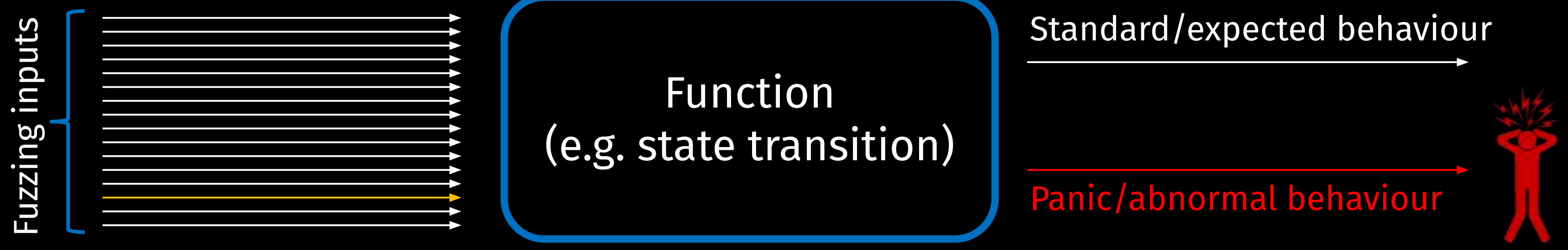
Fun With Fuzzing

Adrian Manning

Fuzzing 101

Fuzzing 101



```
!8;`kZN(d94LGCS*GN}=j)k:,pWwM?+#znJ'7q@%ua'6bL@|{'{!~'~-`@-@&\{~'{:'`
```

Fuzzing in Rust

- **Rust Fuzz Book - https://rust-fuzz.github.io/book/introduction.html**

Targets

```
#![no_main]
#[macro_use] extern crate libfuzzer_sys;
extern crate url;

fuzz_target!(|data: &[u8]| {
    if let Ok(s) = std::str::from_utf8(data) {
        let _ = url::Url::parse(s);
    }
});
```

```
cargo fuzz run <fuzz target name>
```

Fuzzing in Rust

Structured Fuzzing

```rust
// src/lib.rs

#[derive(Clone, Debug)]
#[cfg_attr(feature = "arbitrary", derive(arbitrary::Arbitrary))]
pub struct Rgb {
    pub r: u8,
    pub g: u8,
    pub b: u8,
}
```

```rust
// fuzz/fuzz_targets/rgb_to_hsl_and_back.rs

libfuzzer_sys::fuzz_target!(|color: Rgb| {
    let hsl = color.to_hsl();
    let rgb = hsl.to_rgb();

    // This should be true for all RGB -> HSL -> RGB conversions!
    assert_eq!(color, rgb);
});
```

Beacon Nodes....

