```
MARID Working Group                                           J. Lyon
Internet Draft                                         Microsoft Corp
Document: draft-ietf-marid-core-01.txt                        M. Wong
                                                            pobox.com
Expires: December 2004                                      June 2004
```

MTA Authentication Records in DNS

Status of this Memo

Abstract

Internet mail suffers from the fact that much unwanted mail is sent using spoofed addresses û- "spoofed" in this case means the address is used without the permission of the domain owner. This document describes the following: mechanisms by which a domain owner can publish its set of outgoing MTAs, mechanisms by which SMTP servers can determine what email address is allegedly responsible for most proximately introducing a message into the Internet mail system, and whether that introduction is authorized by the owner of the domain contained in that email address.

The specification is carefully tailored to ensure that the overwhelming majority of legitimate emailers, remailers and mailing list operators are already compliant.

Table of Contents

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

<< Text contained in double angle brackets describes actions that are yet to be taken and decisions that are yet to be made. No such text should survive in the final version of this draft. >>

1. Introduction

Today, a huge majority of unwanted email contains headers that lie about the origin of the mail. This is true of most spam and substantially all of the virus email that is sent.

This document describes a mechanism such that receiving MTAs, MDAs and/or MUAs can recognize mail in the above category and take appropriate action. For example, an MTA might refuse to accept a message, an MDA might discard a message rather than placing it into a mailbox, and an MUA might render that message in some distinctive fashion.

In order to avoid further fragmentation of the Internet email system, it is necessary that the Internet community as a whole come to a consensus as to what mail senders should do to make their mail appear non-spoofed, and how mail receivers should determine whether mail is spoofed. On the other hand, it is not necessary to reach a consensus regarding the actions that various parties take once a message has been determined to be spoofed. This can be done unilaterally -- one agent might decide to discard a spoofed message while another decides to add a disclaimer.

2. Problem Statement

2.1 Positive Problem Statement

Briefly stated, the mechanisms of this document allow one to answer the following question:

When a message is transferred via SMTP between two UNRELATED parties, does the SMTP client host have permission to send mail on behalf of the mailbox that allegedly caused the most recent introduction of the message into the mail delivery system?

As seen from the question, this mechanism applies to unrelated parties: it is useful at the point where a message passes across the

Internet from one organization to another. It is beyond the scope of this document to describe authentication mechanisms that can be deployed within an organization.

The mechanism of this document also seeks to authenticate the mailbox associated with the MOST RECENT introduction of a message into the mail delivery system. In simple cases, this is who the mail is from. However, in the case of a third-party mailer, a forwarder or a mailing list server, the address being authenticated is that of the third party, the forwarder or the mailing list.

This document provides means to authenticate the DOMAIN of the appropriate email address; it makes no attempt to authenticate the local-part. A domain owner gets to determine which SMTP clients speak on behalf of addresses within the domain; a responsible domain owner should not authorize SMTP clients that will lie about local parts.

In the long run, once the domain of the sender is authenticated, it will be possible to use that domain as part of a mechanism to determine the likelihood that a given message is spam, using, for example, reputation and accreditation services. (These services are not the subject of the present mechanism, but it should enable them.)

2.2 Negative Problem Statement

This section describes alternate questions, which this document makes no attempt to solve.

Is the host at a particular IP address authorized to act as an SMTP client? [MTAMARK] attempts to answer this question using a reverse DNS lookup. It suffers from the fact that reverse DNS lookups are poorly implemented in major fractions of the IP address space, from the fact that many ISPs refuse to reverse-delegate to small domains, and from the fact that it's all or nothing: authority to act as an SMTP client conveys authority to send absolutely any email, legitimate or otherwise.

Is an SMTP client authorized to use a particular domain name in its SMTP EHLO command? [CSV] attempts to answer this question. It suffers from the fact that the EHLO name has a tenuous relationship, at best, with the contents of any mail message.

Is an SMTP client authorized to use a particular email address in an SMTP "MAIL FROM:" command? [RMX] and [SPF] attempt to answer this question. These suffer from the fact that the MAIL FROM address really describes where to send NDRs to, and in many third-party and

forwarder situations this address is unrelated to the domain that is resending the message.

Was a message really authored by who it claims to be authored by? [SMIME] and [DK] attempt to answer the question of original authorship. While this is a useful question to answer, these approaches suffer from severe deployment issues.

3. Decision Model

The essence of this specification is:

Given an email message, and given an IP address from which it has been (or will be) received, is the SMTP client at that host address authorized to send that email message?

