Network Working Group
Request for Comments: 4408
Category: Experimental

M. Wong
W. Schlitt

April 2006

Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1

Status of This Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Copyright Notice

IESG Note

The following documents (RFC 4405, RFC 4406, RFC 4407, and RFC 4408) are published simultaneously as Experimental RFCs, although there is no general technical consensus and efforts to reconcile the two approaches have failed. As such, these documents have not received full IETF review and are published "AS-IS" to document the different approaches as they were considered in the MARID working group.

The IESG takes no position about which approach is to be preferred and cautions the reader that there are serious open issues for each approach and concerns about using them in tandem. The IESG believes that documenting the different approaches does less harm than not documenting them.

Note that the Sender ID experiment may use DNS records that may have been created for the current SPF experiment or earlier versions in this set of experiments. Depending on the content of the record, this may mean that Sender-ID heuristics would be applied incorrectly to a message. Depending on the actions associated by the recipient with those heuristics, the message may not be delivered or may be discarded on receipt.

Participants relying on Sender ID experiment DNS records are warned that they may lose valid messages in this set of circumstances. aParticipants publishing SPF experiment DNS records should consider the advice given in section 3.4 of RFC 4406 and may wish to publish both v=spf1 and spf2.0 records to avoid the conflict.

Participants in the Sender-ID experiment need to be aware that the way Resent-* header fields are used will result in failure to receive legitimate email when interacting with standards-compliant systems (specifically automatic forwarders which comply with the standards by not adding Resent-* headers, and systems which comply with RFC 822 but have not yet implemented RFC 2822 Resent-* semantics). It would be inappropriate to advance Sender-ID on the standards track without resolving this interoperability problem.

The community is invited to observe the success or failure of the two approaches during the two years following publication, in order that a community consensus can be reached in the future.

Abstract

E-mail on the Internet can be forged in a number of ways. In particular, existing protocols place no restriction on what a sending host can use as the reverse-path of a message or the domain given on the SMTP HELO/EHLO commands. This document describes version 1 of the Sender Policy Framework (SPF) protocol, whereby a domain may explicitly authorize the hosts that are allowed to use its domain name, and a receiving host may check such authorization.

Table of Contents

1. Introduction

The current E-Mail infrastructure has the property that any host injecting mail into the mail system can identify itself as any domain name it wants. Hosts can do this at a variety of levels: in particular, the session, the envelope, and the mail headers. Although this feature is desirable in some circumstances, it is a major obstacle to reducing Unsolicited Bulk E-Mail (UBE, aka spam). Furthermore, many domain name holders are understandably concerned about the ease with which other entities may make use of their domain names, often with malicious intent.

This document defines a protocol by which domain owners may authorize hosts to use their domain name in the "MAIL FROM" or "HELO" identity. Compliant domain holders publish Sender Policy Framework (SPF) records specifying which hosts are permitted to use their names, and compliant mail receivers use the published SPF records to test the authorization of sending Mail Transfer Agents (MTAs) using a given "HELO" or "MAIL FROM" identity during a mail transaction.

An additional benefit to mail receivers is that after the use of an identity is verified, local policy decisions about the mail can be made based on the sender's domain, rather than the host's IP address. This is advantageous because reputation of domain names is likely to be more accurate than reputation of host IP addresses. Furthermore, if a claimed identity fails verification, local policy can take stronger action against such E-Mail, such as rejecting it.

1.1. Protocol Status

SPF has been in development since the summer of 2003 and has seen deployment beyond the developers beginning in December 2003. The design of SPF slowly evolved until the spring of 2004 and has since stabilized. There have been quite a number of forms of SPF, some written up as documents, some submitted as Internet Drafts, and many discussed and debated in development forums.

The goal of this document is to clearly document the protocol defined by earlier draft specifications of SPF as used in existing implementations. This conception of SPF is sometimes called "SPF Classic". It is understood that particular implementations and

deployments may differ from, and build upon, this work. It is hoped that we have nonetheless captured the common understanding of SPF version 1.

1.2. Terminology

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

This document is concerned with the portion of a mail message commonly called "envelope sender", "return path", "reverse path", "bounce address", "2821 FROM", or "MAIL FROM". Since these terms are either not well defined or often used casually, this document defines the "MAIL FROM" identity in Section 2.2. Note that other terms that may superficially look like the common terms, such as "reverse-path", are used only with the defined meanings from normative documents.

2. Operation

2.1. The HELO Identity

The "HELO" identity derives from either the SMTP HELO or EHLO command (see [RFC2821]). These commands supply the SMTP client (sending host) for the SMTP session. Note that requirements for the domain presented in the EHLO or HELO command are not always clear to the sending party, and SPF clients must be prepared for the "HELO" identity to be malformed or an IP address literal. At the time of this writing, many legitimate E-Mails are delivered with invalid HELO domains.

It is RECOMMENDED that SPF clients not only check the "MAIL FROM" identity, but also separately check the "HELO" identity by applying the check_host() function (Section 4) to the "HELO" identity as the <sender>.

2.2. The MAIL FROM Identity

The "MAIL FROM" identity derives from the SMTP MAIL command (see [RFC2821]). This command supplies the "reverse-path" for a message, which generally consists of the sender mailbox, and is the mailbox to which notification messages are to be sent if there are problems delivering the message.

[RFC2821] allows the reverse-path to be null (see Section 4.5.5 in RFC 2821). In this case, there is no explicit sender mailbox, and such a message can be assumed to be a notification message from the mail system itself. When the reverse-path is null, this document

defines the "MAIL FROM" identity to be the mailbox composed of the localpart "postmaster" and the "HELO" identity (which may or may not have been checked separately before).

SPF clients MUST check the "MAIL FROM" identity. SPF clients check the "MAIL FROM" identity by applying the check_host() function to the "MAIL FROM" identity as the <sender>.

2.3. Publishing Authorization

An SPF-compliant domain MUST publish a valid SPF record as described in Section 3. This record authorizes the use of the domain name in the "HELO" and "MAIL FROM" identities by the MTAs it specifies.

If domain owners choose to publish SPF records, it is RECOMMENDED that they end in "-all", or redirect to other records that do, so that a definitive determination of authorization can be made.

Domain holders may publish SPF records that explicitly authorize no hosts if mail should never originate using that domain.

When changing SPF records, care must be taken to ensure that there is a transition period so that the old policy remains valid until all legitimate E-Mail has been checked.

2.4. Checking Authorization

A mail receiver can perform a set of SPF checks for each mail message it receives. An SPF check tests the authorization of a client host to emit mail with a given identity. Typically, such checks are done by a receiving MTA, but can be performed elsewhere in the mail processing chain so long as the required information is available and reliable. At least the "MAIL FROM" identity MUST be checked, but it is RECOMMENDED that the "HELO" identity also be checked beforehand.

Without explicit approval of the domain owner, checking other identities against SPF version 1 records is NOT RECOMMENDED because there are cases that are known to give incorrect results. For example, almost all mailing lists rewrite the "MAIL FROM" identity (see Section 9.2), but some do not change any other identities in the message. The scenario described in Section 9.3, sub-section 1.2, is another example. Documents that define other identities should define the method for explicit approval.

It is possible that mail receivers will use the SPF check as part of a larger set of tests on incoming mail. The results of other tests may influence whether or not a particular SPF check is performed. For example, finding the sending host’s IP address on a local white

list may cause all other tests to be skipped and all mail from that host to be accepted.

When a mail receiver decides to perform an SPF check, it MUST use a correctly-implemented check_host() function (Section 4) evaluated with the correct parameters. Although the test as a whole is optional, once it has been decided to perform a test it must be performed as specified so that the correct semantics are preserved between publisher and receiver.

To make the test, the mail receiver MUST evaluate the check_host() function with the arguments set as follows:

<ip> - the IP address of the SMTP client that is emitting the mail, either IPv4 or IPv6.

<domain> - the domain portion of the "MAIL FROM" or "HELO" identity.

<sender> - the "MAIL FROM" or "HELO" identity.

Note that the <domain> argument may not be a well-formed domain name. For example, if the reverse-path was null, then the EHLO/HELO domain is used, with its associated problems (see Section 2.1). In these cases, check_host() is defined in Section 4.3 to return a "None" result.

Although invalid, malformed, or non-existent domains cause SPF checks to return "None" because no SPF record can be found, it has long been the policy of many MTAs to reject E-Mail from such domains, especially in the case of invalid "MAIL FROM". In order to prevent the circumvention of SPF records, rejecting E-Mail from invalid domains should be considered.

Implementations must take care to correctly extract the <domain> from the data given with the SMTP MAIL FROM command as many MTAs will still accept such things as source routes (see [RFC2821], Appendix C), the %-hack (see [RFC1123]), and bang paths (see [RFC1983]). These archaic features have been maliciously used to bypass security systems.