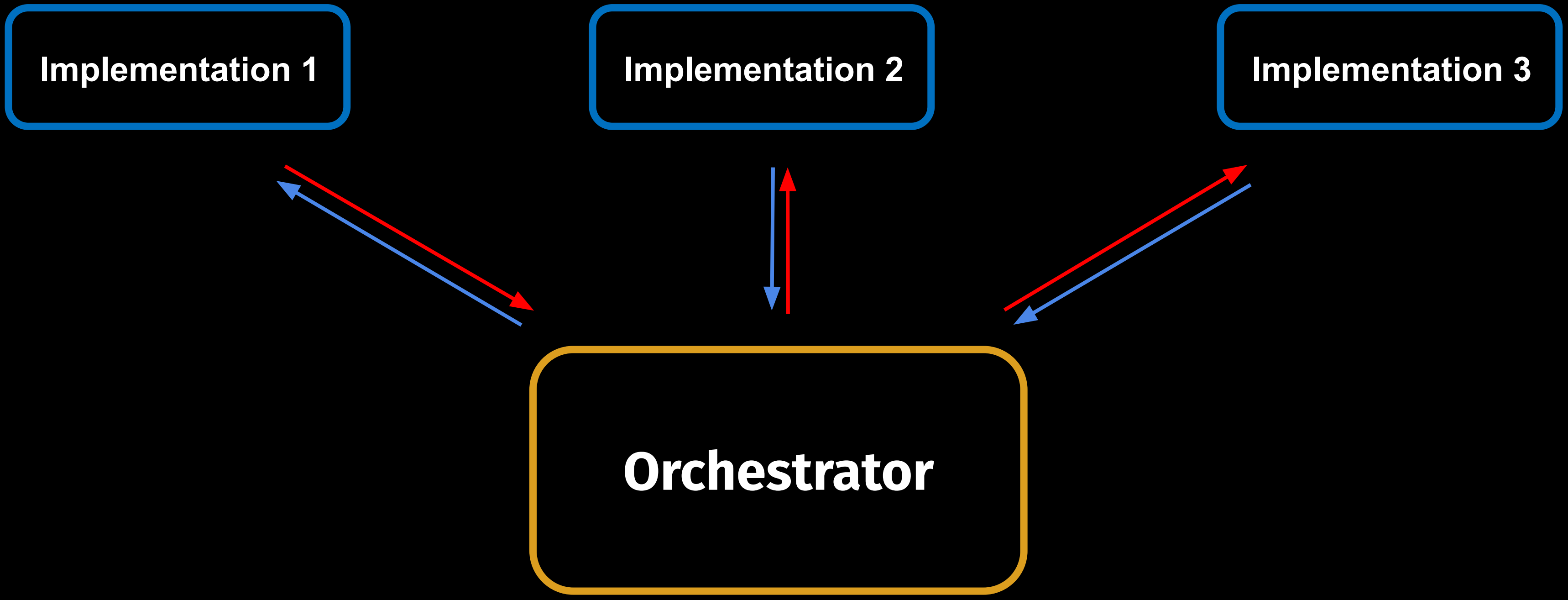
The Beacon Chain

- 5 Production Clients

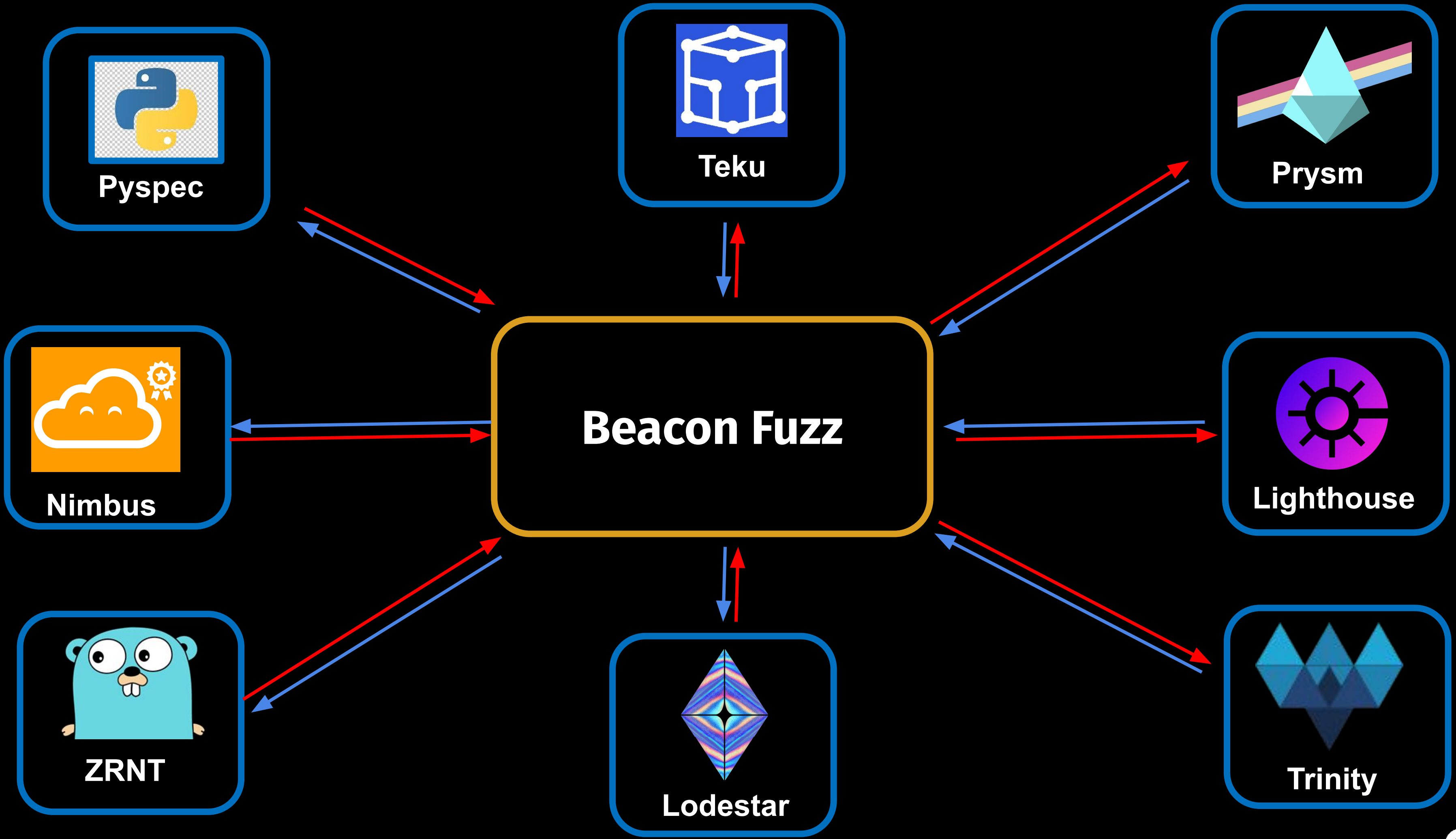
Lighthouse	Lodestar	Nimbus	Prysm	Teku

- Network Launched on December 1st, 2020

Differential Fuzzing



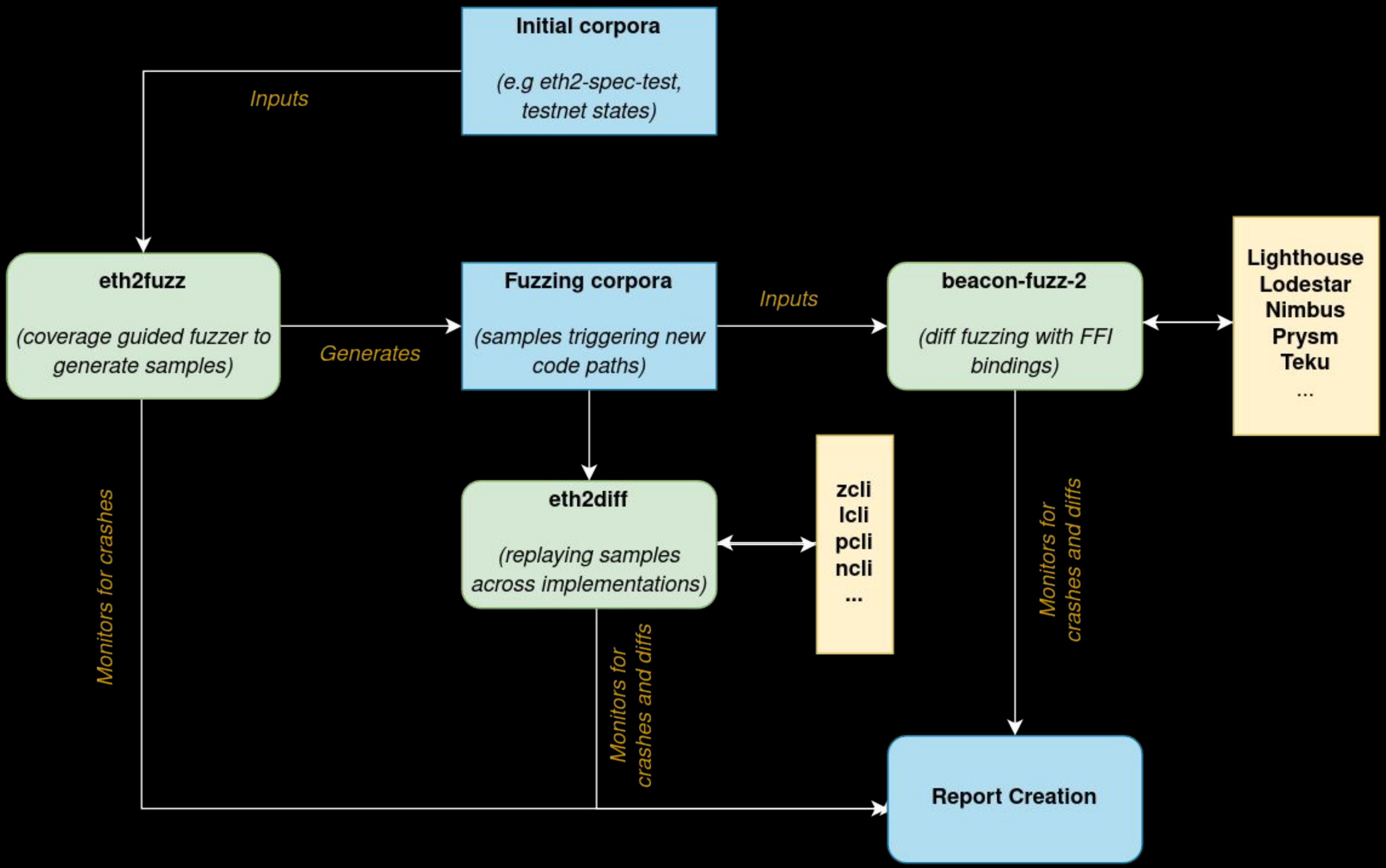
Differential Fuzzing for CL



Beacon Fuzz

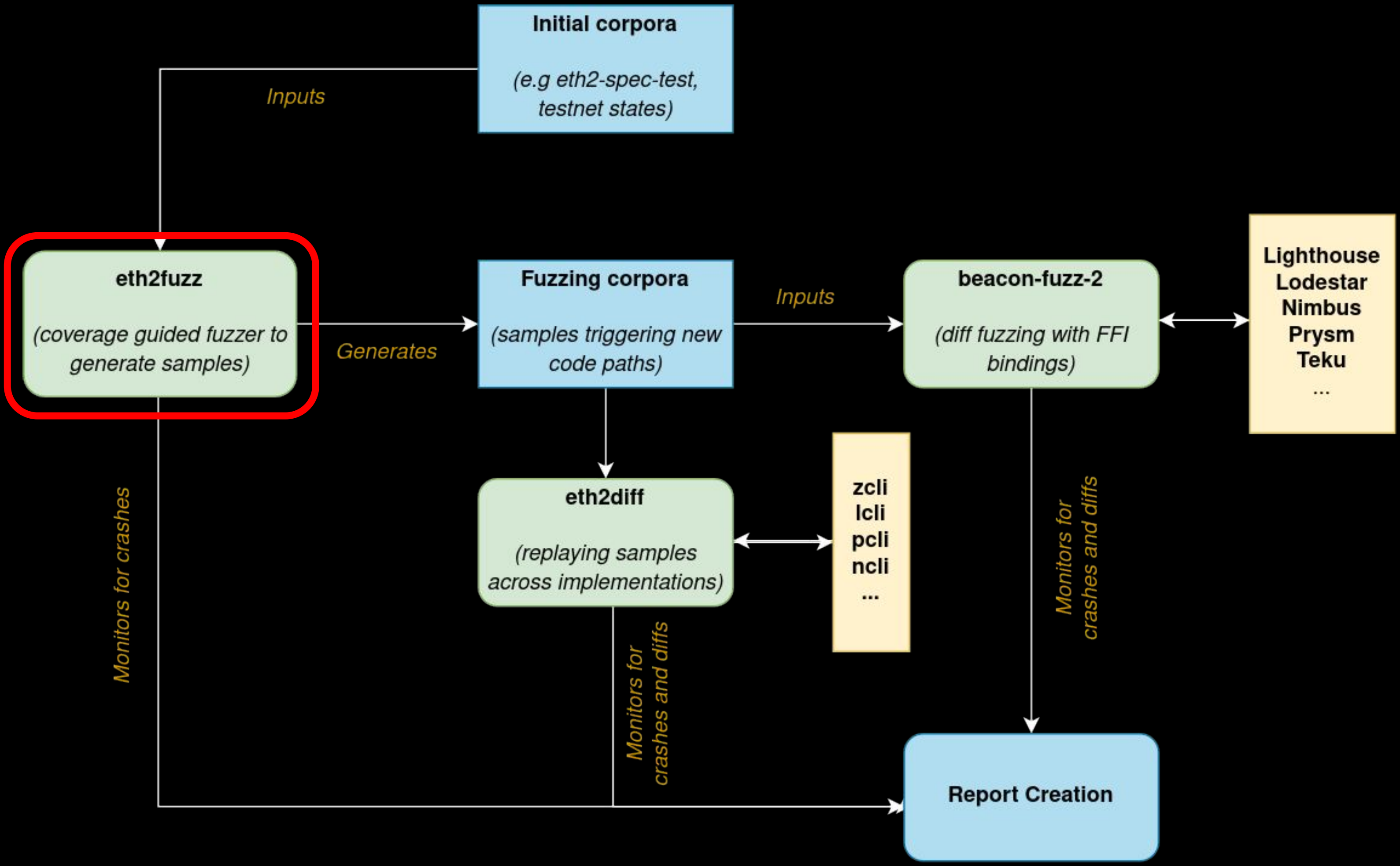
https://github.com/sigp/beacon-fuzz

Beacon Fuzz Architecture



Initial corpora
(e.g eth2-spec-test, testnet states)
Inputs
eth2fuzz
(coverage guided fuzzer to generate samples)
Generates
Fuzzing corpora
(samples triggering new code paths)
Inputs
beacon-fuzz-2
(diff fuzzing with FFI bindings)
Lighthouse
Lodestar
Nimbus
Prysm
Teku
...
eth2diff
(replaying samples across implementations)
zcli
lcli
pcli
ncli
...
Monitors for crashes
Monitors for crashes and diffs
Monitors for crashes and diffs
Report Creation

Beacon Fuzz Architecture



Initial corpora
(e.g eth2-spec-test, testnet states)
Inputs
eth2fuzz
(coverage guided fuzzer to generate samples)
Generates
Fuzzing corpora
(samples triggering new code paths)
Inputs
beacon-fuzz-2
(diff fuzzing with FFI bindings)
Lighthouse
Lodestar
Nimbus
Prysm
Teku
...
eth2diff
(replaying samples across implementations)
zcli
lcli
pcli
ncli
...
Monitors for crashes
Monitors for crashes and diffs
Monitors for crashes and diffs
Report Creation