There are four steps to answering this question:

(1) From the headers of the email message, extract the "purported responsible address". This is the mailbox that the message claims is responsible for the most recent introduction of the message into the delivery system. This step is described in detail in section 4 below. A separate specification, [Submitter], describes an SMTP extension that allows an SMTP server to perform this check at the time of the SMTP MAIL command instead of the SMTP DATA command.

(2) Extract the domain part of the purported responsible address. Call this the "purported responsible domain".

(3) Find the E-mail Policy Document for the purported responsible domain. Section 5.1 below describes the semantics of an E-mail Policy Document. Section 5.2 below describes how to find the document for a domain. The E-mail Policy Document contains a description of a "client authorization function". This function that takes four arguments and returns a seven-valued result. The arguments are:
  a. The local-part of an email address.
  b. A domain name, called the "original domain".
  c. A domain name, called the "current domain".
  d. An IP address (either IPv4 or IPv6).
The result of the function is one of the seven values "pass", "fail", "softFail", "neutral", "transientError", "hardError" or "none".

(4) Execute the function, passing:
  the local-part of the Purported Responsible Address,
  the Purported Responsible Domain (as "original domain"),

the Purported Responsible Domain (as "current domain"), and the IP address from which the mail was received.

Section 6 describes how to interpret the result.

4. Determining the Purported Responsible Address

The purported responsible address (PRA) of a message is determined by the following algorithm:

1. Locate the first non-empty Resent-Sender header in the message. If no such header is found, proceed to step 2. If it is preceded by a non-empty Resent-From header and one or more Received or Return-Path headers occur after said Resent-From header and before the Resent-Sender header, proceed to step 2. If the Resent-Sender header is hopelessly malformed (e.g. it appears to contain multiple mailboxes, or if the single mailbox is hopelessly malformed) then exit, without returning a PRA. Otherwise exit, returning the mailbox from the Resent-Sender header as the PRA.

2. Locate the first non-empty Resent-From header in the message. If no such header is found, proceed to step 3. If it is hopelessly malformed (e.g. one or more mailboxes in the header are hopelessly malformed) then exit without returning a PRA. If it contains multiple mailboxes, then exit without returning a PRA. Otherwise exit, returning the single mailbox from the Resent-From header as the PRA.

3. Locate the first non-empty Delivered-To, X-Envelope-To or Envelope-To header in the message. If no such header is present, proceed to step 4. If it appears to contain multiple mailboxes, then exit without returning a PRA. Otherwise, treat the contents of the header as a mailbox, and exit returning this mailbox as the PRA (unless the mailbox is hopelessly malformed, in which case exit without returning a PRA).

   [Note. This means that a non-standard header that does *not* contain a single valid mailbox will cause the PRA algorithm to fail and may cause the message to be rejected. But anything else means that the process doing the MARID checks might make a different decision as to the validity of the mailbox from a subsequent MUA which attempts to display the purported responsible address by parsing the headers. Maybe it would be better to drop this step, and go back to only considering RFC(2)822 headers?]

4. Locate all the non-empty Sender headers in the message. If there are no such headers, continue to step 5. If there are multiple such headers, exit without returning a PRA. If the single non-empty Sender header is hopelessly malformed (e.g. if it appears to contain multiple mailboxes, or if the single mailbox is hopelessly malformed), exit without returning a PRA. Otherwise, exit returning the mailbox from the Sender header as the PRA.

5. Locate all the non-empty From headers in the message. If there are no such headers, or multiple such headers, exit without returning a PRA. If the single non-empty From header is hopelessly malformed (e.g. it contains one or more mailboxes that are hopelessly malformed) then exit without returning a PRA. If it contains multiple mailboxes, exit without returning a PRA. Otherwise, return the single mailbox from the From header as the PRA.

The purported responsible domain of the message is the domain part of the purported responsible address returned by the above algorithm.

If the above algorithm fails to return a PRA, or if the PRA does not contain a domain, then an MTA SHOULD reject the message as hopelessly malformed.

What constitutes a hopelessly malformed header or a hopelessly malformed mailbox is a matter for local policy.

[Note: such local policy will never cause two implementation to return different PRAs. However it may cause one implementation to return a PRA where another implementation does not. The result is that corner cases may result in messages of questionable deliverability, but they will never result in an MTA doing MARID checks on a different PRA from the address that a PRA-aware MUA chooses to display, even if they make their decisions independently.]

5. E-mail Policy Document

An E-mail Policy Document is modeled by an XML infoset that contains, among other things, a definition of the function described in section 3 above. This function can be used to determine whether a domain owner is willing to take responsibility for e-mail that is sent by a particular SMTP client.