2.5. Interpreting the Result

This section describes how software that performs the authorization should interpret the results of the check_host() function. The authorization check SHOULD be performed during the processing of the SMTP transaction that sends the mail. This allows errors to be returned directly to the sending MTA by way of SMTP replies.

Performing the authorization after the SMTP transaction has finished may cause problems, such as the following: (1) It may be difficult to accurately extract the required information from potentially deceptive headers; (2) legitimate E-Mail may fail because the sender's policy may have since changed.

Generating non-delivery notifications to forged identities that have failed the authorization check is generally abusive and against the explicit wishes of the identity owner.

2.5.1. None

A result of "None" means that no records were published by the domain or that no checkable sender domain could be determined from the given identity. The checking software cannot ascertain whether or not the client host is authorized.

2.5.2. Neutral

The domain owner has explicitly stated that he cannot or does not want to assert whether or not the IP address is authorized. A "Neutral" result MUST be treated exactly like the "None" result; the distinction exists only for informational purposes. Treating "Neutral" more harshly than "None" would discourage domain owners from testing the use of SPF records (see Section 9.1).

2.5.3. Pass

A "Pass" result means that the client is authorized to inject mail with the given identity. The domain can now, in the sense of reputation, be considered responsible for sending the message. Further policy checks can now proceed with confidence in the legitimate use of the identity.

2.5.4. Fail

A "Fail" result is an explicit statement that the client is not authorized to use the domain in the given identity. The checking software can choose to mark the mail based on this or to reject the mail outright.

If the checking software chooses to reject the mail during the SMTP transaction, then it SHOULD use an SMTP reply code of 550 (see [RFC2821]) and, if supported, the 5.7.1 Delivery Status Notification (DSN) code (see [RFC3464]), in addition to an appropriate reply text. The check_host() function may return either a default explanation string or one from the domain that published the SPF records (see Section 6.2). If the information does not originate with the

checking software, it should be made clear that the text is provided by the sender's domain. For example:

```
550-5.7.1 SPF MAIL FROM check failed:
550-5.7.1 The domain example.com explains:
550 5.7.1 Please see http://www.example.com/mailpolicy.html
```

2.5.5. SoftFail

A "SoftFail" result should be treated as somewhere between a "Fail" and a "Neutral". The domain believes the host is not authorized but is not willing to make that strong of a statement. Receiving software SHOULD NOT reject the message based solely on this result, but MAY subject the message to closer scrutiny than normal.

The domain owner wants to discourage the use of this host and thus desires limited feedback when a "SoftFail" result occurs. For example, the recipient's Mail User Agent (MUA) could highlight the "SoftFail" status, or the receiving MTA could give the sender a message using a technique called "greylisting" whereby the MTA can issue an SMTP reply code of 451 (4.3.0 DSN code) with a note the first time the message is received, but accept it the second time.

2.5.6. TempError

A "TempError" result means that the SPF client encountered a transient error while performing the check. Checking software can choose to accept or temporarily reject the message. If the message is rejected during the SMTP transaction for this reason, the software SHOULD use an SMTP reply code of 451 and, if supported, the 4.4.3 DSN code.

2.5.7. PermError

A "PermError" result means that the domain's published records could not be correctly interpreted. This signals an error condition that requires manual intervention to be resolved, as opposed to the TempError result. Be aware that if the domain owner uses macros (Section 8), it is possible that this result is due to the checked identities having an unexpected format.

3. SPF Records

An SPF record is a DNS Resource Record (RR) that declares which hosts are, and are not, authorized to use a domain name for the "HELO" and "MAIL FROM" identities. Loosely, the record partitions all hosts into permitted and not-permitted sets (though some hosts might fall into neither category).

The SPF record is a single string of text. An example record is the following:

```
v=spf1 +mx a:colo.example.com/28 -all
```

This record has a version of "spf1" and three directives: "+mx", "a:colo.example.com/28" (the + is implied), and "-all".

3.1. Publishing

Domain owners wishing to be SPF compliant must publish SPF records for the hosts that are used in the "MAIL FROM" and "HELO" identities. The SPF records are placed in the DNS tree at the host name it pertains to, not a subdomain under it, such as is done with SRV records. This is the same whether the TXT or SPF RR type (see Section 3.1.1) is used.

The example above in Section 3 might be published via these lines in a domain zone file:

```
example.com.          TXT "v=spf1 +mx a:colo.example.com/28 -all"
smtp-out.example.com. TXT "v=spf1 a -all"
```

When publishing via TXT records, beware of other TXT records published there for other purposes. They may cause problems with size limits (see Section 3.1.4).

3.1.1. DNS Resource Record Types

This document defines a new DNS RR of type SPF, code 99. The format of this type is identical to the TXT RR [RFC1035]. For either type, the character content of the record is encoded as [US-ASCII].

It is recognized that the current practice (using a TXT record) is not optimal, but it is necessary because there are a number of DNS server and resolver implementations in common use that cannot handle the new RR type. The two-record-type scheme provides a forward path to the better solution of using an RR type reserved for this purpose.

An SPF-compliant domain name SHOULD have SPF records of both RR types. A compliant domain name MUST have a record of at least one type. If a domain has records of both types, they MUST have identical content. For example, instead of publishing just one record as in Section 3.1 above, it is better to publish:

```
example.com. IN TXT "v=spf1 +mx a:colo.example.com/28 -all"
example.com. IN SPF "v=spf1 +mx a:colo.example.com/28 -all"
```

Example RRs in this document are shown with the TXT record type; however, they could be published with the SPF type or with both types.

3.1.2. Multiple DNS Records

A domain name MUST NOT have multiple records that would cause an authorization check to select more than one record. See Section 4.5 for the selection rules.

3.1.3. Multiple Strings in a Single DNS record

As defined in [RFC1035] sections 3.3.14 and 3.3, a single text DNS record (either TXT or SPF RR types) can be composed of more than one string. If a published record contains multiple strings, then the record MUST be treated as if those strings are concatenated together without adding spaces. For example:

```
IN TXT "v=spf1 .... first" "second string..."
```

MUST be treated as equivalent to

```
IN TXT "v=spf1 .... firstsecond string..."
```

SPF or TXT records containing multiple strings are useful in constructing records that would exceed the 255-byte maximum length of a string within a single TXT or SPF RR record.

3.1.4. Record Size

The published SPF record for a given domain name SHOULD remain small enough that the results of a query for it will fit within 512 octets. This will keep even older DNS implementations from falling over to TCP. Since the answer size is dependent on many things outside the scope of this document, it is only possible to give this guideline: If the combined length of the DNS name and the text of all the records of a given type (TXT or SPF) is under 450 characters, then DNS answers should fit in UDP packets. Note that when computing the sizes for queries of the TXT format, one must take into account any other TXT records published at the domain name. Records that are too long to fit in a single UDP packet MAY be silently ignored by SPF clients.

3.1.5. Wildcard Records

Use of wildcard records for publishing is not recommended. Care must be taken if wildcard records are used. If a domain publishes wildcard MX records, it may want to publish wildcard declarations,

subject to the same requirements and problems. In particular, the declaration must be repeated for any host that has any RR records at all, and for subdomains thereof. For example, the example given in [RFC1034], Section 4.3.3, could be extended with the following:

```
X.COM.          MX      10      A.X.COM
X.COM.          TXT     "v=spf1 a:A.X.COM -all"

*.X.COM.        MX      10      A.X.COM
*.X.COM.        TXT     "v=spf1 a:A.X.COM -all"

A.X.COM.        A       1.2.3.4
A.X.COM.        MX      10      A.X.COM
A.X.COM.        TXT     "v=spf1 a:A.X.COM -all"

*.A.X.COM.      MX      10      A.X.COM
*.A.X.COM.      TXT     "v=spf1 a:A.X.COM -all"
```

Notice that SPF records must be repeated twice for every name within the domain: once for the name, and once with a wildcard to cover the tree under the name.

Use of wildcards is discouraged in general as they cause every name under the domain to exist and queries against arbitrary names will never return RCODE 3 (Name Error).

4. The check_host() Function

The check_host() function fetches SPF records, parses them, and interprets them to determine whether a particular host is or is not permitted to send mail with a given identity. Mail receivers that perform this check MUST correctly evaluate the check_host() function as described here.

Implementations MAY use a different algorithm than the canonical algorithm defined here, so long as the results are the same in all cases.

4.1. Arguments

The check_host() function takes these arguments:

<ip> - the IP address of the SMTP client that is emitting the mail, either IPv4 or IPv6.

<domain> - the domain that provides the sought-after authorization information; initially, the domain portion of the "MAIL FROM" or "HELO" identity.

