Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 7, 2016

W. Koch
GnuPG Project
May 6, 2016

OpenPGP Web Key Service
draft-koch-openpgp-webkey-service-00

Abstract

This specification describes a service to locate OpenPGP keys by mail address using a Web service and the HTTPS protocol. It also provides a method for secure communication between the key owner and the mail provider to publish and revoke the public key.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This memo describes a method to associate OpenPGP keys with a mail address and now to look them up using a web service with a well-known URI. In addition a mail based protocol is given to allow a client to setup such an association and to maintain it.

2. Notational Conventions

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 [RFC2119].

3. Web Key Directory

A major use case for OpenPGP is the encryption of mail. A common difficulty of sending encrypted mails to a new communication partner is to find the appropriate public key of the recipient. Unless an off-channel key exchange has been done, there are no easy ways to discover the required key. The common practice is to search the network of public key servers for a key matching the recipient's mail address. This practise bears the problem that the keyservers are not able to give a positive confirmation that a key actually belongs to the mail addresses given in the key. Further, there are often several keys matching a mail address and thus one needs to pick a key

on good luck. This is clearly not a secure way to setup an end-to-end encryption. Even if the need for a trusted key for an initial mail message is relinquished, a non-authenticated key may be a wrong one and the actual recipient would receive a mail which she can't decrypt, due to the use of a wrong key.

Methods to overcome this problem are

- o sending an initial unencrypted messages with the public key attached,

- o using the OpenPGP DANE protocol to lookup the recipients key via the DNS.

The first method has the obvious problems of not even trying to encrypt the initial mail, an extra mail round-trip, and problems with unattended key discovery.

The latter method works fine but requires that mail providers need to set up a separate DNS resolver to provide the key. The administration of a DNS zone is often not in the hands of small mail installations. Thus an update of the DNS resource records needs to be delegated to the ISP running the DNS service. Further, DNS lookups are not encrypted and missing all confidentially. Even if the participating MUAs are using STARTTLS to encrypt the mail exchange, a DNS lookup for the key unnecessary identifies the local-part of the recipients mail address to any passive eavesdroppers.

This memo specified a new method for key discovery using an encrypted https connection.

3.1. Key Discovery

Although URIs are able to encode all kind of characters, straightforward implementations of a key directory may want to store the "local-part" of a mail address directly in the file system. This forbids the use of certain characters in the "local-part". To allow for such an implementation method the URI uses an encoded form of the "local-part" which can be directly mapped to a file name.

OpenPGP defines its User IDs, and thus the mail address, as UTF-8 strings. To help with the common pattern of using capitalized names (e.g. "Joe.Doe@example.org") for mail addresses, and under the premise that almost all MTAs treat the "local-part" case-insensitive and that the "domain-part" is required to be compared case-insensitive anyway, all upper-case ASCII characters in a User ID are mapped to lowercase. Non-ASCII characters are not changed.

The so mapped "local-part" is hashed using the SHA-1 algorithm. The resulting 160 bit digest is encoded using the Z-Base-32 method as described in [RFC6189], section 5.1.6. The resulting string has a fixed length of 32 octets. To form the URI, the scheme "https://" is concatenated with the mapped "domain-part", the fixed string "./well-known/openpgpkey/hu/", the "domain-part" again, and the above constructed 32 octet string.

For example the URI to lookup the key for Joe.Doe@Example.ORG is:

```
  https://example.org/.well-known/openpgpkey/
  hu/example.org/iy9q119eutrkn8s1mk4r39qejnbu3n5q
```

(line has been wrapped for rendering purposes)

The HTTP GET method MUST return the binary representation of the OpenPGP key for the given mail address. The key needs to carry a User ID packet ([RFC4880]) with that mail address. Note that the key may be revoked or expired - it is up to the client to handle such conditions. The server MUST also accept a HEAD method so that a client may only check for the existence of a key.

The server SHOULD return "application/pgp-key" as the content-type for the data but clients MUST also accept "application/octet-string" as content-type. The server MUST NOT return an ASCII armored version of the key.