This document describes those parts of the XML infoset that define the mail acceptance function. The infoset may contain other information relating to e-mail; this other information may be the subject of future IETF consensus processes. Any program that

interprets the XML Infoset SHOULD ignore any elements whose tags are not understood by the program.

Section 5.1 below contains a detailed definition of the XML infoset. Section 5.2 contains a description of the macro expansion performed on the character data in some of the elements. Section 5.3 below contains the algorithm by which the XML infoset may be obtained.

5.1 Elements of the Infoset

The root of the infoset is always a "root" element.

5.1.1 "root" element

Attributes: none
Child elements: A single "ep" element.

The "root" element carries no semantic information -- it's an artifact of the manner in which the infoset is constructed.

5.1.2 "ep" element

Attributes: "testing" (Boolean)
Child elements: A single optional "out" element.

The "ep" element is intended to describe a domain's e-mail policy. At the present writing, this is limited to describing the client authorization function, but other information may be added in the future.

If the "testing" attribute is "true", then programs SHOULD behave as if the e-mail policy is absent, except by prior arrangement with the owner of the domain.

If the "out" element is present, it describes the client authorization function. If "out" is absent, the client authorization function always returns "unknown".

5.1.3 "out" element

```
Attributes: "default" (one of "pass", "fail", "softFail", "unknown")
            "redirect" (a domain name with macro expansion)
Child elements: A sequence of zero or more "m" elements.
```

The "out" element describes the client authorization function. To evaluate this function, evaluate each of the "m" elements in sequence as described below, until one of them returns a definite result. That result is the result of the client authorization function.

If no child "m" element returns a definite result, then:

1. If the "redirect" attribute is present, extract the domain name from it with macro expansion, obtain that domain's e-mail policy document, and return the result of evaluating that document's client authorization function, passing:
    the local-part that was passed to this function,
    the original domain name that was passed to this function,
    the domain name from the "redirect" attribute
        (after macro expansion), and
    the IP address that was passed to this function.

2. Otherwise, if the "default" attribute is present, the result of the client authorization function is the value of this attribute.

3. Otherwise, the result of the client authorization function is "fail".

5.1.4 "m" element

Attributes: "result" (one of "pass", "fail", "softFail", "unknown")
Child Elements: A sequence of zero or more "a", "exists", "include", "mx" and "r" elements.

Each "m" element describes a portion of the client authorization function. To evaluate the client authorization function, evaluate each of the child elements in sequence, until one of them returns a match. If one of them returns a match, the result of the client authorization function is the value of the "result" attribute (whose default is "pass"). If no child element returns a match, evaluation will continue with the next "m" element.

5.1.5 "a" element

Attributes: None
Child Elements: None
Character Data: An optional domain name or IPv4 or IPv6 address, with macro expansion.

If the character data is empty, replace it with "%D" (as defined below in section 5.2.

If an "a" element contains an IPv4 or IPv6 address, it matches iff that address exactly equals the IP address passed to the client authorization function.

Otherwise, obtain the A or AAAA records for the given domain name (after macro expansion). The "a" element matches if the IP address in any of these records exactly equals the IP address that was passed to the client authorization function.

If the specified name does not exist in DNS, or if no A or AAAA records (as appropriate) exist, the "a" element does not match. If any other error occurs during lookup, the result of the client authorization function is "transientError".

5.1.6 "exists" element

Attributes: None
Child Elements: None
Character Data: A domain name, with macro expansion.

This element matches iff the domain given by the character data (after macro expansion) exists in DNS. If the DNS lookup fails for any reason other than nonexistence, the client authorization function returns "transientError".

5.1.7 "include" element

Attributes: None
Child Elements: None
Character Data: A domain name, with macro expansion.

In order to determine whether this element matches, obtain the e-mail policy record for the specified domain (after macro expansion), and evaluate its client authorization function, passing

    the original local-part
    the original domain-name
    the domain name from the character data (after macro expansion),
    the original IP address.

The "include" element matches iff that domain's client authorization function returns "pass".

Otherwise, if the recursive function returns "transientError" or "hardError", that result is also the result of the current function evaluation.

5.1.8 "mx" element

Attributes: None
Child Elements: None
Character Data: An optional domain name, with macro expansion.

If the character data is empty, replace it with "%D".

This element matches iff the IP address is listed as a mail server for the specified domain. This can occur for either of two reasons:

1. One or more MX records exists for the domain, and one of the MX records references a domain that contains an A or AAAA record that exactly matches the input IP address.

2. No MX records exist for the domain, and an A or AAAA record for the domain exactly matches the input IP address.

If any DNS lookup error other than domain not found occurs, the client authorization function returns "transientError".

5.1.9 "ptr" element

Attributes: None
Child Elements: None
Character Data: A domain name, with macro expansion.