<sender> - the "MAIL FROM" or "HELO" identity.

The domain portion of <sender> will usually be the same as the <domain> argument when check_host() is initially evaluated. However, this will generally not be true for recursive evaluations (see Section 5.2 below).

Actual implementations of the check_host() function may need additional arguments.

4.2. Results

The function check_host() can return one of several results described in Section 2.5. Based on the result, the action to be taken is determined by the local policies of the receiver.

4.3. Initial Processing

If the <domain> is malformed (label longer than 63 characters, zero-length label not at the end, etc.) or is not a fully qualified domain name, or if the DNS lookup returns "domain does not exist" (RCODE 3), check_host() immediately returns the result "None".

If the <sender> has no localpart, substitute the string "postmaster" for the localpart.

4.4. Record Lookup

In accordance with how the records are published (see Section 3.1 above), a DNS query needs to be made for the <domain> name, querying for either RR type TXT, SPF, or both. If both SPF and TXT RRs are looked up, the queries MAY be done in parallel.

If all DNS lookups that are made return a server failure (RCODE 2), or other error (RCODE other than 0 or 3), or time out, then check_host() exits immediately with the result "TempError".

4.5. Selecting Records

Records begin with a version section:

```
record          = version terms *SP
version         = "v=spf1"
```

Starting with the set of records that were returned by the lookup, record selection proceeds in two steps:

1. Records that do not begin with a version section of exactly "v=spf1" are discarded. Note that the version section is terminated either by an SP character or the end of the record. A record with a version section of "v=spf10" does not match and must be discarded.

2. If any records of type SPF are in the set, then all records of type TXT are discarded.

After the above steps, there should be exactly one record remaining and evaluation can proceed. If there are two or more records remaining, then check_host() exits immediately with the result of "PermError".

If no matching records are returned, an SPF client MUST assume that the domain makes no SPF declarations. SPF processing MUST stop and return "None".

4.6. Record Evaluation

After one SPF record has been selected, the check_host() function parses and interprets it to find a result for the current test. If there are any syntax errors, check_host() returns immediately with the result "PermError".

Implementations MAY choose to parse the entire record first and return "PermError" if the record is not syntactically well formed. However, in all cases, any syntax errors anywhere in the record MUST be detected.

4.6.1. Term Evaluation

There are two types of terms: mechanisms and modifiers. A record contains an ordered list of these as specified in the following Augmented Backus-Naur Form (ABNF).

```
terms            = *( 1*SP ( directive / modifier ) )

directive        = [ qualifier ] mechanism
qualifier        = "+" / "-" / "?" / "~"
mechanism        = ( all / include
                   / A / MX / PTR / IP4 / IP6 / exists )
modifier         = redirect / explanation / unknown-modifier
unknown-modifier = name "=" macro-string

name             = ALPHA *( ALPHA / DIGIT / "-" / "_" / "." )
```

Most mechanisms allow a ":" or "/" character after the name.

Modifiers always contain an equals ('=') character immediately after the name, and before any ":" or "/" characters that may be part of the macro-string.

Terms that do not contain any of "=", ":", or "/" are mechanisms, as defined in Section 5.

As per the definition of the ABNF notation in [RFC4234], mechanism and modifier names are case-insensitive.

4.6.2. Mechanisms

Each mechanism is considered in turn from left to right. If there are no more mechanisms, the result is specified in Section 4.7.

When a mechanism is evaluated, one of three things can happen: it can match, not match, or throw an exception.

If it matches, processing ends and the qualifier value is returned as the result of that record. If it does not match, processing continues with the next mechanism. If it throws an exception, mechanism processing ends and the exception value is returned.

The possible qualifiers, and the results they return are as follows:

"+" Pass
"-" Fail
"~" SoftFail
"?" Neutral

The qualifier is optional and defaults to "+".

When a mechanism matches and the qualifier is "-", then a "Fail" result is returned and the explanation string is computed as described in Section 6.2.

The specific mechanisms are described in Section 5.

4.6.3. Modifiers

Modifiers are not mechanisms: they do not return match or not-match. Instead they provide additional information. Although modifiers do not directly affect the evaluation of the record, the "redirect" modifier has an effect after all the mechanisms have been evaluated.

4.7. Default Result

If none of the mechanisms match and there is no "redirect" modifier, then the check_host() returns a result of "Neutral", just as if "?all" were specified as the last directive. If there is a "redirect" modifier, check_host() proceeds as defined in Section 6.1.

Note that records SHOULD always use either a "redirect" modifier or an "all" mechanism to explicitly terminate processing.

For example:

```
   v=spf1 +mx -all
or
   v=spf1 +mx redirect=_spf.example.com
```

4.8. Domain Specification

Several of these mechanisms and modifiers have a <domain-spec> section. The <domain-spec> string is macro expanded (see Section 8). The resulting string is the common presentation form of a fully-qualified DNS name: a series of labels separated by periods. This domain is called the <target-name> in the rest of this document.

Note: The result of the macro expansion is not subject to any further escaping. Hence, this facility cannot produce all characters that are legal in a DNS label (e.g., the control characters). However, this facility is powerful enough to express legal host names and common utility labels (such as "_spf") that are used in DNS.

For several mechanisms, the <domain-spec> is optional. If it is not provided, the <domain> is used as the <target-name>.

5. Mechanism Definitions

This section defines two types of mechanisms.

Basic mechanisms contribute to the language framework. They do not specify a particular type of authorization scheme.

```
   all
   include
```

Designated sender mechanisms are used to designate a set of <ip> addresses as being permitted or not permitted to use the <domain> for sending mail.

```
a
mx
ptr
ip4
ip6
exists
```

The following conventions apply to all mechanisms that perform a comparison between <ip> and an IP address at any point:

If no CIDR-length is given in the directive, then <ip> and the IP address are compared for equality. (Here, CIDR is Classless Inter-Domain Routing.)

If a CIDR-length is specified, then only the specified number of high-order bits of <ip> and the IP address are compared for equality.

When any mechanism fetches host addresses to compare with <ip>, when <ip> is an IPv4 address, A records are fetched, when <ip> is an IPv6 address, AAAA records are fetched. Even if the SMTP connection is via IPv6, an IPv4-mapped IPv6 IP address (see [RFC3513], Section 2.5.5) MUST still be considered an IPv4 address.

Several mechanisms rely on information fetched from DNS. For these DNS queries, except where noted, if the DNS server returns an error (RCODE other than 0 or 3) or the query times out, the mechanism throws the exception "TempError". If the server returns "domain does not exist" (RCODE 3), then evaluation of the mechanism continues as if the server returned no error (RCODE 0) and zero answer records.

5.1. "all"

```
all             = "all"
```

The "all" mechanism is a test that always matches. It is used as the rightmost mechanism in a record to provide an explicit default.

For example:

```
v=spf1 a mx -all
```

Mechanisms after "all" will never be tested. Any "redirect" modifier (Section 6.1) has no effect when there is an "all" mechanism.

5.2. "include"

```
include          = "include"  ":" domain-spec
```

The "include" mechanism triggers a recursive evaluation of check_host(). The domain-spec is expanded as per Section 8. Then check_host() is evaluated with the resulting string as the <domain>. The <ip> and <sender> arguments remain the same as in the current evaluation of check_host().

In hindsight, the name "include" was poorly chosen. Only the evaluated result of the referenced SPF record is used, rather than acting as if the referenced SPF record was literally included in the first. For example, evaluating a "-all" directive in the referenced record does not terminate the overall processing and does not necessarily result in an overall "Fail". (Better names for this mechanism would have been "if-pass", "on-pass", etc.)

The "include" mechanism makes it possible for one domain to designate multiple administratively-independent domains. For example, a vanity domain "example.net" might send mail using the servers of administratively-independent domains example.com and example.org.

Example.net could say

```
IN TXT "v=spf1 include:example.com include:example.org -all"
```

This would direct check_host() to, in effect, check the records of example.com and example.org for a "Pass" result. Only if the host were not permitted for either of those domains would the result be "Fail".

Whether this mechanism matches, does not match, or throws an exception depends on the result of the recursive evaluation of check_host():

A recursive check_host() result of:	Causes the "include" mechanism to:
Pass	match
Fail	not match
SoftFail	not match
Neutral	not match
TempError	throw TempError
PermError	throw PermError
None	throw PermError

The "include" mechanism is intended for crossing administrative boundaries. Although it is possible to use includes to consolidate multiple domains that share the same set of designated hosts, domains are encouraged to use redirects where possible, and to minimize the number of includes within a single administrative domain. For example, if example.com and example.org were managed by the same entity, and if the permitted set of hosts for both domains was "mx:example.com", it would be possible for example.org to specify "include:example.com", but it would be preferable to specify "redirect=example.com" or even "mx:example.com".

