Paramiko

Release

Jun 10, 2022

Contents

1 Changelog **3**

2 Frequently Asked/Answered Questions **37**
- 2.1 Which version should I use? I see multiple active releases. . . . 37
- 2.2 Paramiko doesn't work with my Cisco, Windows or other non-Unix system! . . . 37
- 2.3 I'm having strange issues with my code hanging at shutdown! . . . 37

3 Installing **39**
- 3.1 Paramiko itself . . . 39
- 3.2 Cryptography . . . 40
- 3.3 Optional dependencies for GSS-API / SSPI / Kerberos . . . 40

4 Installing (1.x) **43**
- 4.1 General install notes . . . 43
- 4.2 PyCrypto . . . 43
- 4.3 ActivePython and PyPM . . . 44
- 4.4 Optional dependencies for GSS-API / SSPI / Kerberos . . . 45

5 Contributing **47**
- 5.1 How to get the code . . . 47
- 5.2 How to submit bug reports or new code . . . 47

6 Contact **49**

Index **51**

Paramiko is a pure-Python[1] (2.7, 3.4+) implementation of the SSHv2 protocol[2], providing both client and server functionality. It provides the foundation for the high-level SSH library Fabric, which is what we recommend you use for common client use-cases such as running remote shell commands or transferring files.

Direct use of Paramiko itself is only intended for users who need advanced/low-level primitives or want to run an in-Python sshd.

For installation information, changelogs, FAQs and similar, please visit our main project website; for API details, see the versioned docs. Additionally, the project maintainer keeps a roadmap on his personal site.

[1] Paramiko relies on cryptography for crypto functionality, which makes use of C and Rust extensions but has many precompiled options available. See our installation page for details.

[2] SSH is defined in RFC 4251, RFC 4252, RFC 4253 and RFC 4254. The primary working implementation of the protocol is the OpenSSH project. Paramiko implements a large portion of the SSH feature set, but there are occasional gaps.

CHAPTER 1

Changelog

- : bug:`1637` (via #1599) Raise `SSHException` explicitly when blank private key data is loaded, instead of the natural result of `IndexError`. This should help more bits of Paramiko or Paramiko-adjacent codebases to correctly handle this class of error. Credit: Nicholas Dietz.
- #1822: (via, and relating to, far too many other issues to mention here) Update `SSHClient` so it explicitly closes its wrapped socket object upon encountering socket errors at connection time. This should help somewhat with certain classes of memory leaks, resource warnings, and/or errors (though we hasten to remind everyone that Client and Transport have their own `.close()` methods for use in non-error situations!). Patch courtesy of `@YoavCohen`.
- #866: (also #838) Remove an old test-related file we don't support, and add PyPy to Travis-CI config. Thanks to Pierce Lopez for the final patch and Pedro Rodrigues for an earlier edition.
- #956: Switch code coverage service from coveralls.io to codecov.io (& then disable the latter's auto-comments.) Thanks to Nikolai Røed Kristiansen for the patch.
- #906: Clean up a handful of outdated imports and related tweaks. Thanks to Pierce Lopez.
- #1291: Backport pytest support and application of the `black` code formatter (both of which previously only existed in the 2.4 branch and above) to everything 2.0 and newer. This makes back/forward porting bugfixes significantly easier.
- #1292: Backport changes from #979 (added in Paramiko 2.3) to Paramiko 2.0-2.2, using duck-typing to preserve backwards compatibility. This allows these older versions to use newer Cryptography sign/verify APIs when available, without requiring them (as is the case with Paramiko 2.3+).

  Practically speaking, this change prevents spamming of `CryptographyDeprecationWarning` notices which pop up in the above scenario (older Paramiko, newer Cryptography).

  Note: This is a no-op for Paramiko 2.3+, which have required newer Cryptography releases since they were released.

- #1291: Backport pytest support and application of the `black` code formatter (both of which previously only existed in the 2.4 branch and above) to everything 2.0 and newer. This makes back/forward porting bugfixes significantly easier.

- #1292: Backport changes from #979 (added in Paramiko 2.3) to Paramiko 2.0-2.2, using duck-typing to preserve backwards compatibility. This allows these older versions to use newer Cryptography sign/verify APIs when available, without requiring them (as is the case with Paramiko 2.3+).

  Practically speaking, this change prevents spamming of `CryptographyDeprecationWarning` notices which pop up in the above scenario (older Paramiko, newer Cryptography).

  Note: This is a no-op for Paramiko 2.3+, which have required newer Cryptography releases since they were released.

- #1951: Add SSH config token expansion (eg `%h`, `%p`) when parsing `ProxyJump` directives. Patch courtesy of Bruno Inec.
- #2004: (via #2011) Apply unittest `skipIf` to tests currently using SHA1 in their critical path, to avoid failures on systems starting to disable SHA1 outright in their crypto backends (eg RHEL 9). Report & patch via Paul Howarth.
- #1838: (via #1870/#2028) Update `camelCase` method calls against the `threading` module to be `snake_case`; this and related tweaks should fix some deprecation warnings under Python 3.10. Thanks to Karthikeyan Singaravelan for the report, `@Narendra-Neerukonda` for the patch, and to Thomas Grainger and Jun Omae for patch workshopping.
- #2038: (via #2039) Recent versions of Cryptography have deprecated Blowfish algorithm support; in lieu of an easy method for users to remove it from the list of algorithms Paramiko tries to import and use, we've decided to remove it from our "preferred algorithms" list. This will both discourage use of a weak algorithm, and avoid warnings. Credit for report/patch goes to Mike Roest.
- #2008: (via #2010) Windows-native SSH agent support as merged in 2.10 could encounter `Errno 22 OSError` exceptions in some scenarios (eg server not cleanly closing a relevant named pipe). This has been worked around and should be less problematic. Reported by Danilo Campana Fuchs and patched by Jun Omae.
- #2017: OpenSSH 7.7 and older has a bug preventing it from understanding how to perform SHA2 signature verification for RSA certificates (specifically certs - not keys), so when we added SHA2 support it broke all clients using RSA certificates with these servers. This has been fixed in a manner similar to what OpenSSH's own client does: a version check is performed and the algorithm used is downgraded if needed. Reported by Adarsh Chauhan, with fix suggested by Jun Omae.
- #1933: Align signature verification algorithm with OpenSSH re: zero-padding signatures which don't match their nominal size/length. This shouldn't affect most users, but will help Paramiko-implemented SSH servers handle poorly behaved clients such as PuTTY. Thanks to Jun Omae for catch & patch.
- #2008: (via #2010) Windows-native SSH agent support as merged in 2.10 could encounter `Errno 22 OSError` exceptions in some scenarios (eg server not cleanly closing a relevant named pipe). This has been worked around and should be less problematic. Reported by Danilo Campana Fuchs and patched by Jun Omae.
- #2017: OpenSSH 7.7 and older has a bug preventing it from understanding how to perform SHA2 signature verification for RSA certificates (specifically certs - not keys), so when we added SHA2 support it broke all clients using RSA certificates with these servers. This has been fixed in a manner similar to what OpenSSH's own client does: a version check is performed and the algorithm used is downgraded if needed. Reported by Adarsh Chauhan, with fix suggested by Jun Omae.
- #1933: Align signature verification algorithm with OpenSSH re: zero-padding signatures which don't match their nominal size/length. This shouldn't affect most users, but will help Paramiko-implemented SSH servers handle poorly behaved clients such as PuTTY. Thanks to Jun Omae for catch & patch.
- #2035: Servers offering certificate variants of hostkey algorithms (eg `ssh-rsa-cert-v01@openssh.com`) could not have their host keys verified by Paramiko clients, as it only ever considered non-cert key types for that part of connection handshaking. This has been fixed.

- #1964: (via #2024 as also reported in #2023) `PKey` instances' `__eq__` did not have the usual safety guard in place to ensure they were being compared to another `PKey` object, causing occasional spurious `BadHostKeyException` (among other things). This has been fixed. Thanks to Shengdun Hua for the original report/patch and to Christopher Papke for the final version of the fix.
- #1838: (via #1870/#2028) Update `camelCase` method calls against the `threading` module to be `snake_case`; this and related tweaks should fix some deprecation warnings under Python 3.10. Thanks to Karthikeyan Singaravelan for the report, `@Narendra-Neerukonda` for the patch, and to Thomas Grainger and Jun Omae for patch workshopping.
- #2035: Servers offering certificate variants of hostkey algorithms (eg `ssh-rsa-cert-v01@openssh.com`) could not have their host keys verified by Paramiko clients, as it only ever considered non-cert key types for that part of connection handshaking. This has been fixed.
- #1964: (via #2024 as also reported in #2023) `PKey` instances' `__eq__` did not have the usual safety guard in place to ensure they were being compared to another `PKey` object, causing occasional spurious `BadHostKeyException` (among other things). This has been fixed. Thanks to Shengdun Hua for the original report/patch and to Christopher Papke for the final version of the fix.
- #1838: (via #1870/#2028) Update `camelCase` method calls against the `threading` module to be `snake_case`; this and related tweaks should fix some deprecation warnings under Python 3.10. Thanks to Karthikeyan Singaravelan for the report, `@Narendra-Neerukonda` for the patch, and to Thomas Grainger and Jun Omae for patch workshopping.
- #2002: (via #2003) Switch from module-global to thread-local storage when recording thread IDs for a logging helper; this should avoid one flavor of memory leak for long-running processes. Catch & patch via Richard Kojedzinszky.
- #1963: (via #1977) Certificate-based pubkey auth was inadvertently broken when adding SHA2 support; this has been fixed. Reported by Erik Forsberg and fixed by Jun Omae.
- : (CVE-2022-24302) Creation of new private key files using `PKey` subclasses was subject to a race condition between file creation & mode modification, which could be exploited by an attacker with knowledge of where the Paramiko-using code would write out such files.

  This has been patched by using `os.open` and `os.fdopen` to ensure new files are opened with the correct mode immediately. We've left the subsequent explicit `chmod` in place to minimize any possible disruption, though it may get removed in future backwards-incompatible updates.

  Thanks to Jan Schejbal for the report & feedback on the solution, and to Jeremy Katz at Tidelift for coordinating the disclosure.
- #2001: Fix Python 2 compatibility breakage introduced in 2.10.1. Spotted by Christian Hammond.

> **Warning:** This is almost certainly the last time we will fix Python 2 related errors! Please see the roadmap.

- #2002: (via #2003) Switch from module-global to thread-local storage when recording thread IDs for a logging helper; this should avoid one flavor of memory leak for long-running processes. Catch & patch via Richard Kojedzinszky.
- #1963: (via #1977) Certificate-based pubkey auth was inadvertently broken when adding SHA2 support; this has been fixed. Reported by Erik Forsberg and fixed by Jun Omae.
- #2001: Fix Python 2 compatibility breakage introduced in 2.10.1. Spotted by Christian Hammond.

> **Warning:** This is almost certainly the last time we will fix Python 2 related errors! Please see the roadmap.

- : (CVE-2022-24302) Creation of new private key files using `PKey` subclasses was subject to a race condition between file creation & mode modification, which could be exploited by an attacker with knowledge of where the Paramiko-using code would write out such files.

  This has been patched by using `os.open` and `os.fdopen` to ensure new files are opened with the correct mode immediately. We've left the subsequent explicit `chmod` in place to minimize any possible disruption, though it may get removed in future backwards-incompatible updates.

  Thanks to Jan Schejbal for the report & feedback on the solution, and to Jeremy Katz at Tidelift for coordinating the disclosure.

- #1509: (via #1868, #1837) Add support for OpenSSH's Windows agent as a fallback when Putty/WinPageant isn't available or functional. Reported by `@benj56` with patches/PRs from `@lewgordon` and Patrick Spendrin.

- #1976: Add support for the `%C` token when parsing SSH config files. Foundational PR submitted by `@jbrand42`.

- #892: Significantly speed up low-level read/write actions on `SFTPFile` objects by using `bytearray`/`memoryview`. This is unlikely to change anything for users of the higher level methods like `SFTPClient.get` or `SFTPClient.getfo`, but users of `SFTPClient.open` will likely see orders of magnitude improvements for files larger than a few megabytes in size.

  Thanks to `@jkji` for the original report and to Sevastian Tchernov for the patch.

- #1985: Add `six` explicitly to install-requires; it snuck into active use at some point but has only been indicated by transitive dependency on `bcrypt` until they somewhat-recently dropped it. This will be short-lived until we drop Python 2 support. Thanks to Sondre Lillebø Gundersen for catch & patch.

- : Enhanced log output when connecting to servers that do not support `server-sig-algs` extensions, making the new-as-of-2.9 defaulting to SHA2 pubkey algorithms more obvious when it kicks in.

- : Connecting to servers which support `server-sig-algs` but which have no overlap between that list and what a Paramiko client supports, now raise an exception instead of defaulting to `rsa-sha2-512` (since the use of `server-sig-algs` allows us to know what the server supports).

- #1955: Server-side support for `rsa-sha2-256` and `ssh-rsa` wasn't fully operable after 2.9.0's release (signatures for RSA pubkeys were always run through `rsa-sha2-512` instead). Report and early stab at a fix courtesy of Jun Omae.

- #1643: (also #1925, #1644, #1326) Add support for SHA-2 variants of RSA key verification algorithms (as described in **RFC 8332**) as well as limited SSH extension negotiation (**RFC 8308**).

  > **Warning:** This change is slightly backwards incompatible, insofar as action is required if your target systems do not support either RSA2 or the `server-sig-algs` protocol extension.
  >
  > Specifically, you need to specify `disabled_algorithms={'keys': ['rsa-sha2-256', 'rsa-sha2-512']}` in either `SSHClient` or `Transport`. See below for details on why.

  How SSH servers/clients decide when and how to use this functionality can be complicated; Paramiko's support is as follows:

  - Client verification of server host key during key exchange will now prefer `rsa-sha2-512`, `rsa-sha2-256`, and legacy `ssh-rsa` algorithms, in that order, instead of just `ssh-rsa`.
    - Note that the preference order of other algorithm families such as `ed25519` and `ecdsa` has not changed; for example, those two groups are still preferred over RSA.
  - Server mode will now offer all 3 RSA algorithms for host key verification during key exchange, similar to client mode, if it has been configured with an RSA host key.

  - Client mode key exchange now sends the `ext-info-c` flag signaling support for `MSG_EXT_INFO`, and support for parsing the latter (specifically, its `server-sig-algs` flag) has been added.
  - Client mode, when performing public key authentication with an RSA key or cert, will act as follows:
    * In all cases, the list of algorithms to consider is based on the new `preferred_pubkeys` list (see below) and `disabled_algorithms` (specifically, its `pubkeys` key); this list, like with host keys, prefers SHA2-512, SHA2-256 and SHA1, in that order.
    * When the server does not send `server-sig-algs`, Paramiko will attempt the first algorithm in the above list. Clients connecting to legacy servers should thus use `disabled_algorithms` to turn off SHA2.
    * When the server does send `server-sig-algs`, the first algorithm supported by both ends is used, or if there is none, it falls back to the previous behavior.
  - SSH agent support grew the ability to specify algorithm flags when requesting private key signatures; this is now used to forward SHA2 algorithms when appropriate.
  - Server mode is now capable of pubkey auth involving SHA-2 signatures from clients, provided one's server implementation actually provides for doing so.
    * This includes basic support for sending `MSG_EXT_INFO` (containing `server-sig-algs` only) to clients advertising `ext-info-c` in their key exchange list.