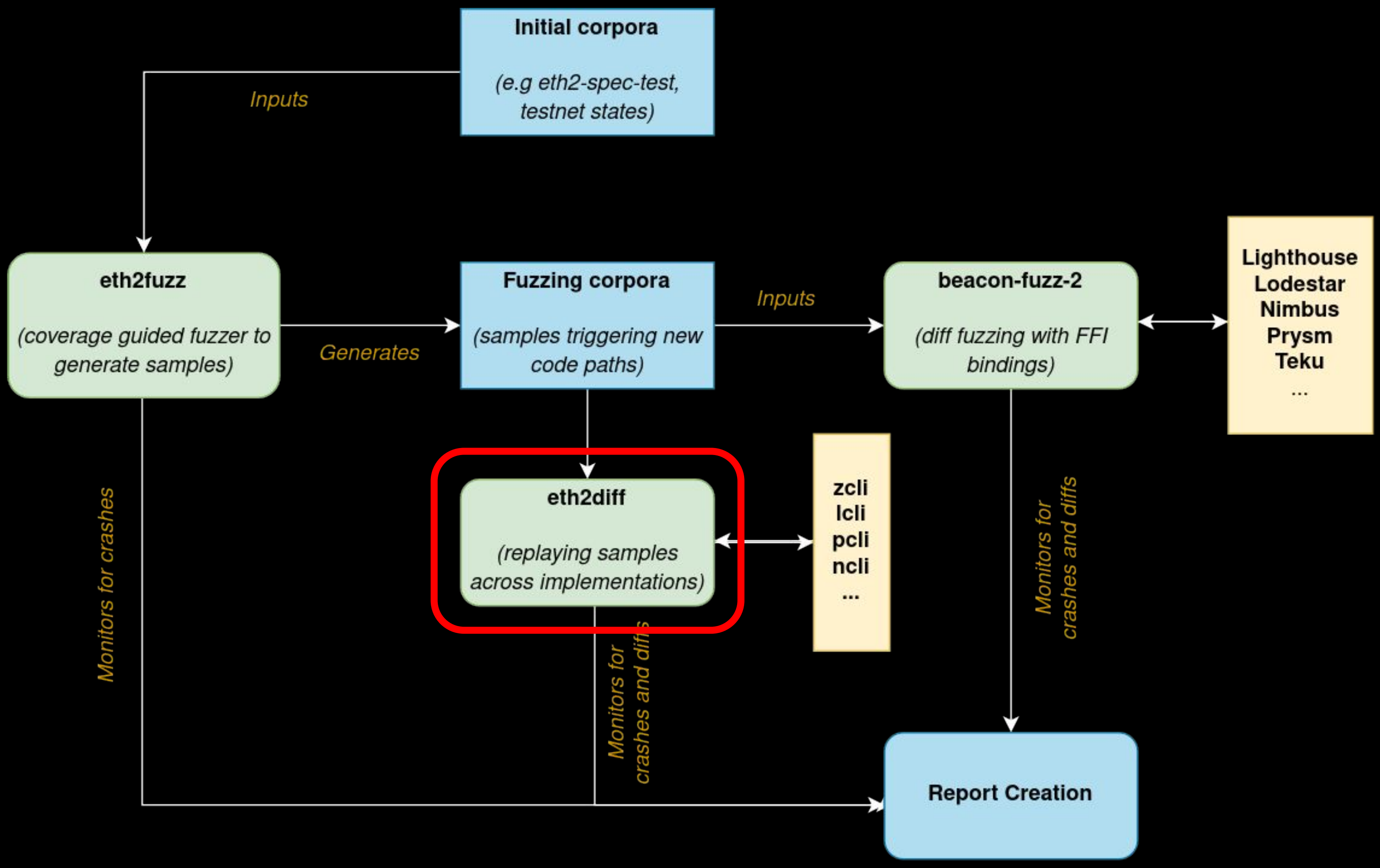
eth2fuzz

- **Standalone, coverage-guided fuzzing**
    - Targets each implementation separately
    - Supports multiple fuzzing engines
        - libFuzzer, AFL/AFL++, honggfuzz, go-fuzz, JQF, Jazzer
    - Fed valid corpora (beacon states and consensus objects)
- Extended to support structural fuzzing (where possible)
    - Move past the deserialisation to actually hit state transitions
    - Use the **arbitrary** trait in Rust

eth2fuzz

- **Types of bugs identified:**
  - Out of bound memory accesses (SSZ)
  - Panics when decoding non-UTF8 characters (ENR)
  - Integer overflows (epoch processing)
  - Memory exhaustion (SSZ)
  - Nil pointer dereference (SSZ)

Beacon Fuzz Architecture



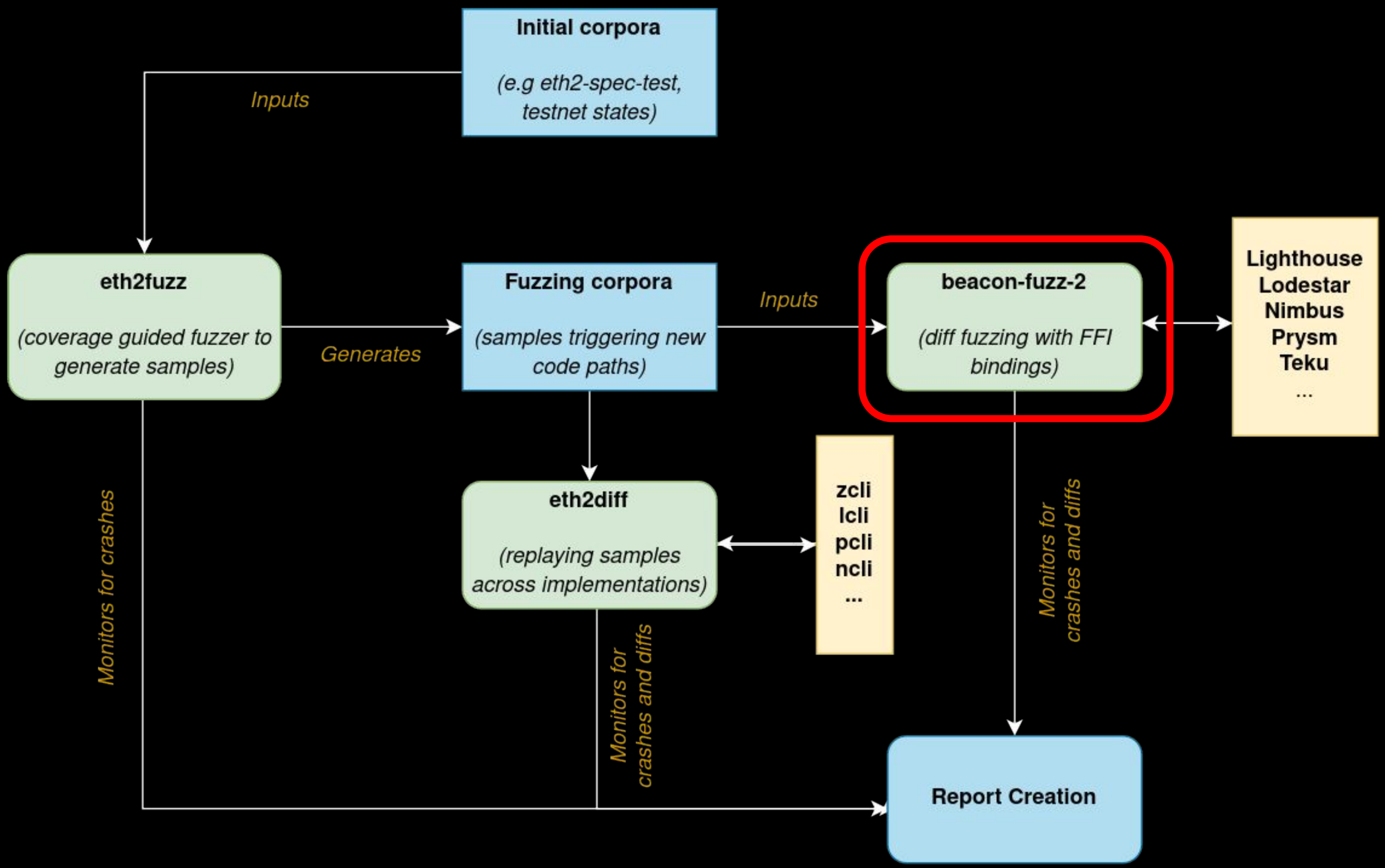
Initial corpora
(e.g eth2-spec-test, testnet states)
Inputs
eth2fuzz
(coverage guided fuzzer to generate samples)
Generates
Fuzzing corpora
(samples triggering new code paths)
Inputs
beacon-fuzz-2
(diff fuzzing with FFI bindings)
Lighthouse
Lodestar
Nimbus
Prysm
Teku
...
eth2diff
(replaying samples across implementations)
zcli
lcli
pcli
ncli
...
Monitors for crashes
Monitors for crashes and diffs
Monitors for crashes and diffs
Report Creation

eth2diff

- **Debugging tool to replay samples across all supported clients**
  - Wrapper around client utilities (thanks implementers <3)
  - Allows us to feed interesting fuzzing inputs to all clients