To determine whether this element matches, first perform a reverse DNS lookup of the IP address. This element matches iff the given domain name is a suffix of any of the domain names that were fetched in the lookup.

If any DNS lookup error other than domain not found occurs, the client authorization function returns "transientError".

5.1.10 "r" element

Attributes: None
Child Elements: None
Character Data: IP address '/' count

This element matches iff the IP address in the character data is of the same form as the input IP address (IPv4 or IPv6), and the two addresses are equal when comparing only the most significant 'count' bits.

5.2 Macro Expansion

This section defines the macro expansion that occurs on many of the domain names in the e-mail policy document.

5.2.1 Macro definitions

```
macro-string = *( macro-char / VCHAR )
macro-char   = ( "%{" ALPHA transformer *delimiter "}" )
               / "%%" / "%_" / "%-"
transformer  = [ *DIGIT ] [ "r" ]
delimiter    = "." / "-" / "+" / "," / "/" / "_" / "="
```

A literal "%" is expressed by "%%".
%_ expands to a single " " space.
%- expands to a URL-encoded space, viz. "%20".

The following macro letters are expanded in directive arguments:

```
l = local-part passed to the client authorization function.
s = Same as "%{l}@%{o}".
o = original domain passed to the client authorization function.
d = current domain passed to the client authorization function.
i = SMTP client IP (nibble format when an IPv6 address).
p = SMTP client domain name
v = client IP version string: "in-addr" for ipv4 or "ip6" for ipv6
r = domain name of the SMTP server.
```

The uppercase versions of all these macros are URL-encoded.

A '%' character not followed by a '{', '%', '-', or '_' character MUST be interpreted as a literal. SPF publishers SHOULD NOT rely on this feature; they MUST escape % literals.

Legal optional transformers are:

```
*DIGIT ; zero or more digits
'r'    ; reverse value, splitting on dots by default
```

If transformers or delimiters are provided, the macro strings are split into parts. After performing any reversal operation or removal of left-hand parts, the parts are rejoined using "." and not the original splitting characters.

By default, strings are split on "." (dots). Macros may specify delimiter characters which are used instead of ".". Delimiters MUST be one or more of the characters:

```
   "." / "-" / "+" / "," / "/" / "_" / "="
```

The 'r' transformer indicates a reversal operation: if the client IP address were 192.0.2.1, the macro %{i} would expand to "192.0.2.1" and the macro %{ir} would expand to "1.2.0.192".

The DIGIT transformer indicates the number of right-hand parts to use after optional reversal. If a DIGIT is specified, it MUST be nonzero. If no DIGITs are specified, or if the value specifies more parts than are available, all the available parts are used. If the DIGIT was 5, and only 3 parts were available, the macro interpreter would pretend the DIGIT was 3. Implementations MAY limit the number, but MUST support at least a value of 128.

For IPv4 addresses, both the "i" and "c" macros expand to the standard dotted-quad format.

For IPv6 addresses, the "i" macro expands to dot-format address; it is intended for use in %{ir}. The "c" macro may expand to any of the hexadecimal colon-format addresses specified in [RFC3513] section 2.2. It is intended for humans to read.

The "p" macro expands to the validated domain name of the SMTP client. The validation procedure is described in section 4.6. If there are no validated domain names, the word "unknown" is substituted. If multiple validated domain names exist, the first one returned in the PTR result is chosen.

The "r" macro expands to the name of the receiving MTA. This SHOULD be a fully qualified domain name, but if one does not exist (as when the checking is done by a script) or if policy restrictions dictate otherwise, the word "unknown" SHOULD be substituted. The domain name MAY be different than the name found in the MX record that the client MTA used to locate the receiving MTA.

The "s" macro expands to the sender email address: a localpart, an @ sign, and a domain. The "o" macro is the domain part of the "s". They remain the same during a recursive "include" or "redirect" subquery.

When the result of macro expansion is used in a domain name query, if the expanded domain name exceeds 255 characters (the maximum length of a domain name), the left side is truncated to fit, by removing successive subdomains until the total length falls below 255 characters.

Uppercased macros are URL escaped.

URL encoding is described in [RFC2396].