  In order to implement the above, the following API additions were made:

  - `PKey.sign_ssh_data`: Grew an extra, optional `algorithm` keyword argument (defaulting to `None` for most subclasses, and to `"ssh-rsa"` for `RSAKey`).
  - A new `SSHException` subclass was added, `IncompatiblePeer`, and is raised in all spots where key exchange aborts due to algorithmic incompatibility.
    * Like all other exceptions in that module, it inherits from `SSHException`, and as we did not change anything else about the raising (i.e. the attributes and message text are the same) this change is backwards compatible.
  - `Transport` grew a `_preferred_pubkeys` attribute and matching `preferred_pubkeys` property to match the other, kex-focused, such members. This allows client pubkey authentication to honor the `disabled_algorithms` feature.

  Thanks to Krisztián Kovács for the report and an early stab at a patch, as well as the numerous users who submitted feedback on the issue, including but not limited to: Christopher Rabotin, Sam Bull, and Manfred Kaiser.

- : (also #908) Update `PKey` and subclasses to compare (`__eq__`) via direct field/attribute comparison instead of hashing (while retaining the existing behavior of `__hash__` via a slight refactor). Big thanks to Josh Snyder and Jun Omae for the reports, and to Josh Snyder for reproduction details & patch.

  Warning: This fixes a security flaw! If you are running Paramiko on 32-bit systems with low entropy (such as any 32-bit Python 2, or a 32-bit Python 3 which is running with `PYTHONHASHSEED=0`) it is possible for an attacker to craft a new keypair from an exfiltrated public key, which Paramiko would consider equal to the original key.

  This could enable attacks such as, but not limited to, the following:

  - Paramiko server processes would incorrectly authenticate the attacker (using their generated private key) as if they were the victim. We see this as the most plausible attack using this flaw.
  - Paramiko client processes would incorrectly validate a connected server (when host key verification is enabled) while subjected to a man-in-the-middle attack. This impacts more users than the server-side

> version, but also carries higher requirements for the attacker, namely successful DNS poisoning or other MITM techniques.

- #1257: (also #1266) Update RSA and ECDSA key decoding subroutines to correctly catch exception types thrown by modern versions of Cryptography (specifically `TypeError` and its internal `UnsupportedAlgorithm`). These exception classes will now become `SSHException` instances instead of bubbling up. Thanks to Ignat Semenov for the report and `@tylergarcianet` for an early patch.
- #1024: Deleting items from `HostKeys` would incorrectly raise `KeyError` even for valid keys, due to a logic bug. This has been fixed. Report & patch credit: Jia Zhang.
- #985: (via #992) Fix listdir failure when server uses a locale. Now on Python 2.7 `SFTPAttributes` will decode abbreviated month names correctly rather than raise `UnicodeDecodeError`. Patch courtesy of Martin Packman.
- #1846: Add a `prefetch` keyword argument to `SFTPClient.get`/`SFTPClient.getfo` so users who need to skip SFTP prefetching are able to conditionally turn it off. Thanks to Github user `@h3ll0r` for the PR.
- #1462: (via #1882) Newer server-side key exchange algorithms not intended to use SHA1 (`diffie-hellman-group14-sha256`, `diffie-hellman-group16-sha512`) were incorrectly using SHA1 after all, due to a bug causing them to ignore the `hash_algo` class attribute. This has been corrected. Big thanks to `@miverson` for the report and to Benno Rice for the patch.
- #1722: Remove leading whitespace from OpenSSH RSA test suite static key fixture, to conform better to spec. Credit: Alex Gaynor.
- #1727: Add missing test suite fixtures directory to MANIFEST.in, reinstating the ability to run Paramiko's tests from an sdist tarball. Thanks to Sandro Tosi for reporting the issue and to Blazej Michalik for the PR.
- : Update our CI to catch issues with sdist generation, installation and testing.
- : Administrivia overhaul, including but not limited to:
  - Migrate CI to CircleCI
  - Primary dev branch is now `main` (renamed)
  - Many README edits for clarity, modernization etc; including a bunch more (and consistent) status badges & unification with main project site index
  - PyPI page much more fleshed out (long_description is now filled in with the README; sidebar links expanded; etc)
  - flake8, pytest configs split out of setup.cfg into their own files
  - Invoke/invocations (used by maintainers/contributors) upgraded to modern versions
- #1723: Fix incorrectly swapped order of `p` and `q` numbers when loading OpenSSH-format RSA private keys. At minimum this should address a slowdown when using such keys, and it also means Paramiko works with Cryptography 3.1 and above (which complains strenuously when this problem appears). Thanks to Alex Gaynor for the patch.
- : Fix incorrect string formatting causing unhelpful error message annotation when using Kerberos/GSSAPI. (Thanks, newer version of flake8!)
- #1722: Remove leading whitespace from OpenSSH RSA test suite static key fixture, to conform better to spec. Credit: Alex Gaynor.
- #1727: Add missing test suite fixtures directory to MANIFEST.in, reinstating the ability to run Paramiko's tests from an sdist tarball. Thanks to Sandro Tosi for reporting the issue and to Blazej Michalik for the PR.
- : Update our CI to catch issues with sdist generation, installation and testing.

- #1565: (via #1566) Fix a bug in support for ECDSA keys under the newly supported OpenSSH key format. Thanks to Pierce Lopez for the patch.
- #1567: The new-style private key format (added in 2.7) suffered from an unpadding bug which had been fixed earlier for Ed25519 (as that key type has always used the newer format). That fix has been refactored and applied to the base key class, courtesy of Pierce Lopez.
- : Add new convenience classmethod constructors to `SSHConfig`: `from_text`, `from_file`, and `from_path`. No more annoying two-step process!
- #897: Implement most 'canonical hostname' `ssh_config` functionality (`CanonicalizeHostname`, `CanonicalDomains`, `CanonicalizeFallbackLocal`, and `CanonicalizeMaxDots`; `CanonicalizePermittedCNAMEs` has **not** yet been implemented). All were previously silently ignored. Reported by Michael Leinartas.
- #717: Implement support for the `Match` keyword in `ssh_config` files. Previously, this keyword was simply ignored & keywords inside such blocks were treated as if they were part of the previous block. Thanks to Michael Leinartas for the initial patchset.

Note: This feature adds a new *optional install dependency*, Invoke, for managing `Match exec` subprocesses.

- : A couple of outright `SSHConfig` parse errors were previously represented as vanilla `Exception` instances; as part of recent feature work a more specific exception class, `ConfigParseError`, has been created. It is now also used in those older spots, which is naturally backwards compatible.
- #602: (via #1343, #1313, #618) Implement support for OpenSSH 6.5-style private key files (typically denoted as having `BEGIN OPENSSH PRIVATE KEY` headers instead of PEM format's `BEGIN RSA PRIVATE KEY` or similar). If you were getting any sort of weird auth error from "modern" keys generated on newer operating system releases (such as macOS Mojave), this is the first update to try.

  Major thanks to everyone who contributed or tested versions of the patch, including but not limited to: Kevin Abel, Michiel Tiller, Pierce Lopez, and Jared Hobbs.
- : Perform deduplication of `IdentityFile` contents during `ssh_config` parsing; previously, if your config would result in the same value being encountered more than once, `IdentityFile` would contain that many copies of the same string.
- : Paramiko's use of `subprocess` for `ProxyCommand` support is conditionally imported to prevent issues on limited interpreter platforms like Google Compute Engine. However, any resulting `ImportError` was lost instead of preserved for raising (in the rare cases where a user tried leveraging `ProxyCommand` in such an environment). This has been fixed.
- : `ssh_config` token expansion used a different method of determining the local username (`$USER` env var), compared to what the (much older) client connection code does (`getpass.getuser`, which includes `$USER` but may check other variables first, and is generally much more comprehensive). Both modules now use `getpass.getuser`.
- : Explicitly document which ssh_config features we currently support. Previously users just had to guess, which is simply no good.
- : Additional *installation* `extras_require` "flavors" (`ed25519`, `invoke`, and `all`) have been added to our packaging metadata; see the install docs for details.
- #1463: Add a new keyword argument to `SSHClient.connect` and `Transport`, `disabled_algorithms`, which allows selectively disabling one or more kex/key/cipher/etc algorithms. This can be useful when disabling algorithms your target server (or client) does not support cleanly, or to work around unpatched bugs in Paramiko's own implementation thereof.

- #322: `SSHClient.exec_command` previously returned a naive `ChannelFile` object for its `stdin` value; such objects don't know to properly shut down the remote end's stdin when they `.close()`. This lead to issues (such as hangs) when running remote commands that read from stdin.

  A new subclass, `ChannelStdinFile`, has been created which closes remote stdin when it itself is closed. `exec_command` has been updated to use that class for its `stdin` return value.

  Thanks to Brandon Rhodes for the report & steps to reproduce.

- #1311: (for #584, replacing #1166) Add backwards-compatible support for the `gssapi` GSSAPI library, as the previous backend (`python-gssapi`) has since become defunct. This change also includes tests for the GSSAPI functionality.

  Big thanks to Anselm Kruis for the patch and to Sebastian Deiß (author of our initial GSSAPI functionality) for review.

  Note: This feature also adds `setup.py` 'extras' support for installing Paramiko as `paramiko[gssapi]`, which pulls in the optional dependencies you had to get by hand previously.

  Note: To be very clear, this patch **does not** remove support for the older `python-gssapi` library. We *may* remove that support in a later release, but for now, either library will work. Please upgrade to `gssapi` when you can, however, as `python-gssapi` is no longer maintained upstream.

- #1440: (with initial fixes via #1460) Tweak many exception classes so their string representations are more human-friendly; this also includes incidental changes to some `super()` calls.

  The definitions of exceptions' `__init__` methods have *not* changed, nor have any log messages been altered, so this should be backwards compatible for everything except the actual exceptions' `__str__()` outputs.

  Thanks to Fabian Büchler for original report & Pierce Lopez for the foundational patch.

- #1306: (via #1400) Fix Ed25519 key handling so certain key comment lengths don't cause `SSHException("Invalid key")` (this was technically a bug in how padding, or lack thereof, is calculated/interpreted). Thanks to `@parke` for the bug report & Pierce Lopez for the patch.

- #1306: (via #1400) Fix Ed25519 key handling so certain key comment lengths don't cause `SSHException("Invalid key")` (this was technically a bug in how padding, or lack thereof, is calculated/interpreted). Thanks to `@parke` for the bug report & Pierce Lopez for the patch.

- #1378: Add support for the modern (as of Python 3.3) import location of `MutableMapping` (used in host key management) to avoid the old location becoming deprecated in Python 3.8. Thanks to Josh Karpel for catch & patch.

- #1212: Updated `SSHConfig.lookup` so it returns a new, type-casting-friendly dict subclass (`SSHConfigDict`) in lieu of dict literals. This ought to be backwards compatible, and allows an easier way to check boolean or int type `ssh_config` values. Thanks to Chris Rose for the patch.

- #532: (via #1384 and #1258) Add support for Curve25519 key exchange (aka `curve25519-sha256@libssh.org`). Thanks to Alex Gaynor and Dan Fuhry for supplying patches.

- #1233: (also #1229, #1332) Add support for encrypt-then-MAC (ETM) schemes (`hmac-sha2-256-etm@openssh.com`, `hmac-sha2-512-etm@openssh.com`) and two newer Diffie-Hellman group key exchange algorithms (`group14`, using SHA256; and `group16`, using SHA512). Patch courtesy of Edgar Sousa.

- #1191: Update our install docs with (somewhat) recently added additional dependencies; we previously only required Cryptography, but the docs never got updated after we incurred `bcrypt` and `pynacl` requirements for Ed25519 key support.

Additionally, `pyasn1` was never actually hard-required; it was necessary during a development branch, and is used by the optional GSSAPI support, but is not required for regular installation. Thus, it has been removed from our `setup.py` and its imports in the GSSAPI code made optional.

Credit to `@stevenwinfield` for highlighting the outdated install docs.

- #1262: Add `*.pub` files to the MANIFEST so distributed source packages contain some necessary test assets. Credit: Alexander Kapshuna.
- #1378: Add support for the modern (as of Python 3.3) import location of `MutableMapping` (used in host key management) to avoid the old location becoming deprecated in Python 3.8. Thanks to Josh Karpel for catch & patch.
- #1379: (also #1369) Raise Cryptography dependency requirement to version 2.5 (from 1.5) and update some deprecated uses of its API.

  This removes a bunch of warnings of the style `CryptographyDeprecationWarning: encode_point has been deprecated on EllipticCurvePublicNumbers and will be removed in a future version. Please use EllipticCurvePublicKey. public_bytes to obtain both compressed and uncompressed point encoding` and similar, which users who had eventually upgraded to Cryptography 2.x would encounter.

  > **Warning:** This change is backwards incompatible **if** you are unable to upgrade your version of Cryptography. Please see Cryptography's own changelog for details on what may change when you upgrade; for the most part the only changes involved dropping older Python versions (such as 2.6, 3.3, or some PyPy editions) which Paramiko itself has already dropped.

- #1283: Fix exploit (CVE-2018-1000805) in Paramiko's server mode (**not** client mode) where hostile clients could trick the server into thinking they were authenticated without actually submitting valid authentication.

  Specifically, steps have been taken to start separating client and server related message types in the message handling tables within `Transport` and `AuthHandler`; this work is not complete but enough has been performed to close off this particular exploit (which was the only obvious such exploit for this particular channel).

  Thanks to Daniel Hoffman for the detailed report.

- : Modify protocol message handling such that `Transport` does not respond to `MSG_UNIMPLEMENTED` with its own `MSG_UNIMPLEMENTED`. This behavior probably didn't cause any outright errors, but it doesn't seem to conform to the RFCs and could cause (non-infinite) feedback loops in some scenarios (usually those involving Paramiko on both ends).

- #1262: Add `*.pub` files to the MANIFEST so distributed source packages contain some necessary test assets. Credit: Alexander Kapshuna.
- #1283: Fix exploit (CVE-2018-1000805) in Paramiko's server mode (**not** client mode) where hostile clients could trick the server into thinking they were authenticated without actually submitting valid authentication.

  Specifically, steps have been taken to start separating client and server related message types in the message handling tables within `Transport` and `AuthHandler`; this work is not complete but enough has been performed to close off this particular exploit (which was the only obvious such exploit for this particular channel).

  Thanks to Daniel Hoffman for the detailed report.

- : Modify protocol message handling such that `Transport` does not respond to `MSG_UNIMPLEMENTED` with its own `MSG_UNIMPLEMENTED`. This behavior probably didn't cause any outright errors, but it doesn't seem to conform to the RFCs and could cause (non-infinite) feedback loops in some scenarios (usually those involving Paramiko on both ends).
- #1262: Add `*.pub` files to the MANIFEST so distributed source packages contain some necessary test assets. Credit: Alexander Kapshuna.

- #1283: Fix exploit (CVE-2018-1000805) in Paramiko's server mode (**not** client mode) where hostile clients could trick the server into thinking they were authenticated without actually submitting valid authentication.

  Specifically, steps have been taken to start separating client and server related message types in the message handling tables within `Transport` and `AuthHandler`; this work is not complete but enough has been performed to close off this particular exploit (which was the only obvious such exploit for this particular channel).

  Thanks to Daniel Hoffman for the detailed report.

- : Modify protocol message handling such that `Transport` does not respond to `MSG_UNIMPLEMENTED` with its own `MSG_UNIMPLEMENTED`. This behavior probably didn't cause any outright errors, but it doesn't seem to conform to the RFCs and could cause (non-infinite) feedback loops in some scenarios (usually those involving Paramiko on both ends).

- #1262: Add `*.pub` files to the MANIFEST so distributed source packages contain some necessary test assets. Credit: Alexander Kapshuna.

- #1283: Fix exploit (CVE-2018-1000805) in Paramiko's server mode (**not** client mode) where hostile clients could trick the server into thinking they were authenticated without actually submitting valid authentication.

  Specifically, steps have been taken to start separating client and server related message types in the message handling tables within `Transport` and `AuthHandler`; this work is not complete but enough has been performed to close off this particular exploit (which was the only obvious such exploit for this particular channel).

