Commit 39258d3

tiran authored on Apr 17, 2021 Verified

```
bpo-43669: PEP 644: Require OpenSSL 1.1.1 or newer (GH-23014)

- Remove HAVE_X509_VERIFY_PARAM_SET1_HOST check
- Update hashopenssl to require OpenSSL 1.1.1
- multissltests only OpenSSL > 1.1.0
- ALPN is always supported
- SNI is always supported
- Remove deprecated NPN code. Python wrappers are no-op.
- ECDH is always supported
- Remove OPENSSL_VERSION_1_1 macro
- Remove locking callbacks
- Drop PY_OPENSSL_1_1_API macro
- Drop HAVE_SSL_CTX_CLEAR_OPTIONS macro
- SSL_CTRL_GET_MAX_PROTO_VERSION is always defined now
- security level is always available now
- get_num_tickets is available with TLS 1.3
- X509_V_ERR MISMATCH is always available now
- Always set SSL_MODE_RELEASE_BUFFERS
- X509_V_FLAG_TRUSTED_FIRST is always available
- get_ciphers is always supported
- SSL_CTX_set_keylog_callback is always available
- Update Modules/Setup with static link example
- Mention PEP in whatsnew
- Drop 1.0.2 and 1.1.0 from GHA tests
```

main (#23014) · v3.15.0a8 ••• v3.10.0b1

1 parent b467d9a commit 39258d3

17 files changed +5310 -8440 lines changed

Top

Filter files...

- .github/workflows
  - build.yml
- Doc

- using
  - unix.rst
- whatsnew
  - 3.10.rst
- Lib
  - ssl.py
- test
  - test_ssl.py
- Misc/NEWS.d/next/Build
  - 2021-03-30-14-19-39.bpo-43669.lWMUYx.rst
- Modules
  - Setup
  - _hashopenssl.c
  - _ssl.c
- _ssl
  - debughelpers.c
- clinic
  - _hashopenssl.c.h
  - _ssl.c.h
- Tools/ssl
  - multissltests.py
- configure.ac
- configure
- pyconfig.h.in
- setup.py

1

17 files changed +5310 -8440 lines changed

Search within code

.github/workflows/build.yml

```
@@ -177,7 +177,7 @@ jobs:
177  177        strategy:
```

```
178  178          fail-fast: false
179  179          matrix:
180       -        openssl_ver: [1.0.2u, 1.1.0l, 1.1.1k, 3.0.0-alpha14]
     180  +        openssl_ver: [1.1.1k, 3.0.0-alpha14]
181  181        env:
182  182          OPENSSL_VER: ${{ matrix.openssl_ver }}
183  183          MULTISSL_DIR: ${{ github.workspace }}/multissl
```

Doc/using/unix.rst

```
@@ -135,6 +135,7 @@ some Unices may not have the :program:`env` command, so you may need to hardcode
135  135
136  136  To use shell commands in your Python scripts, look at the :mod:`subprocess` module.
137  137
     138  + .. _unix_custom_openssl:
138  139
139  140  Custom OpenSSL
140  141  ==============
```

Doc/whatsnew/3.10.rst

```
@@ -65,6 +65,7 @@ Summary -- Release highlights
65   65
66   66   .. PEP-sized items next.
67   67
     68   + * :pep:`644`, require OpenSSL 1.1.1 or newer
68   69
69   70
70   71   New Features
@@ -1438,6 +1439,10 @@ CPython bytecode changes
1438 1439   Build Changes
1439 1440   =============
1440 1441
     1442 + * :pep:`644`: Python now requires OpenSSL 1.1.1 or newer. OpenSSL 1.0.2 is no
     1443 +   longer supported.
     1444 +   (Contributed by Christian Heimes in :issue:`43669`.)
```

```
                1445 +
1441            1446     * The C99 functions :c:func:`snprintf` and :c:func:`vsnprintf` are now
                         required
1442            1447       to build Python.
1443            1448       (Contributed by Victor Stinner in :issue:`36020`.)
                         @@ -1483,7 +1488,6 @@ Build Changes
1483            1488       (Contributed by Christian Heimes in :issue:`43466`.)
1484            1489
1485            1490
1486                     -
1487            1491     C API Changes
1488            1492     =============
1489            1493
```

