

openpgp
Internet-Draft
Intended status: Informational
Expires: November 23, 2019

D. Gillmor
ACLU
May 22, 2019

Common PGP/MIME Message Mangling
draft-dkg-openpgp-pgpmime-message-mangling-00

Abstract

An e-mail message with OpenPGP encryption and/or signature has standard form. However, some mail transport agents damage those forms in transit. A user-friendly mail user agent that encounters such a mangled message may wish to try to process it anyway, despite the message's non-standard structure.

This document aims to be a sort of bestiary of common message mangling patterns, along with guidance for implementers for how to cope with messages structured in these common ways.

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 November 23, 2019.

Copyright Notice

Copyright (c) 2019 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 (<https://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

Internet-Draft

Common PGP/MIME Message Mangling

May 2019

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.

Table of Contents

| | | |
|------------------------|--|--------------------|
| 1. | Introduction | 3 |
| 1.1. | Requirements Language | 3 |
| 1.2. | Terminology | 3 |
| 2. | Problem Statement | 3 |
| 3. | Cryptographic Properties | 4 |
| 4. | Encrypted Messages | 4 |
| 4.1. | "Mixed Up" Encryption | 4 |
| 4.1.1. | Detection of "Mixed Up" Encryption | 5 |
| 4.1.2. | Repair of "Mixed Up" Encryption | 5 |
| 5. | Clearsigned Messages | 5 |
| 5.1. | Extra MIME footer | 6 |
| 5.2. | Content-Re-Encoding | 6 |
| 6. | Performance Considerations | 6 |
| 7. | Security Considerations | 6 |
| 7.1. | Only Repair for Cryptographic Improvement | 6 |
| 7.2. | Avoiding Another E-Fail | 6 |
| 7.3. | Do Not Attempt Repair Inside End-to-End Encryption | 7 |
| 7.4. | Interactions with DKIM Verification | 7 |
| 7.5. | "Helpful" MTA mangling | 7 |
| 8. | Privacy Considerations | 7 |
| 9. | User Considerations | 8 |
| 9.1. | Indications of Mangling Can Be Confusing | 8 |
| 9.2. | Repair Indicators | 8 |
| 9.3. | Undoing Repair | 8 |
| 9.4. | Attempting Decryption During Repair | 8 |
| 10. | IANA Considerations | 8 |
| 11. | Document Considerations | 8 |
| 12. | Example Messages | 9 |
| 12.1. | Example Input Encrypted Message | 9 |
| 12.2. | Input Message Mangled by "Mixed Up" | 9 |
| 13. | Example Implementations | 10 |
| 13.1. | Example: Detecting "Mixed Up" Encryption | 10 |
| 13.2. | Example: Repairing "Mixed Up" Encryption | 11 |
| 14. | References | 11 |
| 14.1. | Normative References | 11 |
| 14.2. | Informative References | 12 |

[1.](#) Introduction

[1.1.](#) Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BCP 14](#) [[RFC2119](#)] [[RFC8174](#)] when, and only when, they appear in all capitals, as shown here.

[1.2.](#) Terminology

- o "Mail User Agent" (or "MUA") refers to a program designed to send and receive e-mail. Common examples include Thunderbird, Outlook, and Mail.app. Webmail implementations, like Roundcube or the Gmail interface can also be considered a MUA.
- o "Mail Transport Agent" (or "MTA") refers to mail-handling software within the network. Common examples include Postfix and Exchange.
- o "PGP/MIME" is the message structure for OpenPGP protections described in [[RFC3156](#)].
- o "encrypted message" is used to mean any message where the outermost PGP/MIME cryptographic feature of the message itself is encryption. Some encrypted messages might be signed, but the inner signature cannot typically be mangled by an "MTA" that can't decrypt.
- o "clearsigned message" is used here to refer specifically to messages that are not encrypted, but are signed with a PGP/MIME signature.
- o "signed message" is used to refer to a message that has a valid OpenPGP cryptographic signature applied by the message sender, whether the message is encrypted or not.

[2.](#) Problem Statement

Some MTAs modify message headers, bodies, and structure in transit. Some of those modifications ("manglings") can transform a message in such a way that the end-to-end cryptographic properties of the message are lost.

Cleartext messages can become unverifiable, and encrypted messages can become impossible to decrypt. MUAs that receive these mangled messages are unable to fulfil their users goals because of damage done to the message by the mangling MTA.

