Network Working Group P. Wouters
Internet-Draft Red Hat
Intended status: Standards Track April 01, 2015
Expires: October 03, 2015

Using DANE to Associate OpenPGP public keys with email addresses
draft-ietf-dane-openpgpkey-03

Abstract

OpenPGP is a message format for email (and file) encryption that lacks a standardized lookup mechanism to securely obtain OpenPGP public keys. This document specifies a method for publishing, locating and verifying OpenPGP public keys in DNS for a specific email address using a new OPENPGPKEY DNS Resource Record. Security is provided via DNSSEC.

Table of Contents

1. Introduction

OpenPGP [RFC4880] public keys are used to encrypt or sign email messages and files. To encrypt an email message, the sender's email client or MTA needs to know where to find the recipient's OpenPGP public key. Once obtained, it needs to find some proof that the OpenPGP public key found actually belongs to the intended recipient.

Similarly, when files on a storage media are signed with an OpenPGP public key that is included on the storage media, this key needs to be independently verified.

Obtaining and verifying an OpenPGP public key is not a straightforward process as there are no trusted standardized locations for publishing OpenPGP public keys indexed by email address. Instead, OpenPGP clients rely on "well-known key servers" that are accessed using the HTTP Keyserver Protocol ("HKP") or manually by users using a variety of differently formatted front-end web pages. Worse, some OpenPGP users announce their OpenPGP public key fingerprint on social media with no method of validation whatsoever.

Currently deployed key servers have no method of validating any uploaded OpenPGP public key. The key servers simply store and publish. Anyone can add public keys with any identities and anyone can add signatures to any other public key using forged malicious identities. Furthermore, once uploaded, public keys cannot be deleted. People who did not pre-sign a key revocation can never remove their public key from these key servers once they lose their private key.

The lack of a secure means to look up a public key for an email address prevents email clients and MUAs from encrypting an outgoing email to the target recipient, forcing the software to send the message unencrypted. Currently deployed MTAs only support encrypting the transport of the email, not the email contents itself, leaving the content of the email exposed to anyone with access to any of the mail or storage servers used to transport the email from the sender to the receiver.

This document describes a mechanism to associate a user's OpenPGP public key with their email address, using a new DNS RRtype.

The proposed new DNS Resource Record type is secured using DNSSEC. This trust model is not meant to replace the Trust Signature model. However, it can be used to encrypt a message that would otherwise have to be sent out unencrypted, where it could be monitored by a third party in transit or located in plaintext on a storage or email server. This method can also be used to obtain the OpenPGP public key which can then be used for manual verification.

1.1. Terminology

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].

This document also makes use of standard DNSSEC and DANE terminology. See DNSSEC [RFC4033], [RFC4034], [RFC4035], and DANE [RFC6698] for these terms.

2. The OPENPGPKEY Resource Record

The OPENPGPKEY DNS resource record (RR) is used to associate an end entity OpenPGP public key with an email address, thus forming a "OpenPGP public key association".

The type value allocated for the OPENPGPKEY RR type is 61. The OPENPGPKEY RR is class independent. The OPENPGPKEY RR has no special TTL requirements.

2.1. The OPENPGPKEY RDATA component

The RDATA portion of an OPENPGPKEY Resource Record contains a single value consisting of a [RFC4880] formatted OpenPGP public keyring.

2.2. The OPENPGPKEY RDATA wire format

The RDATA Wire Format consists of a single OpenPGP public key as defined in Section 5.5.1.1 of [RFC4880]. Note that this format is without ASCII armor or base64 encoding.

2.3. The OPENPGPKEY RDATA presentation format

The RDATA Presentation Format, as visible in textual zone files, consists of a single OpenPGP public key as defined in Section 5.5.1.1. of [RFC4880] encoded in base64 as defined in Section 4 of [RFC4648].

3. Location of the OPENPGPKEY record

The DNS does not allow the use of all characters that are supported in the "local-part" of email addresses as defined in [RFC2822] and [RFC6530]. Therefore, email addresses are mapped into DNS using the following method:

o The user name (the "left-hand side" of the email address, called the "local-part" in the mail message format definition [RFC2822] and the "local part" in the specification for internationalized

email [RFC6530]) should already be encoded in UTF-8 (or its subset ASCII). If it is written in another encoding it should be converted to UTF-8. Next, it is turned into lowercase and hashed using the SHA2-256 [RFC5754] algorithm, with the hash truncated to 28 octets and represented in its hexadecimal representation, to become the left-most label in the prepared domain name. Truncation comes from the right-most octets. This does not include the at symbol ("@") that separates the left and right sides of the email address.

- The string "_openpgpkey" becomes the second left-most label in the prepared domain name.

- The domain name (the "right-hand side" of the email address, called the "domain" in RFC 2822) is appended to the result of step 2 to complete the prepared domain name.