4. Web Key Directory Update Protocol

To put keys into the key directory a protocol to automate the task is desirable. The protocol defined here is entirely based on mail and the assumption that a mail provider can securely deliver mail to the INBOX of a user (e.g. an IMAP folder). Note that the same protocol may also be used for submitting keys for use with OpenPGP DANE.

We assume that the user already created a key for her mail account alice@example.org. To install the key at her provider's Web Key Directory, she performs the following steps:

1. She retrieves a file which contains one line with the mail address used to submit the key to the mail provider. See below for the syntax of that file. For a mail address at the domain "example.org" the URI of the file is

   ```
   https://example.org/.well-known/openpgpkey/submission-address
   ```

2. She sends her key using SMTP (or any other transport mechanism) to the provider using the submission address. The content-type

SHOULD be "application/pgp-key" and the key being a binary attachment (which is then likely base64 encoded). Note that the OpenPGP ASCII armor is not used.

3. The provider checks that the received key has a User ID which

   * matches an account name of the provider,

   * and that the from address matches that account.

4. The provider sends an encrypted message containing a nonce and the fingerprint of the key to the mail account of the user. Note that a similar scheme is used by the well known caff(1) tool to help with key signing parties.

5. A legitimate user will be able to decrypt the message because she created the key and is in charge of the private key. This step verifies that the submitted key has actually been created by the owner of the account.

6. The user sends the decrypted nonce back to the submission address as a confirmation that the private key is owned by her and that the provider may now publish the key. Also technically not required, it is suggested that the mail to the provider is encrypted. The public key for this is retrieved using the key lookup protocol described above.

7. The provider receives the nonce, matches it with its database of pending confirmations and then publishes the key. Finally the provider sends a mail back to the user to notify her of the the publication of her key.

The message data structures used for the above protocol are specified in detail below. In the following sections the string "WELLKNOWN" denotes the first part of an URI specific for a domain. In the examples the domain "example.org" is assumed, thus

```
WELLKNOWN := https://example.org/.well-known/openpgpkey
```

The term "target key" denotes the to be published key, the term "submission key" the key associated with the submission-address of the mail provider.

4.1. The Submission Address

The address of the submission file is

```
WELLKNOWN/submission-address
```

The file consists of exactly one line, terminated by a LF, or the sequence of CR and LF, with the full mail address to be used for submission of a key to the mail provider. For example the content of the file may be

```
  key-submission-example.org@directory.example.org
```

4.2. The Submission Mail

The mail used to submit a key to the mail provider MUST comply to the PGP/MIME specification ([RFC3156], section 7), which states that the Content-Type must be "application/pgp-keys", there are no required or optional parameters, and the body part contains the ASCII-armored transferable Public Key Packets as defined in [RFC4880], section 11.1.

If the mail provider has published an encryption key for the submission-address in the Web Key Directory, the key to be published MUST be submitted using a PGP/MIME encrypted message ([RFC3156], section 4). The message MUST not be signed (because the authenticity of the signing key has not yet been confirmed). After decryption of the message at the mail provider a single "application/pgp-keys" part, as specified above, is expected.

4.3. The Confirmation Request

The mail provider sends a confirmation mail in response to a received key publication request. The message SHOULD be sent from the submission-address of the mail provider to the mail address extracted from the target key. The message needs to be encrypted to the target key and MAY be signed by the submission key. PGP/MIME MUST be used for encryption and signing; the Combined method ([RFC3156], section 6.2) MUST be used if the message is to be signed.

The Content-type used for the plaintext part MUST be "application/vnd.gnupg.wkd". The body consists of name-value pairs with one name-value pair per LF or CR+LF terminated line. Empty lines are allowed and will be ignored by the receiver. A colon is used to terminate a name.

In a confirmation request the following names MUST be send in the specified order:

"type" The value must be "confirmation-request".

"from" This is the mailbox the user is expected to sent the confirmation response to. The value must match the mailbox part of the "From:" address of this request.

"address" The value is the addr-spec part of the target key's mail address. The value SHOULD match the addr-spec part of the recipient's address. The value MUST be be UTF-8 encoded as required for an OpenPGP User ID.

"fingerprint" The value is the fingerprint of the target key. The fingerprint is given in uppercase hex encoding without any interleaving spaces.

