



License-compliant TLS stack for Apertis targets

Contents

2	Goals and requirements	3
3	TLS stack	3
4	GnuTLS	3
5	OpenSSL	4
6	NSS	5
7	Approach	5
8	Summary	5
9	TLS stack	5
10	Appendix	6
11	Previous situation	6
12	Issues	7
13	GnuTLS	7
14	OpenSSL	7
15	Alternative SSL solutions	9
16	BoringSSL	9
17	LibreSSL	10
18	mbed TLS	10
19	MesaLink	10
20	NSS	11
21	wolfSSL	11
22	Possible solutions	11
23	Single stack solutions	11
24	Multi-stack solutions	12

The Apertis distribution provides both a development environment for electronic devices as well as a software stack to be used on them. In line with this goal, the Apertis project strives to provide software components that, where there is intent that they form part of the software stack on the devices themselves, are free from licensing constraints that may make it unsuitable in certain use cases. An example is software licensed under the terms of the GNU [GPL-3](https://www.gnu.org/licenses/gpl-3.0.en.html)¹ (General Public License) or [LGPL-3](https://www.gnu.org/licenses/lgpl-3.0.en.html)² (Lesser General Public License) which are known to present a problem as they sometimes [conflict with regulatory requirements](https://apertis-website-0b3586.pages.apertis.org/policies/license-expectations/#licensing-constraints)³ and thus Apertis will take measures to avoid such packages being provided as part of the “target” [package repositories](https://apertis-website-0b3586.pages.apertis.org/policies/license-expectations/#apertis-repository-component-specific-rules)⁴.

¹<https://www.gnu.org/licenses/gpl-3.0.en.html>

²<https://www.gnu.org/licenses/lgpl-3.0.en.html>

³<https://apertis-website-0b3586.pages.apertis.org/policies/license-expectations/#licensing-constraints>

⁴<https://apertis-website-0b3586.pages.apertis.org/policies/license-expectations/#apertis-repository-component-specific-rules>

35 Goals and requirements

36 The goal here is to provide TLS functionality not just for the packages contained
37 within its own repositories, but to support applications added by those utilizing
38 Apertis as well.

- 39 • **Requirement:** TLS implementation does not require code covered by
40 licenses that are incompatible with the target repositories rules
- 41 • **Requirement:** TLS implementation is licensed under terms that does
42 not preclude its use from existing target applications
- 43 • **Requirement:** TLS implementation is licensed under terms that does
44 not preclude its use from users proprietary applications

45 Given the security sensitive nature of the TLS stack, utilizing unmaintained soft-
46 ware here would be best avoided. Putting maintenance aside, these versions of
47 their respective TLS implementations may not be gaining support for any new
48 ciphers and TLS protocol versions, which will severely limit their usefulness as
49 time progresses. As well as not gaining newer protocol versions, the libraries
50 may not be updated to reflect the frequently changing [recommendations regard-](#)
51 [ing minimal protocol versions](#)⁵ that should be supported, which may result in
52 issues when attempting to access sites following the “Modern” recommendation.
53 Additionally, it is likely that newer versions of the packages utilizing these TLS
54 implementations will begin to require functionality added to newer versions of
55 the TLS libraries thus reducing the ability of Apertis to upgrade to these too.

56 TLS stack

57 In order to have up to date libraries, specially TLS ones which very important
58 for security reasons Apertis based them on Debian as covered in the [Apertis](#)
59 [Release Flow](#)⁶ which present the following issues for Apertis

60 GnuTLS

61 Whilst GnuTLS is licensed under the [LGPL-2.1](#)⁷, it uses [Nettle](#)⁸ and [GMP](#)⁹.
62 Newer versions of both of these dependencies are now licensed as dual GPL-2
63 or LGPL-3, rather than [LGPL-2.1](#).

