"openssl ciphers RSA" omits many ciphers that use RSA authentication #2141

New issue

Closed triska opened this issue on 24 Dec 2016 · 8 comments

triska commented on 24 Dec 2016 — Contributor

The documentation of `openssl ciphers` states:

```
       kRSA, aRSA, RSA
           Cipher suites using RSA key exchange, authentication or either
           respectively.
```

However, `openssl ciphers RSA` shows significantly *fewer* ciphers than `openssl ciphers aRSA`.

In particular, I get with OpenSSL 1.1.0c:

```
$ openssl ciphers RSA
AES256-GCM-SHA384:AES256-CCM8:AES256-CCM:AES128-GCM-SHA256:AES128-CCM8:AES128-CCM:AES256-SHA256:CAMELLIA256-SHA256:AES128-
```

And:

```
$ openssl ciphers aRSA
ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES256
```

Since the documentation states "*or either*", I expect these ciphers to also occur when using `RSA`.

The difference currently comprises the following ciphers:

```
ECDHE-RSA-AES256-GCM-SHA384
DHE-RSA-AES256-GCM-SHA384
ECDHE-RSA-CHACHA20-POLY1305
DHE-RSA-CHACHA20-POLY1305
DHE-RSA-AES256-CCM8
DHE-RSA-AES256-CCM
ECDHE-RSA-AES128-GCM-SHA256
DHE-RSA-AES128-GCM-SHA256
DHE-RSA-AES128-CCM8
DHE-RSA-AES128-CCM
ECDHE-RSA-AES256-SHA384
DHE-RSA-AES256-SHA256
ECDHE-RSA-CAMELLIA256-SHA384
DHE-RSA-CAMELLIA256-SHA256
ECDHE-RSA-AES128-SHA256
DHE-RSA-AES128-SHA256
ECDHE-RSA-CAMELLIA128-SHA256
DHE-RSA-CAMELLIA128-SHA256
ECDHE-RSA-AES256-SHA
DHE-RSA-AES256-SHA
DHE-RSA-CAMELLIA256-SHA
ECDHE-RSA-AES128-SHA
DHE-RSA-AES128-SHA
DHE-RSA-SEED-SHA
DHE-RSA-CAMELLIA128-SHA
ECDHE-RSA-NULL-SHA
RSA-PSK-AES256-GCM-SHA384
RSA-PSK-CHACHA20-POLY1305
RSA-PSK-AES128-GCM-SHA256
SRP-RSA-AES-256-CBC-SHA
RSA-PSK-AES256-CBC-SHA384
RSA-PSK-AES256-CBC-SHA
RSA-PSK-CAMELLIA256-SHA384
SRP-RSA-AES-128-CBC-SHA
RSA-PSK-AES128-CBC-SHA256
RSA-PSK-AES128-CBC-SHA
RSA-PSK-CAMELLIA128-SHA256
RSA-PSK-NULL-SHA384
RSA-PSK-NULL-SHA256
RSA-PSK-NULL-SHA
```

vdukhovni commented on 24 Dec 2016

While the documentation says "or" (and perhaps that's the more desirable meaning) the implementation is "and". And since kRSA requires aRSA, you end up with RSA == kRSA. It seems that this equivalence goes way back to the root commit in the OpenSSL git repository:

```
commit d02b48c63a58ea4367a0e905979f140b7d090f86
Author: Ralf S. Engelschall <rse@openssl.org>
Date:   Mon Dec 21 10:52:47 1998 +0000

    Import of old SSLeay release: SSLeay 0.8.1b
```

So even if "or" is the more useful meaning, at this point it probably makes more sense to update the documentation to match reality.

vdukhovni commented on 24 Dec 2016

Note that of necessity we have `kRSA` ⊆ `aRSA`, so "and" is just `kRSA` while "or" is just `aRSA`. So the issue boils down to whether `RSA` should denote `kRSA` or `aRSA`. There's likely more risk in making an incompatible change than benefit. You can just use `aRSA` to block all use of RSA. I'm still more favourably inclined towards an update of the documentation.

richsalz commented on 24 Dec 2016 — Contributor

We can't rely on 'nobody spoke up so nobody cares.' Few people read the documentation. It is far more likely that users are just relying on the current behavior. We can't change the semantics; we should just fix the docs.

t-j-h commented on 24 Dec 2016 — Member

The documentation is simply wrong and should be fixed - rarely is the documentation considered more correct than the actual code.

And if you read the "original" documentation for the "ciphers" command it actually states that RSA==kRSA (i.e. kRSA, RSA was noted as "cipher suites using the RSA key exchange").

The change to have kRSA, aRSA, RSA and state "or either respectively" came in 2012 in commit `ffa4579`

This is clearly a documentation error - the documentation used to match the code and was updated in a way that doesn't match the code so the documentation should be fixed.

triska commented on 25 Dec 2016 — Contributor, Author

> of necessity we have kRSA ⊆ aRSA

@vdukhovni: As far as I understand, that's not *necessarily* the case (even if it's the case currently): There could be a cipher that uses ephemeral RSA keys for key exchange, and a completely different algorithm for authentication. A provision for this appeared for example in TLS 1.0 (section 7.4.3), and has fallen out of the specification in later versions.

davidben commented on 25 Dec 2016 — Contributor

That turned out to be kind of an unfortunate decision and, given upcoming TLS developments, is not likely to resurface. :-)

triska commented on 25 Dec 2016 — Contributor, Author

Definitely not with such an extreme limitation (I hope).

paulidale mentioned this issue on 2 Mar 2017

Document that RSA == kRSA for cipher strings #2821 — Closed

mattcaswell commented on 2 May 2017 — Member

Documentation was fixed. `cc92ac7` on 1.1.0, `f2bcff4` on master. Closing.

mattcaswell closed this on 2 May 2017

mattcaswell mentioned this issue on 3 May 2017

[rt.openssl.org #3953] Bug: !RSA does not exclude aRSA #2431 — Closed

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked pull requests
Successfully merging a pull request may close this issue.
None yet

6 participants