  Thanks to Daniel Hoffman for the detailed report.

- : Modify protocol message handling such that `Transport` does not respond to `MSG_UNIMPLEMENTED` with its own `MSG_UNIMPLEMENTED`. This behavior probably didn't cause any outright errors, but it doesn't seem to conform to the RFCs and could cause (non-infinite) feedback loops in some scenarios (usually those involving Paramiko on both ends).

- #1262: Add `*.pub` files to the MANIFEST so distributed source packages contain some necessary test assets. Credit: Alexander Kapshuna.

- #1283: Fix exploit (CVE-2018-1000805) in Paramiko's server mode (**not** client mode) where hostile clients could trick the server into thinking they were authenticated without actually submitting valid authentication.

  Specifically, steps have been taken to start separating client and server related message types in the message handling tables within `Transport` and `AuthHandler`; this work is not complete but enough has been performed to close off this particular exploit (which was the only obvious such exploit for this particular channel).

  Thanks to Daniel Hoffman for the detailed report.

- : Modify protocol message handling such that `Transport` does not respond to `MSG_UNIMPLEMENTED` with its own `MSG_UNIMPLEMENTED`. This behavior probably didn't cause any outright errors, but it doesn't seem to conform to the RFCs and could cause (non-infinite) feedback loops in some scenarios (usually those involving Paramiko on both ends).

- #1262: Add `*.pub` files to the MANIFEST so distributed source packages contain some necessary test assets. Credit: Alexander Kapshuna.

- #1039: Ed25519 auth key decryption raised an unexpected exception when given a unicode password string (typical in python 3). Report by Theodor van Nahl and fix by Pierce Lopez.

- #1168: Add newer key classes for Ed25519 and ECDSA to `paramiko.__all__` so that code introspecting that attribute, or using `from paramiko import *` (such as some IDEs) sees them. Thanks to `@patriksevallius` for the patch.

- #1175: Fix a security flaw (CVE-2018-7750) in Paramiko's server mode (emphasis on **server** mode; this does **not** impact *client* use!) where authentication status was not checked before processing channel-open and other requests typically only sent after authenticating. Big thanks to Matthijs Kooijman for the report.

- #1108: Rename a private method keyword argument (which was named `async`) so that we're compatible with the upcoming Python 3.7 release (where `async` is a new keyword.) Thanks to `@vEpiphyte` for the report.
- #1039: Ed25519 auth key decryption raised an unexpected exception when given a unicode password string (typical in python 3). Report by Theodor van Nahl and fix by Pierce Lopez.
- #1168: Add newer key classes for Ed25519 and ECDSA to `paramiko.__all__` so that code introspecting that attribute, or using `from paramiko import *` (such as some IDEs) sees them. Thanks to `@patriksevallius` for the patch.
- #1175: Fix a security flaw (CVE-2018-7750) in Paramiko's server mode (emphasis on **server** mode; this does **not** impact *client* use!) where authentication status was not checked before processing channel-open and other requests typically only sent after authenticating. Big thanks to Matthijs Kooijman for the report.
- : Include LICENSE file in wheel archives.
- #1071: Certificate support broke the no-certificate case for Ed25519 keys (symptom is an `AttributeError` about `public_blob`.) This went uncaught due to cert autoload behavior (i.e. our test suite never actually ran the no-cert case, because the cert existed!) Both issues have been fixed. Thanks to John Hu for the report.
- #1108: Rename a private method keyword argument (which was named `async`) so that we're compatible with the upcoming Python 3.7 release (where `async` is a new keyword.) Thanks to `@vEpiphyte` for the report.
- #1039: Ed25519 auth key decryption raised an unexpected exception when given a unicode password string (typical in python 3). Report by Theodor van Nahl and fix by Pierce Lopez.
- #1168: Add newer key classes for Ed25519 and ECDSA to `paramiko.__all__` so that code introspecting that attribute, or using `from paramiko import *` (such as some IDEs) sees them. Thanks to `@patriksevallius` for the patch.
- #1175: Fix a security flaw (CVE-2018-7750) in Paramiko's server mode (emphasis on **server** mode; this does **not** impact *client* use!) where authentication status was not checked before processing channel-open and other requests typically only sent after authenticating. Big thanks to Matthijs Kooijman for the report.
- : Include LICENSE file in wheel archives.
- #1071: Certificate support broke the no-certificate case for Ed25519 keys (symptom is an `AttributeError` about `public_blob`.) This went uncaught due to cert autoload behavior (i.e. our test suite never actually ran the no-cert case, because the cert existed!) Both issues have been fixed. Thanks to John Hu for the report.
- #1108: Rename a private method keyword argument (which was named `async`) so that we're compatible with the upcoming Python 3.7 release (where `async` is a new keyword.) Thanks to `@vEpiphyte` for the report.
- #1039: Ed25519 auth key decryption raised an unexpected exception when given a unicode password string (typical in python 3). Report by Theodor van Nahl and fix by Pierce Lopez.
- #1168: Add newer key classes for Ed25519 and ECDSA to `paramiko.__all__` so that code introspecting that attribute, or using `from paramiko import *` (such as some IDEs) sees them. Thanks to `@patriksevallius` for the patch.
- #1175: Fix a security flaw (CVE-2018-7750) in Paramiko's server mode (emphasis on **server** mode; this does **not** impact *client* use!) where authentication status was not checked before processing channel-open and other requests typically only sent after authenticating. Big thanks to Matthijs Kooijman for the report.
- : Include LICENSE file in wheel archives.
- #1071: Certificate support broke the no-certificate case for Ed25519 keys (symptom is an `AttributeError` about `public_blob`.) This went uncaught due to cert autoload behavior (i.e. our test suite never actually ran the no-cert case, because the cert existed!) Both issues have been fixed. Thanks to John Hu for the report.
- #1108: Rename a private method keyword argument (which was named `async`) so that we're compatible with the upcoming Python 3.7 release (where `async` is a new keyword.) Thanks to `@vEpiphyte` for the report.

- #1039: Ed25519 auth key decryption raised an unexpected exception when given a unicode password string (typical in python 3). Report by Theodor van Nahl and fix by Pierce Lopez.
- #1168: Add newer key classes for Ed25519 and ECDSA to `paramiko.__all__` so that code introspecting that attribute, or using `from paramiko import *` (such as some IDEs) sees them. Thanks to `@patriksevallius` for the patch.
- #1175: Fix a security flaw (CVE-2018-7750) in Paramiko's server mode (emphasis on **server** mode; this does **not** impact *client* use!) where authentication status was not checked before processing channel-open and other requests typically only sent after authenticating. Big thanks to Matthijs Kooijman for the report.
- : Include LICENSE file in wheel archives.
- #1108: Rename a private method keyword argument (which was named `async`) so that we're compatible with the upcoming Python 3.7 release (where `async` is a new keyword.) Thanks to `@vEpiphyte` for the report.
- #1175: Fix a security flaw (CVE-2018-7750) in Paramiko's server mode (emphasis on **server** mode; this does **not** impact *client* use!) where authentication status was not checked before processing channel-open and other requests typically only sent after authenticating. Big thanks to Matthijs Kooijman for the report.
- #1055: (also #1056, #1057, #1058, #1059) Fix up host-key checking in our GSSAPI support, which was previously using an incorrect API call. Thanks to Anselm Kruis for the patches.
- #1108: Rename a private method keyword argument (which was named `async`) so that we're compatible with the upcoming Python 3.7 release (where `async` is a new keyword.) Thanks to `@vEpiphyte` for the report.
- #1175: Fix a security flaw (CVE-2018-7750) in Paramiko's server mode (emphasis on **server** mode; this does **not** impact *client* use!) where authentication status was not checked before processing channel-open and other requests typically only sent after authenticating. Big thanks to Matthijs Kooijman for the report.
- : Add a new `passphrase` kwarg to `SSHClient.connect` so users may disambiguate key-decryption passphrases from password-auth passwords. (This is a backwards compatible change; `password` will still pull double duty as a passphrase when `passphrase` is not given.)
- #1070: Drop Python 2.6 and Python 3.3 support; now only 2.7 and 3.4+ are supported. If you're unable to upgrade from 2.6 or 3.3, please stick to the Paramiko 2.3.x (or below) release lines.
- : Include LICENSE file in wheel archives.
- #1100: Updated the test suite & related docs/metadata/config to be compatible with pytest instead of using the old, custom, crufty unittest-based `test.py`.

  This includes marking known-slow tests (mostly the SFTP ones) so they can be filtered out by `inv test`'s default behavior; as well as other minor tweaks to test collection and/or display (for example, GSSAPI tests are collected, but skipped, instead of not even being collected by default as in `test.py`.)
- : Update `tearDown` of client test suite to avoid hangs due to eternally blocking `accept()` calls on the internal server thread (which can occur when test code raises an exception before actually connecting to the server.)
- #1071: Certificate support broke the no-certificate case for Ed25519 keys (symptom is an `AttributeError` about `public_blob`.) This went uncaught due to cert autoload behavior (i.e. our test suite never actually ran the no-cert case, because the cert existed!) Both issues have been fixed. Thanks to John Hu for the report.
- #1042: (also partially #531) Implement basic client-side certificate authentication (as per the OpenSSH vendor extension.)

  The core implementation is `PKey.load_certificate` and its corresponding `.public_blob` attribute on key objects, which is honored in the auth and transport modules. Additionally, `SSHClient.connect` will now automatically load certificate data alongside private key data when one has appropriately-named cert files (e.g. `id_rsa-cert.pub`) - see its docstring for details.

  Thanks to Jason Rigby for a first draft (#531) and to Paul Kapp for the second draft, upon which the current functionality has been based (with modifications.)

Note: This support is client-focused; Paramiko-driven server code is capable of handling cert-bearing pubkey auth packets, *but* it does not interpret any cert-specific fields, so the end result is functionally identical to a vanilla pubkey auth process (and thus requires e.g. prepopulated authorized-keys data.) We expect full server-side cert support to follow later.

- #1013: Added pre-authentication banner support for the server interface (`ServerInterface.get_banner` plus related support in `Transport`/`AuthHandler`.) Patch courtesy of Dennis Kaarsemaker.
- #1026: Update `Ed25519Key` so its constructor offers the same `file_obj` parameter as its sibling key classes. Credit: Michal Kuffa.
- #1063: Add a `gss_trust_dns` option to `Client` and `Transport` to allow explicitly setting whether or not DNS canonicalization should occur when using GSSAPI. Thanks to Richard E. Silverman for the report & Sebastian Deiß for initial patchset.
- #60: (via #1037) Paramiko originally defaulted to zlib compression level 9 (when one connects with `compression=True`; it defaults to off.) This has been found to be quite wasteful and tends to cause much longer transfers in most cases, than is necessary.

  OpenSSH defaults to compression level 6, which is a much more reasonable setting (nearly identical compression characteristics but noticeably, sometimes significantly, faster transmission); Paramiko now uses this value instead.

  Thanks to Damien Dubé for the report and `@DrNeutron` for investigating & submitting the patch.
- #1012: (via #1016) Enhance documentation around the new `SFTP.posix_rename` method so it's referenced in the 'standard' `rename` method for increased visibility. Thanks to Marius Flage for the report.
- #1041: Modify logic around explicit disconnect messages, and unknown-channel situations, so that they rely on centralized shutdown code instead of running their own. This is at worst removing some unnecessary code, and may help with some situations where Paramiko hangs at the end of a session. Thanks to Paul Kapp for the patch.
- : Display exception type and message when logging auth-rejection messages (ones reading `Auth rejected: unsupported or mangled public key`); previously this error case had a bare except and did not display exactly why the key failed. It will now append info such as `KeyError: 'some-unknown-type-string'` or similar.
- : Ed25519 keys never got proper API documentation support; this has been fixed.
- #979: Update how we use Cryptography's signature/verification methods so we aren't relying on a deprecated API. Thanks to Paul Kehrer for the patch.

  Warning: This bumps the minimum Cryptography version from 1.1 to 1.5. Such an upgrade should be backwards compatible and easy to do. See their changelog for additional details.
- #945: (backport of #910 and re: #865) SSHClient now requests the type of host key it has (e.g. from known_hosts) and does not consider a different type to be a "Missing" host key. This fixes a common case where an ECDSA key is in known_hosts and the server also has an RSA host key. Thanks to Pierce Lopez.
- #1055: (also #1056, #1057, #1058, #1059) Fix up host-key checking in our GSSAPI support, which was previously using an incorrect API call. Thanks to Anselm Kruis for the patches.
- #1060: Fix key exchange (kex) algorithm list for GSSAPI authentication; previously, the list used solely out-of-date algorithms, and now contains newer ones listed preferentially before the old. Credit: Anselm Kruis.

- #1061: Clean up GSSAPI authentication procedures so they do not prevent normal fallback to other authentication methods on failure. (In other words, presence of GSSAPI functionality on a target server precluded use of _any_ other auth type if the user was unable to pass GSSAPI auth.) Patch via Anselm Kruis.
- #1065: Add rekeying support to GSSAPI connections, which was erroneously missing. Without this fix, any attempt to renegotiate the transport keys for a `gss-kex`-authed `Transport` would cause a MIC failure and terminate the connection. Thanks to Sebastian Deiß and Anselm Kruis for the patch.
- #990: The (added in 2.2.0) `bcrypt` dependency should have been on version 3.1.3 or greater (was initially set to 3.0.0 or greater.) Thanks to Paul Howarth for the report.
- #993: Ed25519 host keys were not comparable/hashable, causing an exception if such a key existed in a `known_hosts` file. Thanks to Oleh Prypin for the report and Pierce Lopez for the fix.
- #945: (backport of #910 and re: #865) SSHClient now requests the type of host key it has (e.g. from known_hosts) and does not consider a different type to be a "Missing" host key. This fixes a common case where an ECDSA key is in known_hosts and the server also has an RSA host key. Thanks to Pierce Lopez.
- #1055: (also #1056, #1057, #1058, #1059) Fix up host-key checking in our GSSAPI support, which was previously using an incorrect API call. Thanks to Anselm Kruis for the patches.
- #1060: Fix key exchange (kex) algorithm list for GSSAPI authentication; previously, the list used solely out-of-date algorithms, and now contains newer ones listed preferentially before the old. Credit: Anselm Kruis.
- #1061: Clean up GSSAPI authentication procedures so they do not prevent normal fallback to other authentication methods on failure. (In other words, presence of GSSAPI functionality on a target server precluded use of _any_ other auth type if the user was unable to pass GSSAPI auth.) Patch via Anselm Kruis.
- #1065: Add rekeying support to GSSAPI connections, which was erroneously missing. Without this fix, any attempt to renegotiate the transport keys for a `gss-kex`-authed `Transport` would cause a MIC failure and terminate the connection. Thanks to Sebastian Deiß and Anselm Kruis for the patch.
- #990: The (added in 2.2.0) `bcrypt` dependency should have been on version 3.1.3 or greater (was initially set to 3.0.0 or greater.) Thanks to Paul Howarth for the report.
- #993: Ed25519 host keys were not comparable/hashable, causing an exception if such a key existed in a `known_hosts` file. Thanks to Oleh Prypin for the report and Pierce Lopez for the fix.
- #945: (backport of #910 and re: #865) SSHClient now requests the type of host key it has (e.g. from known_hosts) and does not consider a different type to be a "Missing" host key. This fixes a common case where an ECDSA key is in known_hosts and the server also has an RSA host key. Thanks to Pierce Lopez.
- #1055: (also #1056, #1057, #1058, #1059) Fix up host-key checking in our GSSAPI support, which was previously using an incorrect API call. Thanks to Anselm Kruis for the patches.
- #1060: Fix key exchange (kex) algorithm list for GSSAPI authentication; previously, the list used solely out-of-date algorithms, and now contains newer ones listed preferentially before the old. Credit: Anselm Kruis.
- #1061: Clean up GSSAPI authentication procedures so they do not prevent normal fallback to other authentication methods on failure. (In other words, presence of GSSAPI functionality on a target server precluded use of _any_ other auth type if the user was unable to pass GSSAPI auth.) Patch via Anselm Kruis.
- #1065: Add rekeying support to GSSAPI connections, which was erroneously missing. Without this fix, any attempt to renegotiate the transport keys for a `gss-kex`-authed `Transport` would cause a MIC failure and terminate the connection. Thanks to Sebastian Deiß and Anselm Kruis for the patch.
- #945: (backport of #910 and re: #865) SSHClient now requests the type of host key it has (e.g. from known_hosts) and does not consider a different type to be a "Missing" host key. This fixes a common case where an ECDSA key is in known_hosts and the server also has an RSA host key. Thanks to Pierce Lopez.
- #1055: (also #1056, #1057, #1058, #1059) Fix up host-key checking in our GSSAPI support, which was previously using an incorrect API call. Thanks to Anselm Kruis for the patches.

