Contributing to mbed TLS

ARM

Simon Butcher
Principal Security Engineer
mbed TLS Tech Lead

Wyboston Lakes
30th March 2017

Agenda

- mbed TLS as an Open Source Project
- Why Contribute?
- How to make a code contribution
- Where to contribute code – mbed OS and mbed TLS
- How to get your code accepted
- Common mistakes
- Overview of source code directory structure
- Review criteria
- Implementation tips
- Reporting Security Issues

mbed TLS is an Open Source Project

- All source code is hosted on github in the open, with no binary distributions or undocumented APIs
- Development of new features is done in the open transparently
- Community contributions are gratefully received, as long as contributors have signed a Contributor License Agreements or are mbed Partners, and code meets our standards, with tests and appropriate documentation

(…although Security defects are developed on a private github repository)

Why Contribute?

- To ensure ***your*** hardware is fully supported
- To provide a feature your business needs, which isn't supported
- To fix a bug in the platform that is directly impacting you
- ..or even to fix that typo that's annoying you

How to make a code contribution

1. Develop a great new feature or fix a bug
2. Push it as a feature branch to your own 'personal' or 'corporate' repository on github
3. Create a pull request against the mbed OS or mbed TLS repositories
4. Work with us as we test and review the code and give feedback!
5. Watch the change get merged

…and that's it!

ARM

…but, we don't accept all code that's contributed.

How to get your code accepted

- Follow our guidelines, as described in these slides and online to avoid possible issues in review
- Implement all relevant API's as they are specified, including options
- Test your code as widely as possible
- Provide code that passes the mbed OS or mbed TLS CI test systems
- Review your code internally
  - If we spot issues that require rework, it will delay acceptance
- Provide your code as a pull request
  - Don't provide it as a .zip, tarball of your repository or patch if you can avoid it
- Provide code that is easy to follow and well commented

…and what causes patches to be delayed or rejected

- Ignore feedback, or don't change or reply to review feedback
  - We are open-minded, and don't mind discussing review feedback, but you need to tell us if you disagree
- Submit a pull request that changes 1000's of lines and 10's of files
  - **Big** pull requests are more complex and take longer to review. A series of smaller PR's are likely to be adopted more quickly than one big one
- Submit a pull request to mbed OS, that is intended to change the mbed TLS library (as described in a later slide)
- Change an interface without previous discussion with us
- Change the architecture without previous discussion with us
- Break someone else's platform, target or feature
- Submit code that obviously doesn't follow our guidelines

Where to place changes and new code

- Changes to mbed TLS the library should always be done upstream in the mbed TLS repository
- Hardware Acceleration code should be placed in the TARGETS directory in the mbed TLS feature directory in mbed OS
- Entropy devices are part of the mbed OS HAL, and code should go in the mbed OS HAL TARGETS directory

Different projects, different landing points

- mbed TLS Hardware Acceleration drivers are platform specific and are held along with mbed OS specific platform code are held in the mbed OS github repository
  - https://github.com/ARMmbed/mbed-os
- mbed TLS is a standalone project, and is held in the mbed TLS github repository
  - https://github.com/ARMmbed/mbedtls

Structure of mbed TLS inside mbed OS

mbed-os / features / mbedtls

importer	Script to import mbed TLS from it's own repo
inc / mbedtls*	Header files for the library
platform	mbed OS Platform integration code
src*	mbed TLS library source files
targets	Directory for Target specific mbed TLS code
README.md	Readme
VERSION.txt	mbed TLS Version file

** Imported library code cannot be modified in mbed OS*

ARM

Review Criteria - General

- What we look for in code reviews
  1. Does it pass the CI tests?
  2. Is the code well commented, using doxygen where appropriate?
  3. The code is logically correct, makes sense and is not too hard to follow
  4. Does the source code follow the style guide defined in the coding standards?
     - *Note, the coding standards are currently different for mbedTLS from mbed OS*
  5. Error handling is properly done in all cases, and the correct error is reported

Review Criteria – mbed TLS specific

- What we look for in code reviews *for mbed TLS contributions:*
  1. If the code introduces a new feature or fixes a bug, a new test case should always be considered. If one isn't added, it needs to be explained why not
  2. Does it pass the CI tests?
  3. Well commented code, using doxygen
  4. Source code that follows the styles defined in the coding standards
  5. All key material and calculation byproducts are zeroed before exit
  6. Hardware acceleration driver code is thread safe
     - *Calls to mbed TLS can be from any thread, and the alternative implementation must consider resource contention*
  7. No calls to libc or any other external library can be made, and all such calls must go through the platform abstraction layer

Secure coding tips

- Always check buffer bounds
  - Buffer overflows
- Validate input wherever possible
  - Buffer overflows
  - Key misuse
- Pay attention to possible arithmetic overflows
  - Buffer overflows
  - Bypassing validation
- Compile code with highest warning levels
- Fail securely
  - Verify and handle every possible error
  - Leave the device in a secure state when exiting
  - Clean buffers and registers *(see first bullet-point)*

Crypto coding tips

- Erase buffers containing secret data after use
  - Buffers in memory – stack and heap
  - Registers/memory on the device
- Buffer overreads are dangerous too!
  - HEARTBLEED
  - Always bounds check data read as well as data written
- Secure by default
  - Disable vulnerable/legacy options by default
  - Deny access by default
- Be aware of side-channels
  - Compare buffers in constant time
  - In general aim for constant time implementation when handling secret data

TRNG pitfalls

- Returned bits != bits of entropy
- A random sensor + deskew != TRNG
- Selftest is a very basic sanity test
- Passing selftest does not mean that the entropy source is strong

What to do if you find a security issue?

- If you find a security issue, they can be confidentially reported using our whitehat email alias:

  whitehat@polarssl.org

- To encrypt any bug reports, you can use our PGP key herehttps://tls.mbed.org/public/whitehat.pub

Questions?