5.3. "a"

This mechanism matches if <ip> is one of the <target-name>'s IP addresses.

```
A               = "a"      [ ":" domain-spec ] [ dual-cidr-length ]
```

An address lookup is done on the <target-name>. The <ip> is compared to the returned address(es). If any address matches, the mechanism matches.

5.4. "mx"

This mechanism matches if <ip> is one of the MX hosts for a domain name.

```
MX              = "mx"     [ ":" domain-spec ] [ dual-cidr-length ]
```

check_host() first performs an MX lookup on the <target-name>. Then it performs an address lookup on each MX name returned. The <ip> is compared to each returned IP address. To prevent Denial of Service (DoS) attacks, more than 10 MX names MUST NOT be looked up during the evaluation of an "mx" mechanism (see Section 10). If any address matches, the mechanism matches.

Note regarding implicit MXs: If the <target-name> has no MX records, check_host() MUST NOT pretend the target is its single MX, and MUST NOT default to an A lookup on the <target-name> directly. This behavior breaks with the legacy "implicit MX" rule. See [RFC2821], Section 5. If such behavior is desired, the publisher should specify an "a" directive.

5.5. "ptr"

This mechanism tests whether the DNS reverse-mapping for <ip> exists and correctly points to a domain name within a particular domain.

```
PTR             = "ptr"    [ ":" domain-spec ]
```

First, the <ip>'s name is looked up using this procedure: perform a DNS reverse-mapping for <ip>, looking up the corresponding PTR record in "in-addr.arpa." if the address is an IPv4 one and in "ip6.arpa." if it is an IPv6 address. For each record returned, validate the domain name by looking up its IP address. To prevent DoS attacks, more than 10 PTR names MUST NOT be looked up during the evaluation of a "ptr" mechanism (see Section 10). If <ip> is among the returned IP addresses, then that domain name is validated. In pseudocode:

```
sending-domain_names := ptr_lookup(sending-host_IP); if more than 10
sending-domain_names are found, use at most 10.  for each name in
(sending-domain_names) {
  IP_addresses := a_lookup(name);
  if the sending-domain_IP is one of the IP_addresses {
    validated-sending-domain_names += name;
  } }
```

Check all validated domain names to see if they end in the <target-name> domain. If any do, this mechanism matches. If no validated domain name can be found, or if none of the validated

domain names end in the <target-name>, this mechanism fails to match. If a DNS error occurs while doing the PTR RR lookup, then this mechanism fails to match. If a DNS error occurs while doing an A RR lookup, then that domain name is skipped and the search continues.

Pseudocode:

```
for each name in (validated-sending-domain_names) {
  if name ends in <domain-spec>, return match.
  if name is <domain-spec>, return match.
}
return no-match.
```

This mechanism matches if the <target-name> is either an ancestor of a validated domain name or if the <target-name> and a validated domain name are the same. For example: "mail.example.com" is within the domain "example.com", but "mail.bad-example.com" is not.

Note: Use of this mechanism is discouraged because it is slow, it is not as reliable as other mechanisms in cases of DNS errors, and it places a large burden on the arpa name servers. If used, proper PTR records must be in place for the domain's hosts and the "ptr" mechanism should be one of the last mechanisms checked.

5.6. "ip4" and "ip6"

These mechanisms test whether <ip> is contained within a given IP network.

```
IP4              = "ip4"      ":" ip4-network   [ ip4-cidr-length ]
IP6              = "ip6"      ":" ip6-network   [ ip6-cidr-length ]

ip4-cidr-length  = "/" 1*DIGIT
ip6-cidr-length  = "/" 1*DIGIT
dual-cidr-length = [ ip4-cidr-length ] [ "/" ip6-cidr-length ]

ip4-network      = qnum "." qnum "." qnum "." qnum
qnum             = DIGIT                 ; 0-9
                 / %x31-39 DIGIT         ; 10-99
                 / "1" 2DIGIT            ; 100-199
                 / "2" %x30-34 DIGIT     ; 200-249
                 / "25" %x30-35          ; 250-255
         ; as per conventional dotted quad notation.  e.g., 192.0.2.0
ip6-network      = <as per [RFC 3513], section 2.2>
         ; e.g., 2001:DB8::CD30
```

The <ip> is compared to the given network. If CIDR-length high-order bits match, the mechanism matches.

If ip4-cidr-length is omitted, it is taken to be "/32". If ip6-cidr-length is omitted, it is taken to be "/128". It is not permitted to omit parts of the IP address instead of using CIDR notations. That is, use 192.0.2.0/24 instead of 192.0.2.

5.7. "exists"

This mechanism is used to construct an arbitrary domain name that is used for a DNS A record query. It allows for complicated schemes involving arbitrary parts of the mail envelope to determine what is permitted.

```
exists          = "exists"   ":" domain-spec
```

The domain-spec is expanded as per Section 8. The resulting domain name is used for a DNS A RR lookup. If any A record is returned, this mechanism matches. The lookup type is A even when the connection type is IPv6.

Domains can use this mechanism to specify arbitrarily complex queries. For example, suppose example.com publishes the record:

```
v=spf1 exists:%{ir}.%{l1r+-}._spf.%{d} -all
```

The <target-name> might expand to "1.2.0.192.someuser._spf.example.com". This makes fine-grained decisions possible at the level of the user and client IP address.

This mechanism enables queries that mimic the style of tests that existing anti-spam DNS blacklists (DNSBL) use.

6. Modifier Definitions

Modifiers are name/value pairs that provide additional information. Modifiers always have an "=" separating the name and the value.

The modifiers defined in this document ("redirect" and "exp") MAY appear anywhere in the record, but SHOULD appear at the end, after all mechanisms. Ordering of these two modifiers does not matter. These two modifiers MUST NOT appear in a record more than once each. If they do, then check_host() exits with a result of "PermError".

Unrecognized modifiers MUST be ignored no matter where in a record, or how often. This allows implementations of this document to gracefully handle records with modifiers that are defined in other specifications.

6.1. redirect: Redirected Query

If all mechanisms fail to match, and a "redirect" modifier is present, then processing proceeds as follows:

```
redirect         = "redirect" "=" domain-spec
```

The domain-spec portion of the redirect section is expanded as per the macro rules in Section 8. Then check_host() is evaluated with the resulting string as the <domain>. The <ip> and <sender> arguments remain the same as current evaluation of check_host().

The result of this new evaluation of check_host() is then considered the result of the current evaluation with the exception that if no SPF record is found, or if the target-name is malformed, the result is a "PermError" rather than "None".

Note that the newly-queried domain may itself specify redirect processing.

This facility is intended for use by organizations that wish to apply the same record to multiple domains. For example:

```
  la.example.com. TXT "v=spf1 redirect=_spf.example.com"
  ny.example.com. TXT "v=spf1 redirect=_spf.example.com"
  sf.example.com. TXT "v=spf1 redirect=_spf.example.com"
_spf.example.com. TXT "v=spf1 mx:example.com -all"
```

In this example, mail from any of the three domains is described by the same record. This can be an administrative advantage.

Note: In general, the domain "A" cannot reliably use a redirect to another domain "B" not under the same administrative control. Since the <sender> stays the same, there is no guarantee that the record at domain "B" will correctly work for mailboxes in domain "A", especially if domain "B" uses mechanisms involving localparts. An "include" directive may be more appropriate.

For clarity, it is RECOMMENDED that any "redirect" modifier appear as the very last term in a record.

6.2. exp: Explanation

```
explanation      = "exp" "=" domain-spec
```

If check_host() results in a "Fail" due to a mechanism match (such as "-all"), and the "exp" modifier is present, then the explanation string returned is computed as described below. If no "exp" modifier

is present, then either a default explanation string or an empty explanation string may be returned.

The <domain-spec> is macro expanded (see Section 8) and becomes the <target-name>. The DNS TXT record for the <target-name> is fetched.

If <domain-spec> is empty, or there are any DNS processing errors (any RCODE other than 0), or if no records are returned, or if more than one record is returned, or if there are syntax errors in the explanation string, then proceed as if no exp modifier was given.

The fetched TXT record's strings are concatenated with no spaces, and then treated as an <explain-string>, which is macro-expanded. This final result is the explanation string. Implementations MAY limit the length of the resulting explanation string to allow for other protocol constraints and/or reasonable processing limits. Since the explanation string is intended for an SMTP response and [RFC2821] Section 2.4 says that responses are in [US-ASCII], the explanation string is also limited to US-ASCII.

Software evaluating check_host() can use this string to communicate information from the publishing domain in the form of a short message or URL. Software SHOULD make it clear that the explanation string comes from a third party. For example, it can prepend the macro string "%{o} explains: " to the explanation, such as shown in Section 2.5.4.