5.2.2 Expansion Examples

```
The <responsible-sender> is internet-draft@email.example.com.
The IPv4 SMTP client IP is 192.0.2.3.
The IPv6 SMTP client IP is 5f05:2000:80ad:5800::1.
The PTR domain name of the client IP is mx.example.org.

macro                 expansion
-------------------------------
%{s} internet-draft@email.example.com
%{o}                email.example.com
%{d}                email.example.com
%{d4}               email.example.com
%{d3}               email.example.com
%{d2}                     example.com
%{d1}                             com
%{p}                   mx.example.org
%{p2}                     example.org
%{dr}               com.example.email
%{d2r}                  example.email
%{l}                   internet-draft
%{l-}                  internet.draft
%{lr}                  internet-draft
%{lr-}                 draft-internet
%{l1r-}                         draft

macro-string                                   expansion
---------------------------------------------------------------------
%{ir}.%{v}._spf.%{d2}             3.2.0.192.in-addr._spf.example.com
%{lr-}.lp._spf.%{d2}              draft.internet.lp._spf.example.com

%{lr-}.lp.%{ir}.%{v}._spf.%{d2}
                 Draft.internet.lp.3.2.0.192.in-addr._spf.example.com

%{ir}.%{v}.%{l1r-}.lp._spf.%{d2}
                       3.2.0.192.in-addr.internet.lp._spf.example.com

%{p2}.trusted-domains.example.net
                              example.org.trusted-domains.example.net

IPv6:
%{ir}.example.org              1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.
                                5.d.a.0.8.0.0.0.2.5.0.f.5.example.org
```

5.3 Determining the E-mail Policy Document for a Domain

In order to find the E-mail policy document for a domain, the following algorithm (or an algorithm that always yields the same result) MUST be used. In this algorithm, "$DOMAIN" stands for the domain whose E-mail policy document is desired.

1. From DNS, obtain the set of TXT records for domain "_ep.$DOMAIN". If the set contains exactly one TXT record, apply the transformations in section 5.3.1 to that record.

   If the set contains more than one record, the E-mail policy document cannot be obtained, and the result of validation is "hardError".

   If the set is empty, or if the domain "_ep.$DOMAIN" does not exist, continue to step 2.

   If any DNS error other than NXDOMAIN occurs, the e-mail policy document cannot be obtained, and the result of validation is "transientError".

2. From DNS, obtain the set of TXT records for domain "$DOMAIN". If the set contains exactly one record whose first string starts with the characters "v=spf1 ", apply the transformations in section 5.3.2 to that record to obtain the EMail policy document.

   If the record set contains more than one record whose initial string starts with "v=spf1", the EMail policy document cannot be obtained, and the result of validation is "hardError".

   If the record set contains no records whose initial string starts with "v=spf1", the EMail policy document cannot be obtained, and the result of validation is "none".

   If the domain "$DOMAIN" does not exist, the EMail policy document cannot be obtains, and the result of validation is "fail".

   If any DNS error other than NXDOMAIN occurs, the EMail policy document cannot be obtained, and the result of validation is "transientError".

5.3.1 From an XML TXT Record

Given a DNS TXT record from the "_ep" subdomain of the desired domain, the EMail policy document is obtained as follows:

1. Append a CRLF to each string in the DNS record.
2. Concatenate all of the resulting strings together, in order.
3. Prefix the resulting string with the following:

```
      <?xml version="1.0" encoding="UTF-8"?>
```

```
<root xmlns="urn:ietf:params:xml:schema:marid-1"
      xmlns:m2="urn:ietf:params:xml:schema:marid-2"
      xmlns:m3="urn:ietf:params:xml:schema:marid-3"
      xmlns:m4="urn:ietf:params:xml:schema:marid-4"
      xmlns:m5="urn:ietf:params:xml:schema:marid-5"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
```

4. Suffix the resulting string with the following:
   `</root>`

5. If the resulting string is not a well-formed XML document, the domain's EMail policy document cannot be obtained, and the result of validation is "hardError".

6. If the resulting string, treated as an XML document, is not valid with respect to the schema published in Appendix A, the EMail policy document cannot be obtained, and the result of validation is "hardError".

7. A processor MAY attempt to validate the resulting XML document with respect to any other XML schemas of which it is aware. It MAY return a "hardError" result if any such validation fails. A processor SHOULD NOT attempt to dynamically locate schemas for namespaces whose semantics are unknown to the processor.

8. The Infoset corresponding to the resulting string is the domain's EMail policy document.

XML namespace "urn:ietf:params:xml:schema:marid-1" is defined by this specification, the "...:marid-2" through "...:marid-5" namespaces are reserved for definition by future IETF consensus processes, and the namespaces "http://www.w3.org/2001/XMLSchema" and "http://www.w3.org/2000/09/xmldsig#" are defined by [XMLSchema] and [RFC3275], respectively.

Note: Even though the implied header for the XML document specifies <encoding="UTF-8">, domain that publish records SHOULD restrict themselves to the 7-bit US-ASCII subset, because there exist DNS resolvers and caching DNS servers that incorrectly reject TXT records with any characters whose high bits are set. Nevertheless, should an MTA receive a record containing any characters with high bits set, the MTA MUST treat those characters as UTF-8 encoded.