64 To avoid including GnuTLS under LGPL-3 terms since it is against [Apertis](#)
65 [license expectations](#)¹⁰, Apertis would need to utilize it under the GPL-2 terms.
66 This would result in the binary GnuTLS library effectively being used under

⁵https://wiki.mozilla.org/Security/Server_Side_TLS

⁶<https://apertis-website-0b3586.pages.apertis.org/policies/release-flow/#apertis-release-flow>

⁷<https://www.gnu.org/licenses/old-licenses/lgpl-2.1.en.html>

⁸<https://www.lysator.liu.se/~nisse/nettle/nettle.html>

⁹<https://gmplib.org/>

¹⁰<https://apertis-website-0b3586.pages.apertis.org/policies/license-expectations/>

67 the terms of the GPL-2 rather than LGPL-2.1. This would restrict Apertis
68 users from using this Apertis provided TLS implementation either directly or
69 indirectly from any non-GPL-2 compatible applications they wish to integrate
70 into their systems, for example in proprietary applications, where it would have
71 the effect of requiring the app to also be GPL-2 licensed.

72 In such a scenario, a newer GnuTLS library could be allowed by accepting its
73 dependencies under the GPL-2 license and restricting its use to places where
74 this license wouldn't be problematic, such as existing GPL-2 software. As the
75 existing applications written exclusively to use GnuTLS are GPL-2 or tolerant
76 of GPL-2, this is viable.

77 OpenSSL

78 The OpenSSL version 1.1 available in v2022 and v2023 is licensed under a custom
79 GPL-incompatible license. OpenSSL 3.0 available from v2024dev2 is licensed
80 under the [Apache 2.0](https://www.apache.org/licenses/LICENSE-2.0)¹¹ license, which is compatible with the GPL-3, but not
81 GPL-2. This means that GPL-2 tools like `systemd` cannot use the newer versions
82 of OpenSSL without effectively becoming GPL-3 licensed or through these up-
83 stream projects applying a license exceptions (for example as [OpenVPN](https://openvpn.net)¹² has).

84 Fortunately, the GPL [states](https://www.gnu.org/licenses/old-licenses/gpl-2.0.html#section3)¹³ “as a special exception, the source code dis-
85 tributed need not include anything that is normally distributed (in either source
86 or binary form) with the major components (compiler, kernel, and so on) of the
87 operating system on which the executable runs, unless that component itself
88 accompanies the executable”. If the library is distributed as part of the OS and
89 can be considered a major component of it, then this clause doesn't require the
90 library to be considered as part of the software and therefore falls outside of the
91 scope of the license. A counter argument to this is that because the application
92 may also be considered to be distributed as part of the operating system this
93 exception doesn't apply especially in embedded devices where the software is
94 distributed preinstalled as a complete entity.

95 Currently, most distributions such as Fedora and Debian consider OpenSSL a
96 system library overcoming the incompatibility.

97 In relation to proprietary code, OpenSSL 1.1 is licensed under OpenSSL license
98 a BSD-style license with additional advertising clauses. This license falls under
99 the permissive category since it does not imposes many restrictions. OpenSSL
100 3.0 is licensed under the standard Apache 2.0, which is also a permissive one.
101 In conclusion, the use of OpenSSL is suitable in proprietary code.

¹¹<https://www.apache.org/licenses/LICENSE-2.0>

¹²<https://spdx.org/licenses/openvpn-openssl-exception.html>

¹³<https://www.gnu.org/licenses/old-licenses/gpl-2.0.html#section3>

NSS

[Network Security Services](#)¹⁴ (NSS) is a set of security libraries developed by Mozilla. NSS provides its own API, which is currently only supported by a few of the applications which use TLS in Apertis. It is licensed as [MPL-2.0](#)¹⁵.

Approach

In order to fulfil the requirements the approach taken has been to upgrade GnuTLS to a new version for those applications that can use it licensed as GPL-2. With OpenSSL upgraded and retained as a system library, utilizing it, inline with the approach taken by other distributions that have documented a specific policy covering this.

The one outlier is the printing support in GTK which uses GnuTLS and which potentially ends up causing GPL-2 dependencies in GTK. Whilst Debian have also declared CUPS as a system library, we feel that the differing use cases for Debian and Apertis make this less of a realistic position to take. We have therefore dropped printing support from GTK in order to remove this dependency as we don't feel that this functionality is critical to Apertis'aim.

