Closed Issue created 2 years ago by hdevalence

Potential consensus divergence from Ed25519 edge cases

Ed25519 poses risks in consensus-critical applications, because (a) the spec does not require that implementations agree on whether signatures are valid and (b) in practice, implementations differ from the spec and from each other.

In the context of working to address this issue in Zcash (resulting in ZIP215), I created a set of 196 test vectors, consisting of hex-encoded (public key, signature) pairs on the message `b"Zcash"`. Running these test vectors across various other Ed25519 implementations reveals a wide divergence in behaviour (see 1 2 for additional context).

From a quick look at the Tor source and some tips from Teor, it looks like Tor has four different verification codepaths: ref10 `open`, ref10 `open_batch`, donna `open`, and donna `open_batch`. But I'm not entirely sure whether these are all used, because that requires a deeper knowledge of the codebase than I have.

The test vectors can be found in C-friendly format (thanks to Patrick Steuer) here: https://github.com/p-steuer/ossl-eddsa-tests

To upload designs, you'll need to enable LFS and have an admin enable hashed storage. More information

Tasks 0

No tasks are currently assigned. Use tasks to break down this issue into smaller parts.

Linked items 0

Link issues together to show that they're related. Learn more.

Related merge requests 1

Disable ed25519-donna's batch verification.

!414

Activity

Roger Dingledine @arma · 2 years ago Reporter

Tagging @asn and @sysrqb as two people that I recently saw talking about the ed key formats (in the context of onion services, and we should think about both onion services and relay identities here).

H

hdevalence @hdevalence · 2 years ago Author

I hooked the test vectors I linked above into Tor's test suite (testing code is available here, though it's very much works-once quality), and tested the `donna` and `ref10` implementations in single verification and the `donna` batch verification implementation.

1. At least on these test vectors, it appears that both `ref10` and `donna` produce identical results on single verification. Neither agree with RFC8032, but this is unsurprising because the RFC was written several years later and included extra incompatible checks.
2. Neither `ref10` nor `donna` appear to check that the `s` component of the signature is canonically encoded (though I did not test this), which may open the door for a signature malleability attack.
3. The batch verification implementation in `donna` does not match the single verification. There are signatures that verify in a batch but not individually, and vice versa.
4. The batch verification implementation does not multiply by the cofactor, so it is nondeterministic in practice (not just with negligible probability), and will return different results on different test runs.
5. A DoS attack allows an adversary who can control the input to batch verification to crash Tor with `SIGABRT`. This happens because of (3) and the fact that the batch verification implementation falls back to single verification and then sanity-checks the results of the fallback. Passing signatures that verify individually but not as part of a batch (e.g., `[case, case, case, case]` where `case` is index `16` of the test vector list) triggers an assertion when the sanity check fails. A longer snippet is posted below.
6. Batch verification is only used when the batch size is greater than 3, because this is the crossover point in performance for `donna`'s implementation (but not for other implementations, where batches of size 1 can be just as fast as single verification). However, the batch verification codepath is never used because no batch of size greater than 3 is ever passed in. The only caller that doesn't hardcode the batch size is `if (ed25519_checksig_batch(NULL, check, n_checkable) < 0) {` in `torcert.c:635`, but this function builds the `check` list using an `ADDCERT` macro that is called at most twice.

Because batch verification is never actually used, neither the DoS vector nor the verification inconsistency is exploitable, but only by accident -- if batch verification had been used, or is used in the future, it might be.

One fix is to upgrade to using ZIP215 rules, but this would require swapping out the ed25519 implementation. Another, easier alternative is to delete the unused batch verification code.