- #990: The (added in 2.2.0) `bcrypt` dependency should have been on version 3.1.3 or greater (was initially set to 3.0.0 or greater.) Thanks to Paul Howarth for the report.
- #993: Ed25519 host keys were not comparable/hashable, causing an exception if such a key existed in a `known_hosts` file. Thanks to Oleh Prypin for the report and Pierce Lopez for the fix.
- #325: (via #972) Add Ed25519 support, for both host keys and user authentication. Big thanks to Alex Gaynor for the patch.

Note: This change adds the `bcrypt` and `pynacl` Python libraries as dependencies. No C-level dependencies beyond those previously required (for Cryptography) have been added.

- #951: Add support for ECDH key exchange (kex), specifically the algorithms `ecdh-sha2-nistp256`, `ecdh-sha2-nistp384`, and `ecdh-sha2-nistp521`. They now come before the older `diffie-hellman-*` family of kex algorithms in the preferred-kex list. Thanks to Shashank Veerapaneni for the patch & Pierce Lopez for a follow-up.
- #857: Allow `SSHClient.set_missing_host_key_policy` to accept policy classes _or_ instances, instead of only instances, thus fixing a long-standing gotcha for unaware users.
- #869: Add an `auth_timeout` kwarg to `SSHClient.connect` (default: 30s) to avoid hangs when the remote end becomes unresponsive during the authentication step. Credit to `@timsavage`.

Note: This technically changes behavior, insofar as very slow auth steps >30s will now cause timeout exceptions instead of completing. We doubt most users will notice; those affected can simply give a higher value to `auth_timeout`.

- #65: (via #471) Add support for OpenSSH's SFTP `posix-rename` protocol extension (section 3.3 of OpenSSH's protocol extension document), via a new `posix_rename` method in `SFTPClient` and `SFTPServerInterface`. Thanks to Wren Turkal for the initial patch & Mika Pflüger for the enhanced, merged PR.
- #866: (also #838) Remove an old test-related file we don't support, and add PyPy to Travis-CI config. Thanks to Pierce Lopez for the final patch and Pedro Rodrigues for an earlier edition.
- #974: Overhaul the codebase to be PEP-8, etc, compliant (i.e. passes the maintainer's preferred flake8 configuration) and add a `flake8` step to the Travis config. Big thanks to Dorian Pula!
- : A big formatting pass to clean up an enormous number of invalid Sphinx reference links, discovered by switching to a modern, rigorous nitpicking doc-building mode.
- #956: Switch code coverage service from coveralls.io to codecov.io (& then disable the latter's auto-comments.) Thanks to Nikolai Røed Kristiansen for the patch.
- #921: Tighten up the `__hash__` implementation for various key classes; less code is good code. Thanks to Francisco Couzo for the patch.
- #906: Clean up a handful of outdated imports and related tweaks. Thanks to Pierce Lopez.
- #683: Make `util.log_to_file` append instead of replace. Thanks to `@vlcinsky` for the report.
- #949: SSHClient and Transport could cause a memory leak if there's a connection problem or protocol error, even if `Transport.close()` is called. Thanks Kyle Agronick for the discovery and investigation, and Pierce Lopez for assistance.
- #794: (via #981) Prior support for `ecdsa-sha2-nistp(384|521)` algorithms didn't fully extend to covering host keys, preventing connection to hosts which only offer these key types and no others. This is now fixed. Thanks to `@ncoult` and `@kasdoe` for reports and Pierce Lopez for the patch.

- #900: (via #911) Prefer newer `ecdsa-sha2-nistp` keys over RSA and DSA keys during host key selection. This improves compatibility with OpenSSH, both in terms of general behavior, and also re: ability to properly leverage OpenSSH-modified `known_hosts` files. Credit: `@kasdoe` for original report/PR and Pierce Lopez for the second draft.
- #667: The RC4/arcfour family of ciphers has been broken since version 2.0; but since the algorithm is now known to be completely insecure, we are opting to remove support outright instead of fixing it. Thanks to Alex Gaynor for catch & patch.
- #983: Move `sha1` above the now-arguably-broken `md5` in the list of preferred MAC algorithms, as an incremental security improvement for users whose target systems offer both. Credit: Pierce Lopez.
- #741: (also #809, #772; all via #912) Writing encrypted/password-protected private key files was silently broken since 2.0 due to an incorrect API call; this has been fixed.

  Includes a directly related fix, namely adding the ability to read `AES-256-CBC` ciphered private keys (which is now what we tend to write out as it is Cryptography's default private key cipher.)

  Thanks to `@virlos` for the original report, Chris Harris and `@ibuler` for initial draft PRs, and `@jhgorrell` for the final patch.
- #971: Allow any type implementing the buffer API to be used with `BufferedFile`, `Channel`, and `SFTPFile`. This resolves a regression introduced in 1.13 with the Python 3 porting changes, when using types such as `memoryview`. Credit: Martin Packman.
- #984: Enhance default cipher preference order such that `aes(192|256)-cbc` are preferred over `blowfish-cbc`. Thanks to Alex Gaynor.
- #865: SSHClient now requests the type of host key it has (e.g. from known_hosts) and does not consider a different type to be a "Missing" host key. This fixes a common case where an ECDSA key is in known_hosts and the server also has an RSA host key. Thanks to Pierce Lopez.
- #974: Overhaul the codebase to be PEP-8, etc, compliant (i.e. passes the maintainer's preferred flake8 configuration) and add a `flake8` step to the Travis config. Big thanks to Dorian Pula!
- : A big formatting pass to clean up an enormous number of invalid Sphinx reference links, discovered by switching to a modern, rigorous nitpicking doc-building mode.
- #956: Switch code coverage service from coveralls.io to codecov.io (& then disable the latter's auto-comments.) Thanks to Nikolai Røed Kristiansen for the patch.
- #683: Make `util.log_to_file` append instead of replace. Thanks to `@vlcinsky` for the report.
- #949: SSHClient and Transport could cause a memory leak if there's a connection problem or protocol error, even if `Transport.close()` is called. Thanks Kyle Agronick for the discovery and investigation, and Pierce Lopez for assistance.
- #794: (via #981) Prior support for `ecdsa-sha2-nistp(384|521)` algorithms didn't fully extend to covering host keys, preventing connection to hosts which only offer these key types and no others. This is now fixed. Thanks to `@ncoult` and `@kasdoe` for reports and Pierce Lopez for the patch.
- #900: (via #911) Prefer newer `ecdsa-sha2-nistp` keys over RSA and DSA keys during host key selection. This improves compatibility with OpenSSH, both in terms of general behavior, and also re: ability to properly leverage OpenSSH-modified `known_hosts` files. Credit: `@kasdoe` for original report/PR and Pierce Lopez for the second draft.
- #667: The RC4/arcfour family of ciphers has been broken since version 2.0; but since the algorithm is now known to be completely insecure, we are opting to remove support outright instead of fixing it. Thanks to Alex Gaynor for catch & patch.
- #983: Move `sha1` above the now-arguably-broken `md5` in the list of preferred MAC algorithms, as an incremental security improvement for users whose target systems offer both. Credit: Pierce Lopez.

- #741: (also #809, #772; all via #912) Writing encrypted/password-protected private key files was silently broken since 2.0 due to an incorrect API call; this has been fixed.

  Includes a directly related fix, namely adding the ability to read `AES-256-CBC` ciphered private keys (which is now what we tend to write out as it is Cryptography's default private key cipher.)

  Thanks to `@virlos` for the original report, Chris Harris and `@ibuler` for initial draft PRs, and `@jhgorrell` for the final patch.

- #971: Allow any type implementing the buffer API to be used with `BufferedFile`, `Channel`, and `SFTPFile`. This resolves a regression introduced in 1.13 with the Python 3 porting changes, when using types such as `memoryview`. Credit: Martin Packman.

- #984: Enhance default cipher preference order such that `aes(192|256)-cbc` are preferred over `blowfish-cbc`. Thanks to Alex Gaynor.

- #865: SSHClient now requests the type of host key it has (e.g. from known_hosts) and does not consider a different type to be a "Missing" host key. This fixes a common case where an ECDSA key is in known_hosts and the server also has an RSA host key. Thanks to Pierce Lopez.

- #974: Overhaul the codebase to be PEP-8, etc, compliant (i.e. passes the maintainer's preferred flake8 configuration) and add a `flake8` step to the Travis config. Big thanks to Dorian Pula!

- : A big formatting pass to clean up an enormous number of invalid Sphinx reference links, discovered by switching to a modern, rigorous nitpicking doc-building mode.

- #956: Switch code coverage service from coveralls.io to codecov.io (& then disable the latter's auto-comments.) Thanks to Nikolai Røed Kristiansen for the patch.

- #683: Make `util.log_to_file` append instead of replace. Thanks to `@vlcinsky` for the report.

- #949: SSHClient and Transport could cause a memory leak if there's a connection problem or protocol error, even if `Transport.close()` is called. Thanks Kyle Agronick for the discovery and investigation, and Pierce Lopez for assistance.

- #971: Allow any type implementing the buffer API to be used with `BufferedFile`, `Channel`, and `SFTPFile`. This resolves a regression introduced in 1.13 with the Python 3 porting changes, when using types such as `memoryview`. Credit: Martin Packman.

- #956: Switch code coverage service from coveralls.io to codecov.io (& then disable the latter's auto-comments.) Thanks to Nikolai Røed Kristiansen for the patch.

- #683: Make `util.log_to_file` append instead of replace. Thanks to `@vlcinsky` for the report.

- #949: SSHClient and Transport could cause a memory leak if there's a connection problem or protocol error, even if `Transport.close()` is called. Thanks Kyle Agronick for the discovery and investigation, and Pierce Lopez for assistance.

- #971: Allow any type implementing the buffer API to be used with `BufferedFile`, `Channel`, and `SFTPFile`. This resolves a regression introduced in 1.13 with the Python 3 porting changes, when using types such as `memoryview`. Credit: Martin Packman.

- #956: Switch code coverage service from coveralls.io to codecov.io (& then disable the latter's auto-comments.) Thanks to Nikolai Røed Kristiansen for the patch.

- #895: Fix a bug in server-mode concerning multiple interactive auth steps (which were incorrectly responded to). Thanks to Dennis Kaarsemaker for catch & patch.

- #44: (via #891) `SSHClient` now gives its internal `Transport` a handle on itself, preventing garbage collection of the client until the session is closed. Without this, some code which returns stream or transport objects without the client that generated them, would result in premature session closure when the client was GCd. Credit: `@w31rd0` for original report, Omer Anson for the patch.

- #862: (via #863) Avoid test suite exceptions on platforms lacking `errno.ETIME` (which seems to be some FreeBSD and some Windows environments.) Thanks to Sofian Brabez.
- #853: Tweak how `RSAKey.__str__` behaves so it doesn't cause `TypeError` under Python 3. Thanks to Francisco Couzo for the report.
- #866: (also #838) Remove an old test-related file we don't support, and add PyPy to Travis-CI config. Thanks to Pierce Lopez for the final patch and Pedro Rodrigues for an earlier edition.
- #895: Fix a bug in server-mode concerning multiple interactive auth steps (which were incorrectly responded to). Thanks to Dennis Kaarsemaker for catch & patch.
- #44: (via #891) `SSHClient` now gives its internal `Transport` a handle on itself, preventing garbage collection of the client until the session is closed. Without this, some code which returns stream or transport objects without the client that generated them, would result in premature session closure when the client was GCd. Credit: `@w31rd0` for original report, Omer Anson for the patch.
- #862: (via #863) Avoid test suite exceptions on platforms lacking `errno.ETIME` (which seems to be some FreeBSD and some Windows environments.) Thanks to Sofian Brabez.
- #853: Tweak how `RSAKey.__str__` behaves so it doesn't cause `TypeError` under Python 3. Thanks to Francisco Couzo for the report.
- #866: (also #838) Remove an old test-related file we don't support, and add PyPy to Travis-CI config. Thanks to Pierce Lopez for the final patch and Pedro Rodrigues for an earlier edition.
- #895: Fix a bug in server-mode concerning multiple interactive auth steps (which were incorrectly responded to). Thanks to Dennis Kaarsemaker for catch & patch.
- #713: (via #714 and #889) Don't pass initialization vectors to PyCrypto when dealing with counter-mode ciphers; newer PyCrypto versions throw an exception otherwise (older ones simply ignored this parameter altogether). Thanks to `@jmh045000` for report & patches.
- #44: (via #891) `SSHClient` now gives its internal `Transport` a handle on itself, preventing garbage collection of the client until the session is closed. Without this, some code which returns stream or transport objects without the client that generated them, would result in premature session closure when the client was GCd. Credit: `@w31rd0` for original report, Omer Anson for the patch.
- #862: (via #863) Avoid test suite exceptions on platforms lacking `errno.ETIME` (which seems to be some FreeBSD and some Windows environments.) Thanks to Sofian Brabez.
- #853: Tweak how `RSAKey.__str__` behaves so it doesn't cause `TypeError` under Python 3. Thanks to Francisco Couzo for the report.
- #866: (also #838) Remove an old test-related file we don't support, and add PyPy to Travis-CI config. Thanks to Pierce Lopez for the final patch and Pedro Rodrigues for an earlier edition.
- #895: Fix a bug in server-mode concerning multiple interactive auth steps (which were incorrectly responded to). Thanks to Dennis Kaarsemaker for catch & patch.
- #713: (via #714 and #889) Don't pass initialization vectors to PyCrypto when dealing with counter-mode ciphers; newer PyCrypto versions throw an exception otherwise (older ones simply ignored this parameter altogether). Thanks to `@jmh045000` for report & patches.
- #44: (via #891) `SSHClient` now gives its internal `Transport` a handle on itself, preventing garbage collection of the client until the session is closed. Without this, some code which returns stream or transport objects without the client that generated them, would result in premature session closure when the client was GCd. Credit: `@w31rd0` for original report, Omer Anson for the patch.
- #862: (via #863) Avoid test suite exceptions on platforms lacking `errno.ETIME` (which seems to be some FreeBSD and some Windows environments.) Thanks to Sofian Brabez.