This approach was introduced initially in v2022, and after being tested the changes were backported to v2021 and v2020, to make older releases comply with Apertis license expectations.

Summary

The tables below summarize the use of TLS libraries in various releases of Apertis target images. We would expect proprietary applications to either utilize the OpenSSL or NSS libraries as deemed appropriate by the individual projects.

TLS stack

Component	License	OpenSSL	GnuTLS	Notes
apt	GPL-2+		X	
connman	GPL-2		X	
curl	curl and BSD-3-Clause and BSD-4-Clause-UC and ISC	X	X	also produced after rebase
glib-networking	LGPL-2.1+ and LGPL-2.1+ with OpenSSL exception	X		
liboauth	Expat/MIT	curl		
libmicrohttpd	LGPL-2.1+		X	removed s
libmicrohttpd	LGPL-2.1+		X	
neon27	LGPL-2.1+	X	X	
openjpeg	BSD-2	curl		package li

¹⁴<https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS>

¹⁵<https://www.mozilla.org/en-US/MPL/2.0/>

Component	License	OpenSSL	GnuTLS	Notes
openldap	OLDAP-2.8	X		
rtmpdump	GPL-2+ (tools), LGPL-2.1+ (library)		X	removed s
systemd	LGPL-2.1+ and GPL-2[+] and PD X	curl		package sy
tumbler	LGPL-2.1+ and GPL-2+	curl		

Appendix

Previous situation

The “target” section of Apertis ships a variety of packages which use TLS from a provided library. There are a number of software libraries that provide competing TLS implementations and which are provided under various licensing terms. However, these projects do not always provide the same programming interfaces, thus do not provide a drop in replacement for each other. Whilst some users of TLS libraries may provide some level of abstraction to support more than one TLS library, others may support only one and thus Apertis currently provides [GnuTLS](#)¹⁶, [OpenSSL](#)¹⁷ and [NSS](#)¹⁸.

- **GnuTLS:** Apertis currently provides GnuTLS version 3.4.10. This is an approximately four-year-old version of GnuTLS as shipped in Ubuntu Xenial and thus is currently supported by Ubuntu and is expected to be until 2022. GnuTLS is used directly or indirectly via libcurl in just more than a dozen packages in target. Debian Buster, the current main upstream of Apertis, includes a newer version of GnuTLS (currently 3.6.7) though upgrading to this has already been avoided due to licensing issues that will be discussed below.
- **OpenSSL:** Apertis currently provides OpenSSL version 1.1.1. This is a relatively recent release in the 1.1.1 series and is packaged as part of Debian Buster. The 1.1.1 series is [currently supported](#)¹⁹ as an LTS release by the OpenSSL project until September 2023. Support for Debian Buster is [expected](#)²⁰ until June 2024.
- **NSS:** Apertis currently provides NSS version 3.42.1. This version is approximately a year and a half old, and is packaged as part of Debian Buster. As with OpenSSL, support for Debian Buster is expected until June 2024.

Some of the packages requiring TLS support only support one of the currently provided TLS implementations, often due to licensing compatibility. Other

¹⁶<https://www.gnutls.org/>

¹⁷<https://www.openssl.org/>

¹⁸<https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS>

¹⁹<https://www.openssl.org/policies/releasestrat.html>

²⁰<https://wiki.debian.org/LTS>

155 packages, most notably libraries, support multiple TLS backends, frequently
156 including both GnuTLS and OpenSSL as options.

157 Issues

158 The TLS libraries used in Apertis were supported, though this will not re-
159 main the case indefinitely, with Ubuntu dropping support for the currently
160 used GnuTLS in 2022, NSS and OpenSSL 1.1 losing support in 2024.