For example, to request an OPENPGPKEY resource record for a user whose email address is "hugh@example.com", an OPENPGPKEY query would be placed for the following QNAME: "c93f1e400f26708f98cb19d936620da35 eec8f72e57f9eec01c1afd6._openpgpkey.example.com". The corresponding RR in the example.com zone might look like (key shortened for formatting):

```
c9[..]d6._openpgpkey.example.com. IN OPENPGPKEY <base64 public key>
```

3.1. Email address variants

Mail systems usually handle variant forms of local-parts. The most common variants are upper and lower case, which are now invariably treated as equivalent. But many other variants are possible. Some systems allow and ignore "noise" characters such as dots, so local parts johnsmith and John.Smith would be equivalent. Many systems allow "extensions" such as john-ext or mary+ext where john or mary is treated as the effective local-part, and the ext is passed to the recipient for further handling. This can complicate finding the OPENPGPKEY record associated with the dynamically created email address.

[RFC5321] and its predecessors have always made it clear that only the recipient MTA is allowed to interpret the local-part of an address. A client supporting OPENPGPKEY therefor MUST NOT perform any kind of mapping rules based on the email address. As the local-part is converted to lowercase before hashing, case sensitivity will not cause problems for the OPENPGPKEY lookup.

4. Application use of OPENPGPKEY

The OPENPGPKEY record allows an application or service to obtain or verify an OpenPGP public key. The lookup result MUST pass DNSSEC validation; if validation reaches any state other than "Secure", the verification MUST be treated as a failure.

4.1. Obtaining an OpenPGP key for a specific email address

If no OpenPGP public keys are known for an email address, an OPENPGPKEY lookup can be performed to discover the OpenPGP public key that belongs to a specific email address. This public key can then be used to verify a received signed message or can be used to send out an encrypted email message.

4.2. Confirming the validity of an OpenPGP key

Locally stored OpenPGP public keys are not automatically refreshed. If the owner of that key creates a new OpenPGP public key, that owner is unable to securely notify all users and applications that have its old OpenPGP public key. Applications and users can perform an OPENPGPKEY lookup to confirm the locally stored OpenPGP public key is still the correct key to use. If verifying a locally stored OpenPGP public key and the OpenPGP public key found through DNS is different from the locally stored OpenPGP public key, the verification MUST be treated as a failure. An application that can interact with the user MAY ask the user for guidance.

4.3. Verifying an unknown OpenPGP signature

Storage media can be signed using an OpenPGP public key. Even if the OpenPGP public key is included on the storage media, it needs to be independently validated. OpenPGP public keys contain one or more IDs than can have the syntax of an email address. An application can perform a lookup for an OPENPGPKEY at the expected location for the specific email address to confirm the validity of the OpenPGP public key. Once the key has been validated, all files on the storage media that have been signed by this key can now be verified.

5. OpenPGP Key size and DNS

Due to the expected size of the OPENPGPKEY record, it is recommended to perform DNS queries for the OPENPGPKEY record using TCP, not UDP.

Although the reliability of the transport of large DNS Resource Records has improved in the last years, it is still recommended to keep the DNS records as small as possible without sacrificing the security properties of the public key. The algorithm type and key

size of OpenPGP keys should not be modified to accommodate this section.

OpenPGP supports various attributes that do not contribute to the security of a key, such as an embedded image file. It is recommended that these properties are not exported to OpenPGP public keyrings that are used to create OPENPGPKEY Resource Records. Some OpenPGP software, for example GnuPG, have support for a "minimal key export" that is well suited to use as OPENPGPKEY RDATA. See Appendix A.

6. Security Considerations

OPENPGPKEY usage considerations are published in [OPENPGPKEY-USAGE].

6.1. Response size

To prevent amplification attacks, an Authoritative DNS server MAY wish to prevent returning OPENPGPKEY records over UDP unless the source IP address has been verified with [DNS-COOKIES]. Such servers MUST NOT return REFUSED, but answer the query with an empty Answer Section and the truncation flag set ("TC=1").

6.2. Email address information leak

Email addresses are not secret. Using them causes their publication. The hashing of the user name in this document is not a security feature. Publishing OPENPGPKEY records however, will create a list of hashes of valid email addresses, which could simplify obtaining a list of valid email addresses for a particular domain. It is desirable to not ease the harvesting of email addresses where possible.

The domain name part of the email address is not used as part of the hash so that hashes can be used in multiple zones deployed using DNAME [RFC6672]. This does makes it slightly easier and cheaper to brute-force the SHA2-224 hashes into common and short user names, as single rainbow tables can be re-used across domains. This can be somewhat countered by using NSEC3.

DNS zones that are signed with DNSSEC using NSEC for denial of existence are susceptible to zone-walking, a mechanism that allows someone to enumerate all the OPENPGPKEY hashes in a zone. This can be used in combination with previously hashed common or short user names (in rainbow tables) to deduce valid email addresses. DNSSEC-signed zones using NSEC3 for denial of existence instead of NSEC are significantly harder to brute-force after performing a zone-walk.