Suppose example.com has this record:

```
v=spf1 mx -all exp=explain._spf.%{d}
```

Here are some examples of possible explanation TXT records at explain._spf.example.com:

```
"Mail from example.com should only be sent by its own servers."
   -- a simple, constant message

"%{i} is not one of %{d}'s designated mail servers."
   -- a message with a little more information, including the IP
      address that failed the check

"See http://%{d}/why.html?s=%{S}&i=%{I}"
   -- a complicated example that constructs a URL with the
      arguments to check_host() so that a web page can be
      generated with detailed, custom instructions
```

Note: During recursion into an "include" mechanism, an exp= modifier from the <target-name> MUST NOT be used. In contrast, when executing

a "redirect" modifier, an exp= modifier from the original domain MUST NOT be used.

7. The Received-SPF Header Field

It is RECOMMENDED that SMTP receivers record the result of SPF processing in the message header. If an SMTP receiver chooses to do so, it SHOULD use the "Received-SPF" header field defined here for each identity that was checked. This information is intended for the recipient. (Information intended for the sender is described in Section 6.2, Explanation.)

The Received-SPF header field is a trace field (see [RFC2822] Section 3.6.7) and SHOULD be prepended to the existing header, above the Received: field that is generated by the SMTP receiver. It MUST appear above all other Received-SPF fields in the message. The header field has the following format:

```
header-field     = "Received-SPF:" [CFWS] result FWS [comment FWS]
                   [ key-value-list ] CRLF

result           = "Pass" / "Fail" / "SoftFail" / "Neutral" /
                   "None" / "TempError" / "PermError"

key-value-list   = key-value-pair *( ";" [CFWS] key-value-pair )
                   [";"]

key-value-pair   = key [CFWS] "=" ( dot-atom / quoted-string )

key              = "client-ip" / "envelope-from" / "helo" /
                   "problem" / "receiver" / "identity" /
                    mechanism / "x-" name / name

identity         = "mailfrom"   ; for the "MAIL FROM" identity
                   / "helo"     ; for the "HELO" identity
                   / name       ; other identities

dot-atom         = <unquoted word as per [RFC2822]>
quoted-string    = <quoted string as per [RFC2822]>
comment          = <comment string as per [RFC2822]>
CFWS             = <comment or folding white space as per [RFC2822]>
FWS              = <folding white space as per [RFC2822]>
CRLF             = <standard end-of-line token as per [RFC2822]>
```

The header field SHOULD include a "(...)" style <comment> after the result, conveying supporting information for the result, such as <ip>, <sender>, and <domain>.

The following key-value pairs are designed for later machine parsing. SPF clients SHOULD give enough information so that the SPF results can be verified. That is, at least "client-ip", "helo", and, if the "MAIL FROM" identity was checked, "envelope-from".

client-ip	the IP address of the SMTP client
envelope-from	the envelope sender mailbox
helo	the host name given in the HELO or EHLO command
mechanism	the mechanism that matched (if no mechanisms matched, substitute the word "default")
problem	if an error was returned, details about the error
receiver	the host name of the SPF client
identity	the identity that was checked; see the <identity> ABNF rule

Other keys may be defined by SPF clients. Until a new key name becomes widely accepted, new key names should start with "x-".

SPF clients MUST make sure that the Received-SPF header field does not contain invalid characters, is not excessively long, and does not contain malicious data that has been provided by the sender.

Examples of various header styles that could be generated are the following:

```
Received-SPF: Pass (mybox.example.org: domain of
 myname@example.com designates 192.0.2.1 as permitted sender)
    receiver=mybox.example.org; client-ip=192.0.2.1;
    envelope-from=<myname@example.com>; helo=foo.example.com;

Received-SPF: Fail (mybox.example.org: domain of
                  myname@example.com does not designate
                  192.0.2.1 as permitted sender)
                  identity=mailfrom; client-ip=192.0.2.1;
                  envelope-from=<myname@example.com>;
```

8. Macros

8.1. Macro Definitions

Many mechanisms and modifiers perform macro expansion on part of the term.

```
domain-spec      = macro-string domain-end
domain-end       = ( "." toplabel [ "." ] ) / macro-expand

toplabel         = ( *alphanum ALPHA *alphanum ) /
                   ( 1*alphanum "-" *( alphanum / "-" ) alphanum )
                   ; LDH rule plus additional TLD restrictions
                   ; (see [RFC3696], Section 2)
alphanum         = ALPHA / DIGIT

explain-string   = *( macro-string / SP )

macro-string     = *( macro-expand / macro-literal )
macro-expand     = ( "%{" macro-letter transformers *delimiter "}" )
                   / "%%" / "%_" / "%-"
macro-literal    = %x21-24 / %x26-7E
                   ; visible characters except "%"
macro-letter     = "s" / "l" / "o" / "d" / "i" / "p" / "h" /
                   "c" / "r" / "t"
transformers     = *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 term arguments:

s = <sender>
l = local-part of <sender>
o = domain of <sender>
d = <domain>
i = <ip>
p = the validated domain name of <ip>
v = the string "in-addr" if <ip> is ipv4, or "ip6" if <ip> is ipv6
h = HELO/EHLO domain

The following macro letters are allowed only in "exp" text:

```
c = SMTP client IP (easily readable format)
r = domain name of host performing the check
t = current timestamp
```

A '%' character not followed by a '{', '%', '-', or '_' character is a syntax error. So

```
-exists:%(ir).sbl.spamhaus.example.org
```

is incorrect and will cause check_host() to return a "PermError". Instead, say

```
-exists:%{ir}.sbl.spamhaus.example.org
```

Optional transformers are the following:

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

If transformers or delimiters are provided, the replacement value for a macro letter is split into parts. After performing any reversal operation and/or removal of left-hand parts, the parts are rejoined using "." and not the original splitting characters.

By default, strings are split on "." (dots). Note that no special treatment is given to leading, trailing, or consecutive delimiters, and so the list of parts may contain empty strings. Older implementations of SPF prohibit trailing dots in domain names, so trailing dots should not be published by domain owners, although they must be accepted by implementations conforming to this document. Macros may specify delimiter characters that are used instead of ".".

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, the value 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 MUST support at least a value of 128, as that is the maximum number of labels in a domain name.

The "s" macro expands to the <sender> argument. It is an E-Mail address with a localpart, an "@" character, and a domain. The "l" macro expands to just the localpart. The "o" macro expands to just the domain part. Note that these values remain the same during recursive and chained evaluations due to "include" and/or "redirect". Note also that if the original <sender> had no localpart, the localpart was set to "postmaster" in initial processing (see Section 4.3).

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

For IPv6 addresses, the "i" macro expands to a 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 <ip>. The procedure for finding the validated domain name is defined in Section 5.5. If the <domain> is present in the list of validated domains, it SHOULD be used. Otherwise, if a subdomain of the <domain> is present, it SHOULD be used. Otherwise, any name from the list may be used. If there are no validated domain names or if a DNS error occurs, the string "unknown" is used.

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 MUA) or if policy restrictions dictate otherwise, the word "unknown" SHOULD be substituted. The domain name may be different from the name found in the MX record that the client MTA used to locate the receiving MTA.

The "t" macro expands to the decimal representation of the approximate number of seconds since the Epoch (Midnight, January 1, 1970, UTC). This is the same value as is returned by the POSIX time() function in most standards-compliant libraries.

When the result of macro expansion is used in a domain name query, if the expanded domain name exceeds 253 characters (the maximum length of a domain name), the left side is truncated to fit, by removing successive domain labels until the total length does not exceed 253 characters.

Uppercased macros expand exactly as their lowercased equivalents, and are then URL escaped. URL escaping must be performed for characters not in the "uric" set, which is defined in [RFC3986].

Note: Care must be taken so that macro expansion for legitimate E-Mail does not exceed the 63-character limit on DNS labels. The localpart of E-Mail addresses, in particular, can have more than 63 characters between dots.

Note: Domains should avoid using the "s", "l", "o", or "h" macros in conjunction with any mechanism directive. Although these macros are powerful and allow per-user records to be published, they severely limit the ability of implementations to cache results of check_host() and they reduce the effectiveness of DNS caches.

Implementations should be aware that if no directive processed during the evaluation of check_host() contains an "s", "l", "o", or "h" macro, then the results of the evaluation can be cached on the basis of <domain> and <ip> alone for as long as the shortest Time To Live (TTL) of all the DNS records involved.

8.2. Expansion Examples

The <sender> is strong-bad@email.example.com.
The IPv4 SMTP client IP is 192.0.2.3.
The IPv6 SMTP client IP is 2001:DB8::CB01.
The PTR domain name of the client IP is mx.example.org.

macro	expansion
%{s}	strong-bad@email.example.com
%{o}	email.example.com
%{d}	email.example.com
%{d4}	email.example.com
%{d3}	email.example.com
%{d2}	example.com
%{d1}	com
%{dr}	com.example.email
%{d2r}	example.email
%{l}	strong-bad
%{l-}	strong.bad
%{lr}	strong-bad
%{lr-}	bad.strong
%{l1r-}	strong