Lib/ssl.py

```
                         @@ -909,15 +909,12 @@ def selected_npn_protocol(self):
909             909              """Return the currently selected NPN protocol as a string, or
                         ``None``
910             910              if a next protocol was not negotiated or if NPN is not supported by
                         one
911             911              of the peers."""
912                      -        if _ssl.HAS_NPN:
913                      -            return self._sslobj.selected_npn_protocol()
914             912
915             913          def selected_alpn_protocol(self):
916             914              """Return the currently selected ALPN protocol as a string, or
                         ``None``
917             915              if a next protocol was not negotiated or if ALPN is not supported by
                         one
918             916              of the peers."""
919                      -        if _ssl.HAS_ALPN:
920                      -            return self._sslobj.selected_alpn_protocol()
                    917  +        return self._sslobj.selected_alpn_protocol()
921             918
922             919          def cipher(self):
923             920              """Return the currently selected cipher as a 3-tuple ``(name,
                         @@ -1126,10 +1123,7 @@ def getpeercert(self, binary_form=False):
```

```
1126  1123          @_sslcopydoc
1127  1124          def selected_npn_protocol(self):
1128  1125              self._checkClosed()
1129        -         if self._sslobj is None or not _ssl.HAS_NPN:
1130        -             return None
1131        -         else:
1132        -             return self._sslobj.selected_npn_protocol()
      1126  +         return None
1133  1127
1134  1128          @_sslcopydoc
1135  1129          def selected_alpn_protocol(self):
```