6.3. Storage of OPENPGPKEY data

Users may have a local key store with OpenPGP public keys. An application supporting the use of OPENPGPKEY DNS records MUST NOT modify the local key store without explicit confirmation of the user, as the application is unaware of the user's personal policy for adding, removing or updating their local key store. An application MAY warn the user if an OPENPGPKEY record does not match the OpenPGP public key in the local key store.

OpenPGP public keys obtained via OPENPGPKEY records should not be stored beyond their DNS TTL value.

6.4. Forward security of OpenPGP versus DNSSEC

DNSSEC key sizes are chosen based on the fact that these keys can be rolled with next to no requirement for security in the future. If one doubts the strength or security of the DNSSEC key for whatever reason, one simply rolls to a new DNSSEC key with a stronger algorithm or larger key size. On the other hand, OpenPGP key sizes are chosen based on how many years (or decades) their encryption should remain unbreakable by adversaries that own large scale computational resources.

This effectively means that anyone who can obtain a DNSSEC private key of a domain name via coercion, theft or brute force calculations, can replace any OPENPGPKEY record in that zone and all of the delegated child zones, irrespective of the key size of the OpenPGP keypair. Any future messages encrypted with the malicious OpenPGP key could then be read.

Therefore, an OpenPGP key obtained via an OPENPGPKEY record can only be trusted as much as the DNS domain can be trusted, and is no substitute for in-person key verification of the "Web of Trust". See [OPENPGPKEY-USAGE] for more in-depth information on safe usage of OPENPGPKEY based OpenPGP keys.

7. IANA Considerations

7.1. OPENPGPKEY RRtype

This document uses a new DNS RR type, OPENPGPKEY, whose value 61 has been allocated by IANA from the Resource Record (RR) TYPEs subregistry of the Domain Name System (DNS) Parameters registry.

8. Acknowledgments

This document is based on RFC-4255 and draft-ietf-dane-smime whose authors are Paul Hoffman, Jacob Schlyter and W. Griffin. Olafur Gudmundsson provided feedback and suggested various improvements. Willem Toorop contributed the gpg and hexdump command options. Edwin Taylor contributed language improvements for various iterations of this document. Text regarding email mappings was taken from draft-levine-dns-mailbox whose author is John Levine.

9. References

9.1. Normative References

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

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

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

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

[RFC5754] Turner, S., "Using SHA2 Algorithms with Cryptographic Message Syntax", RFC 5754, January 2010.

9.2. Informative References

[DNS-COOKIES]
Eastlake, Donald., "Domain Name System (DNS) Cookies", draft-ietf-dnsop-cookies (work in progress), February 2015.

[OPENPGPKEY-USAGE]
Wouters, P., "Usage considerations with the DNS OPENPGPKEY record", draft-dane-openpgpkey-usage (work in progress), October 2014.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, July 1997.

[RFC2822] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

[RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, September 2003.

[RFC5233] Murchison, K., "Sieve Email Filtering: Subaddress Extension", RFC 5233, January 2008.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, October 2008.

[RFC6530] Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 6530, February 2012.

[RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, June 2012.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, August 2012.

[RFC7129] Gieben, R. and W. Mekking, "Authenticated Denial of Existence in the DNS", RFC 7129, February 2014.

Appendix A. Generating OPENPGPKEY records

The commonly available GnuPG software can be used to generate the RRdata portion of an OPENPGPKEY record:

```
gpg --export --export-options export-minimal \
    hugh@example.com | base64
```

The --armor or -a option of the gpg command should NOT be used, as it adds additional markers around the armored key.

When DNS software reading or signing the zone file does not yet support the OPENPGPKEY RRtype, the Generic Record Syntax of [RFC3597] can be used to generate the RDATA. One needs to calculate the number of octets and the actual data in hexadecimal:

```
gpg --export --export-options export-minimal \
    hugh@example.com | wc -c

gpg --export --export-options export-minimal \
    hugh@example.com | hexdump -e \
       '"\t" /1 "%.2x"' -e '/32 "\n"'
```

These values can then be used to generate a generic record (line break has been added for formatting):

```
<SHA2-256-trunc(hugh)>._openpgpkey.example.com. IN TYPE61 \# \
    <numOctets> <keydata in hex>
```

The openpgpkey command in the hash-slinger software can be used to generate complete OPENPGPKEY records

```
~> openpgpkey --output rfc hugh@example.com
c9[..]d6._openpgpkey.example.com. IN OPENPGPKEY mQCNAzIG[...]

~> openpgpkey --output generic hugh@example.com
c9[..]d6._openpgpkey.example.com. IN TYPE61 \# 2313 99008d03[...]
```

Author's Address

Paul Wouters
Red Hat

Email: pwouters@redhat.com