- #853: Tweak how `RSAKey.__str__` behaves so it doesn't cause `TypeError` under Python 3. Thanks to Francisco Couzo for the report.
- #866: (also #838) Remove an old test-related file we don't support, and add PyPy to Travis-CI config. Thanks to Pierce Lopez for the final patch and Pedro Rodrigues for an earlier edition.
- #859: (via #860) A tweak to the original patch implementing #398 was not fully applied, causing calls to `invoke_shell` to fail with `AttributeError`. This has been fixed. Patch credit: Kirk Byers.
- : Accidentally merged the new features from 1.18.0 into the 2.0.x bugfix-only branch. This included merging a bug in one of those new features (breaking `invoke_shell` with an `AttributeError`.) The offending code has been stripped out of the 2.0.x line (but of course, remains in 2.1.x and above.)
- #859: (via #860) A tweak to the original patch implementing #398 was not fully applied, causing calls to `invoke_shell` to fail with `AttributeError`. This has been fixed. Patch credit: Kirk Byers.
- #859: (via #860) A tweak to the original patch implementing #398 was not fully applied, causing calls to `invoke_shell` to fail with `AttributeError`. This has been fixed. Patch credit: Kirk Byers.
- : Accidentally merged the new features from 1.18.0 into the 2.0.x bugfix-only branch. This included merging a bug in one of those new features (breaking `invoke_shell` with an `AttributeError`.) The offending code has been stripped out of the 2.0.x line (but of course, remains in 2.1.x and above.)
- #859: (via #860) A tweak to the original patch implementing #398 was not fully applied, causing calls to `invoke_shell` to fail with `AttributeError`. This has been fixed. Patch credit: Kirk Byers.
- #859: (via #860) A tweak to the original patch implementing #398 was not fully applied, causing calls to `invoke_shell` to fail with `AttributeError`. This has been fixed. Patch credit: Kirk Byers.
- #398: Add an `environment` dict argument to `Client.exec_command` (plus the lower level `Channel.update_environment` and `Channel.set_environment_variable` methods) which implements the `env` SSH message type. This means the remote shell environment can be set without the use of `VARNAME=value` shell tricks, provided the server's `AcceptEnv` lists the variables you need to set. Thanks to Philip Lorenz for the pull request.
- #780: (also #779, and may help users affected by #520) Add an optional `timeout` parameter to `Transport.start_client` (and feed it the value of the configured connection timeout when used within `SSHClient`.) This helps prevent situations where network connectivity isn't timing out, but the remote server is otherwise unable to service the connection in a timely manner. Credit to `@sanseihappa`.
- #854: Fix incorrect docstring/param-list for `Transport.auth_gssapi_keyex` so it matches the real signature. Caught by `@Score_Under`.
- #792: Minor updates to the README and demos; thanks to Alan Yee.
- #801: Skip a Unix-only test when on Windows; thanks to Gabi Davar.
- #681: Fix a Python3-specific bug re: the handling of read buffers when using `ProxyCommand`. Thanks to Paul Kapp for catch & patch.
- #334: Make the `subprocess` import in `proxy.py` lazy so users on platforms without it (such as Google App Engine) can import Paramiko successfully. (Relatedly, make it easier to tweak an active socket check timeout [in `Transport`] which was previously hardcoded.) Credit: Shinya Okano.
- #789: Add a missing `.closed` attribute (plus `._closed` because reasons) to `ProxyCommand` so the earlier partial fix for #520 works in situations where one is gatewaying via `ProxyCommand`.
- #742: (also re: #559) Catch `AssertionError` thrown by Cryptography when attempting to load bad ECDSA keys, turning it into an `SSHException`. This moves the behavior in line with other "bad keys" situations, re: Paramiko's main auth loop. Thanks to MengHuan Yu for the patch.
- #824: Fix the implementation of `PKey.write_private_key_file` (this method is only publicly defined on subclasses; the fix was in the private real implementation) so it passes the correct params to `open()`. This

bug apparently went unnoticed and unfixed for 12 entire years. Congrats to John Villalovos for noticing & submitting the patch!

- #802: (via #804) Update our vendored Windows API module to address errors of the form `AttributeError: 'module' object has no attribute 'c_ssize_t'`. Credit to Jason R. Coombs.
- #854: Fix incorrect docstring/param-list for `Transport.auth_gssapi_keyex` so it matches the real signature. Caught by `@Score_Under`.
- #792: Minor updates to the README and demos; thanks to Alan Yee.
- #801: Skip a Unix-only test when on Windows; thanks to Gabi Davar.
- #398: Add an `environment` dict argument to `Client.exec_command` (plus the lower level `Channel.update_environment` and `Channel.set_environment_variable` methods) which implements the `env` SSH message type. This means the remote shell environment can be set without the use of `VARNAME=value` shell tricks, provided the server's `AcceptEnv` lists the variables you need to set. Thanks to Philip Lorenz for the pull request.
- #780: (also #779, and may help users affected by #520) Add an optional `timeout` parameter to `Transport.start_client` (and feed it the value of the configured connection timeout when used within `SSHClient`.) This helps prevent situations where network connectivity isn't timing out, but the remote server is otherwise unable to service the connection in a timely manner. Credit to `@sanseihappa`.
- #819: Document how lacking `gmp` headers at install time can cause a significant performance hit if you build PyCrypto from source. (Most system-distributed packages already have this enabled.)
- #854: Fix incorrect docstring/param-list for `Transport.auth_gssapi_keyex` so it matches the real signature. Caught by `@Score_Under`.
- #792: Minor updates to the README and demos; thanks to Alan Yee.
- #801: Skip a Unix-only test when on Windows; thanks to Gabi Davar.
- #681: Fix a Python3-specific bug re: the handling of read buffers when using `ProxyCommand`. Thanks to Paul Kapp for catch & patch.
- #334: Make the `subprocess` import in `proxy.py` lazy so users on platforms without it (such as Google App Engine) can import Paramiko successfully. (Relatedly, make it easier to tweak an active socket check timeout [in `Transport`] which was previously hardcoded.) Credit: Shinya Okano.
- #789: Add a missing `.closed` attribute (plus `._closed` because reasons) to `ProxyCommand` so the earlier partial fix for #520 works in situations where one is gatewaying via `ProxyCommand`.
- #824: Fix the implementation of `PKey.write_private_key_file` (this method is only publicly defined on subclasses; the fix was in the private real implementation) so it passes the correct params to `open()`. This bug apparently went unnoticed and unfixed for 12 entire years. Congrats to John Villalovos for noticing & submitting the patch!
- #802: (via #804) Update our vendored Windows API module to address errors of the form `AttributeError: 'module' object has no attribute 'c_ssize_t'`. Credit to Jason R. Coombs.
- #819: Document how lacking `gmp` headers at install time can cause a significant performance hit if you build PyCrypto from source. (Most system-distributed packages already have this enabled.)
- #854: Fix incorrect docstring/param-list for `Transport.auth_gssapi_keyex` so it matches the real signature. Caught by `@Score_Under`.
- #792: Minor updates to the README and demos; thanks to Alan Yee.
- #801: Skip a Unix-only test when on Windows; thanks to Gabi Davar.

- #758: Apply type definitions to `_winapi` module from jaraco.windows 3.6.1. This should address issues on Windows platforms that often result in errors like `ArgumentError: [...] int too long to convert`. Thanks to `@swohlerLL` for the report and Jason R. Coombs for the patch.
- #774: Add a `_closed` private attribute to `Channel` objects so that they continue functioning when used as proxy sockets under Python 3 (e.g. as `direct-tcpip` gateways for other Paramiko connections.)
- #673: (via #681) Fix protocol banner read errors (`SSHException`) which would occasionally pop up when using `ProxyCommand` gatewaying. Thanks to `@Depado` for the initial report and Paul Kapp for the fix.
- #758: Apply type definitions to `_winapi` module from jaraco.windows 3.6.1. This should address issues on Windows platforms that often result in errors like `ArgumentError: [...] int too long to convert`. Thanks to `@swohlerLL` for the report and Jason R. Coombs for the patch.
- #774: Add a `_closed` private attribute to `Channel` objects so that they continue functioning when used as proxy sockets under Python 3 (e.g. as `direct-tcpip` gateways for other Paramiko connections.)
- #673: (via #681) Fix protocol banner read errors (`SSHException`) which would occasionally pop up when using `ProxyCommand` gatewaying. Thanks to `@Depado` for the initial report and Paul Kapp for the fix.
- #758: Apply type definitions to `_winapi` module from jaraco.windows 3.6.1. This should address issues on Windows platforms that often result in errors like `ArgumentError: [...] int too long to convert`. Thanks to `@swohlerLL` for the report and Jason R. Coombs for the patch.
- #774: Add a `_closed` private attribute to `Channel` objects so that they continue functioning when used as proxy sockets under Python 3 (e.g. as `direct-tcpip` gateways for other Paramiko connections.)
- #673: (via #681) Fix protocol banner read errors (`SSHException`) which would occasionally pop up when using `ProxyCommand` gatewaying. Thanks to `@Depado` for the initial report and Paul Kapp for the fix.
- #537: Fix a bug in `BufferedPipe.set_event` which could cause deadlocks/hangs when one uses `select.select` against `Channel` objects (or otherwise calls `Channel.fileno` after the channel has closed). Thanks to Przemysław Strzelczak for the report & reproduction case, and to Krzysztof Rusek for the fix.
- #520: (Partial fix) Fix at least one instance of race condition driven threading hangs at end of the Python interpreter session. (Includes a docs update as well - always make sure to `.close()` your clients!)
- #537: Fix a bug in `BufferedPipe.set_event` which could cause deadlocks/hangs when one uses `select.select` against `Channel` objects (or otherwise calls `Channel.fileno` after the channel has closed). Thanks to Przemysław Strzelczak for the report & reproduction case, and to Krzysztof Rusek for the fix.
- #520: (Partial fix) Fix at least one instance of race condition driven threading hangs at end of the Python interpreter session. (Includes a docs update as well - always make sure to `.close()` your clients!)
- #537: Fix a bug in `BufferedPipe.set_event` which could cause deadlocks/hangs when one uses `select.select` against `Channel` objects (or otherwise calls `Channel.fileno` after the channel has closed). Thanks to Przemysław Strzelczak for the report & reproduction case, and to Krzysztof Rusek for the fix.
- #520: (Partial fix) Fix at least one instance of race condition driven threading hangs at end of the Python interpreter session. (Includes a docs update as well - always make sure to `.close()` your clients!)
- #394: Replace PyCrypto with the Python Cryptographic Authority (PyCA) 'Cryptography' library suite. This improves security, installability, and performance; adds PyPy support; and much more.

  There aren't enough ways to thank Alex Gaynor for all of his work on this, and then his patience while the maintainer let his PR grow moss for a year and change. Paul Kehrer came in with an assist, and I think I saw Olle Lundberg, `@techtonik` and `@johnthagen` supplying backup as well. Thanks to all!

> **Warning: This is a backwards incompatible change.**
>
> However, **it should only affect installation** requirements; **no API changes are intended or expected**. Please report any such breakages as bugs.
>
> See our updated *installation docs* for details on what is now required to install Paramiko; many/most users should be able to simply `pip install -U paramiko` (especially if you **upgrade to pip 8**).

- #731: (working off the earlier #611) Add support for 384- and 512-bit elliptic curve groups in ECDSA key types (aka `ecdsa-sha2-nistp384` / `ecdsa-sha2-nistp521`). Thanks to Michiel Tiller and `@CrazyCasta` for the patches.
- #588: Add missing file-like object methods for `BufferedFile` and `SFTPFile`. Thanks to Adam Meily for the patch.
- #649: Update the module in charge of handling SSH moduli so it's consistent with OpenSSH behavior re: prime number selection. Thanks to Damien Tournoud for catch & patch.
- #636: Clean up and enhance the README (and rename it to `README.rst` from just `README`). Thanks to `@LucasRMehl`.
- #697: Remove whitespace in our `setup.py`'s `install_requires` as it triggers occasional bugs in some versions of `setuptools`. Thanks to Justin Lecher for catch & original patch.
- #612: Identify & work around a race condition in the test for handshake timeouts, which was causing frequent test failures for a subset of contributors as well as Travis-CI (usually, but not always, limited to Python 3.5). Props to Ed Kellett for assistance during some of the troubleshooting.
- #621: Annotate some public attributes on `Channel` such as `.closed`. Thanks to Sergey Vasilyev for the report.
- #729: Clean up `setup.py` to always use `setuptools`, not doing so was a historical artifact from bygone days. Thanks to Alex Gaynor.
- #652: Fix behavior of `gssapi-with-mic` auth requests so they fail gracefully (allowing followup via other auth methods) instead of raising an exception. Patch courtesy of `@jamercee`.
- #499: Strip trailing/leading whitespace from lines when parsing SSH config files - this brings things in line with OpenSSH behavior. Thanks to Alfredo Esteban for the original report and Nick Pillitteri for the patch.
- #632: Fix logic bug in the SFTP client's callback-calling functionality; previously there was a chance the given callback would fire twice at the end of a transfer. Thanks to `@ab9-er` for catch & original patch.
- #613: (via #619) Update to `jaraco.windows` 3.4.1 to fix some errors related to `ctypes` on Windows platforms. Credit to Jason R. Coombs.
- #617: (aka fabric/fabric#1429; via #679; related: #678, #685, #615 & #616) Fix up `NoValidConnectionsError` so it pickles correctly, and fix a related Python 3 compatibility issue. Thanks to Rebecca Schlussel for the report & Marius Gedminas for the patch.
- #716: Fix a Python 3 compatibility issue when handling two-factor authentication. Thanks to Mateusz Kowalski for the catch & original patch.
- #577: (via #578; should also fix #718, #560) Fix stalled/hung SFTP downloads by cleaning up some threading lock issues. Thanks to Stephen C. Pope for the patch.
- #676: (via #677) Fix a backwards incompatibility issue that cropped up in `SFTPFile.prefetch` re: the erroneously non-optional `file_size` parameter. Should only affect users who manually call `prefetch`. Thanks to `@stevevanhooser` for catch & patch.
- #670: Due to an earlier bugfix, less-specific `Host` blocks' `ProxyCommand` values were overriding `ProxyCommand none` in more-specific `Host` blocks. This has been fixed in a backwards compatible manner

(i.e. `ProxyCommand none` continues to appear as a total lack of any `proxycommand` key in parsed config structures). Thanks to Pat Brisbin for the catch.

- #636: Clean up and enhance the README (and rename it to `README.rst` from just `README`). Thanks to `@LucasRMehl`.
- #697: Remove whitespace in our `setup.py`'s `install_requires` as it triggers occasional bugs in some versions of `setuptools`. Thanks to Justin Lecher for catch & original patch.
- #612: Identify & work around a race condition in the test for handshake timeouts, which was causing frequent test failures for a subset of contributors as well as Travis-CI (usually, but not always, limited to Python 3.5). Props to Ed Kellett for assistance during some of the troubleshooting.
- #621: Annotate some public attributes on `Channel` such as `.closed`. Thanks to Sergey Vasilyev for the report.
- #729: Clean up `setup.py` to always use `setuptools`, not doing so was a historical artifact from bygone days. Thanks to Alex Gaynor.
- #401: Fix line number reporting in log output regarding invalid `known_hosts` line entries. Thanks to Dylan Thacker-Smith for catch & patch.
- #652: Fix behavior of `gssapi-with-mic` auth requests so they fail gracefully (allowing followup via other auth methods) instead of raising an exception. Patch courtesy of `@jamercee`.
- #499: Strip trailing/leading whitespace from lines when parsing SSH config files - this brings things in line with OpenSSH behavior. Thanks to Alfredo Esteban for the original report and Nick Pillitteri for the patch.
- #632: Fix logic bug in the SFTP client's callback-calling functionality; previously there was a chance the given callback would fire twice at the end of a transfer. Thanks to `@ab9-er` for catch & original patch.
- #613: (via #619) Update to `jaraco.windows` 3.4.1 to fix some errors related to `ctypes` on Windows platforms. Credit to Jason R. Coombs.
- #617: (aka fabric/fabric#1429; via #679; related: #678, #685, #615 & #616) Fix up `NoValidConnectionsError` so it pickles correctly, and fix a related Python 3 compatibility issue. Thanks to Rebecca Schlussel for the report & Marius Gedminas for the patch.
- #716: Fix a Python 3 compatibility issue when handling two-factor authentication. Thanks to Mateusz Kowalski for the catch & original patch.
- #577: (via #578; should also fix #718, #560) Fix stalled/hung SFTP downloads by cleaning up some threading lock issues. Thanks to Stephen C. Pope for the patch.
- #676: (via #677) Fix a backwards incompatibility issue that cropped up in `SFTPFile.prefetch` re: the erroneously non-optional `file_size` parameter. Should only affect users who manually call `prefetch`. Thanks to `@stevevanhooser` for catch & patch.
- #670: Due to an earlier bugfix, less-specific `Host` blocks' `ProxyCommand` values were overriding `ProxyCommand none` in more-specific `Host` blocks. This has been fixed in a backwards compatible manner (i.e. `ProxyCommand none` continues to appear as a total lack of any `proxycommand` key in parsed config structures). Thanks to Pat Brisbin for the catch.
- #525: Update the vendored Windows API addon to a more recent edition. Also fixes #193, #488, #498. Thanks to Jason Coombs.