161 Releases of Apertis would be expected to be based on newer versions of Debian
162 (as covered in the [Apertis Release Flow](#)²¹. As could be expected, newer ver-
163 sions of Debian have integrated newer versions of these TLS libraries. Whilst
164 upgrading to newer versions of NSS does not appear to present any issues, the
165 GnuTLS or OpenSSL may present issues for Apertis:

166 GnuTLS

167 Whilst GnuTLS is licensed under the [LGPL-2.1](#)²², it uses [Nettle](#)²³ and [GMP](#)²⁴.
168 Newer versions of both of these dependencies are now licensed as dual GPL-2
169 or LGPL-3, rather than LGPL-2.1.

170 To avoid including GnuTLS under LGPL-3 terms, should Apertis integrate a
171 newer version it would need to be utilized under the GPL-2 terms. This would
172 result in the binary GnuTLS library effectively being used under the terms of
173 the GPL-2 rather than LGPL-2.1. This would restrict Apertis users from using
174 this Apertis provided TLS implementation either directly or indirectly from any
175 non-GPL-2 compatible applications they wish to integrate into their systems, for
176 example in proprietary applications, where it would have the effect of requiring
177 the app to also be GPL-2 licensed.

178 OpenSSL

179 The currently used version of OpenSSL is licensed under a custom GPL-
180 incompatible license. OpenSSL 3.0 (the next major version of OpenSSL) will
181 be licensed under the [Apache 2.0](#)²⁵ license, which is compatible with the
182 GPL-3, but not GPL-2. This means that GPL-2 tools like `tumbler`, `connman`, `apt`
183 or `systemd-journal-remote` cannot use the newer versions of OpenSSL without
184 effectively becoming GPL-3 licensed or through these upstream projects
185 applying a license exceptions (for example as [OpenVPN](#)²⁶ has). The OpenSSL
186 project do not seem to hold a strong opinion on the compatibility, though

²¹<https://apertis-website-0b3586.pages.apertis.org/policies/release-flow/#apertis-release-flow>

²²<https://www.gnu.org/licenses/old-licenses/lgpl-2.1.en.html>

²³<https://www.lysator.liu.se/~nisse/nettle/nettle.html>

²⁴<https://gmplib.org/>

²⁵<https://www.apache.org/licenses/LICENSE-2.0>

²⁶<https://spdx.org/licenses/openvpn-openssl-exception.html>

187 [suggest](#)²⁷ either not using the GPL or applying an exception should you wish
188 to gain some legal certainty.

189 Given the security sensitive nature of the TLS stack, utilizing unmaintained soft-
190 ware here would be best avoided. Putting maintenance aside, these versions of
191 their respective TLS implementations may not be gaining support for any new
192 ciphers and TLS protocol versions, which will severely limit their usefulness as
193 time progresses. As well as not gaining newer protocol versions, the libraries
194 may not be updated to reflect the frequently changing [recommendations regard-](#)
195 [ing minimal protocol versions](#)²⁸ that should be supported, which may result in
196 issues when attempting to access sites following the “Modern” recommendation.
197 Additionally, it is likely that newer versions of the packages utilizing these TLS
198 implementations will begin to require functionality added to newer versions of
199 the TLS libraries thus reducing the ability of Apertis to upgrade to these too.

200 It is therefore imperative that a way forward is agreed upon that is acceptable
201 to Apertis stakeholders.

202 The OpenSSL project do not seem to hold a strong opinion on the compatibility,
203 though [suggest](#)²⁹ either not using the GPL or applying an exception should you
204 wish to gain some legal certainty.

205 The compatibility between the current OpenSSL licensing and GPL-2 is based
206 on the premise that:

- 207 1. The [OpenSSL license](#)³⁰ contains licensing terms not in the GPL (such as
208 the need to mention use of the software in all advertising material and
209 derivatives not being able to be called OpenSSL).
- 210 2. Linking OpenSSL with a GPL-2 application creates a derivative work
211 formed from the two pieces of code.
- 212 3. The GPL expressly [states](#)³¹ that one can’t “impose any further restrictions
213 on the recipients’ exercise of the rights granted herein” to the GPL licensed
214 work.