```
macro-string                                                  expansion
-------------------------------------------------------------------------
%{ir}.%{v}._spf.%{d2}            3.2.0.192.in-addr._spf.example.com
%{lr-}.lp._spf.%{d2}                 bad.strong.lp._spf.example.com

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

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

%{d2}.trusted-domains.example.net
                              example.com.trusted-domains.example.net

IPv6:
%{ir}.%{v}._spf.%{d2}                                 1.0.B.C.0.0.0.0.
0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.B.D.0.1.0.0.2.ip6._spf.example.com
```

9. Implications

This section outlines the major implications that adoption of this document will have on various entities involved in Internet E-Mail. It is intended to make clear to the reader where this document knowingly affects the operation of such entities. This section is not a "how-to" manual, or a "best practices" document, and it is not a comprehensive list of what such entities should do in light of this document.

This section is non-normative.

9.1. Sending Domains

Domains that wish to be compliant with this specification will need to determine the list of hosts that they allow to use their domain name in the "HELO" and "MAIL FROM" identities. It is recognized that forming such a list is not just a simple technical exercise, but involves policy decisions with both technical and administrative considerations.

It can be helpful to publish records that include a "tracking exists:" mechanism. By looking at the name server logs, a rough list may then be generated. For example:

```
v=spf1 exists:_h.%{h}._l.%{l}._o.%{o}._i.%{i}._spf.%{d} ?all
```

9.2. Mailing Lists

Mailing lists must be aware of how they re-inject mail that is sent to the list. Mailing lists MUST comply with the requirements in [RFC2821], Section 3.10, and [RFC1123], Section 5.3.6, that say that the reverse-path MUST be changed to be the mailbox of a person or other entity who administers the list. Whereas the reasons for changing the reverse-path are many and long-standing, SPF adds enforcement to this requirement.

In practice, almost all mailing list software in use already complies with this requirement. Mailing lists that do not comply may or may not encounter problems depending on how access to the list is restricted. Such lists that are entirely internal to a domain (only people in the domain can send to or receive from the list) are not affected.

9.3. Forwarding Services and Aliases

Forwarding services take mail that is received at a mailbox and direct it to some external mailbox. At the time of this writing, the near-universal practice of such services is to use the original "MAIL FROM" of a message when re-injecting it for delivery to the external mailbox. [RFC1123] and [RFC2821] describe this action as an "alias" rather than a "mail list". This means that the external mailbox's MTA sees all such mail in a connection from a host of the forwarding service, and so the "MAIL FROM" identity will not, in general, pass authorization.

There are three places that techniques can be used to ameliorate this problem.

1. The beginning, when E-Mail is first sent.

    1. "Neutral" results could be given for IP addresses that may be forwarders, instead of "Fail" results. For example:

        "v=spf1 mx -exists:%{ir}.sbl.spamhaus.example.org ?all"

        This would cause a lookup on an anti-spam DNS blacklist (DNSBL) and cause a result of "Fail" only for E-Mail coming from listed sources. All other E-Mail, including E-Mail sent through forwarders, would receive a "Neutral" result. By checking the DNSBL after the known good sources, problems with incorrect listing on the DNSBL are greatly reduced.

2. The "MAIL FROM" identity could have additional information in the localpart that cryptographically identifies the mail as coming from an authorized source. In this case, such an SPF record could be used:

   "v=spf1 mx exists:%{l}._spf_verify.%{d} -all"

   Then, a specialized DNS server can be set up to serve the _spf_verify subdomain that validates the localpart. Although this requires an extra DNS lookup, this happens only when the E-Mail would otherwise be rejected as not coming from a known good source.

   Note that due to the 63-character limit for domain labels, this approach only works reliably if the localpart signature scheme is guaranteed either to only produce localparts with a maximum of 63 characters or to gracefully handle truncated localparts.

3. Similarly, a specialized DNS server could be set up that will rate-limit the E-Mail coming from unexpected IP addresses.

   "v=spf1 mx exists:%{ir}._spf_rate.%{d} -all"

4. SPF allows the creation of per-user policies for special cases. For example, the following SPF record and appropriate wildcard DNS records can be used:

   "v=spf1 mx redirect=%{l1r+}._at_.%{o}._spf.%{d}"

2. The middle, when E-Mail is forwarded.

1. Forwarding services can solve the problem by rewriting the "MAIL FROM" to be in their own domain. This means that mail bounced from the external mailbox will have to be re-bounced by the forwarding service. Various schemes to do this exist though they vary widely in complexity and resource requirements on the part of the forwarding service.

2. Several popular MTAs can be forced from "alias" semantics to "mailing list" semantics by configuring an additional alias with "owner-" prepended to the original alias name (e.g., an alias of "friends: george@example.com, fred@example.org" would need another alias of the form "owner-friends: localowner").

3. The end, when E-Mail is received.

    1. If the owner of the external mailbox wishes to trust the forwarding service, he can direct the external mailbox's MTA to skip SPF tests when the client host belongs to the forwarding service.

    2. Tests against other identities, such as the "HELO" identity, may be used to override a failed test against the "MAIL FROM" identity.

    3. For larger domains, it may not be possible to have a complete or accurate list of forwarding services used by the owners of the domain's mailboxes. In such cases, whitelists of generally-recognized forwarding services could be employed.

9.4. Mail Services

Service providers that offer mail services to third-party domains, such as sending of bulk mail, may want to adjust their setup in light of the authorization check described in this document. If the "MAIL FROM" identity used for such E-Mail uses the domain of the service provider, then the provider needs only to ensure that its sending host is authorized by its own SPF record, if any.

If the "MAIL FROM" identity does not use the mail service provider's domain, then extra care must be taken. The SPF record format has several options for the third-party domain to authorize the service provider's MTAs to send mail on its behalf. For mail service providers, such as ISPs, that have a wide variety of customers using the same MTA, steps should be taken to prevent cross-customer forgery (see Section 10.4).

9.5. MTA Relays

The authorization check generally precludes the use of arbitrary MTA relays between sender and receiver of an E-Mail message.

Within an organization, MTA relays can be effectively deployed. However, for purposes of this document, such relays are effectively transparent. The SPF authorization check is a check between border MTAs of different domains.

For mail senders, this means that published SPF records must authorize any MTAs that actually send across the Internet. Usually, these are just the border MTAs as internal MTAs simply forward mail to these MTAs for delivery.

Mail receivers will generally want to perform the authorization check at the border MTAs, specifically including all secondary MXs. This allows mail that fails to be rejected during the SMTP session rather than bounced. Internal MTAs then do not perform the authorization test. To perform the authorization test other than at the border, the host that first transferred the message to the organization must be determined, which can be difficult to extract from the message header. Testing other than at the border is not recommended.

10. Security Considerations

10.1. Processing Limits

As with most aspects of E-Mail, there are a number of ways that malicious parties could use the protocol as an avenue for a Denial-of-Service (DoS) attack. The processing limits outlined here are designed to prevent attacks such as the following:

- A malicious party could create an SPF record with many references to a victim's domain and send many E-Mails to different SPF clients; those SPF clients would then create a DoS attack. In effect, the SPF clients are being used to amplify the attacker's bandwidth by using fewer bytes in the SMTP session than are used by the DNS queries. Using SPF clients also allows the attacker to hide the true source of the attack.

- Whereas implementations of check_host() are supposed to limit the number of DNS lookups, malicious domains could publish records that exceed these limits in an attempt to waste computation effort at their targets when they send them mail. Malicious domains could also design SPF records that cause particular implementations to use excessive memory or CPU usage, or to trigger bugs.

- Malicious parties could send a large volume of mail purporting to come from the intended target to a wide variety of legitimate mail hosts. These legitimate machines would then present a DNS load on the target as they fetched the relevant records.

Of these, the case of a third party referenced in the SPF record is the easiest for a DoS attack to effectively exploit. As a result, limits that may seem reasonable for an individual mail server can still allow an unreasonable amount of bandwidth amplification. Therefore, the processing limits need to be quite low.

SPF implementations MUST limit the number of mechanisms and modifiers that do DNS lookups to at most 10 per SPF check, including any lookups caused by the use of the "include" mechanism or the

"redirect" modifier. If this number is exceeded during a check, a PermError MUST be returned. The "include", "a", "mx", "ptr", and "exists" mechanisms as well as the "redirect" modifier do count against this limit. The "all", "ip4", and "ip6" mechanisms do not require DNS lookups and therefore do not count against this limit. The "exp" modifier does not count against this limit because the DNS lookup to fetch the explanation string occurs after the SPF record has been evaluated.

When evaluating the "mx" and "ptr" mechanisms, or the %{p} macro, there MUST be a limit of no more than 10 MX or PTR RRs looked up and checked.

SPF implementations SHOULD limit the total amount of data obtained from the DNS queries. For example, when DNS over TCP or EDNS0 are available, there may need to be an explicit limit to how much data will be accepted to prevent excessive bandwidth usage or memory usage and DoS attacks.

MTAs or other processors MAY also impose a limit on the maximum amount of elapsed time to evaluate check_host(). Such a limit SHOULD allow at least 20 seconds. If such a limit is exceeded, the result of authorization SHOULD be "TempError".

Domains publishing records SHOULD try to keep the number of "include" mechanisms and chained "redirect" modifiers to a minimum. Domains SHOULD also try to minimize the amount of other DNS information needed to evaluate a record. This can be done by choosing directives that require less DNS information and placing lower-cost mechanisms earlier in the SPF record.

For example, consider a domain set up as follows:

```
example.com.      IN MX   10 mx.example.com.
mx.example.com.   IN A    192.0.2.1
a.example.com.    IN TXT  "v=spf1 mx:example.com -all"
b.example.com.    IN TXT  "v=spf1 a:mx.example.com -all"
c.example.com.    IN TXT  "v=spf1 ip4:192.0.2.1 -all"
```

Evaluating check_host() for the domain "a.example.com" requires the MX records for "example.com", and then the A records for the listed hosts. Evaluating for "b.example.com" requires only the A records. Evaluating for "c.example.com" requires none.

However, there may be administrative considerations: using "a" over "ip4" allows hosts to be renumbered easily. Using "mx" over "a" allows the set of mail hosts to be changed easily.

10.2. SPF-Authorized E-Mail May Contain Other False Identities

The "MAIL FROM" and "HELO" identity authorizations must not be construed to provide more assurance than they do. It is entirely possible for a malicious sender to inject a message using his own domain in the identities used by SPF, to have that domain's SPF record authorize the sending host, and yet the message can easily list other identities in its header. Unless the user or the MUA takes care to note that the authorized identity does not match the other more commonly-presented identities (such as the From: header field), the user may be lulled into a false sense of security.

10.3. Spoofed DNS and IP Data

There are two aspects of this protocol that malicious parties could exploit to undermine the validity of the check_host() function:

- The evaluation of check_host() relies heavily on DNS. A malicious attacker could attack the DNS infrastructure and cause check_host() to see spoofed DNS data, and then return incorrect results. This could include returning "Pass" for an <ip> value where the actual domain's record would evaluate to "Fail". See [RFC3833] for a description of DNS weaknesses.

- The client IP address, <ip>, is assumed to be correct. A malicious attacker could spoof TCP sequence numbers to make mail appear to come from a permitted host for a domain that the attacker is impersonating.

10.4. Cross-User Forgery

By definition, SPF policies just map domain names to sets of authorized MTAs, not whole E-Mail addresses to sets of authorized users. Although the "l" macro (Section 8) provides a limited way to define individual sets of authorized MTAs for specific E-Mail addresses, it is generally impossible to verify, through SPF, the use of specific E-Mail addresses by individual users of the same MTA.

It is up to mail services and their MTAs to directly prevent cross-user forgery: based on SMTP AUTH ([RFC2554]), users should be restricted to using only those E-Mail addresses that are actually under their control (see [RFC4409], Section 6.1). Another means to verify the identity of individual users is message cryptography such as PGP ([RFC2440]) or S/MIME ([RFC3851]).

10.5. Untrusted Information Sources

SPF uses information supplied by third parties, such as the "HELO" domain name, the "MAIL FROM" address, and SPF records. This information is then passed to the receiver in the Received-SPF: trace fields and possibly returned to the client MTA in the form of an SMTP rejection message. This information must be checked for invalid characters and excessively long lines.

When the authorization check fails, an explanation string may be included in the reject response. Both the sender and the rejecting receiver need to be aware that the explanation was determined by the publisher of the SPF record checked and, in general, not the receiver. The explanation may contain malicious URLs, or it may be offensive or misleading.

This is probably less of a concern than it may initially seem since such messages are returned to the sender, and the explanation strings come from the sender policy published by the domain in the identity claimed by that very sender. As long as the DSN is not redirected to someone other than the actual sender, the only people who see malicious explanation strings are people whose messages claim to be from domains that publish such strings in their SPF records. In practice, DSNs can be misdirected, such as when an MTA accepts an E-Mail and then later generates a DSN to a forged address, or when an E-Mail forwarder does not direct the DSN back to the original sender.

10.6. Privacy Exposure

Checking SPF records causes DNS queries to be sent to the domain owner. These DNS queries, especially if they are caused by the "exists" mechanism, can contain information about who is sending E-Mail and likely to which MTA the E-Mail is being sent. This can introduce some privacy concerns, which may be more or less of an issue depending on local laws and the relationship between the domain owner and the person sending the E-Mail.

11. Contributors and Acknowledgements

This document is largely based on the work of Meng Weng Wong and Mark Lentczner. Although, as this section acknowledges, many people have contributed to this document, a very large portion of the writing and editing are due to Meng and Mark.

This design owes a debt of parentage to [RMX] by Hadmut Danisch and to [DMP] by Gordon Fecyk. The idea of using a DNS record to check the legitimacy of an E-Mail address traces its ancestry further back through messages on the namedroppers mailing list by Paul Vixie

[Vixie] (based on suggestion by Jim Miller) and by David Green [Green].

Philip Gladstone contributed the concept of macros to the specification, multiplying the expressiveness of the language and making per-user and per-IP lookups possible.

The authors would also like to thank the literally hundreds of individuals who have participated in the development of this design. They are far too numerous to name, but they include the following:

The folks on the spf-discuss mailing list.
The folks on the SPAM-L mailing list.
The folks on the IRTF ASRG mailing list.
The folks on the IETF MARID mailing list.
The folks on #perl.

12. IANA Considerations

12.1. The SPF DNS Record Type

The IANA has assigned a new Resource Record Type and Qtype from the DNS Parameters Registry for the SPF RR type with code 99.

12.2. The Received-SPF Mail Header Field

Per [RFC3864], the "Received-SPF:" header field is added to the IANA Permanent Message Header Field Registry. The following is the registration template:

Header field name: Received-SPF
Applicable protocol: mail ([RFC2822])
Status: Experimental
Author/Change controller: IETF
Specification document(s): RFC 4408
Related information:
Requesting SPF Council review of any proposed changes and additions to this field are recommended. For information about the SPF Council see http://www.openspf.org/Council

13. References

13.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

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

[RFC2821] Klensin, J., "Simple Mail Transfer Protocol", RFC 2821, April 2001.

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

[RFC3464] Moore, K. and G. Vaudreuil, "An Extensible Message Format for Delivery Status Notifications", RFC 3464, January 2003.

[RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[US-ASCII] American National Standards Institute (formerly United States of America Standards Institute), "USA Code for Information Interchange, X3.4", 1968.

ANSI X3.4-1968 has been replaced by newer versions with slight modifications, but the 1968 version remains definitive for the Internet.

13.2 Informative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1983] Malkin, G., "Internet Users' Glossary", RFC 1983, August 1996.

[RFC2440] Callas, J., Donnerhacke, L., Finney, H., and R. Thayer, "OpenPGP Message Format", RFC 2440, November 1998.

[RFC2554] Myers, J., "SMTP Service Extension for Authentication", RFC 2554, March 1999.

[RFC3696] Klensin, J., "Application Techniques for Checking and Transformation of Names", RFC 3696, February 2004.

[RFC3833] Atkins, D. and R. Austein, "Threat Analysis of the Domain Name System (DNS)", RFC 3833, August 2004.

[RFC3851] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004.

[RFC4409] Gellens, R. and J. Klensin, "Message Submission for Mail", RFC 4409, April 2006.

[RMX] Danish, H., "The RMX DNS RR Type for light weight sender authentication", Work In Progress

[DMP] Fecyk, G., "Designated Mailers Protocol", Work In Progress

[Vixie] Vixie, P., "Repudiating MAIL FROM", 2002.

[Green] Green, D., "Domain-Authorized SMTP Mail", 2002.

Appendix A. Collected ABNF

This section is normative and any discrepancies with the ABNF fragments in the preceding text are to be resolved in favor of this grammar.

See [RFC4234] for ABNF notation. Please note that as per this ABNF definition, literal text strings (those in quotes) are case-insensitive. Hence, "mx" matches "mx", "MX", "mX", and "Mx".