- #636: Clean up and enhance the README (and rename it to `README.rst` from just `README`). Thanks to `@LucasRMehl`.
- #697: Remove whitespace in our `setup.py`'s `install_requires` as it triggers occasional bugs in some versions of `setuptools`. Thanks to Justin Lecher for catch & original patch.

- #612: Identify & work around a race condition in the test for handshake timeouts, which was causing frequent test failures for a subset of contributors as well as Travis-CI (usually, but not always, limited to Python 3.5). Props to Ed Kellett for assistance during some of the troubleshooting.
- #621: Annotate some public attributes on `Channel` such as `.closed`. Thanks to Sergey Vasilyev for the report.
- #729: Clean up `setup.py` to always use `setuptools`, not doing so was a historical artifact from bygone days. Thanks to Alex Gaynor.
- #167: Add `get_hostnames` for easier introspection of a loaded SSH config file or object. Courtesy of Søren Løvborg.
- #356: (also #596, #365, #341, #164, #581, and a bunch of other duplicates besides) Add support for SHA-2 based key exchange (kex) algorithm `diffie-hellman-group-exchange-sha256` and (H)MAC algorithms `hmac-sha2-256` and `hmac-sha2-512`.

  This change includes tweaks to debug-level logging regarding algorithm-selection handshakes; the old all-in-one log line is now multiple easier-to-read, printed-at-handshake-time log lines.

  Thanks to the many people who submitted patches for this functionality and/or assisted in testing those patches. That list includes but is not limited to, and in no particular order: Matthias Witte, Dag Wieers, Ash Berlin, Etienne Perot, Gert van Dijk, `@GuyShaanan`, Aaron Bieber, `@cyphase`, and Eric Brown.
- #604: Add support for the `aes192-ctr` and `aes192-cbc` ciphers. Thanks to Michiel Tiller for noticing it was as easy as tweaking some key sizes :D
- #467: (also #139, #412) Fully enable two-factor authentication (e.g. when a server requires `AuthenticationMethods pubkey,keyboard-interactive`). Thanks to `@perryjrandall` for the patch and to `@nevins-b` and Matt Robenolt for additional support.
- #22: Try harder to connect to multiple network families (e.g. IPv4 vs IPv6) in case of connection issues; this helps with problems such as hosts which resolve both IPv4 and IPv6 addresses but are only listening on IPv4. Thanks to Dries Desmet for original report and Torsten Landschoff for the foundational patchset.
- #502: Fix 'exec' requests in server mode to use `get_string` instead of `get_text` to avoid `UnicodeDecodeError` on non-UTF-8 input. Thanks to Anselm Kruis for the patch & discussion.
- #194: (also #562, #530, #576) Streamline use of `stat` when downloading SFTP files via `SFTPClient.get`; this avoids triggering bugs in some off-spec SFTP servers such as IBM Sterling. Thanks to `@muraleee` for the initial report and to Torkil Gustavsen for the patch.
- #419: Modernize a bunch of the codebase internals to leverage decorators. Props to `@beckjake` for realizing we're no longer on Python 2.2 :D
- #421: Modernize threading calls to use newer API. Thanks to Olle Lundberg.
- #422: Clean up some unused imports. Courtesy of Olle Lundberg.
- #431: Replace handrolled `ssh_config` parsing code with use of the `shlex` module. Thanks to Yan Kalchevskiy.
- #582: Fix some old `setup.py` related helper code which was breaking `bdist_dumb` on Mac OS X. Thanks to Peter Odding for the patch.
- #516: Document `AgentRequestHandler`. Thanks to `@toejough` for report & suggestions.
- #554: Fix inaccuracies in the docstring for the ECDSA key class. Thanks to Jared Hance for the patch.
- #594: Correct some post-Python3-port docstrings to specify `bytes` type instead of `str`. Credit to `@redixin`.
- #525: Update the vendored Windows API addon to a more recent edition. Also fixes #193, #488, #498. Thanks to Jason Coombs.

- #565: Don't explode with `IndexError` when reading private key files lacking an `-----END <type> PRIVATE KEY-----` footer. Patch courtesy of Prasanna Santhanam.
- #359: Use correct attribute name when trying to use Python 3's `int.bit_length` method; prior to fix, the Python 2 custom fallback implementation was always used, even on Python 3. Thanks to Alex Gaynor.
- #366: Fix `SFTPAttributes` so its string representation doesn't raise exceptions on empty/initialized instances. Patch by Ulrich Petri.
- #594: Correct some post-Python3-port docstrings to specify `bytes` type instead of `str`. Credit to `@redixin`.
- #402: Check to see if an SSH agent is actually present before trying to forward it to the remote end. This replaces what was usually a useless `TypeError` with a human-readable `AuthenticationException`. Credit to Ken Jordan for the fix and Yvan Marques for original report.
- #353: (via #482) Fix a bug introduced in the Python 3 port which caused `OverFlowError` (and other symptoms) in SFTP functionality. Thanks to `@dboreham` for leading the troubleshooting charge, and to Scott Maxwell for the final patch.
- #469: (also #488, #461 and like a dozen others) Fix a typo introduced in the 1.15 release which broke WinPageant support. Thanks to everyone who submitted patches, and to Steve Cohen who was the lucky winner of the cherry-pick lottery.
- #404: Print details when displaying `BadHostKeyException` objects (expected vs received data) instead of just "hey shit broke". Patch credit: Loic Dachary.
- #490: Skip invalid/unparseable lines in `known_hosts` files, instead of raising `SSHException`. This brings Paramiko's behavior more in line with OpenSSH, which silently ignores such input. Catch & patch courtesy of Martin Topholm.
- #491: (combines #62 and #439) Implement timeout functionality to address hangs from dropped network connections and/or failed handshakes. Credit to `@vazir` and `@dacut` for the original patches and to Olle Lundberg for reimplementation.
- #565: Don't explode with `IndexError` when reading private key files lacking an `-----END <type> PRIVATE KEY-----` footer. Patch courtesy of Prasanna Santhanam.
- #359: Use correct attribute name when trying to use Python 3's `int.bit_length` method; prior to fix, the Python 2 custom fallback implementation was always used, even on Python 3. Thanks to Alex Gaynor.
- #366: Fix `SFTPAttributes` so its string representation doesn't raise exceptions on empty/initialized instances. Patch by Ulrich Petri.
- #516: Document `AgentRequestHandler`. Thanks to `@toejough` for report & suggestions.
- #554: Fix inaccuracies in the docstring for the ECDSA key class. Thanks to Jared Hance for the patch.
- #594: Correct some post-Python3-port docstrings to specify `bytes` type instead of `str`. Credit to `@redixin`.
- #402: Check to see if an SSH agent is actually present before trying to forward it to the remote end. This replaces what was usually a useless `TypeError` with a human-readable `AuthenticationException`. Credit to Ken Jordan for the fix and Yvan Marques for original report.
- #353: (via #482) Fix a bug introduced in the Python 3 port which caused `OverFlowError` (and other symptoms) in SFTP functionality. Thanks to `@dboreham` for leading the troubleshooting charge, and to Scott Maxwell for the final patch.
- #469: (also #488, #461 and like a dozen others) Fix a typo introduced in the 1.15 release which broke WinPageant support. Thanks to everyone who submitted patches, and to Steve Cohen who was the lucky winner of the cherry-pick lottery.
- #404: Print details when displaying `BadHostKeyException` objects (expected vs received data) instead of just "hey shit broke". Patch credit: Loic Dachary.

- #490: Skip invalid/unparseable lines in `known_hosts` files, instead of raising `SSHException`. This brings Paramiko's behavior more in line with OpenSSH, which silently ignores such input. Catch & patch courtesy of Martin Topholm.
- #491: (combines #62 and #439) Implement timeout functionality to address hangs from dropped network connections and/or failed handshakes. Credit to `@vazir` and `@dacut` for the original patches and to Olle Lundberg for reimplementation.
- #565: Don't explode with `IndexError` when reading private key files lacking an `-----END <type> PRIVATE KEY-----` footer. Patch courtesy of Prasanna Santhanam.
- #359: Use correct attribute name when trying to use Python 3's `int.bit_length` method; prior to fix, the Python 2 custom fallback implementation was always used, even on Python 3. Thanks to Alex Gaynor.
- #366: Fix `SFTPAttributes` so its string representation doesn't raise exceptions on empty/initialized instances. Patch by Ulrich Petri.
- #516: Document `AgentRequestHandler`. Thanks to `@toejough` for report & suggestions.
- #554: Fix inaccuracies in the docstring for the ECDSA key class. Thanks to Jared Hance for the patch.
- #594: Correct some post-Python3-port docstrings to specify `bytes` type instead of `str`. Credit to `@redixin`.
- #402: Check to see if an SSH agent is actually present before trying to forward it to the remote end. This replaces what was usually a useless `TypeError` with a human-readable `AuthenticationException`. Credit to Ken Jordan for the fix and Yvan Marques for original report.
- #353: (via #482) Fix a bug introduced in the Python 3 port which caused `OverFlowError` (and other symptoms) in SFTP functionality. Thanks to `@dboreham` for leading the troubleshooting charge, and to Scott Maxwell for the final patch.
- #469: (also #488, #461 and like a dozen others) Fix a typo introduced in the 1.15 release which broke WinPageant support. Thanks to everyone who submitted patches, and to Steve Cohen who was the lucky winner of the cherry-pick lottery.
- #404: Print details when displaying `BadHostKeyException` objects (expected vs received data) instead of just "hey shit broke". Patch credit: Loic Dachary.
- #490: Skip invalid/unparseable lines in `known_hosts` files, instead of raising `SSHException`. This brings Paramiko's behavior more in line with OpenSSH, which silently ignores such input. Catch & patch courtesy of Martin Topholm.
- #491: (combines #62 and #439) Implement timeout functionality to address hangs from dropped network connections and/or failed handshakes. Credit to `@vazir` and `@dacut` for the original patches and to Olle Lundberg for reimplementation.
- #496: Fix a handful of small but critical bugs in Paramiko's GSSAPI support (note: this includes switching from PyCrypo's Random to `os.urandom`). Thanks to Anselm Kruis for catch & patch.
- #516: Document `AgentRequestHandler`. Thanks to `@toejough` for report & suggestions.
- #554: Fix inaccuracies in the docstring for the ECDSA key class. Thanks to Jared Hance for the patch.
- #320: Update our win_pageant module to be Python 3 compatible. Thanks to `@sherbang` and `@adamkerz` for the patches.
- #429: Server-level debug message logging was overlooked during the Python 3 compatibility update; Python 3 clients attempting to log SSH debug packets encountered type errors. This is now fixed. Thanks to `@mjmaenpaa` for the catch.
- #459: Tighten up agent connection closure behavior to avoid spurious `ResourceWarning` display in some situations. Thanks to `@tkrapp` for the catch.

- #266: Change numbering of `Transport` channels to start at 0 instead of 1 for better compatibility with OpenSSH & certain server implementations which break on 1-indexed channels. Thanks to `@egroeper` for catch & patch.
- #415: Fix `ssh_config` parsing to correctly interpret `ProxyCommand none` as the lack of a proxy command, instead of as a literal command string of `"none"`. Thanks to Richard Spiers for the catch & Sean Johnson for the fix.
- #428: Fix an issue in `BufferedFile` (primarily used in the SFTP modules) concerning incorrect behavior by `readlines` on files whose size exceeds the buffer size. Thanks to `@achapp` for catch & patch.
- #455: Tweak packet size handling to conform better to the OpenSSH RFCs; this helps address issues with interactive program cursors. Courtesy of Jeff Quast.
- #413: (also #414, #420, #454) Be significantly smarter about polling & timing behavior when running proxy commands, to avoid unnecessary (often 100%!) CPU usage. Major thanks to Jason Dunsmore for report & initial patchset and to Chris Adams & John Morrissey for followup improvements.
- #419: Modernize a bunch of the codebase internals to leverage decorators. Props to `@beckjake` for realizing we're no longer on Python 2.2 :D
- #421: Modernize threading calls to use newer API. Thanks to Olle Lundberg.
- #422: Clean up some unused imports. Courtesy of Olle Lundberg.
- #431: Replace handrolled `ssh_config` parsing code with use of the `shlex` module. Thanks to Yan Kalchevskiy.
- #399: SSH agent forwarding (potentially other functionality as well) would hang due to incorrect values passed into the new window size arguments for `Transport` (thanks to a botched merge). This has been corrected. Thanks to Dylan Thacker-Smith for the report & patch.
- #320: Update our win_pageant module to be Python 3 compatible. Thanks to `@sherbang` and `@adamkerz` for the patches.
- #429: Server-level debug message logging was overlooked during the Python 3 compatibility update; Python 3 clients attempting to log SSH debug packets encountered type errors. This is now fixed. Thanks to `@mjmaenpaa` for the catch.
- #459: Tighten up agent connection closure behavior to avoid spurious `ResourceWarning` display in some situations. Thanks to `@tkrapp` for the catch.
- #266: Change numbering of `Transport` channels to start at 0 instead of 1 for better compatibility with OpenSSH & certain server implementations which break on 1-indexed channels. Thanks to `@egroeper` for catch & patch.
- #415: Fix `ssh_config` parsing to correctly interpret `ProxyCommand none` as the lack of a proxy command, instead of as a literal command string of `"none"`. Thanks to Richard Spiers for the catch & Sean Johnson for the fix.
- #428: Fix an issue in `BufferedFile` (primarily used in the SFTP modules) concerning incorrect behavior by `readlines` on files whose size exceeds the buffer size. Thanks to `@achapp` for catch & patch.
- #455: Tweak packet size handling to conform better to the OpenSSH RFCs; this helps address issues with interactive program cursors. Courtesy of Jeff Quast.
- #413: (also #414, #420, #454) Be significantly smarter about polling & timing behavior when running proxy commands, to avoid unnecessary (often 100%!) CPU usage. Major thanks to Jason Dunsmore for report & initial patchset and to Chris Adams & John Morrissey for followup improvements.
- #249: Consolidate version information into one spot. Thanks to Gabi Davar for the reminder.
- #378: Minor code cleanup in the SSH config module courtesy of Olle Lundberg.