215 Likewise, the Apache 2.0 license, to which version 3 of OpenSSL will be release
216 under, contains clauses such as its [patent litigation license termination clause](#)³².

217 While the argument made in step (2) is widely held by many, others disagree
218 with this interpretation, especially when the library is dynamically linked to
219 the application. For instance, it might be [claimed](#)³³ that a dynamically linked
220 library is only truly combined with the application when run, not when dis-
221 tributed, so it would only become a derivative at that point, or it [might be](#)

²⁷<https://www.openssl.org/docs/faq.html#LEGAL2>

²⁸https://wiki.mozilla.org/Security/Server_Side_TLS

²⁹<https://www.openssl.org/docs/faq.html#LEGAL2>

³⁰<https://www.openssl.org/source/license-openssl-ssleay.txt>

³¹<https://www.gnu.org/licenses/old-licenses/gpl-2.0.html#section6>

³²<http://www.apache.org/licenses/LICENSE-2.0#patent>

³³<https://lwn.net/Articles/548216/>

222 [claimed](#)³⁴ as this is the intended interface for interacting with a library this is
223 excluded either due to fair use laws in some jurisdictions or explicitly allowed
224 by the GPL when it [states](#)³⁵ “the act of running the Program is not restricted”.

225 A further argument is that the GPL [states](#)³⁶ “as a special exception, the source
226 code distributed need not include anything that is normally distributed (in either
227 source or binary form) with the major components (compiler, kernel, and so on)
228 of the operating system on which the executable runs, unless that component
229 itself accompanies the executable”. If the library is distributed as part of the
230 OS and can be considered a major component of it, then this clause doesn’t
231 require the library to be considered as part of the software and therefore falls
232 outside of the scope of the license. A counter argument to this is that because
233 the application may also be considered to be distributed as part of the operating
234 system this exception doesn’t apply especially in embedded devices where the
235 software is distributed preinstalled as a complete entity.

236 Most distributions seem to either ignore this potential issue or do not consider a
237 policy to be needed. The Fedora project have deemed OpenSSL to be a [system](#)
238 [library](#)³⁷ as defined by the GPL and thus there is no incompatibility. Debian
239 historically decided that a linked library creates a derivative work and all the
240 packages it ships should be considered a combined work, though the decision
241 has [recently been taken](#)³⁸ to follow Fedora’s lead here.

242 **Alternative SSL solutions**

243 In addition to GnuTLS and OpenSSL, there are a number of other TLS libraries
244 available, including:

245 **BoringSSL**

246 [BoringSSL](#)³⁹ is a fork of OpenSSL being actively maintained by Google for
247 internal use. It currently provides an OpenSSL based API, but explicitly states
248 it comes with no API-ABI guarantees, users should expect API changes as
249 deemed suitable for Googles internal users. BoringSSL maintains the current
250 licensing state, though as it’s developed the amount of OpenSSL-licensed code
251 is reduced, in part through the removal of legacy code. Googles additions are
252 currently provided under the ISC license.

³⁴<https://www.linuxjournal.com/article/6366>

³⁵<https://www.gnu.org/licenses/old-licenses/gpl-2.0.html#section0>

³⁶<https://www.gnu.org/licenses/old-licenses/gpl-2.0.html#section3>

³⁷[https://fedoraproject.org/wiki/Licensing:FAQ?rd=Licensing/FAQ#What.27s_the_deal
with_the_OpenSSL_license.3F](https://fedoraproject.org/wiki/Licensing:FAQ?rd=Licensing/FAQ#What.27s_the_deal_with_the_OpenSSL_license.3F)

³⁸<https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=924937#105>

³⁹<https://boringssl.googlesource.com/boringssl/>

253 LibreSSL

254 LibreSSL⁴⁰ is maintained by OpenBSD, it is a fork of OpenSSL v1.0.1, made
255 as a result of the poor maintenance of OpenSSL at the time (but which has
256 since improved). It aims to modernize the code base, improve security, and
257 apply best practice development process. As a result of these goals a lot of
258 legacy code has been removed. LibreSSL maintains the current licensing state,
259 with new additions provided under the ISC license. LibreSSL does not appear
260 to have gained significant adoption which will limit the developer attention it
261 receives.