```
record           = version terms *SP
version          = "v=spf1"

terms            = *( 1*SP ( directive / modifier ) )

directive        = [ qualifier ] mechanism
qualifier        = "+" / "-" / "?" / "~"
mechanism        = ( all / include
                   / A / MX / PTR / IP4 / IP6 / exists )

all              = "all"
include          = "include"  ":" domain-spec
A                = "a"      [ ":" domain-spec ] [ dual-cidr-length ]
MX               = "mx"     [ ":" domain-spec ] [ dual-cidr-length ]
PTR              = "ptr"    [ ":" domain-spec ]
IP4              = "ip4"      ":" ip4-network   [ ip4-cidr-length ]
IP6              = "ip6"      ":" ip6-network   [ ip6-cidr-length ]
exists           = "exists"   ":" domain-spec

modifier         = redirect / explanation / unknown-modifier
redirect         = "redirect" "=" domain-spec
explanation      = "exp" "=" domain-spec
unknown-modifier = name "=" macro-string

ip4-cidr-length  = "/" 1*DIGIT
ip6-cidr-length  = "/" 1*DIGIT
dual-cidr-length = [ ip4-cidr-length ] [ "/" ip6-cidr-length ]

ip4-network      = qnum "." qnum "." qnum "." qnum
qnum             = DIGIT                 ; 0-9
                   / %x31-39 DIGIT       ; 10-99
                   / "1" 2DIGIT          ; 100-199
                   / "2" %x30-34 DIGIT   ; 200-249
                   / "25" %x30-35        ; 250-255
          ; conventional dotted quad notation.  e.g., 192.0.2.0
ip6-network      = <as per [RFC 3513], section 2.2>
          ; e.g., 2001:DB8::CD30
```

```
   domain-spec      = macro-string domain-end
   domain-end       = ( "." toplabel [ "." ] ) / macro-expand
   toplabel         = ( *alphanum ALPHA *alphanum ) /
                      ( 1*alphanum "-" *( alphanum / "-" ) alphanum )
                      ; LDH rule plus additional TLD restrictions
                      ; (see [RFC3696], Section 2)

   alphanum         = ALPHA / DIGIT

   explain-string   = *( macro-string / SP )

   macro-string     = *( macro-expand / macro-literal )
   macro-expand     = ( "%{" macro-letter transformers *delimiter "}" )
                      / "%%" / "%_" / "%-"
   macro-literal    = %x21-24 / %x26-7E
                      ; visible characters except "%"
   macro-letter     = "s" / "l" / "o" / "d" / "i" / "p" / "h" /
                      "c" / "r" / "t"
   transformers     = *DIGIT [ "r" ]
   delimiter        = "." / "-" / "+" / "," / "/" / "_" / "="

   name             = ALPHA *( ALPHA / DIGIT / "-" / "_" / "." )

   header-field     = "Received-SPF:" [CFWS] result FWS [comment FWS]
                      [ key-value-list ] CRLF

   result           = "Pass" / "Fail" / "SoftFail" / "Neutral" /
                      "None" / "TempError" / "PermError"

   key-value-list   = key-value-pair *( ";" [CFWS] key-value-pair )
                      [";"]

   key-value-pair   = key [CFWS] "=" ( dot-atom / quoted-string )

   key              = "client-ip" / "envelope-from" / "helo" /
                      "problem" / "receiver" / "identity" /
                       mechanism / "x-" name / name

   identity         = "mailfrom"   ; for the "MAIL FROM" identity
                      / "helo"     ; for the "HELO" identity
                      / name       ; other identities

   dot-atom         = <unquoted word as per [RFC2822]>
   quoted-string    = <quoted string as per [RFC2822]>
   comment          = <comment string as per [RFC2822]>
   CFWS             = <comment or folding white space as per [RFC2822]>
   FWS              = <folding white space as per [RFC2822]>
   CRLF             = <standard end-of-line token as per [RFC2822]>
```

Appendix B. Extended Examples

These examples are based on the following DNS setup:

```
; A domain with two mail servers, two hosts
; and two servers at the domain name
$ORIGIN example.com.
@           MX  10 mail-a
            MX  20 mail-b
            A   192.0.2.10
            A   192.0.2.11
amy         A   192.0.2.65
bob         A   192.0.2.66
mail-a      A   192.0.2.129
mail-b      A   192.0.2.130
www         CNAME example.com.

; A related domain
$ORIGIN example.org.
@           MX  10 mail-c
mail-c      A   192.0.2.140

; The reverse IP for those addresses
$ORIGIN 2.0.192.in-addr.arpa.
10          PTR example.com.
11          PTR example.com.
65          PTR amy.example.com.
66          PTR bob.example.com.
129         PTR mail-a.example.com.
130         PTR mail-b.example.com.
140         PTR mail-c.example.org.

; A rogue reverse IP domain that claims to be
; something it's not
$ORIGIN 0.0.10.in-addr.arpa.
4           PTR bob.example.com.
```

B.1. Simple Examples

These examples show various possible published records for example.com and which values if <ip> would cause check_host() to return "Pass". Note that <domain> is "example.com".

```
v=spf1 +all
   -- any <ip> passes

v=spf1 a -all
   -- hosts 192.0.2.10 and 192.0.2.11 pass
```

```
v=spf1 a:example.org -all
   -- no sending hosts pass since example.org has no A records

v=spf1 mx -all
   -- sending hosts 192.0.2.129 and 192.0.2.130 pass

v=spf1 mx:example.org -all
   -- sending host 192.0.2.140 passes

v=spf1 mx mx:example.org -all
   -- sending hosts 192.0.2.129, 192.0.2.130, and 192.0.2.140 pass

v=spf1 mx/30 mx:example.org/30 -all
   -- any sending host in 192.0.2.128/30 or 192.0.2.140/30 passes

v=spf1 ptr -all
   -- sending host 192.0.2.65 passes (reverse DNS is valid and is in
      example.com)
   -- sending host 192.0.2.140 fails (reverse DNS is valid, but not
      in example.com)
   -- sending host 10.0.0.4 fails (reverse IP is not valid)

v=spf1 ip4:192.0.2.128/28 -all
   -- sending host 192.0.2.65 fails
   -- sending host 192.0.2.129 passes
```

B.2. Multiple Domain Example

These examples show the effect of related records:

```
   example.org: "v=spf1 include:example.com include:example.net -all"
```

This record would be used if mail from example.org actually came through servers at example.com and example.net. Example.org's designated servers are the union of example.com's and example.net's designated servers.

```
   la.example.org: "v=spf1 redirect=example.org"
   ny.example.org: "v=spf1 redirect=example.org"
   sf.example.org: "v=spf1 redirect=example.org"
```

These records allow a set of domains that all use the same mail system to make use of that mail system's record. In this way, only the mail system's record needs to be updated when the mail setup changes. These domains' records never have to change.

B.3. DNSBL Style Example

Imagine that, in addition to the domain records listed above, there are these:

```
$ORIGIN _spf.example.com.  mary.mobile-users                    A
127.0.0.2 fred.mobile-users                    A 127.0.0.2
15.15.168.192.joel.remote-users     A 127.0.0.2
16.15.168.192.joel.remote-users     A 127.0.0.2
```

The following records describe users at example.com who mail from arbitrary servers, or who mail from personal servers.

example.com:

```
v=spf1 mx
       include:mobile-users._spf.%{d}
       include:remote-users._spf.%{d}
       -all
```

mobile-users._spf.example.com:

```
v=spf1 exists:%{l1r+}.%{d}
```

remote-users._spf.example.com:

```
v=spf1 exists:%{ir}.%{l1r+}.%{d}
```

B.4. Multiple Requirements Example

Say that your sender policy requires both that the IP address is within a certain range and that the reverse DNS for the IP matches. This can be done several ways, including the following:

```
example.com.           SPF  ( "v=spf1 "
                              "-include:ip4._spf.%{d} "
                              "-include:ptr._spf.%{d} "
                              "+all" )
ip4._spf.example.com.  SPF  "v=spf1 -ip4:192.0.2.0/24 +all"
ptr._spf.example.com.  SPF  "v=spf1 -ptr +all"
```

This example shows how the "-include" mechanism can be useful, how an SPF record that ends in "+all" can be very restrictive, and the use of De Morgan’s Law.

Authors' Addresses

Meng Weng Wong
Singapore

EMail: mengwong+spf@pobox.com

Wayne Schlitt
4615 Meredeth #9
Lincoln Nebraska, NE 68506
United States of America

EMail: wayne@schlitt.net
URI: http://www.schlitt.net/spf/

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).