- #419: Modernize a bunch of the codebase internals to leverage decorators. Props to `@beckjake` for realizing we're no longer on Python 2.2 :D
- #421: Modernize threading calls to use newer API. Thanks to Olle Lundberg.
- #422: Clean up some unused imports. Courtesy of Olle Lundberg.
- #431: Replace handrolled `ssh_config` parsing code with use of the `shlex` module. Thanks to Yan Kalchevskiy.
- #399: SSH agent forwarding (potentially other functionality as well) would hang due to incorrect values passed into the new window size arguments for `Transport` (thanks to a botched merge). This has been corrected. Thanks to Dylan Thacker-Smith for the report & patch.
- #320: Update our win_pageant module to be Python 3 compatible. Thanks to `@sherbang` and `@adamkerz` for the patches.
- #429: Server-level debug message logging was overlooked during the Python 3 compatibility update; Python 3 clients attempting to log SSH debug packets encountered type errors. This is now fixed. Thanks to `@mjmaenpaa` for the catch.
- #459: Tighten up agent connection closure behavior to avoid spurious `ResourceWarning` display in some situations. Thanks to `@tkrapp` for the catch.
- #266: Change numbering of `Transport` channels to start at 0 instead of 1 for better compatibility with OpenSSH & certain server implementations which break on 1-indexed channels. Thanks to `@egroeper` for catch & patch.
- #415: Fix `ssh_config` parsing to correctly interpret `ProxyCommand none` as the lack of a proxy command, instead of as a literal command string of `"none"`. Thanks to Richard Spiers for the catch & Sean Johnson for the fix.
- #428: Fix an issue in `BufferedFile` (primarily used in the SFTP modules) concerning incorrect behavior by `readlines` on files whose size exceeds the buffer size. Thanks to `@achapp` for catch & patch.
- #455: Tweak packet size handling to conform better to the OpenSSH RFCs; this helps address issues with interactive program cursors. Courtesy of Jeff Quast.
- #413: (also #414, #420, #454) Be significantly smarter about polling & timing behavior when running proxy commands, to avoid unnecessary (often 100%!) CPU usage. Major thanks to Jason Dunsmore for report & initial patchset and to Chris Adams & John Morrissey for followup improvements.
- #249: Consolidate version information into one spot. Thanks to Gabi Davar for the reminder.
- #378: Minor code cleanup in the SSH config module courtesy of Olle Lundberg.
- #419: Modernize a bunch of the codebase internals to leverage decorators. Props to `@beckjake` for realizing we're no longer on Python 2.2 :D
- #421: Modernize threading calls to use newer API. Thanks to Olle Lundberg.
- #422: Clean up some unused imports. Courtesy of Olle Lundberg.
- #431: Replace handrolled `ssh_config` parsing code with use of the `shlex` module. Thanks to Yan Kalchevskiy.
- #399: SSH agent forwarding (potentially other functionality as well) would hang due to incorrect values passed into the new window size arguments for `Transport` (thanks to a botched merge). This has been corrected. Thanks to Dylan Thacker-Smith for the report & patch.
- #131: Add a `listdir_iter` method to `SFTPClient` allowing for more efficient, async/generator based file listings. Thanks to John Begeman.
- #184: Support quoted values in SSH config file parsing. Credit to Yan Kalchevskiy.

- #218: Add support for ECDSA private keys on the client side. Thanks to `@aszlig` for the patch.
- #372: Update default window & packet sizes to more closely adhere to the pertinent RFC; also expose these settings in the public API so they may be overridden by client code. This should address some general speed issues such as #175. Big thanks to Olle Lundberg for the update.
- #362: Allow users to control the SSH banner timeout. Thanks to Cory Benfield.
- #267: (also #250, #241, #228) Add GSS-API / SSPI (e.g. Kerberos) key exchange and authentication support (*installation docs here*). Mega thanks to Sebastian Deiß, with assist by Torsten Landschoff.

Note: Unix users should be aware that the `python-gssapi` library (a requirement for using this functionality) only appears to support Python 2.7 and up at this time.

- #335: Fix ECDSA key generation (generation of brand new ECDSA keys was broken previously). Thanks to `@solarw` for catch & patch.
- #234: Lower logging levels for a few overly-noisy log messages about secure channels. Thanks to David Pursehouse for noticing & contributing the fix.
- #298: Don't perform point validation on ECDSA keys in `known_hosts` files, since a) this can cause significant slowdown when such keys exist, and b) `known_hosts` files are implicitly trustworthy. Thanks to Kieran Spear for catch & patch.

Note: This change bumps up the version requirement for the `ecdsa` library to `0.11`.

- #373: Attempt to fix a handful of issues (such as #354) related to infinite loops and threading deadlocks. Thanks to Olle Lundberg as well as a handful of community members who provided advice & feedback via IRC.
- #346: Fix an issue in private key files' encryption salts that could cause tracebacks and file corruption if keys were re-encrypted. Credit to Xavier Nunn.
- #371: Add Travis support & docs update for Python 3.4. Thanks to Olle Lundberg.
- #169: Minor refactor of `paramiko.sftp_client.SFTPClient.put` thanks to Abhinav Upadhyay.
- #229: Fix a couple of incorrectly-copied docstrings' `.. versionadded::` RST directives. Thanks to Aarni Koskela for the catch.
- #324: A bevvy of documentation typo fixes, courtesy of Roy Wellington.
- #249: Consolidate version information into one spot. Thanks to Gabi Davar for the reminder.
- #378: Minor code cleanup in the SSH config module courtesy of Olle Lundberg.
- #377: Factor `Channel` openness sanity check into a decorator. Thanks to Olle Lundberg for original patch.
- #374: (also #375) Old code cleanup courtesy of Olle Lundberg.
- #393: Replace internal use of PyCrypto's `SHA.new` with the stdlib's `hashlib.sha1`. Thanks to Alex Gaynor.
- #285: (also #352) Update our Python 3 `b()` compatibility shim to handle `buffer` objects correctly; this fixes a frequently reported issue affecting many users, including users of the `bzr` software suite. Thanks to `@basictheprogram` for the initial report, Jelmer Vernooij for the fix and Andrew Starr-Bochicchio & Jeremy T. Bouse (among others) for discussion & feedback.
- #239: Add Windows-style CRLF support to SSH config file parsing. Props to Christopher Swenson.
- #272: Fix a bug where `known_hosts` parsing hashed the input hostname as well as the hostnames from the `known_hosts` file, on every comparison. Thanks to `@sigmunau` for final patch and `@ostacey` for the original report.

- #312: paramiko.transport.Transport had a bug in its __repr__ which surfaces during errors encountered within its __init__, causing problematic tracebacks in such situations. Thanks to Simon Percivall for catch & patch.
- #376: Be less aggressive about expanding variables in ssh_config files, which results in a speedup of SSH config parsing. Credit to Olle Lundberg.
- #169: Minor refactor of paramiko.sftp_client.SFTPClient.put thanks to Abhinav Upadhyay.
- #229: Fix a couple of incorrectly-copied docstrings' .. versionadded:: RST directives. Thanks to Aarni Koskela for the catch.
- #324: A bevvy of documentation typo fixes, courtesy of Roy Wellington.
- #285: (also #352) Update our Python 3 b() compatibility shim to handle buffer objects correctly; this fixes a frequently reported issue affecting many users, including users of the bzr software suite. Thanks to @basictheprogram for the initial report, Jelmer Vernooij for the fix and Andrew Starr-Bochicchio & Jeremy T. Bouse (among others) for discussion & feedback.
- #239: Add Windows-style CRLF support to SSH config file parsing. Props to Christopher Swenson.
- #272: Fix a bug where known_hosts parsing hashed the input hostname as well as the hostnames from the known_hosts file, on every comparison. Thanks to @sigmunau for final patch and @ostacey for the original report.
- #312: paramiko.transport.Transport had a bug in its __repr__ which surfaces during errors encountered within its __init__, causing problematic tracebacks in such situations. Thanks to Simon Percivall for catch & patch.
- #376: Be less aggressive about expanding variables in ssh_config files, which results in a speedup of SSH config parsing. Credit to Olle Lundberg.
- #169: Minor refactor of paramiko.sftp_client.SFTPClient.put thanks to Abhinav Upadhyay.
- #229: Fix a couple of incorrectly-copied docstrings' .. versionadded:: RST directives. Thanks to Aarni Koskela for the catch.
- #324: A bevvy of documentation typo fixes, courtesy of Roy Wellington.
- #284: Add Python language trove identifiers to setup.py. Thanks to Alex Gaynor for catch & patch.
- #290: (also #292) Add support for building universal (Python 2+3 compatible) wheel files during the release process. Courtesy of Alex Gaynor.
- #295: Swap out a bunch of PyCrypto hash functions with use of hashlib. Thanks to Alex Gaynor.
- #297: Replace PyCrypto's Random with os.urandom for improved speed and security. Thanks again to Alex.
- #299: Use deterministic signatures for ECDSA keys for improved security. Thanks to Alex Gaynor.
- #235: Improve string type testing in a handful of spots (e.g. s/if type(x) is str/if isinstance(x, basestring)/g.) Thanks to @ksamuel for the report.
- #308: Fix regression in dsskey.py that caused sporadic signature verification failures. Thanks to Chris Rose.
- : Fix logging error in sftp_client for filenames containing the '%' character. Thanks to Antoine Brenner.
- : Added self.args for exception classes. Used for unpickling. Related to (Fabric #986, Fabric #714). Thanks to Alex Plugaru.
- : paramiko.file.BufferedFile.read incorrectly returned text strings after the Python 3 migration, despite bytes being more appropriate for file contents (which may be binary or of an unknown encoding.) This has been addressed.

Note: `paramiko.file.BufferedFile.readline` continues to return strings, not bytes, as "lines" only make sense for textual data. It assumes UTF-8 by default.

This should fix this issue raised on the Obnam mailing list. Thanks to Antoine Brenner for the patch.

- #235: Improve string type testing in a handful of spots (e.g. `s/if type(x) is str/if isinstance(x, basestring)/g`.) Thanks to `@ksamuel` for the report.
- #308: Fix regression in dsskey.py that caused sporadic signature verification failures. Thanks to Chris Rose.
- : Fix logging error in sftp_client for filenames containing the '%' character. Thanks to Antoine Brenner.
- : Added self.args for exception classes. Used for unpickling. Related to (Fabric #986, Fabric #714). Thanks to Alex Plugaru.
- : `paramiko.file.BufferedFile.read` incorrectly returned text strings after the Python 3 migration, despite bytes being more appropriate for file contents (which may be binary or of an unknown encoding.) This has been addressed.

Note: `paramiko.file.BufferedFile.readline` continues to return strings, not bytes, as "lines" only make sense for textual data. It assumes UTF-8 by default.

This should fix this issue raised on the Obnam mailing list. Thanks to Antoine Brenner for the patch.

- #235: Improve string type testing in a handful of spots (e.g. `s/if type(x) is str/if isinstance(x, basestring)/g`.) Thanks to `@ksamuel` for the report.
- #308: Fix regression in dsskey.py that caused sporadic signature verification failures. Thanks to Chris Rose.
- : Fix logging error in sftp_client for filenames containing the '%' character. Thanks to Antoine Brenner.
- : Added self.args for exception classes. Used for unpickling. Related to (Fabric #986, Fabric #714). Thanks to Alex Plugaru.
- : `paramiko.file.BufferedFile.read` incorrectly returned text strings after the Python 3 migration, despite bytes being more appropriate for file contents (which may be binary or of an unknown encoding.) This has been addressed.

Note: `paramiko.file.BufferedFile.readline` continues to return strings, not bytes, as "lines" only make sense for textual data. It assumes UTF-8 by default.

This should fix this issue raised on the Obnam mailing list. Thanks to Antoine Brenner for the patch.

- #58: Allow client code to access the stored SSH server banner via `Transport.get_banner`. Thanks to `@Jhoanor` for the patch.
- #16: **Python 3 support!** Our test suite passes under Python 3, and it (& Fabric's test suite) continues to pass under Python 2. **Python 2.5 is no longer supported with this change!**

  The merged code was built on many contributors' efforts, both code & feedback. In no particular order, we thank Daniel Goertzen, Ivan Kolodyazhny, Tomi Pieviläinen, Jason R. Coombs, Jan N. Schulze, `@Lazik`, Dorian Pula, Scott Maxwell, Tshepang Lekhonkhobe, Aaron Meurer, and Dave Halter.
- #256: Convert API documentation to Sphinx, yielding a new API docs website to replace the old Epydoc one. Thanks to Olle Lundberg for the initial conversion work.
- : Use constant-time hash comparison operations where possible, to protect against timing-based attacks. Thanks to Alex Gaynor for the patch.

- #256: Convert API documentation to Sphinx, yielding a new API docs website to replace the old Epydoc one. Thanks to Olle Lundberg for the initial conversion work.
- : Use constant-time hash comparison operations where possible, to protect against timing-based attacks. Thanks to Alex Gaynor for the patch.
- #256: Convert API documentation to Sphinx, yielding a new API docs website to replace the old Epydoc one. Thanks to Olle Lundberg for the initial conversion work.
- : Use constant-time hash comparison operations where possible, to protect against timing-based attacks. Thanks to Alex Gaynor for the patch.
- #256: Convert API documentation to Sphinx, yielding a new API docs website to replace the old Epydoc one. Thanks to Olle Lundberg for the initial conversion work.
- #193: (and its attentant PRs #230 & #253) Fix SSH agent problems present on Windows. Thanks to David Hobbs for initial report and to Aarni Koskela & Olle Lundberg for the patches.
- #34: (PR #35) Fix SFTP prefetching incompatibility with some SFTP servers regarding request/response ordering. Thanks to Richard Kettlewell.
- #268: Fix some missed renames of `ProxyCommand` related error classes. Thanks to Marius Gedminas for catch & patch.
- #252: (Fabric #1020) Enhanced the implementation of `ProxyCommand` to avoid a deadlock/hang condition that frequently occurs at `Transport` shutdown time. Thanks to Mateusz Kobos, Matthijs van der Vleuten and Guillaume Zitta for the original reports and to Marius Gedminas for helping test nontrivial use cases.
- #193: (and its attentant PRs #230 & #253) Fix SSH agent problems present on Windows. Thanks to David Hobbs for initial report and to Aarni Koskela & Olle Lundberg for the patches.
- #34: (PR #35) Fix SFTP prefetching incompatibility with some SFTP servers regarding request/response ordering. Thanks to Richard Kettlewell.
- #268: Fix some missed renames of `ProxyCommand` related error classes. Thanks to Marius Gedminas for catch & patch.
- #252: (Fabric #1020) Enhanced the implementation of `ProxyCommand` to avoid a deadlock/hang condition that frequently occurs at `Transport` shutdown time. Thanks to Mateusz Kobos, Matthijs van der Vleuten and Guillaume Zitta for the original reports and to Marius Gedminas for helping test nontrivial use cases.
- #193: (and its attentant PRs #230 & #253) Fix SSH agent problems present on Windows. Thanks to David Hobbs for initial report and to Aarni Koskela & Olle Lundberg for the patches.
- #34: (PR #35) Fix SFTP prefetching incompatibility with some SFTP servers regarding request/response ordering. Thanks to Richard Kettlewell.
- #268: Fix some missed renames of `ProxyCommand` related error classes. Thanks to Marius Gedminas for catch & patch.
- #252: (Fabric #1020) Enhanced the implementation of `ProxyCommand` to avoid a deadlock/hang condition that frequently occurs at `Transport` shutdown time. Thanks to Mateusz Kobos, Matthijs van der Vleuten and Guillaume Zitta for the original reports and to Marius Gedminas for helping test nontrivial use cases.
- #176: Fix AttributeError bugs in known_hosts file (re)loading. Thanks to Nathan Scowcroft for the patch & Martin Blumenstingl for the initial test case.
- #225: Note ecdsa requirement in README. Thanks to Amaury Rodriguez for the catch.
- #176: Fix AttributeError bugs in known_hosts file (re)loading. Thanks to Nathan Scowcroft for the patch & Martin Blumenstingl for the initial test case.
- #176: Fix AttributeError bugs in known_hosts file (re)loading. Thanks to Nathan Scowcroft for the patch & Martin Blumenstingl for the initial test case.