Here's a more complete snippet of the assertion failure:

```
Aug 02 22:51:18.466 [err] tor_assertion_failed_(): Bug: src/lib/crypt_ops/crypto_ed25519.c:436: ed25519_checksig_batch__re
Aug 02 22:51:18.467 [err] Bug: Tor 0.4.3.5 (git-5ceb60e5ca61fec9): Assertion ((res == 0) && !all_ok) || ((res < 0) && all_
Aug 02 22:51:18.467 [err] Bug:     ./src/test/test(log_backtrace_impl+0x57) [0x560a1f294a17] (on Tor 0.4.3.5 5ceb60e5ca61
Aug 02 22:51:18.467 [err] Bug:     ./src/test/test(tor_assertion_failed_+0x148) [0x560a1f2900c8] (on Tor 0.4.3.5 5ceb60e5
Aug 02 22:51:18.467 [err] Bug:     ./src/test/test(ed25519_checksig_batch__real+0x377) [0x560a1f232c17] (on Tor 0.4.3.5 5
Aug 02 22:51:18.467 [err] Bug:     ./src/test/test(+0x1ba83a) [0x560a1ee6783a] (on Tor 0.4.3.5 5ceb60e5ca61fec9)
Aug 02 22:51:18.467 [err] Bug:     ./src/test/test(+0x41d398) [0x560a1f0ca398] (on Tor 0.4.3.5 5ceb60e5ca61fec9)
Aug 02 22:51:18.467 [err] Bug:     ./src/test/test(testcase_run_one+0x74) [0x560a1f0ca474] (on Tor 0.4.3.5 5ceb60e5ca61fe
Aug 02 22:51:18.467 [err] Bug:     ./src/test/test(tinytest_main+0x114) [0x560a1f0cad24] (on Tor 0.4.3.5 5ceb60e5ca61fec9
Aug 02 22:51:18.467 [err] Bug:     ./src/test/test(main+0x3b7) [0x560a1ed3a117] (on Tor 0.4.3.5 5ceb60e5ca61fec9)
Aug 02 22:51:18.467 [err] Bug:     /lib64/libc.so.6(__libc_start_main+0xf2) [0x7fb0c8e1d042] (on Tor 0.4.3.5 5ceb60e5ca61
Aug 02 22:51:18.467 [err] Bug:     ./src/test/test(_start+0x2e) [0x560a1ed3a22e] (on Tor 0.4.3.5 5ceb60e5ca61fec9)
fish: "./src/test/test crypto/zip215ve…" terminated by signal SIGABRT (Abort)
```

George Kadianakis @asn · 2 years ago Contributor

Hello. Came back from vacations a few days ago. This is on my list and trying to get to it.

George Kadianakis assigned to @asn 2 years ago

Nick Mathewson added Bug label 2 years ago

George Kadianakis @asn · 2 years ago Contributor

Hello Henry! Thanks for this bug report and for doing all this work. Sorry for the late response.

My intuition right now is to take the easy way out. That is, remove the unused batch verification code for now; and if we ever need it we should spend some extra time to make it work right based on the findings of this ticket. However, I'd like to wait for @nickm to give his opinion here, because it might be that we do need batch verification right now and it's just accidental that it never gets activated.

Other than this, I think the other actionable item from the original post is the part about the `s` component of the signature is canonically encoded. Can you give me some more information on this? I looked at ZIP215 but didn't find anything about the `s` canonical encoding. Is it the part about `S` representing an integer less than `l`?

Thanks a lot :)

George Kadianakis added Doing label 1 year ago

George Kadianakis @asn · 1 year ago Contributor

OK I took some time for this ticket again:

a) The batch verification logic of donna is indeed problematic and the way that Tor does the fallback (see `fallback` label when the batch verification check fails) in `ed25519_checksig_batch()` can indeed cause crashes. The good thing is that the batch verification is never actually called (it's only enabled if four or more sigs are passed which never happens). Fixing the math would be a pretty big move since upstream is also broken, so I opted for removing the batching functionality. Using a proper library like ed25519-dalek in arti would allow us to do batch verification again. See Henry's excellent blog post for more details.

See !414 (closed) for the fix commit. I also pushed https://gitlab.torproject.org/asn/tor/-/tree/bug40078-test-vectors which makes sure that the remaining ZIP125 test vectors don't break tor. It's up to us whether we want to include the test vectors in tor. I saw that OpenSSL did not include them so I took the easy way out myself.

b) Henry's point (2) is also valid. We are indeed not doing the right check to see if the received `S` is canonical. In particular, ed25519-donna does the weak check for `S` canonicity (checking just the three most-significant bits) as described in the section `Checking for non-canonical S` of this paper.

Our check will catch most non-canonical signatures but there are a bunch of them that can escape. In particular, the paper says that *for honestly generated signatures, the probability that the fourth most significant bit (252th bit) is set is very small, roughly 1/2^{128}* .

This means that an attacker can craft signatures that are malleable (for each such evil signature, there is another one that verifies). And also there is an **extremely** small chance that non-attacker signatures can be malleable. I'm not sure what are the implications of ed25519 signature malleability in our protocol. Certainly one can go around replay-caches; what other attacks are there?

Fixing this would require us to implement the full canonicity check as described by that paper. I didn't find any other C codebase that implements the full canonicity check (see `Table 5` on the paper) so I opted to not try to do it at this point.

George Kadianakis unassigned @asn 1 year ago

Nick Mathewson @nickm · 1 year ago Owner

I believe this turned into #40446 (closed) and !414 (closed).

Nick Mathewson closed 1 year ago

Gaba added BugSmashFund label 5 months ago