"nonce" The value is a string with a minimum length of 16 octets and a maximum length of 64 octets. The string must entirely be made up of random ASCII letters or digits. This nonce will be sent back to the mail provider as proof that the recipient is the legitimate owner of the target-key.

The receiver of the message decrypts the message, checks that the "fingerprint" matches the target key, checks that the "address" matches a User ID of the target key, and checks the other constrains of the request format. If any constraint is not asserted, or the fingerprint or User ID do not match the target key, or there is no pending publication requests (i.e. a mail recently sent o the submission address), the user MAY be notified about this fake confirmation attempt.

In other cases the confirmation request is legitimate and the MUA shall silently send a response as described in the next section.

4.4. The Confirmation Response

A response to a confirmation request MUST only be send in the positive case; there is no negative confirmation response. A mail service provider is expected to cancel a pending key submission after a suitable time without a confirmation. The mail service provider SHOULD not retry the sending of a confirmation request after the first request has been send successfully.

The user MUST send the confirmation response from her target mail address to the "from" address of the confirmation request. The message MUST be signed and SHOULD be encrypted. The PGP/MIME Combined format MUST be used for encryption and signing ([RFC3156], section 6.2). The encryption key can be taken from the Web Key Directory.

The Content-type used for the plaintext message MUST also be "application/vnd.gnupg.wkd". The format is the same as described above for the Confirmation Request. The body must contain three name-value pairs in this order:

"type" The value must be "confirmation-response".

"from" The value must match the mailbox part of the "From:" address of this response.

"nonce" The value is the value of the "nonce" parameter from the confirmation request.

4.5. Policy Flags

For key generation and submission it is sometimes useful to tell the client about certain properties of the mail provider in advance. This can be done with a file at the URL

```
  WELLKNOWN/policy
```

The file contains keywords, one per line with each line terminated by a LF or the sequence of CR and LF. Empty lines and lines starting with a '#' character are considered comment lines. A keyword is made up of lowercase letters, digits, hyphens, or dots. An underscore is allowed as a name space delimiters; see below. The first character must be a letter. Clients MUST use case-insensitive matching.

Currently defined keywords are:

"mailbox-only" The mail server provider does only accept keys with only a mailbox in the User ID. In particular User IDs with a real name in addition to the mailbox will be rejected as invalid.

More keywords will be defined in updates to this I-D. There is no registry yet except for this document. For experimental use of new features or for provider specific settings, keywords MUST be prefixed with a domain name and an underscore.

5. Security Considerations

The use of SHA-1 for the mapping of the "local-part" to a fixed string is not a security feature but merely used to map the local-part to a fixed-sized string made from a well defined set of characters. It is not intended to conceal information about a mail address.

The domain name part of the mail address is not part of the hash to avoid problems with internationalized domain names. Instead a separate web service is required for each domain name.

6. IANA Considerations

6.1. Well-Known URI

IANA is requested to assign a well-known URI in the "Well-Known URIs" registry as defined by [RFC5785]:

URI suffix: openpgpkey

Change controller: IETF

Specification document: This

7. Normative References

[RFC0822] Crocker, D., "STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES", STD 11, RFC 822, DOI 10.17487/RFC0822, August 1982, <http://www.rfc-editor.org/info/rfc822>.

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

[RFC3156] Elkins, M., Del Torto, D., Levien, R., and T. Roessler, "MIME Security with OpenPGP", RFC 3156, August 2001.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, November 2007.

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

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <http://www.rfc-editor.org/info/rfc5785>.

[RFC6189] Zimmermann, P., Johnston, A., Ed., and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP", RFC 6189, DOI 10.17487/RFC6189, April 2011, <http://www.rfc-editor.org/info/rfc6189>.

Appendix A. Test Vectors

For help implementing this specification a non-normative example is given:

A.1. Sample key

TODO

A.2. Software Notes

GnuPG supports the key discovery described in this document since version 2.1.12. To use it, the new method "wkd" needs to be used with the --auto-key-locate option.

Appendix B. Changes

- This is the initial draft.

B.1. TODO

- What about authenticated submission?

- Describe how to handle a key with several User IDs.

Author's Address

Werner Koch
GnuPG Project

Email: wk@gnupg.org
URI: https://gnupg.org