Qx2Lb70XCL5eivdokMh2lRint9cfxC2W
fJ6QOnJgsN9XIQGTUAk3cLvmrKg3UOmJXXq+Q6djVB/3Op3+TFzsGS2ORMel9r6o
kAHYG/qdairlW9uTDsnwUP8UtE0lidhSXLGIAy71eMDbDg/c/yyrWTvysXf5kAiJ
CzTnyvY3nQKBgBt+Va5IbH3jxyxWxQM7Qf0kfaMHTe6R4ZMCShY8C6WIZRZhjCti
UA3JlzRU+9J/KFJHVH52OH1iUZWSMsopwMCuaju0aZq4MHKS6Hf04k1bzM4Pyui4
AEwx1KNnMB579IwL4y+ysYgtG4LQDO6YkMZb3KcG03ehhOB2HwJkH33HAoGATOw3
8bQ3v4OG970r/lcjEZsTYqnhA5qJg3yzgdmQbGmbhOX5CLNi5dQ4S3x3KSnilNvC
dO/DjcjbzKnWhsSFkzKQhRV50ZH3JbTqHQT5QLqA3nCKVPFJQJ90+ONLoXTrWIHd
J1rvakRtLE6tc4GartRcDMib2PcymmDxHZpA4/0CgYEAs0XF1G0gmnef8oEYuwZT
c+vr4wnD7YCP1h8nsNSgRHLk1e7k727iHGvruX3qrKsY26RHKi2+i1P6A39I4F5s
3Dme4HGXTyoc/qKp+/GAx5XYVG4c3Z3sdBejkpkhPTSlsSsDOHbjaiFV1zCyEdg5
fOPfIBX8uLc3UtOm0+Gn1IQ=
-----END PRIVATE KEY-----
```

and public key in PEM format is:

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAjpE1PEzilvyGesIYcoYP
tSOLtI5tNfc+ElVCgMCsjrRT7aBL0Rj3SmFKC2j2Zlcjf1U2Jbm1Z+3RPCGKYm8J
A4bN/AypsqNL3Vo1GqKUDmsoLSwfj2ddpf12kghNXFrJeOUoVL5LXFUfwwysL+w1
hCGa0UyysujPvDXuMfvpCl5sG0o/FrDUOJFOLhYUviVx106Gb4b6EC5E7OPOl1z5
+EiTy8kzetm+ih0yzNb2+0WOIuFOzeoUtVFnu2yMWTWdL53+L2ewOkke1pzN9Q6G
luVwgR+wi+ejEnZ+3ZrpLEYpnzZp7BJrwV2ZrkDB51GYIQSzBzwvcZh4U/Kt6UmO
YwIDAQAB
-----END PUBLIC KEY-----
```

Using the binary format (DER) and hashed using SHA256 gives this hex value:

```
21ebc0d00e98e3cb289738e2c091e532c4ad8240e0365b22067a1449693e5a18
```

Converting that hex value to binary and base32 encoded (without trailing "=") gives:

EHV4BUAOTDR4WKEXHDRMBEPFGLCK3ASA4A3FWIQGPIKES2J6LIMA

After the hash algorithm identification digit (0 for SHA512/256) is prefixed to that base64url, resulting in:

0EHV4BUAOTDR4WKEXHDRMBEPFGLCK3ASA4A3FWIQGPIKES2J6LIMA

Finally, if this is a printer named "printer" advertised using ".local", the full FQDN for its unique name would be:

printer.0EHV4BUAOTDR4WKEXHDRMBEPFGLCK3ASA4A3FWIQGPIKES2J6LIMA.local

and the full FQDN for its short name would be "printer.local".

Acknowledgments

This draft was inspired by a document published by Martin Thomson in 2007; however, this draft takes a different approach by using unique names over the wire.

Other systems have also utilized public key hashes in an identifier including Tor and Freenet’s Content Hash Key.

Thanks to Sridharan Rajagopalan for reviews and feedback.

Author’s Address

Dan Wing
Citrix
United States of America
Email: danwing@gmail.com