Lib/test/test_ssl.py

```
                  @@ -40,7 +40,6 @@
40    40          PROTOCOLS = sorted(ssl._PROTOCOL_NAMES)
41    41          HOST = socket_helper.HOST
42    42          IS_LIBRESSL = ssl.OPENSSL_VERSION.startswith('LibreSSL')
43        -       IS_OPENSSL_1_1_0 = not IS_LIBRESSL and ssl.OPENSSL_VERSION_INFO >= (1, 1, 0)
44    43          IS_OPENSSL_1_1_1 = not IS_LIBRESSL and ssl.OPENSSL_VERSION_INFO >= (1, 1, 1)
45    44          IS_OPENSSL_3_0_0 = not IS_LIBRESSL and ssl.OPENSSL_VERSION_INFO >= (3, 0, 0)
46    45          PY_SSL_DEFAULT_CIPHERS = sysconfig.get_config_var('PY_SSL_DEFAULT_CIPHERS')
                  @@ -270,18 +269,6 @@ def handle_error(prefix):
270   269              if support.verbose:
271   270                  sys.stdout.write(prefix + exc_format)
272   271
273       - def can_clear_options():
274       -     # 0.9.8m or higher
275       -     return ssl._OPENSSL_API_VERSION >= (0, 9, 8, 13, 15)
276       -
277       - def no_sslv2_implies_sslv3_hello():
278       -     # 0.9.7h or higher
279       -     return ssl.OPENSSL_VERSION_INFO >= (0, 9, 7, 8, 15)
280       -
281       - def have_verify_flags():
282       -     # 0.9.8 or higher
283       -     return ssl.OPENSSL_VERSION_INFO >= (0, 9, 8, 0, 15)
284       -
285   272      def _have_secp_curves():
```

```
286  273          if not ssl.HAS_ECDH:
287  274              return False
     @@ -372,17 +359,15 @@ def test_constants(self):
372  359              ssl.OP_SINGLE_DH_USE
373  360              if ssl.HAS_ECDH:
374  361                  ssl.OP_SINGLE_ECDH_USE
375       -          if ssl.OPENSSL_VERSION_INFO >= (1, 0):
376       -              ssl.OP_NO_COMPRESSION
         362 +          ssl.OP_NO_COMPRESSION
377  363              self.assertIn(ssl.HAS_SNI, {True, False})
378  364              self.assertIn(ssl.HAS_ECDH, {True, False})
379  365              ssl.OP_NO_SSLv2
380  366              ssl.OP_NO_SSLv3
381  367              ssl.OP_NO_TLSv1
382  368              ssl.OP_NO_TLSv1_3
383       -          if ssl.OPENSSL_VERSION_INFO >= (1, 0, 1):
384       -              ssl.OP_NO_TLSv1_1
385       -              ssl.OP_NO_TLSv1_2
         369 +          ssl.OP_NO_TLSv1_1
         370 +          ssl.OP_NO_TLSv1_2
386  371              self.assertEqual(ssl.PROTOCOL_TLS, ssl.PROTOCOL_SSLv23)
387  372
388  373          def test_private_init(self):
     @@ -1161,7 +1146,6 @@ def test_python_ciphers(self):
1161 1146              self.assertNotIn("RC4", name)
1162 1147              self.assertNotIn("3DES", name)
1163 1148
1164      -    @unittest.skipIf(ssl.OPENSSL_VERSION_INFO < (1, 0, 2, 0, 0), 'OpenSSL too
     old')
1165 1149          def test_get_ciphers(self):
1166 1150              ctx = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
1167 1151              ctx.set_ciphers('AESGCM')
     @@ -1181,15 +1165,11 @@ def test_options(self):
1181 1165              self.assertEqual(default, ctx.options)
1182 1166              ctx.options |= ssl.OP_NO_TLSv1
1183 1167              self.assertEqual(default | ssl.OP_NO_TLSv1, ctx.options)
1184      -            if can_clear_options():
1185      -                ctx.options = (ctx.options & ~ssl.OP_NO_TLSv1)
```

```
1186      -             self.assertEqual(default, ctx.options)
1187      -             ctx.options = 0
1188      -             # Ubuntu has OP_NO_SSLv3 forced on by default
1189      -             self.assertEqual(0, ctx.options & ~ssl.OP_NO_SSLv3)
1190      -         else:
1191      -             with self.assertRaises(ValueError):
1192      -                 ctx.options = 0
          1168  +         ctx.options = (ctx.options & ~ssl.OP_NO_TLSv1)
          1169  +         self.assertEqual(default, ctx.options)
          1170  +         ctx.options = 0
          1171  +         # Ubuntu has OP_NO_SSLv3 forced on by default
          1172  +         self.assertEqual(0, ctx.options & ~ssl.OP_NO_SSLv3)
1193      1173
1194      1174      def test_verify_mode_protocol(self):
1195      1175          ctx = ssl.SSLContext(ssl.PROTOCOL_TLS)
                  @@ -1327,8 +1307,6 @@ def test_security_level(self):
1327      1307          }
1328      1308          self.assertIn(ctx.security_level, security_level_range)
1329      1309
1330      -     @unittest.skipUnless(have_verify_flags(),
1331      -                          "verify_flags need OpenSSL > 0.9.8")
1332      1310      def test_verify_flags(self):
1333      1311          ctx = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
1334      1312          # default value
                  @@ -1797,7 +1775,6 @@ class MySSLObject(ssl.SSLObject):
1797      1775          obj = ctx.wrap_bio(ssl.MemoryBIO(), ssl.MemoryBIO())
1798      1776          self.assertIsInstance(obj, MySSLObject)
1799      1777
1800      -     @unittest.skipUnless(IS_OPENSSL_1_1_1, "Test requires OpenSSL 1.1.1")
1801      1778      def test_num_tickest(self):
1802      1779          ctx = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
1803      1780          self.assertEqual(ctx.num_tickets, 2)
                  @@ -2956,8 +2933,6 @@ def test_getpeercert(self):
2956      2933                  after = ssl.cert_time_to_seconds(cert['notAfter'])
2957      2934                  self.assertLess(before, after)
2958      2935
2959      -     @unittest.skipUnless(have_verify_flags(),
```

```
2960      -                       "verify_flags need OpenSSL > 0.9.8")
2961 2936        def test_crl_check(self):
2962 2937            if support.verbose:
2963 2938                sys.stdout.write("\n")
          @@ -3859,12 +3834,7 @@ def test_version_basic(self):
3859 3834                    self.assertIs(s.version(), None)
3860 3835                    self.assertIs(s._sslobj, None)
3861 3836                    s.connect((HOST, server.port))
3862      -                  if IS_OPENSSL_1_1_1 and has_tls_version('TLSv1_3'):
3863      -                      self.assertEqual(s.version(), 'TLSv1.3')
3864      -                  elif ssl.OPENSSL_VERSION_INFO >= (1, 0, 2):
3865      -                      self.assertEqual(s.version(), 'TLSv1.2')
3866      -                  else:  # 0.9.8 to 1.0.1
3867      -                      self.assertIn(s.version(), ('TLSv1', 'TLSv1.2'))
     3837 +                  self.assertEqual(s.version(), 'TLSv1.3')
3868 3838                self.assertIs(s._sslobj, None)
3869 3839                self.assertIs(s.version(), None)
3870 3840
          @@ -3966,8 +3936,6 @@ def test_default_ecdh_curve(self):
3966 3936            # explicitly using the 'ECCdraft' cipher alias.  Otherwise,
3967 3937            # our default cipher list should prefer ECDH-based ciphers
3968 3938            # automatically.
3969      -          if ssl.OPENSSL_VERSION_INFO < (1, 0, 0):
3970      -              context.set_ciphers("ECCdraft:ECDH")
3971 3939            with ThreadedEchoServer(context=context) as server:
3972 3940                with context.wrap_socket(socket.socket()) as s:
3973 3941                    s.connect((HOST, server.port))
          @@ -4099,15 +4067,11 @@ def test_ecdh_curve(self):
4099 4067            server_context.set_ciphers("ECDHE:!eNULL:!aNULL")
4100 4068            server_context.options |= ssl.OP_NO_TLSv1 | ssl.OP_NO_TLSv1_1
4101 4069            try:
4102      -              stats = server_params_test(client_context, server_context,
4103      -                                         chatty=True, connectionchatty=True,
4104      -                                         sni_name=hostname)
     4070 +              server_params_test(client_context, server_context,
     4071 +                                 chatty=True, connectionchatty=True,
     4072 +                                 sni_name=hostname)
```

```
4105  4073            except ssl.SSLError:
4106        -               pass
4107        -           else:
4108        -               # OpenSSL 1.0.2 does not fail although it should.
4109        -               if IS_OPENSSL_1_1_0:
4110        -                   self.fail("mismatch curve did not fail")
      4074  +               self.fail("mismatch curve did not fail")
4111  4075
4112  4076        def test_selected_alpn_protocol(self):
4113  4077            # selected_alpn_protocol() is None unless ALPN is used.
          @@ -4117,7 +4081,6 @@ def test_selected_alpn_protocol(self):
4117  4081                                        sni_name=hostname)
4118  4082            self.assertIs(stats['client_alpn_protocol'], None)
4119  4083
4120        -     @unittest.skipUnless(ssl.HAS_ALPN, "ALPN support required")
4121  4084        def test_selected_alpn_protocol_if_server_uses_alpn(self):
4122  4085            # selected_alpn_protocol() is None unless ALPN is used by the client.
4123  4086            client_context, server_context, hostname = testing_context()
          @@ -4127,7 +4090,6 @@ def
          test_selected_alpn_protocol_if_server_uses_alpn(self):
4127  4090                                        sni_name=hostname)
4128  4091            self.assertIs(stats['client_alpn_protocol'], None)
4129  4092
4130        -     @unittest.skipUnless(ssl.HAS_ALPN, "ALPN support needed for this test")
4131  4093        def test_alpn_protocols(self):
4132  4094            server_protocols = ['foo', 'bar', 'milkshake']
4133  4095            protocol_tests = [
          @@ -4150,22 +4112,17 @@ def test_alpn_protocols(self):
4150  4112                except ssl.SSLError as e:
4151  4113                    stats = e
4152  4114
4153        -           if (expected is None and IS_OPENSSL_1_1_0
4154        -                   and ssl.OPENSSL_VERSION_INFO < (1, 1, 0, 6)):
4155        -               # OpenSSL 1.1.0 to 1.1.0e raises handshake error
4156        -               self.assertIsInstance(stats, ssl.SSLError)
4157        -           else:
4158        -               msg = "failed trying %s (s) and %s (c).\n" \
4159        -                     "was expecting %s, but got %%s from the %%s" \
4160        -                     % (str(server_protocols), str(client_protocols),
4161        -                        str(expected))
```

```
4162      -                client_result = stats['client_alpn_protocol']
4163      -                self.assertEqual(client_result, expected,
4164      -                                 msg % (client_result, "client"))
4165      -                server_result = stats['server_alpn_protocols'][-1] \
4166      -                    if len(stats['server_alpn_protocols']) else 'nothing'
4167      -                self.assertEqual(server_result, expected,
4168      -                                 msg % (server_result, "server"))
          4115 +            msg = "failed trying %s (s) and %s (c).\n" \
          4116 +                "was expecting %s, but got %%s from the %%s" \
          4117 +                    % (str(server_protocols), str(client_protocols),
          4118 +                       str(expected))
          4119 +            client_result = stats['client_alpn_protocol']
          4120 +            self.assertEqual(client_result, expected,
          4121 +                             msg % (client_result, "client"))
          4122 +            server_result = stats['server_alpn_protocols'][-1] \
          4123 +                if len(stats['server_alpn_protocols']) else 'nothing'
          4124 +            self.assertEqual(server_result, expected,
          4125 +                             msg % (server_result, "server"))
4169      4126
4170      4127        def test_selected_npn_protocol(self):
4171      4128            # selected_npn_protocol() is None unless NPN is used
                    @@ -4175,31 +4132,8 @@ def test_selected_npn_protocol(self):
4175      4132                                        sni_name=hostname)
4176      4133            self.assertIs(stats['client_npn_protocol'], None)
4177      4134
4178      -    @unittest.skipUnless(ssl.HAS_NPN, "NPN support needed for this test")
4179      4135        def test_npn_protocols(self):
4180      -        server_protocols = ['http/1.1', 'spdy/2']
4181      -        protocol_tests = [
4182      -            (['http/1.1', 'spdy/2'], 'http/1.1'),
4183      -            (['spdy/2', 'http/1.1'], 'http/1.1'),
4184      -            (['spdy/2', 'test'], 'spdy/2'),
4185      -            (['abc', 'def'], 'abc')
4186      -        ]
4187      -        for client_protocols, expected in protocol_tests:
4188      -            client_context, server_context, hostname = testing_context()
4189      -            server_context.set_npn_protocols(server_protocols)
4190      -            client_context.set_npn_protocols(client_protocols)
4191      -            stats = server_params_test(client_context, server_context,
4192      -                                       chatty=True, connectionchatty=True,
```

```
4193        -                                       sni_name=hostname)
4194        -            msg = "failed trying %s (s) and %s (c).\n" \
4195        -                  "was expecting %s, but got %%s from the %%s" \
4196        -                      % (str(server_protocols), str(client_protocols),
4197        -                         str(expected))
4198        -            client_result = stats['client_npn_protocol']
4199        -            self.assertEqual(client_result, expected, msg % (client_result,
              "client"))
4200        -            server_result = stats['server_npn_protocols'][-1] \
4201        -                if len(stats['server_npn_protocols']) else 'nothing'
4202        -            self.assertEqual(server_result, expected, msg % (server_result,
              "server"))
            4136    +         assert not ssl.HAS_NPN
4203        4137
4204        4138        def sni_contexts(self):
4205        4139            server_context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
                    ↓
                    ↑   @@ -4369,8 +4303,7 @@ def test_session(self):
4369        4303            self.assertGreater(session.time, 0)
4370        4304            self.assertGreater(session.timeout, 0)
4371        4305            self.assertTrue(session.has_ticket)
4372        -           if ssl.OPENSSL_VERSION_INFO > (1, 0, 1):
4373        -               self.assertGreater(session.ticket_lifetime_hint, 0)
            4306    +           self.assertGreater(session.ticket_lifetime_hint, 0)
4374        4307            self.assertFalse(stats['session_reused'])
4375        4308            sess_stat = server_context.session_stats()
4376        4309            self.assertEqual(sess_stat['accept'], 1)
                    ↓
```