Examples:

```
example.com. IN TXT "<ep><out><m><mx/></m></out></ep>"
```

This record says that example.com's outbound mail servers are exactly the same as its inbound servers.

```
example.com. IN TXT "<ep><out><m><r>10.1.1.0/24</r></m></out></ep>"
```
This record says that example.com's outbound mail servers are given by the IP address range 10.1.1.0/24.

5.3.2 From an SPF TXT Record

<< This text really needs to be rewritten. What follows gives the concept, but isn't legalistic enough. It probably needs to stand alone instead of referencing the SPF spec. >>

Given a TXT record from the desired domain that starts with the characters "v=spf1 ", the EMail policy document is obtained via the following algorithm. The namespace of all elements and attributes in the resulting XML infoset is "urn:ietf:params:xml:schema:marid-1".

1. Start with the infoset described by the following XML text:
   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <root xmlns="urn:ietf:params:xml:schema:marid-1">
     <ep>
       <out default="unknown"/>
     </ep>
   </root>
   ```

2. For each SPF mechanism in the input, create a new "m" node under the "out" node. The "result" attribute of the "m" node is "pass", "fail", "softFail" or "unknown", depending on whether the mechanism's prefix is "+", "-", "~", or "?".

3. Unless the mechanism name is "all", create a new element under the new "m" node, of a type as follows:

mechanism	element
a	a
exists	exists
include	include
ip4 without "/"	a
ip4 with "/"	r
ip6 without "/"	a
ip6 with "/"	r
mx	mx
ptr	ptr

   Place everything after the ":" (if any) into the character data of the new element.

4. Treat any unknown mechanism as if it said "?all".

5. If there's a "redirect" modifier, set the "redirect" attribute of the "out" node.

The resulting infoset is now complete.

5.4 Recursion Limitations

Evaluation of many of the mechanisms in section 5.1 will require additional DNS lookups. To avoid infinite recursion, and to avoid certain denial of service attacks, an MTA or other processor SHOULD limit the total number of DNS lookups that it is willing to perform in the course of a single authentication. Such a limit SHOULD allow for at least 20 lookups. If such a limit is exceeded, the result of authentication MUST be "hardError".

MTAs or other processors MAY also impose a limit on the maximum amount of elapsed time to perform an authentication. Such a limit SHOULD allow at least 10 seconds. If such a limit is exceeded, the result of authentication SHOULD be "transientError".

Domains publishing records SHOULD keep the number of DNS lookups to less than 20. Domains publishing records that are intended for use as the target of "indirect" elements SHOULD keep the number of DNS lookups to less than 10.

6. Actions Based on the Decision

This section describes the meaning of each possible result of the authentication function.

6.1 pass

A "pass" result from the check means that the domain has authorized the sending of the message in question. An SMTP server receiving this result SHOULD accept the message.

6.2 fail

A "fail" result from the check means that the domain disclaims authorization of the message. An SMTP server receiving this result SHOULD reject the message with a 550 SMTP error code.

6.3 softFail

A "softFail" result from the check means that the message did not originate from the domain's primary mail servers, but the domain is

unable to definitively state that the message was not authorized. An SMTP server receiving this result SHOULD NOT reject the message for this reason alone, but MAY subject the message to heightened scrutiny by other anti-spam measures, and MAY reject the message as a result of this heightened scrutiny. A message for which the result is "softFail" is less likely to be authentic than a message for which the result is "neutral".

6.4 neutral

A "neutral" result from the check means that the domain can offer no information about whether the message was authorized. An SMTP server receiving this result SHOULD NOT reject the message for this reason alone, but MAY subject the message to heightened scrutiny by other anti-spam measures, and MAY reject the message as a result of this heightened scrutiny.

6.5 transientError

A "transientError" result from the check means that the check was aborted because of inability to get a definitive answer from a DNS lookup. An SMTP server receiving this result MAY reject the message with a 450 error code (transient failure). Alternatively, an SMTP server receiving this result MAY accept a message and optionally subject it to heightened scrutiny by other anti-spam measures.

6.6 hardError

A "hardError" result from the check means that the check was aborted because the published information was ill-formed or because too much processing would be required to achieve a definitive answer. An SMTP server SHOULD treat this result identically to a "neutral" result.

6.7 none

A "none" result from the check means that the check was aborted because the domain did not publish an e-mail policy document. An SMTP server SHOULD treat this identically to a "neutral" result.

7. Security Considerations

This entire document describes a new mechanism for mitigating spoofed email, which is today a pervasive security problem in the Internet.

Assuming that this mechanism is widely deployed, the following sections describe counter-attacks that could be used to defeat this mechanism.

