Base Spec Errata

IETF 73

Tony Hansen

http://www.rfc-editor.org/errata_search.php?rfc=4871&rec_status=15&presentation=table

(1376)	3.4.3/.4	Tony Hansen	2008-03-21
(1377)	3.4.4	Tony Hansen	2008-03-21
(1378)	3.5	Tony Hansen	2008-03-21
(1379)	3.5	Tony Hansen	2008-03-21
(1380)	3.5	Tony Hansen	2008-03-21
(1381)	3.5/3.6.1	Tony Hansen	2008-03-21
(1382)	3.6.1	Tony Hansen	2008-03-21
(1383)	3,6.1	Tony Hansen	2008-03-21
(1384)	4.3.4	Tony Hansen	2008-03-22
(1385)	3.6.1	Murray S. Kucherawy	2008-03-23
(1386)	3.5	Mark Delany	2008-03-24
(1461)	3.5	Frank Ellermann	2008-07-04
(1487)	3.6.1	Murray S. Kucherawy	2008-08-14
(1532)	3.6.1	Tony Hansen	2008-09-30
(1596)	2.4/3.7	Tony Hansen	2008-11-17

Errata 1383

- Errata says to add examples to g= description

  Section 3,6.1 says:

  g= Granularity of the key (plain-text; OPTIONAL, default is "*"). .... Wildcarding allows matching for addresses such as "user+*" or "*-offer". An empty "g=" value never matches any addresses.

  It should say:

  g= Granularity of the key (plain-text; OPTIONAL, default is "*"). .... Wildcarding allows matching for addresses such as "user+*", "*-offer", "foo*bar", "ex*am*ple" or "*exam*ple". An empty "g=" value never matches any addresses.
- This was prompted by someone at the interop noting that some people had not coded for the example of a * in the middle.
- The text says “the *single*, optional ‘*’ character”
- The ABNF allows single * at beginning, middle or end.

Choices

- Choice 1: The errata is wrong with respect to multiple wildcards. However, it would still be useful to add an example of foo*bar.
- Choice 2: One implementer said:
  - we … allow “*” anywhere, and more than once, in the “g=“ value.

Errata 1378

- Is “a=” required or optional?
  - §3.3 says that rsa-sha256 is the default if no algorithm is specified
  - §3.5 says “a=” is REQUIRED
- Need to pick one

- One response:
  - We currently require a= when verifying, but are willing to change

Errata 1532

- There should be a note added somewhere to section 3.6.1 saying that if a v= is not found at the beginning of the DKIM key record, the DNS key record should be interpreted as for DomainKeys and described in RFC 4870. In addition, a note should be added about the difference in the interpretation of an empty "g=", which is the only incompatible tag.

Discussion

- Not right direction:
  - people should not be using empty g= tags in DK keys
  - Makes all existing verifiers non-compliant
  - Compatibility note for DK recommending against using g=;
- Do nothing
- DK people *are* adding DKIM signatures, without updating keys with g=; in them
  - And does not need to be a MUST

Suggestion for updated text

- Compatibility Note for DomainKeys
  - The definition given here for the key record is upwardly compatible with what is used for DomainKeys, with the exception of the "g=" value. In DomainKeys, a key record empty "g=" value is equivalent to "g=*", while DKIM treats that value as matching nothing. The value "g=*" means the same in both DomainKeys and DKIM.
  - DomainKeys deployers are encouraged to at least switch their key records to using the equivalent "g=*" value, which works equivalently for both DomainKeys and DKIM.
  - A DKIM implementation MAY choose to use the lack of a v= value at the beginning of the key record as an indicator that the key record is a DomainKeys key record, and interpret an empty "g=" value as if it were written "g=*".

Errata 1596

- When calculating hash, what to do with WSP in bh=
  - bh=$^{\text{WSP}}$a$^{\text{WSP}}$b$^{\text{WSP}}$c$^{\text{WSP}}$;
  - bh=$^{\text{WSP}}$a$^{\text{WSP}}$b$^{\text{WSP}}$c$^{\text{WSP}}$$_{\text{<end-of-header>}}$

§ 3.5

- § 3.5 “b=“ deletion description talks about adding FWS “in” the value, but not “before” or “after”.
  - `b= The signature data (base64; REQUIRED). Whitespace is ignored` *`in this value`* `and MUST be ignored when reassembling the original signature. In particular, the signing process can safely insert FWS` *`in this value`* `in arbitrary places to conform to line-length limits.`

§3.2

- Notice that the §3.2 definition of tag-val

  ```
  tag-spec = [FWS] tag-name [FWS] "=" [FWS] tag-
  value [FWS]
  tag-value = [ tval 0*( 1*(WSP / FWS) tval ) ]
  ; WSP and FWS prohibited at beginning and end
  ```

  explicitly does *not* include either the FWS before its value or after.

  And the text in section 3.2 explicitly says that the surrounding WSP is not part of the value.

Section 3.5 grammar for sig-b-tag-data

- And notice that the section 3.5 grammar around sig-b-tag-data

```
sig-b-tag = %x62 [FWS] "=" [FWS] sig-b-tag-data
sig-b-tag-data = base64string
```

  explicitly mentions FWS as being separate from the data.

Conclusions from those

- By the above definitions, tag-val and sig-b-tag-data explicitly do *not* include the FWS either before or after it.

Base64string

- However, the definition of base64string

  ```
  base64string = 1*(ALPHA / DIGIT / "+" / "/" / [FWS])
  [ "=" [FWS] [ "=" [FWS] ] ]
  ```

  tosses FWS in to its production. So it is ambiguous from the grammar whether the leading/trailing FWS is part of sig-b-tag-data or part of base64string. (This grammar ambiguity is in *all* of the uses of base64string in sections 3.5 and 3.6.1.)

Back to section 3.5

- In addition, the text in the section 3.5 b= description certainly implies that white space before and after the hash should not affect the verification.

Back to the problem

- So by these, “with the value of the 'b=' tag deleted” could mean
  1. everything after the "=" which ***includes*** the leading/trailing white space,
  2. the *tag-value* grammar production which ***excludes*** leading/trailing white space, or
  3. the *sig-b-tag-data* grammar production that ***may or may not include*** leading/trailing white space.

Suggestion in Errata

- Add text "(including all surrounding whitespace)" to the description of deleting the b= value.
- 3.7. Computing the Message Hashes
- 2. The DKIM-Signature header field that exists (verifying) or will be inserted (signing) in the message, with the value of the "b=" tag *(including all surrounding whitespace)* deleted (i.e., treated as the empty string), canonicalized using the header canonicalization algorithm specified in the "c=" tag, and without a trailing CRLF.
- Fix the ambiguity in the base64string grammar to remove leading and trailing FWS:

```
ALPHADIGITPS = (ALPHA / DIGIT / "+" / "/")
base64string = ALPHADIGITPS *([FWS] ALPHADIGITPS)
                 [ [FWS] "=" [ [FWS] "=" ] ]
```