Gillmor

Expires November 23, 2019

[Page 3]

Internet-Draft

Common PGP/MIME Message Mangling

May 2019

In some cases, those manglings can take a regular enough form that a clever MUA can "repair" them, restoring the cryptographic properties of the message for the user.

This document aims to collect common manglings, and document how an MUA can detect them and (if possible) repair them safely for the user.

[3.](#) Cryptographic Properties

TODO: describe message-wide cryptographic envelope vs. cryptographic payload and their relevance to this draft.

TODO: we're assuming a single layer of cryptographic protection. These steps get more complicated if there are multiple nested layers (encrypted and then signed? signed and then encrypted? triple-wrapped?). Should this document address that situation?

[4.](#) Encrypted Messages

This section aims to collect examples of repairable mangling known to be performed by some currently-active MTA on encrypted messages. Each section offers a brief description of the mangling, a way to detect it, and a proposed method of repair.

[4.1.](#) "Mixed Up" Encryption

One common mangling takes an incoming multipart/encrypted e-mail and transforms it into a multipart/mixed e-mail, shuffling body parts at

the same time.

An encrypted message typically has this clean structure C (see example in [Section 12.1](#)):

```
C    multipart/encrypted
C.1  application/pgp-encrypted
C.2  application/octet-stream
```

But after processing, it has mangled structure M (see example in [Section 12.2](#)):

```
M    multipart/mixed
M.1  text/plain (0 bytes)
M.2  application/pgp-encrypted
M.3  application/octet-stream
```

Some versions of Microsoft Exchange are known to do this.

[4.1.1](#). Detection of "Mixed Up" Encryption

Check that all of the following conditions hold:

- o The message itself (M) is Content-Type: multipart/mixed.
- o The message has exactly three MIME subparts, all direct children of the message itself.
- o The first part (M.1) is Content-Type: text/plain, and has 0 bytes.
- o The second part (M.2) is Content-Type: application/pgp-encrypted, and contains (after removing any Content-Transfer-Encoding) exactly the string "Version: 1"
- o The third part (M.3) is Content-Type: application/octet-stream.
- o After removing any Content-Transfer-Encoding, the decoded third part (M.3) is an ASCII-armored OpenPGP block of type "PGP MESSAGE" (see see [section 6.2 of \[RFC4880\]](#))

Please see [Section 13.1](#) for an example implementation.

[4.1.2.](#) Repair of "Mixed Up" Encryption

If the detection is successful, repair can be attempted with the following steps:

- o Convert the message's Content-Type: value to multipart/encrypted, while leaving any other parameters untouched.
- o Add (or reset) the parameter protocol=application/pgp-encrypted to the Content-Type: header.
- o Drop the first sub-part (M.1).

Please see [Section 13.2](#) for an example implementation.

[5.](#) Clearsigned Messages

This section aims to collect examples of repairable mangling known to be performed by some currently-active MTA on clearsigned messages. Each section offers a brief description of the mangling, a way to detect it, and a proposed method of repair.

[5.1.](#) Extra MIME footer

TODO: describe the mailman case

[5.2.](#) Content-Re-Encoding

TODO: despite the content of multipart/signed parts being ostensibly invariant (see page 4 of [\[RFC1847\]](#)), some MTAs mangle them. Describe some reversible manglings, like gratuitous application of Content-Transfer-Encoding.

[6.](#) Performance Considerations

Trying multiple repair techniques can be expensive. A resource-constrained MUA may want to limit itself to detection of possible

mangling, rather than trying every possible repair it knows of and indicating

[7.](#) Security Considerations

There are several ways that attempting repair on a mangled message can go dangerously wrong.

[7.1.](#) Only Repair for Cryptographic Improvement

If a message repair produces a seemingly-better-structured message, but the repaired message still cannot be decrypted or verified, then the MUA should abandon the repair and instead render the message in its original form. Additional message modification that does not provide a cryptographic advantage over the received form of the message offers no benefit to the user.

TODO: is a decryptable, unsigned message "better" than a verifiably signed message? What is "an improvement" isn't necessarily clear here.

[7.2.](#) Avoiding Another E-Fail

Promiscuous message repair may inject new vulnerabilities, or modify a message beyond recognition. An MUA attempting message repair should use caution to avoid re-combining potentially malicious material with material that has better cryptographic guarantees.

In particular, a responsible MUA should take care not to claim cryptographic protections over material that does not have them.