7.1 DNS Attacks

The new mechanism is entirely dependent of DNS lookups, and is therefore only as secure as DNS. An attacker bent on spoofing messages could attempt to get his messages accepted by sending forged answers to DNS queries.

An MTA could largely defeat such an attack by using a properly paranoid DNS resolver. DNSSEC may ultimately provide a way to completely neutralize this class of attacks.

7.2 TCP Attacks

This mechanism is designed to be used in conjunction with SMTP over TCP. A sufficiently resourceful attacker might be able to send TCP packets with forged from-addresses, and thus execute an entire SMTP session that appears to come from somewhere other than its true origin.

Such an attack requires guessing what TCP sequence numbers an SMTP server will use. It also requires transmitting completely in the blind û the attack will be unable hear any of the server's side of the conversation.

Attacks of this sort can be ameliorated if IP gateways refuse to forward packets when the source address is clearly bogus.

7.3 Forged Resent-From Attacks

This mechanism chooses a purported responsible address from one of a number of message headers, and then uses that address for validation. A message with a true Resent-From header (for example), but a forged From header will be accepted. Since many MUAs do not display all of the headers of received messages, the message will appear to be forged when displayed.

In order to avoid this attack, MUAs will need to start displaying at least the header that was verified.

8. Extensibility Considerations

Many of the XML elements are defined to allow any attribute and any child element. A program MUST ignore any element or attribute whose meaning it does not understand. It will be easy to use new elements and attributes to describe other pieces of email policy that are not

currently specified. For example, it may become appropriate to define elements that allow a domain to specify how to complain about spam from that domain, which accreditation agencies can vouch for the domain, whether the domain is interested in answering challenges in challenge / response systems, and so forth.

While it will be easy to add new kinds of information to the e-mail policy document, future implementers must proceed with extreme care if they attempt to modify the semantics of the client authorization function. Code that is not aware of the new semantics will completely ignore any newly invented elements.

It is anticipated that extensions will only be defined for the XML version of the DNS record, not for the SPF version.

With the final draft of this document, the "urn:ietf:params:xml:schema:marid-1" namespace schema becomes frozen for all time. Future extensions will need to use other namespaces. In order to avoid having to write very long namespaces in DNS TXT records, the namespace prefixes "m2" through "m5" are defined at this time, even though their referents will only be defined in the future. Their referents, the namespaces "urn:ietf:params:xml:schema:marid-2" through "...-5" are of the form that they can only be defined by a future IETF consensus process.

9. Applicability Statement

This section describes the actions that certain members of the Internet email ecosystem must take to be compliant with this specification.

<< Be more precise and prescriptive about what header to insert where. >>

9.1 Simple E-mailers

A domain that injects original email into the Internet, using its own name in From headers, need do nothing to be compliant. However, such domains SHOULD publish e-mail policy records in DNS.

9.2 E-mail Forwarders

A program that forwards received mail to other addresses MUST add an appropriate header that contains an email address that it is authorized to use. Such programs SHOULD use the Resent-From header for this purpose.

Many of today's forwarders already add an appropriate header (although most of them use Delivered-To or some other nonstandard header rather than Resent-From.)

9.3 Mailing List Servers

A mailing list server MUST add an appropriate header that contains an email address that it is authorized to use. Such programs SHOULD use the Resent-From header for this purpose.

Most of today's mailing list software already adds an appropriate header (although most of them use Sender rather than Resent-From).

9.4 Third-Party Mailers

A program that sends mail on behalf of another user MUST add an appropriate header than contains an email address that it is authorized to use. Such programs SHOULD use the Sender header for this purpose.

Many, but not all, of today's third-party mailers are already compliant.

9.5 MTA Implementers

MTAs that are acting as SMTP servers SHOULD implement the checks described in this document.

An MTA SHOULD limit the number of DNS lookups (or the time spent performing the lookup) that it will perform in the process of checking a message. Such a limit SHOULD permit at least 20 DNS lookups.

9.6 MUA Implementers

When displaying a received message, an MUA SHOULD display the purported responsible address as defined by this document whenever that address differs from the From address. This display SHOULD be in addition to the From address.

When a received message contains multiple headers that might be used for the purported responsible address determination, an MUA should consider displaying all of them. That is, if a message contains several Resent-From's, a Sender and a From, an MUA should consider displaying all of them.