- **Type of bugs identified:**
  - Incorrect attestation validation
  - Incorrect signature validation
  - Lack of Merkle Proof validation

Beacon Fuzz Architecture

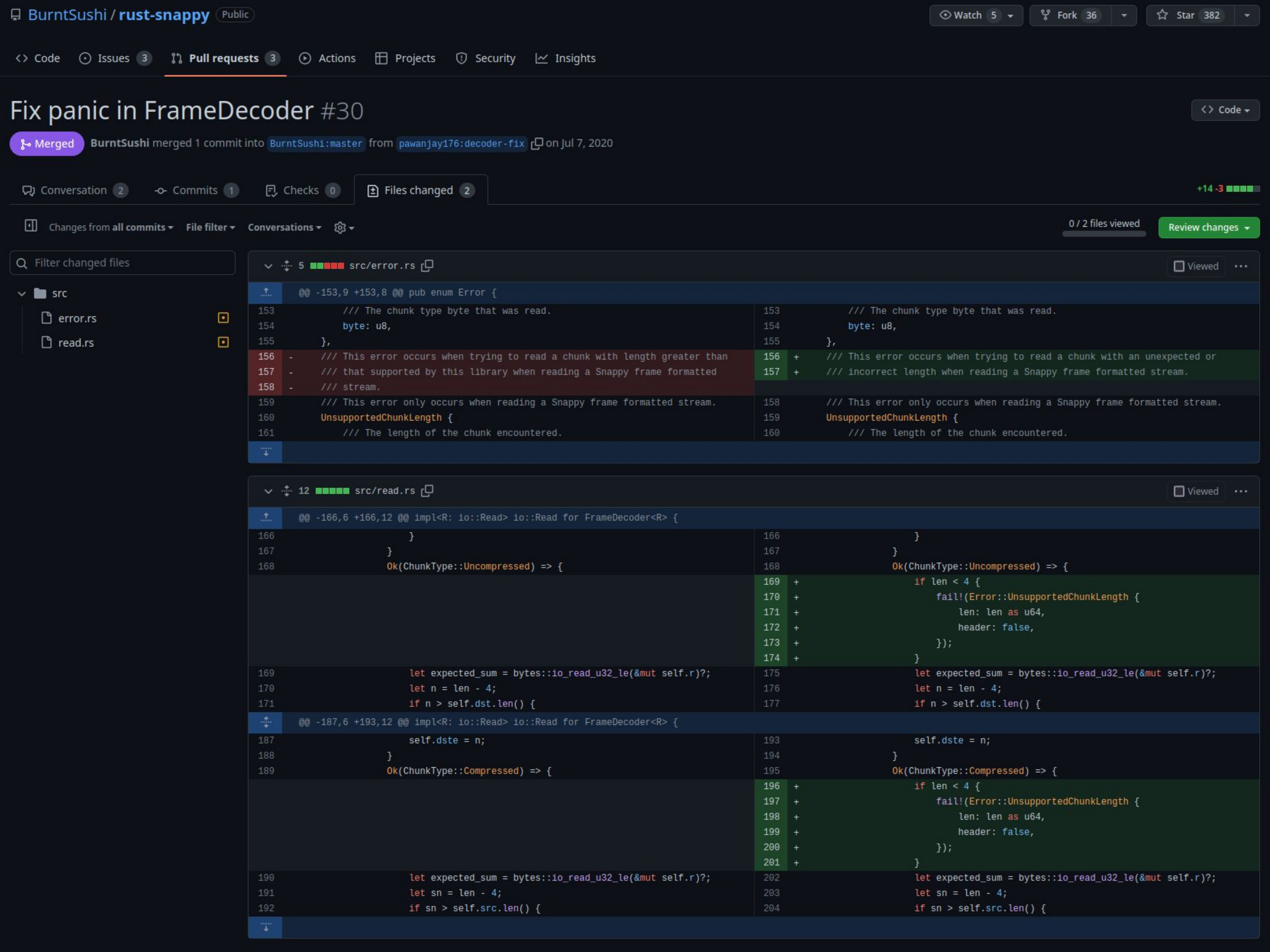


Initial corpora
(e.g eth2-spec-test, testnet states)
Inputs
eth2fuzz
(coverage guided fuzzer to generate samples)
Generates
Fuzzing corpora
(samples triggering new code paths)
Inputs
beacon-fuzz-2
(diff fuzzing with FFI bindings)
Lighthouse
Lodestar
Nimbus
Prysm
Teku
...
eth2diff
(replaying samples across implementations)
zcli
lcli
pcli
ncli
...
Monitors for crashes
Monitors for crashes and diffs
Monitors for crashes and diffs
Report Creation

beacon-fuzz-v2

- **Differential Fuzzing Engine:**
  - Foreign Function Interfaces (FFI) bindings
  - Structural fuzzing support with inputs generated by Lighthouse fuzzers
  - Most effective technique for catching consensus-splitting bugs