[7.3.](#) Do Not Attempt Repair Inside End-to-End Encryption

If a message or part of a message arrived end-to-end encrypted, none of the recommendations in this draft should be read as encouraging their application to any cleartext material within that end-to-end encryption. These techniques are specifically for recovering from damage done by the MTA, which by design does not have access to the cleartext of an end-to-end encrypted message.

[7.4.](#) Interactions with DKIM Verification

A mangling MTA may also add DKIM headers after mangling. A subsequent MTA that handles the message may verify the message by checking DKIM headers, storing the results of that check in an Authentication-Results header (see [[RFC7601](#)]). If the MUA performs message repair, the repaired message might not pass DKIM check. An MUA that relies on Authentication-Results headers for any purpose on such a repaired message should consider re-validating the DKIM header itself directly after message repair (though this introduces new problems for stored/old messages, as older DKIM signing keys may no longer be published in the DNS).

Note that a repaired message may have cryptographic properties that obviates the need for DKIM verification; an MUA that requires messages to have a valid Authentication-Results: header might waive that requirement for a signed message (whether any repair routine was invoked or not).

[7.5.](#) "Helpful" MTA mangling

An MTA may intend to help the user by modifying the message. For example, in a clearsigned message, the MTA may modify a potentially-malicious URL in the text, or it might change an attachment name or MIME-type in an attachment. If an MUA attempts message repair in a way that reverses these modifications, it may negate whatever security advantages the MTA hoped to offer.

It is not clear that this is a risk for repair of an encrypted message, though, since the MTA by design cannot perform such a modification on the cleartext of an encrypted message.

[8.](#) Privacy Considerations

TODD: privacy considerations?

[9.](#) User Considerations

Cryptographic properties of messages are only useful if the user understands them.

9.1. Indications of Mangling Can Be Confusing

- o TODO: about how indicators of mangled messages can cause user-experience confusion (e.g. parts of <https://efail.de/> and <https://github.com/RUB-NDS/Johnny-You-Are-Fired>)

9.2. Repair Indicators

- o TODO: how should a repairing MUA indicate that a repair has happened (or is possible) to the user?

9.3. Undoing Repair

- o TODO: Should a repairing MUA allow the user to reverse the repair and see the broken message? If so, what else should happen if the user takes this action (e.g., how is a subsequent "Reply" or "Forward" action affected?)

9.4. Attempting Decryption During Repair

Attempting decryption may require access to secret key material, which itself may cause interaction with the user. If multiple repairs are attempted, each attempt to decrypt may require multiple decryptions. In some scenarios, multiple use of the secret key may be annoying to the user. It might be preferable to try to decrypt the apparent underlying block of text exactly once, regardless of message structure or repair, and if successful, either stash the session key, or store the cleartext in memory, while manipulating exterior structure.

10. IANA Considerations

This document requires no actions from IANA.

11. Document Considerations

[RFC Editor: please remove this section before publication]

This document is currently edited as markdown. Minor editorial changes can be suggested via merge requests at <https://gitlab.com/dkg/draft-openpgp-pgpmime-message-mangling> or by e-mail to the author. Please direct all significant commentary to the public IETF OpenPGP mailing list: openpgp@ietf.org

[12.](#) Example Messages

[12.1.](#) Example Input Encrypted Message

```
From: Michael Elkins <elkins@aero.org>
To: Michael Elkins <elkins@aero.org>
Mime-Version: 1.0
Content-Type: multipart/encrypted; boundary=foo;
    protocol="application/pgp-encrypted"
```

```
--foo
Content-Type: application/pgp-encrypted
```

```
Version: 1
```

```
--foo
Content-Type: application/octet-stream
```

```
-----BEGIN PGP MESSAGE-----
```

```
hIwDY32hYGCE8MkBA/wOu7d45aUxF4Q0RKJprD3v5Z9K1YcRJ2fve87lMlDlx40j
eW4GDdBfLbJE7VUpp13N19GL8e/AqbyyjHH4aS0YoTk10QQ9nnRvjY8nZL3MPXSZ
g9VGQxFeGqzykzmykU6A26MSMexR4Apee0N6xzZWfo+0y0qAq6lb46wsvldZ96YA
AABH78hyX7YX4uT1tNCWEIIBoqqvCeIMpp7UQ2IzBrXg6GtukS8NxbukLeamqVW3
1yt21DY0juLzcMNe/JNsD9vDVCv00G30Ci8=
=zzaA
```

```
-----END PGP MESSAGE-----
```

```
--foo--
```