- #136: Add server-side support for the SSH protocol's 'env' command. Thanks to Benjamin Pollack for the patch.
- #152: Add tentative support for ECDSA keys. **This adds the ecdsa module as a new dependency of Paramiko.** The module is available at warner/python-ecdsa on Github and ecdsa on PyPI.
  - Note that you might still run into problems with key negotiation – Paramiko picks the first key that the server offers, which might not be what you have in your known_hosts file.
  - Mega thanks to Ethan Glasser-Camp for the patch.
- #199: Typo fix in the license header cross-project. Thanks to Armin Ronacher for catch & patch.
- #200: Fix an exception-causing typo in `demo_simple.py`. Thanks to Alex Buchanan for catch & Dave Foster for patch.
- #179: Fix a missing variable causing errors when an ssh_config file has a non-default AddressFamily set. Thanks to Ed Marshall & Tomaz Muraus for catch & patch.
- #156: Fix potential deadlock condition when using Channel objects as sockets (e.g. when using SSH gatewaying). Thanks to Steven Noonan and Frank Arnold for catch & patch.
- #199: Typo fix in the license header cross-project. Thanks to Armin Ronacher for catch & patch.
- #200: Fix an exception-causing typo in `demo_simple.py`. Thanks to Alex Buchanan for catch & Dave Foster for patch.
- #179: Fix a missing variable causing errors when an ssh_config file has a non-default AddressFamily set. Thanks to Ed Marshall & Tomaz Muraus for catch & patch.
- #168: Update config handling to properly handle multiple 'localforward' and 'remoteforward' keys. Thanks to Emre Yılmaz for the patch.
- #36: Fix the port-forwarding demo to avoid file descriptor errors. Thanks to Jonathan Halcrow for catch & patch.
- #162: Clean up HMAC module import to avoid deadlocks in certain uses of SSHClient. Thanks to Gernot Hillier for the catch & suggested fix.
- #168: Update config handling to properly handle multiple 'localforward' and 'remoteforward' keys. Thanks to Emre Yılmaz for the patch.
- #36: Fix the port-forwarding demo to avoid file descriptor errors. Thanks to Jonathan Halcrow for catch & patch.
- #162: Clean up HMAC module import to avoid deadlocks in certain uses of SSHClient. Thanks to Gernot Hillier for the catch & suggested fix.
- #87: Ensure updates to `known_hosts` files account for any updates to said files after Paramiko initially read them. (Includes related fix to guard against duplicate entries during subsequent `known_hosts` loads.) Thanks to `@sunweaver` for the contribution.
- #98: On Windows, when interacting with the PuTTY PAgeant, Paramiko now creates the shared memory map with explicit Security Attributes of the user, which is the same technique employed by the canonical PuTTY library to avoid permissions issues when Paramiko is running under a different UAC context than the PuTTY Ageant process. Thanks to Jason R. Coombs for the patch.
- #100: Remove use of PyWin32 in `win_pageant` module. Module was already dependent on ctypes for constructing appropriate structures and had ctypes implementations of all functionality. Thanks to Jason R. Coombs for the patch.
- #146: Indentation fixes for readability. Thanks to Abhinav Upadhyay for catch & patch.
- #153: (also #67) Warn on parse failure when reading known_hosts file. Thanks to `@glasserc` for patch.

- #154: (Fabric #876) Forwarded SSH agent connections left stale local pipes lying around, which could cause local (and sometimes remote or network) resource starvation when running many agent-using remote commands. Thanks to Kevin Tegtmeier for catch & patch.
- #142: (Fabric #811) SFTP put of empty file will still return the attributes of the put file. Thanks to Jason R. Coombs for the patch.
- #80: Expose the internal "is closed" property of the file transfer class `BufferedFile` as `.closed`, better conforming to Python's file interface. Thanks to `@smunaut` and James Hiscock for catch & patch.
- #113: Add `timeout` parameter to `SSHClient.exec_command` for easier setting of the command's internal channel object's timeout. Thanks to Cernov Vladimir for the patch.
- #71: Add `SFTPClient.putfo` and `.getfo` methods to allow direct uploading/downloading of file-like objects. Thanks to Eric Buehl for the patch.
- #115: Add convenience `get_pty` kwarg to `Client.exec_command` so users not manually controlling a channel object can still toggle PTY creation. Thanks to Michael van der Kolff for the patch.
- #116: Limit `Message.get_bytes` to an upper bound of 1MB to protect against potential DoS vectors. Thanks to `@mvschaik` for catch & patch.
- #127: Turn `SFTPFile` into a context manager. Thanks to Michael Williamson for the patch.
- #128: Defer FQDN resolution until needed, when parsing SSH config files. Thanks to Parantapa Bhattacharya for catch & patch.
- #110: Honor SSH config `AddressFamily` setting when looking up local host's FQDN. Thanks to John Hensley for the patch.
- #93: Overhaul SSH config parsing to be in line with `man ssh_config` (& the behavior of `ssh` itself), including addition of parameter expansion within config values. Thanks to Olle Lundberg for the patch.
- #66: Batch SFTP writes to help speed up file transfers. Thanks to Olle Lundberg for the patch.
- #102: Forego random padding for packets when running under `*-ctr` ciphers. This corrects some slowdowns on platforms where random byte generation is inefficient (e.g. Windows). Thanks to `@warthog618` for catch & patch, and Michael van der Kolff for code/technique review.
- #133: Fix handling of window-change events to be on-spec and not attempt to wait for a response from the remote sshd; this fixes problems with less common targets such as some Cisco devices. Thanks to Phillip Heller for catch & patch.
- #94: Remove duplication of SSH port constant. Thanks to Olle Lundberg for the catch.

CHAPTER 2

Frequently Asked/Answered Questions

2.1 Which version should I use? I see multiple active releases.

Please see *the installation docs* which have an explicit section about this topic.

2.2 Paramiko doesn't work with my Cisco, Windows or other non-Unix system!

In an ideal world, the developers would love to support every possible target system. Unfortunately, volunteer development time and access to non-mainstream platforms are limited, meaning that we can only fully support standard OpenSSH implementations such as those found on the average Linux distribution (as well as on Mac OS X and *BSD.)

Because of this, **we typically close bug reports for nonstandard SSH implementations or host systems**.

However, **closed does not imply locked** - affected users can still post comments on such tickets - and **we will always consider actual patch submissions for these issues**, provided they can get +1s from similarly affected users and are proven to not break existing functionality.

2.3 I'm having strange issues with my code hanging at shutdown!

Make sure you explicitly `.close()` your connection objects (usually `SSHClient`) if you're having any sort of hang/freeze at shutdown time!

Doing so isn't strictly necessary 100% of the time, but it is almost always the right solution if you run into the various corner cases that cause race conditions, etc.

CHAPTER 3

Installing

Note: These instructions cover Paramiko 2.0 and above. If you're looking to install Paramiko 1.x, see *Installing (1.x)*. However, **the 1.x line relies on insecure dependencies** so upgrading is strongly encouraged.

3.1 Paramiko itself

The recommended way to get Paramiko is to **install the latest stable release** via pip:

```
$ pip install paramiko
```

We currently support **Python 2.7, 3.4+, and PyPy**. Users on Python 2.6 or older (or 3.3 or older) are urged to upgrade.

Paramiko has only a few **direct dependencies**:

- The big one, with its own sub-dependencies, is Cryptography; see *its specific note below* for more details;
- bcrypt, for Ed25519 key support;
- pynacl, also for Ed25519 key support.

There are also a number of **optional dependencies** you may install using setuptools 'extras':

- If you want all optional dependencies at once, use `paramiko[all]`.
- For `Match exec` config support, use `paramiko[invoke]` (which installs Invoke).
- For GSS-API / SSPI support, use `paramiko[gssapi]`, though also see *the below subsection on it* for details.
- `paramiko[ed25519]` references the dependencies for Ed25519 key support.
  - As of Paramiko 2.x this doesn't technically do anything, as those dependencies are core installation requirements.
  - However, you should use this for forwards compatibility; 3.0 will drop those dependencies from core, leaving them purely optional.

3.1.1 Release lines

Users desiring stability may wish to pin themselves to a specific release line once they first start using Paramiko; to assist in this, we guarantee bugfixes for the last 2-3 releases including the latest stable one.

This typically spans major & minor versions, so even if e.g. 3.1 is the latest stable release, it's likely that bugfixes will occasionally come out for the latest 2.x and perhaps even 1.x releases, as well as for 3.0. New feature releases for previous major-version lines are less likely but not unheard of.

If you're unsure which version to install:

- **Completely new users** should always default to the **latest stable release** (as above, whatever is newest / whatever shows up with `pip install paramiko`.)
- **Users upgrading from a much older version** (e.g. 1.7.x through 1.10.x) should probably get the **oldest actively supported line** (check the *Changelog* for recent releases).
- **Everybody else** is hopefully already "on" a given version and can carefully upgrade to whichever version they care to, when their release line stops being supported.

3.2 Cryptography

Cryptography provides the low-level (C-based) encryption algorithms we need to implement the SSH protocol. It has detailed installation instructions (and an FAQ) which you should read carefully.

In general, you'll need one of the following setups:

- On Windows or Mac OS X, provided your `pip` is modern (8.x+): nothing else is required. `pip` will install statically compiled binary archives of Cryptography & its dependencies.
- On Linux, or on other platforms with older versions of `pip`: you'll need a C build toolchain, plus development headers for Python, OpenSSL and `libffi`. Again, see Cryptography's install docs; these requirements may occasionally change.

  > **Warning:** If you go this route, note that **OpenSSL 1.0.1 or newer is effectively required**. Cryptography 1.3 and older technically allow OpenSSL 0.9.8, but 1.4 and newer - which Paramiko will gladly install or upgrade, if you e.g. `pip install -U` - drop that support.

- Similarly, Cryptography 3.4 and above require Rust language tooling to install from source; once again see Cryptography's documentation for details here, such as their Rust install section and this FAQ entry.

3.3 Optional dependencies for GSS-API / SSPI / Kerberos

In order to use GSS-API/Kerberos & related functionality, additional dependencies are required. It hopefully goes without saying but **all platforms** need **a working installation of GSS-API itself**, e.g. Heimdal.

Note: If you use Microsoft SSPI for kerberos authentication and credential delegation, make sure that the target host is trusted for delegation in the active directory configuration. For details see: http://technet.microsoft.com/en-us/library/cc738491%28v=ws.10%29.aspx

---

3.3.1 The `gssapi` "extra" install flavor

If you're installing via `pip` (recommended), you should be able to get the optional Python package requirements by changing your installation to refer to `paramiko[gssapi]` (from simply `paramiko`), e.g.:

```
pip install "paramiko[gssapi]"
```

(Or update your `requirements.txt`, or etc.)

3.3.2 Manual dependency installation

If you're not using `pip` or your `pip` is too old to support the "extras" functionality, the optional dependencies are as follows:

- All platforms need pyasn1 `0.1.7` or later.
- **Unix** needs: gssapi `1.4.1` or better.
  - An alternative is the python-gssapi library (`0.6.1` or above), though it is no longer maintained upstream, and Paramiko's support for its API may eventually become deprecated.
- **Windows** needs pywin32 `2.1.8` or better.

CHAPTER 4

Installing (1.x)

Note: Installing Paramiko 2.0 or above? See *Installing* instead.

This document includes legacy notes on installing Paramiko 1.x (specifically, 1.13 and up). Users are strongly encouraged to upgrade to 2.0 when possible; PyCrypto (the dependency covered below) is no longer maintained and contains security vulnerabilities.

4.1 General install notes

- Python 2.6+ and 3.3+ are supported; Python <=2.5 and 3.0-3.2 are **not supported**.
- See the note in the main install doc about *Release lines* for details on specific versions you may want to install.

  Note: 1.x will eventually be entirely end-of-lifed.

- Paramiko 1.7-1.14 have only one dependency: *PyCrypto*.
- Paramiko 1.15+ (not including 2.x and above) add a second, pure-Python dependency: the `ecdsa` module, trivially installable via PyPI.
- Paramiko 1.15+ (again, not including 2.x and up) also allows you to optionally install a few more dependencies to gain support for *GSS-API/Kerberos*.
- Users on Windows may want to opt for the *ActivePython and PyPM* approach.

4.2 PyCrypto

PyCrypto provides the low-level (C-based) encryption algorithms we need to implement the SSH protocol. There are a couple gotchas associated with installing PyCrypto: its compatibility with Python's package tools, and the fact that

it is a C-based extension.

4.2.1 C extension

Unless you are installing from a precompiled source such as a Debian apt repository or RedHat RPM, or using *pypm*, you will also need the ability to build Python C-based modules from source in order to install PyCrypto. Users on **Unix-based platforms** such as Ubuntu or Mac OS X will need the traditional C build toolchain installed (e.g. Developer Tools / XCode Tools on the Mac, or the `build-essential` package on Ubuntu or Debian Linux – basically, anything with `gcc`, `make` and so forth) as well as the Python development libraries, often named `python-dev` or similar.

Slow vs fast crypto math

PyCrypto attempts to use the `gmp` C math library if it is present on your system, which enables what it internally calls "fastmath" (`_fastmath.so`). When those headers are not available, it falls back to "slowmath" (`_slowmath.py`) which is a pure-Python implementation.

Real-world tests have shown significant benefits to using the C version of this code; thus we strongly recommend you install the `gmp` development headers **before** installing Paramiko/PyCrypto. E.g.:

```
$ apt-get install libgmp-dev # or just apt
$ yum install gmp-devel # or dnf
$ brew install gmp
```

If you're unsure which version of math you've ended up with, a quick way to check is to examine whether `_fastmath.so` or `_slowmath.py` appears in the output of:

```
from Crypto.PublicKey import RSA
print(RSA._impl._math)
```

Windows

For **Windows** users we recommend using *ActivePython and PyPM*, installing a C development environment such as Cygwin or obtaining a precompiled Win32 PyCrypto package from voidspace's Python modules page.

Note: Some Windows users whose Python is 64-bit have found that the PyCrypto dependency `winrandom` may not install properly, leading to ImportErrors. In this scenario, you'll probably need to compile `winrandom` yourself via e.g. MS Visual Studio. See Fabric #194 for info.

4.3 ActivePython and PyPM

Windows users who already have ActiveState's ActivePython distribution installed may find Paramiko is best installed with its package manager, PyPM. Below is example output from an installation of Paramiko via `pypm`:

```
C:\> pypm install paramiko
The following packages will be installed into "%APPDATA%\Python" (2.7):
 paramiko-1.7.8 pycrypto-2.4
Get: [pypm-free.activestate.com] paramiko 1.7.8
Get: [pypm-free.activestate.com] pycrypto 2.4
Installing paramiko-1.7.8
```

```
Installing pycrypto-2.4
C:\>
```

4.4 Optional dependencies for GSS-API / SSPI / Kerberos

First, see the main install doc's notes: *Optional dependencies for GSS-API / SSPI / Kerberos* - everything there is required for Paramiko 1.x as well.

Additionally, users of Paramiko 1.x, on all platforms, need a final dependency: pyasn1 `0.1.7` or better.

CHAPTER 5

Contributing

5.1 How to get the code

Our primary Git repository is on Github at paramiko/paramiko; please follow their instructions for cloning to your local system. (If you intend to submit patches/pull requests, we recommend forking first, then cloning your fork. Github has excellent documentation for all this.)

5.2 How to submit bug reports or new code

Please see this project-agnostic contribution guide - we follow it explicitly. Again, our code repository and bug tracker is on Github.

Our current changelog is located in `sites/www/changelog.rst` - the top level files like `ChangeLog.*` and `NEWS` are historical only.

CHAPTER 6

Contact

You can get in touch with the developer & user community in any of the following ways:

- Submit contributions on Github - see the *Contributing* page.
- Follow `@bitprophet` on Twitter, though it's not a dedicated account and mostly just retweets funny pictures.
- Subscribe to the `paramiko` category on the developer's blog: http://bitprophet.org/categories/paramiko/

Index

R

RFC
    RFC 8308, 6
    RFC 8332, 6