262 mbed TLS

263 mbed TLS⁴¹ is a TLS implementation with a small footprint, targeting embed-
264 ded systems. The mbed TLS library does not provide either the OpenSSL or
265 GnuTLS API, it provides an API at a slightly lower level, requiring more man-
266 ual operations⁴² and thus wrappers or porting effort would be required to use
267 it. It is available in two versions, one distributed under the Apache-2.0 license
268 and another separately licensed as GPL-2+, though it's understood that it will
269 drop the GPL-2+ license in the next major release.

270 MesaLink

271 MesaLink⁴³ is an OpenSSL-compatible TLS library written in Rust⁴⁴. With
272 it being implemented in Rust it can be assumed to have some resilience due
273 to this languages focus on safety and MesaLink recently underwent a third-
274 party security audit with excellent results⁴⁵. However, MesaLink only supports
275 modern TLS standards and thus connectivity with older and less secure servers
276 may be impacted. MesaLink is licensed as BSD-3-Clause, however it uses a
277 large number of third party libraries, licensed as follows:

- 278 • base64: Apache-2.0/MIT
- 279 • bitflags: Apache-2.0/MIT
- 280 • env_logger: Apache-2.0/MIT
- 281 • enum_to_u8_slice_derive: BSD-3-Clause
- 282 • libc: Apache-2.0/MIT
- 283 • parking_lot: Apache-2.0/MIT
- 284 • ring: Based on BoringSSL and thus has parts licensed under the ISC and
285 original OpenSSL licensing
- 286 • rustls: Apache-2.0/ISC/MIT
- 287 • sct: Apache-2.0/ISC/MIT

⁴⁰<https://www.libressl.org/>

⁴¹<https://tls.mbed.org/>

⁴²<https://github.com/warmcat/libwebsockets/commit/9da75727858b4d60750cfefc1673f6783e8719d>

⁴³<https://mesalink.io/>

⁴⁴<https://www.rust-lang.org/>

⁴⁵<https://github.com/ctz/rustls/blob/master/audit/TLS-01-report.pdf>

- webpki, untrusted: ISC
- webpki-roots: MPL-2.0

NSS

Network Security Services⁴⁶ (NSS) is a set of security libraries developed by Mozilla. NSS provides its own API, which is currently only supported by a few of the applications which use TLS in Apertis, thus its use would require wrappers to be created or porting effort. It is licensed as MPL-2.0⁴⁷.

wolfSSL

The wolfSSL⁴⁸ cryptographic library provides some compatibility with OpenSSL via a compatibility header, which maps a subset of the most commonly used OpenSSL commands to its native API. It provides up-to-date standards support. wolfSSL has already been packaged in Debian. It is available under a dual license, GPL-2+ and commercial⁴⁹ licensing terms.

Possible solutions

We have considered the following options to meet Apertis' requirements.

Single stack solutions

Despite the relatively large number of TLS implementations, the required application compatibility and licensing requirements mean that there is not a single solution that will work without investing at least some development effort.

Attempting to standardize on a TLS implementation, such as by using the single stack solutions detailed below would therefore result in Apertis carrying significant changes to its packages without any guarantees that these changes could be upstreamed. These changes would thus need to be maintained as part of Apertis.

Standardize on GnuTLS, replace use of problematic dependencies

GnuTLS used to use libgcrypt as a cryptographic backend and the code is mostly structured to abstract the backend details. Reverting to using libgcrypt would result in a LGPL-2.1 licensed solution that may be viable for all desired use cases.

A preliminary investigation suggests that GnuTLS may have started to use Nettle outside of the abstracted code, which would complicate conversion back to libgcrypt. More investigation would be required to confirm this.

