service_identity Documentation

Release 1.0.0

Hynek Schlawack

June 15, 2014

Contents

Release v1.0.0 (*What's new?*).

TL;DR: Use this package if you use pyOpenSSL and don't want to be MITMed.

`service_identity` aspires to give you all the tools you need for verifying whether a certificate is valid for the intended purposes.

In the simplest case, this means *host name verification*. However, `service_identity` implements RFC 6125 fully and plans to add other relevant RFCs too.

`service_identity`'s documentation lives at Read the Docs, the code on GitHub.

CHAPTER 1

User's Guide

1.1 Installation and Requirements

1.1.1 Installation

```
$ pip install service_identity
```

1.1.2 Requirements

Python 2.6, 2.7, 3.3 and later, as well as PyPy are supported.

Additionally, the following PyPI modules are required:

- characteristic
- pyOpenSSL `>= 0.12` (`0.14` strongly recommended)
- pyasn1
- pyasn1-modules

Optionally, idna `>= 0.6` can be used for internationalized domain names (IDN), i.e. non-ASCII domains. Unfortunately it's required because Python's IDN support in the standard library is outdated even in the latest releases.

If you need Python 3.2 support, you will have to use the latest 0.2.x release. It will receive bug fix releases if necessary but other than that no further development is planned.

1.2 Implemented Standards

1.2.1 Present

- `dNSName` with fallback to `CN` (DNS-ID, aka host names, RFC 6125).
- `uniformResourceIdentifier` (URI-ID, RFC 6125).
- SRV-ID (RFC 6125)

1.2.2 Future

- `xmppAddr` (RFC 3920).
- `iPAddress` (RFC 2818).
- `nameConstraints` extensions (RFC 3280).

1.3 API

Note: The APIs for RFC 6125 verification beyond DNS-IDs (i.e. hostnames) aren't public yet. They are in place and used by the documented high-level APIs though. Eventually they will become public. If you'd like to play with them and provide feedback have a look at the `verify_service_identity` function in the _common module.

`service_identity.pyopenssl.`**`verify_hostname`**(*connection*, *hostname*)

Verify whether the certificate of *connection* is valid for *hostname*.

Parameters

- **connection** (`OpenSSL.SSL.Connection`) – A pyOpenSSL connection object.
- **hostname** (`unicode`) – The hostname that *connection* should be connected to.

Raises

- **service_identity.VerificationError** – If *connection* does not provide a certificate that is valid for *hostname*.
- **service_identity.CertificateError** – If the certificate chain of *connection* contains a certificate that contains invalid/unexpected data.

Returns *None*

In practice, this may look like the following:

```
from __future__ import absolute_import, division, print_function

import socket

from OpenSSL import SSL
from service_identity import VerificationError
from service_identity.pyopenssl import verify_hostname


ctx = SSL.Context(SSL.SSLv23_METHOD)
ctx.set_verify(SSL.VERIFY_PEER, lambda conn, cert, errno, depth, ok: ok)
ctx.set_default_verify_paths()

hostname = u"twistedmatrix.com"
conn = SSL.Connection(ctx, socket.socket(socket.AF_INET, socket.SOCK_STREAM))
conn.connect((hostname, 443))

try:
   conn.do_handshake()
   verify_hostname(conn, hostname)
   # Do your super-secure stuff here.
except SSL.Error as e:
   print("TLS Handshake failed: {0!r}.".format(e.args[0]))
except VerificationError:
   print("Presented certificate is not valid for {0}.".format(hostname))
finally:
   conn.shutdown()
   conn.close()
```

exception `service_identity.`**`VerificationError`**

Verification failed.

exception `service_identity.`**`CertificateError`**

A certificate contains invalid or unexpected data.

1.4 Project Information

1.4.1 License and Hall of Fame

`service_identity` is licensed under the permissive MIT license. The full license text can be also found in the source code repository.

Authors

`service_identity` is currently maintained by Hynek Schlawack.

The development is kindly supported by Variomedia AG.

If you think you've found a security-relevant bug, please contact me privately and ideally encrypt your message using PGP. I will then work with you on a responsible resolution. You can find my contact information and PGP data on my homepage.

The following wonderful people contributed directly or indirectly to this project:

- Alex Stapleton
- Glyph
- Paul Kehrer

Please add yourself here alphabetically when you submit your first pull request.

1.4.2 How To Contribute

Every open source project lives from the generous help by contributors that sacrifice their time and `service_identity` is no different.

To make participation as pleasant as possible, this project adheres to the Code of Conduct by the Python Software Foundation.

Here are a few hints and rules to get you started:

- Add yourself to the AUTHORS.rst file in an alphabetical fashion. Every contribution is valuable and shall be credited.
- If your change is noteworthy, add an entry to the changelog.
- No contribution is too small; please submit as many fixes for typos and grammar bloopers as you can!
- Don't *ever* break backward compatibility. If it ever *has* to happen for higher reasons, `service_identity` will follow the proven procedures of the Twisted project.

- *Always* add tests and docs for your code. This is a hard rule; patches with missing tests or documentation won't be merged. If a feature is not tested or documented, it doesn't exist.
- Obey PEP 8 and PEP 257.
- Write good commit messages.
- Ideally, squash your commits, i.e. make your pull requests just one commit.

Note: If you have something great but aren't sure whether it adheres – or even can adhere – to the rules above: **please submit a pull request anyway**!

In the best case, we can mold it into something, in the worst case the pull request gets politely closed. There's absolutely nothing to fear.

Thank you for considering to contribute to `service_identity`! If you have any question or concerns, feel free to reach out to me. I can usually be found on the `#cryptography-dev` channel on freenode.

1.4.3 History

1.0.0 (2014-06-15)

- Move into the Python Cryptography Authority's GitHub account.
- Drop support for Python 3.2. There is no justification to add complexity and unnecessary function calls for a Python version that nobody uses.
- Move exceptions into `service_identity.exceptions` so tracebacks don't contain private module names.
- Promoting to stable since Twisted 14.0 is optionally depending on `service_identity` now.
- Use characteristic instead of a home-grown solution.
- `idna` 0.6 did some backward-incompatible fixes that broke Python 3 support. This has been fixed now therefore `service_identity` only works with `idna` 0.6 and later. Unfortunately since `idna` doesn't offer version introspection, `service_identity` can't warn about it.

0.2.0 (2014-04-06)

This release contains multiple backward-incompatible changes.

- Refactor into a multi-module package. Most notably, `verify_hostname` and `extract_ids` live in the `service_identity.pyopenssl` module now.
- `verify_hostname` now takes an `OpenSSL.SSL.Connection` for the first argument.
- Less false positives in IP address detection.
- Officially support Python 3.4 too.
- More strict checks for URI_IDs.

0.1.0 (2014-03-03)

- Initial release.

CHAPTER 2

Indices and tables

- *genindex*
- *search*