...ld/2021-03-30-14-19-39.bpo-43669.lWMUYx.rst

```
            •••     @@ -0,0 +1 @@
            1     + Implement :pep:`644`. Python now requires OpenSSL 1.1.1 or newer.
```

Modules/Setup

```
                    ↑   @@ -207,11 +207,23 @@ _symtable symtablemodule.c
207         207     #_socket socketmodule.c
208         208
209         209     # Socket module helper for SSL support; you must comment out the other
210         -       # socket line above, and possibly edit the SSL variable:
```

```
211       - #SSL=/usr/local/ssl
212       - #_ssl _ssl.c \
213       - #   -DUSE_SSL -I$(SSL)/include -I$(SSL)/include/openssl \
214       - #   -L$(SSL)/lib -lssl -lcrypto
      210 + # socket line above, and edit the OPENSSL variable:
      211 + # OPENSSL=/path/to/openssl/directory
      212 + # _ssl _ssl.c \
      213 + #     -I$(OPENSSL)/include -L$(OPENSSL)/lib \
      214 + #     -lssl -lcrypto
      215 + #_hashlib _hashopenssl.c \
      216 + #     -I$(OPENSSL)/include -L$(OPENSSL)/lib \
      217 + #     -lcrypto
      218 +
      219 + # To statically link OpenSSL:
      220 + # _ssl _ssl.c \
      221 + #     -I$(OPENSSL)/include -L$(OPENSSL)/lib \
      222 + #     -l:libssl.a -Wl,--exclude-libs,libssl.a \
      223 + #     -l:libcrypto.a -Wl,--exclude-libs,libcrypto.a
      224 + #_hashlib _hashopenssl.c \
      225 + #     -I$(OPENSSL)/include -L$(OPENSSL)/lib \
      226 + #     -l:libcrypto.a -Wl,--exclude-libs,libcrypto.a
215   227
216   228     # The crypt module is now disabled by default because it breaks builds
217   229     # on many systems (where -lcrypt is needed), e.g. Linux (I believe).
```

Comments 0