- **Type of bugs identified:**
  - Off-by-one errors (state transitions)
  - BLS signature malleability
  - Insufficient attestation validation



BurntSushi / **rust-snappy** Public

Watch 5 · Fork 36 · Star 382

Code · Issues 3 · **Pull requests 3** · Actions · Projects · Security · Insights

Fix panic in FrameDecoder #30

Merged — **BurntSushi** merged 1 commit into `BurntSushi:master` from `pawanjay176:decoder-fix` on Jul 7, 2020

Conversation 2 · Commits 1 · Checks 0 · **Files changed 2** · +14 -3

Changes from **all commits** · File filter · Conversations · 0 / 2 files viewed · Review changes

Filter changed files

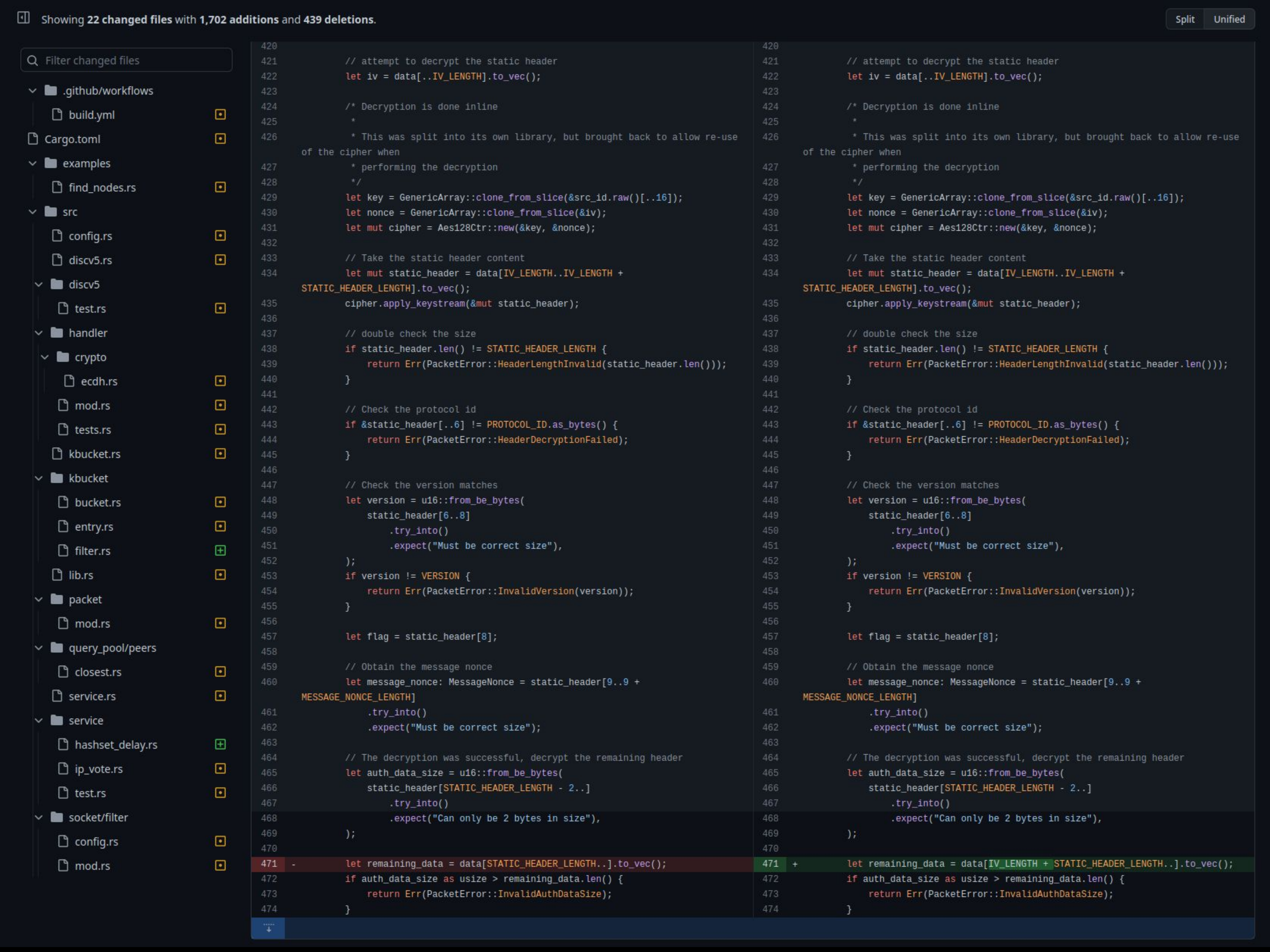
- src
  - error.rs
  - read.rs

5 src/error.rs — Viewed

```diff
@@ -153,9 +153,8 @@ pub enum Error {
         /// The chunk type byte that was read.
         byte: u8,
     },
-    /// This error occurs when trying to read a chunk with length greater than
-    /// that supported by this library when reading a Snappy frame formatted
-    /// stream.
+    /// This error occurs when trying to read a chunk with an unexpected or
+    /// incorrect length when reading a Snappy frame formatted stream.
     /// This error only occurs when reading a Snappy frame formatted stream.
     UnsupportedChunkLength {
         /// The length of the chunk encountered.
```