10. IANA Considerations

The IANA is requested to register the following URI:

URI:	urn:ietf:params:xml:schema:marid-1
Registrant Contact:	IESG.
XML:	The contents of Appendix A of this document

11. Acknowledgements

Variations on the idea of using a DNS record to check the legitimacy of an email address have occurred multiple times. The earliest known work is [RMX]; others include [SPF} and [CallerID].

The current document borrows heavily from each of the above, and incorporates many ideas proposed by many members of the MARID working group.

12. References

12.1 Normative References

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

[RFC2026]   S. Bradner, "The Internet Standards Process û Revision 3", RFC 2026.

[RFC3275]   D. Eastlake et al, "XML-Signature Syntax and Processing", RFC 3275.

[XMLSchema] "XML Schema Part 1: Structures", W3C Recommendation 2 May 2001, http://www.w3c.org/TR/2001/REC-xmlschema-1-20010502/

12.2 Informative References

[CallerID] Microsoft Corporation, Caller ID for E-Mail Technical Specification, http://www.microsoft.com/mscorp/twc/privacy/spam_callerid .mspx.

[CSV]   D. Crocker, "Client SMTP Validation (CSV)", draft-crocker-marid-smtp-validate-01. Work in progress.

[DK]   M. Delany, "Domain-based Email Authentication Using Public-Keys Advertised in the DNS (DomainKeys)", draft-delany-domainkeys-base-00. Work in progress.

[MTAMARK]   M. Stumpf, and S. Hoehne, "Marking Mail Transfer Agents in Reverse DNS with TXT RRs", draft-stumpf-dns-mtamark-02. Work in progress.

[RMX]   H. Danisch, "The RMX DNS RR and method for lightweight SMTP sender authorization", draft-danisch-dns-rr-smtp-04. Work in progress.

[SMIME]   B. Ramsdell (editor), "S/MIME Version 3 Message Specification", RFC 2633.

[SPF]   M. Lentczner and M. Weng, "Sender Policy Framework (SPF): A Convention to Describe Hosts Authorized to Send SMTP Traffic", draft-mengwong-spf-01. Work in progress.

[Submitter] E. Allman and H. Katz, "SMTP Service Extension for Indicating the Responsible Submitter of an E-mail Message", draft-ietf-marid-submitter-00. Work in progress.

13. Authors' Addresses

Jim Lyon
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
USA
jimlyon@microsoft.com

Meng Weng Wong
Singapore
mengwong@dumbo.pobox.com

14. Full Copyright Statement

Appendix A: XML Schema for urn:ietf:params:xml:schema:marid-1

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:ietf:params:xml:schema:marid-1"
elementFormDefault="qualified" attributeFormDefault="unqualified"
blockDefault="#all" xmlns="urn:ietf:params:xml:schema:marid-1"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="root">
    <xs:complexType>
      <xs:all>
        <xs:element name="ep">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="out" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="m" minOccurs="0"
                    maxOccurs="unbounded">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:choice minOccurs="0"
                          maxOccurs="unbounded">
                            <xs:element name="a"
                            type="ExtensibleString"/>
                            <xs:element name="exists"
                            type="ExtensibleString"/>
                            <xs:element name="include"
                            type="ExtensibleString"/>
                            <xs:element name="mx"
                            type="ExtensibleString"/>
                            <xs:element name="ptr"
                            type="ExtensibleString"/>
                            <xs:element name="r"
                            type="ExtensibleString"/>
                          </xs:choice>
                          <xs:any namespace="##other"
                          processContents="lax" minOccurs="0"
                          maxOccurs="unbounded"/>
                        </xs:sequence>
                        <xs:attribute name="result" type="resultType"
                        use="optional" default="pass"/>
                        <xs:anyAttribute namespace="##other"
                        processContents="lax"/>
                      </xs:complexType>
                    </xs:element>
                    <xs:any namespace="##other" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
          <xs:attribute name="default" type="resultType"
```

```
                use="optional" default="fail"/>
                <xs:attribute name="redirect" type="xs:string"
                use="optional"/>
                <xs:anyAttribute namespace="##other"
                processContents="lax"/>
              </xs:complexType>
            </xs:element>
            <xs:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="testing" type="xs:boolean"
          use="optional" default="false"/>
          <xs:anyAttribute namespace="##other"
          processContents="lax"/>
        </xs:complexType>
      </xs:element>
    </xs:all>
  </xs:complexType>
</xs:element>
<xs:complexType name="ExtensibleString">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:simpleType name="resultType">
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="collapse"/>
    <xs:enumeration value="pass"/>
    <xs:enumeration value="fail"/>
    <xs:enumeration value="softFail"/>
    <xs:enumeration value="unknown"/>
  </xs:restriction>
</xs:simpleType>
```
```
</xs:schema>
```