⁴⁶<https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS>

⁴⁷<https://www.mozilla.org/en-US/MPL/2.0/>

⁴⁸<https://www.wolfssl.com/>

⁴⁹<https://www.wolfssl.com/license/>

320 If libgcrypt is deemed unsuitable, an alternative may be to port GnuTLS to a dif-
321 ferent cryptographic library such as libtomcrypt (Public Domain) or libsodium
322 (ISC). The effort required to achieve this has not been investigated.

323 It is likely that any resulting changes would need to be maintained as part of
324 Apertis. It's not clear the upstream GnuTLS project would be interested in
325 maintaining another backend.

326 **Standardize on an OpenSSL-compatible library** As many of the appli-
327 cations already utilize OpenSSL, another possible approach would be writing a
328 wrapper for a library which provides OpenSSL compatibility to also provide the
329 GnuTLS API.

330 As GnuTLS itself comes with a wrapper providing OpenSSL API, it is believed
331 that the reverse should also be possible. However, this presents some significant
332 effort as the APIs are quite different.

333 An alternative approach may be to port those apps which only support GnuTLS
334 to utilize the OpenSSL API. The effort required to achieve this has not been
335 estimated.

336 Such an approach is of limited benefit as the more widely used and mature
337 solutions providing an OpenSSL API are also licensed in such a way as to be
338 incompatible with the GPL-2, which happens to be the license used by the most
339 critical applications currently using GnuTLS.

340 **Wrapping a non-GnuTLS/OpenSSL-compatible library to provide**
341 **both APIs** Standardizing on NSS would fall into this category. This would
342 also be true for mbed TLS, but the Apache-2.0 license that it is future version
343 are likely to be solely licensed under would be problematic for GPL-2-licensed
344 applications. This option would require significant effort (creating wrappers for
345 both GnuTLS and OpenSSL APIs) and would be a high risk strategy.

346 **Multi-stack solutions**

347 Attempting to choose a TLS implementation that is licensed in a manner that
348 will work for the GPL-2-licensed applications through to Apertis'users propri-
349 etary applications massively limits the choice of library. Most of the available
350 choices only satisfy one or other end of this spectrum, with NSS and MesaLink
351 being the only solutions that may be suitably licensed, but which also lacks
352 compatibility with critical applications.

353 As there does not appear to be any single TLS solutions meeting all use cases
354 without significant work, we will consider keeping a multi stack solution as
355 currently employed.

356 In such a scenario, a newer GnuTLS library could be allowed by accepting its
357 dependencies under the GPL-2 license and restricting its use to places where
358 this license wouldn't be problematic, such as existing GPL-2 software. As the

359 existing applications written exclusively to use GnuTLS are GPL-2 or tolerant
360 of GPL-2, this is viable.

361 **Replace OpenSSL with compatible alternative** A number of alternative
362 TLS implementations provide an “OpenSSL-compatible” interface of one form or
363 other. Whilst a number of these solutions are not compatible with the GPL-2,
364 as this solution would require the continued availability of GnuTLS, the choice
365 of replacement can be picked without needing to provide GPL-2 compatibility.
366 This would suggest BoringSSL, LibreSSL and MesaLink as options (wolfSSL
367 being immediately unsuitable due to licensing).

- 368 • **BoringSSL:** Whilst actively maintained by Google for its own products,
369 the lack of API/ABI guarantees make its adoption risky.
- 370 • **LibreSSL:** It’s use inside OpenBSD suggests this will be maintained at
371 least in the mid-term.
- 372 • **MesaLink:** As Rust is good at detecting many security related issues at
373 compile time, its use here brings many advantages, though this needs to
374 be weighed against its lack of support of older TLS standards which may
375 prove problematic in some use cases.

376 Picking an API-compatible replacement for OpenSSL may provide a solution
377 for the mid-term, however with OpenSSL set to release its new version at some
378 point in the future, it is likely that we may start to see applications requiring
379 compatibility with OpenSSL 3.0 APIs, thus requiring Apertis to reconsider its
380 solution. Additionally, while these libraries claim OpenSSL compatibility, a
381 different implementation may result in hard to diagnose bugs being uncovered
382 in applications expecting OpenSSL.