12 src/read.rs — Viewed

```diff
@@ -166,6 +166,12 @@ impl<R: io::Read> io::Read for FrameDecoder<R> {
                     }
                 }
                 Ok(ChunkType::Uncompressed) => {
+                    if len < 4 {
+                        fail!(Error::UnsupportedChunkLength {
+                            len: len as u64,
+                            header: false,
+                        });
+                    }
                     let expected_sum = bytes::io_read_u32_le(&mut self.r)?;
                     let n = len - 4;
                     if n > self.dst.len() {
@@ -187,6 +193,12 @@ impl<R: io::Read> io::Read for FrameDecoder<R> {
                     self.dste = n;
                 }
                 Ok(ChunkType::Compressed) => {
+                    if len < 4 {
+                        fail!(Error::UnsupportedChunkLength {
+                            len: len as u64,
+                            header: false,
+                        });
+                    }
                     let expected_sum = bytes::io_read_u32_le(&mut self.r)?;
                     let sn = len - 4;
                     if sn > self.src.len() {
```



Showing **22 changed files** with **1,702 additions** and **439 deletions**.

Split Unified

Filter changed files

- .github/workflows
  - build.yml
- Cargo.toml
- examples
  - find_nodes.rs
- src
  - config.rs
  - discv5.rs
  - discv5
    - test.rs
  - handler
    - crypto
      - ecdh.rs
    - mod.rs
    - tests.rs
  - kbucket.rs
  - kbucket
    - bucket.rs
    - entry.rs
    - filter.rs
  - lib.rs
  - packet
    - mod.rs
  - query_pool/peers
    - closest.rs
  - service.rs
  - service
    - hashset_delay.rs
    - ip_vote.rs
    - test.rs
  - socket/filter
    - config.rs
    - mod.rs

```

        // attempt to decrypt the static header
        let iv = data[..IV_LENGTH].to_vec();

        /* Decryption is done inline
         *
         * This was split into its own library, but brought back to allow re-use of the cipher when
         * performing the decryption
         */
        let key = GenericArray::clone_from_slice(&src_id.raw()[..16]);
        let nonce = GenericArray::clone_from_slice(&iv);
        let mut cipher = Aes128Ctr::new(&key, &nonce);

        // Take the static header content
        let mut static_header = data[IV_LENGTH..IV_LENGTH + STATIC_HEADER_LENGTH].to_vec();
        cipher.apply_keystream(&mut static_header);

        // double check the size
        if static_header.len() != STATIC_HEADER_LENGTH {
            return Err(PacketError::HeaderLengthInvalid(static_header.len()));
        }

        // Check the protocol id
        if &static_header[..6] != PROTOCOL_ID.as_bytes() {
            return Err(PacketError::HeaderDecryptionFailed);
        }

        // Check the version matches
        let version = u16::from_be_bytes(
            static_header[6..8]
                .try_into()
                .expect("Must be correct size"),
        );
        if version != VERSION {
            return Err(PacketError::InvalidVersion(version));
        }

        let flag = static_header[8];

        // Obtain the message nonce
        let message_nonce: MessageNonce = static_header[9..9 + MESSAGE_NONCE_LENGTH]
            .try_into()
            .expect("Must be correct size");

        // The decryption was successful, decrypt the remaining header
        let auth_data_size = u16::from_be_bytes(
            static_header[STATIC_HEADER_LENGTH - 2..]
                .try_into()
                .expect("Can only be 2 bytes in size"),
        );

-       let remaining_data = data[STATIC_HEADER_LENGTH..].to_vec();
        if auth_data_size as usize > remaining_data.len() {
            return Err(PacketError::InvalidAuthDataSize);
        }
```

```

        // attempt to decrypt the static header
        let iv = data[..IV_LENGTH].to_vec();

        /* Decryption is done inline
         *
         * This was split into its own library, but brought back to allow re-use of the cipher when
         * performing the decryption
         */
        let key = GenericArray::clone_from_slice(&src_id.raw()[..16]);
        let nonce = GenericArray::clone_from_slice(&iv);
        let mut cipher = Aes128Ctr::new(&key, &nonce);

        // Take the static header content
        let mut static_header = data[IV_LENGTH..IV_LENGTH + STATIC_HEADER_LENGTH].to_vec();
        cipher.apply_keystream(&mut static_header);

        // double check the size
        if static_header.len() != STATIC_HEADER_LENGTH {
            return Err(PacketError::HeaderLengthInvalid(static_header.len()));
        }

        // Check the protocol id
        if &static_header[..6] != PROTOCOL_ID.as_bytes() {
            return Err(PacketError::HeaderDecryptionFailed);
        }

        // Check the version matches
        let version = u16::from_be_bytes(
            static_header[6..8]
                .try_into()
                .expect("Must be correct size"),
        );
        if version != VERSION {
            return Err(PacketError::InvalidVersion(version));
        }

        let flag = static_header[8];

        // Obtain the message nonce
        let message_nonce: MessageNonce = static_header[9..9 + MESSAGE_NONCE_LENGTH]
            .try_into()
            .expect("Must be correct size");

        // The decryption was successful, decrypt the remaining header
        let auth_data_size = u16::from_be_bytes(
            static_header[STATIC_HEADER_LENGTH - 2..]
                .try_into()
                .expect("Can only be 2 bytes in size"),
        );

+       let remaining_data = data[IV_LENGTH + STATIC_HEADER_LENGTH..].to_vec();
        if auth_data_size as usize > remaining_data.len() {
            return Err(PacketError::InvalidAuthDataSize);
        }
```

Questions?