[12.2.](#) Input Message Mangled by "Mixed Up"

Internet-Draft

Common PGP/MIME Message Mangling

May 2019

```
From: Michael Elkins <elkins@aero.org>
To: Michael Elkins <elkins@aero.org>
Mime-Version: 1.0
Content-Type: multipart/mixed; boundary=foo
```

```
--foo
Content-Type: text/plain; charset="us-ascii"
```

```
--foo
Content-Type: application/pgp-encrypted
```

```
Version: 1
```

```
--foo
Content-Type: application/octet-stream
```

```
-----BEGIN PGP MESSAGE-----
```

```
hIwDY32hYGCE8MkBA/wOu7d45aUxF4Q0RKJprD3v5Z9K1YcRJ2fve87lMlDlx40j
eW4GDdBfLbJE7VUpp13N19GL8e/AqbyyjHH4aS0YoTk10QQ9nnRvjY8nZL3MPXSZ
g9VGQxFeGqzykzmykU6A26MSMexR4Apee0N6xzZWfo+0y0qAq6lb46wsvldZ96YA
AABH78hyX7YX4uT1tNCWEIIBoqqvCeIMpp7UQ2IzBrXg6GtukS8NxbukLeamqVW3
1yt21DY0juLzcMNe/JNsD9vDVCv00G30Ci8=
=zzaA
```

```
-----END PGP MESSAGE-----
```

```
--foo--
```

[13.](#) Example Implemenations

These examples use python3. As Code Examples, they are licensed under the Simplified BSD License as noted above.

[13.1.](#) Example: Detecting "Mixed Up" Encryption

Internet-Draft

Common PGP/MIME Message Mangling

May 2019

```
import email.message

def is_mixed_up(msg: email.message.Message) -> bool:
    if msg.get_content_type() == 'multipart/mixed':
        pts = msg.get_payload()
        if len(pts) == 3 and \
            pts[0].get_content_type() == 'text/plain' and \
            pts[0].get_payload(decode=True) == b'' and \
            pts[1].get_content_type() == 'application/pgp-encrypted' and \
            pts[1].get_payload(decode=True).strip() == b'Version: 1' and \
            pts[2].get_content_type() == 'application/octet-stream':
            encpart = pts[2].get_payload(decode=True)
            # FIXME: this is not a true check for RFC4880 ASCII armor:
            lines = encpart.replace(b'\r\n', b'\n').split(b'\n')
            if lines[0] == b'-----BEGIN PGP MESSAGE-----' and \
                lines[-1] == b'-----END PGP MESSAGE-----':
                return True
    return False
```

[13.2.](#) Example: Repairing "Mixed Up" Encryption

```
import copy
import email.message
import typing

def un_mix_up(msg: email.message.Message) -> \
    typing.Optional[email.message.Message]:
    if is_mixed_up(msg):
        ret = copy.deepcopy(msg)
        ret.set_type('multipart/encrypted')
        ret.set_param('protocol', 'application/pgp-encrypted')
        ret.set_payload(msg.get_payload()[1:])
```

return ret
return None

14. References

14.1. Normative References

- [RFC1847] Galvin, J., Murphy, S., Crocker, S., and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", [RFC 1847](#), DOI 10.17487/RFC1847, October 1995, <<https://www.rfc-editor.org/info/rfc1847>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

Gillmor

Expires November 23, 2019

[Page 11]

Internet-Draft

Common PGP/MIME Message Mangling

May 2019

- [RFC3156] Elkins, M., Del Torto, D., Levien, R., and T. Roessler, "MIME Security with OpenPGP", [RFC 3156](#), DOI 10.17487/RFC3156, August 2001, <<https://www.rfc-editor.org/info/rfc3156>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in [RFC 2119](#) Key Words", [BCP 14](#), [RFC 8174](#), DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

14.2. Informative References

- [RFC4880] Callas, J., Donnerhackle, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", [RFC 4880](#), DOI 10.17487/RFC4880, November 2007, <<https://www.rfc-editor.org/info/rfc4880>>.
- [RFC7601] Kucherawy, M., "Message Header Field for Indicating Message Authentication Status", [RFC 7601](#), DOI 10.17487/RFC7601, August 2015, <<https://www.rfc-editor.org/info/rfc7601>>.

Author's Address

Daniel Kahn Gillmor
American Civil Liberties Union

125 Broad St.
New York, NY 10004
USA

Email: dkg@fifthhorseman.net