

Snapshot Fuzzing for the Lightning Network

ek · bitcoin++ Berlin 2026

Topics

- About Me
- Fuzzing?
- Challenges
- Advisories
- Plans



About Me

abusix

mail intelligence

2020-2023

lightning app dev

2023-2025

chaincode

BOSS program

Jan-Mar 2026



About Me

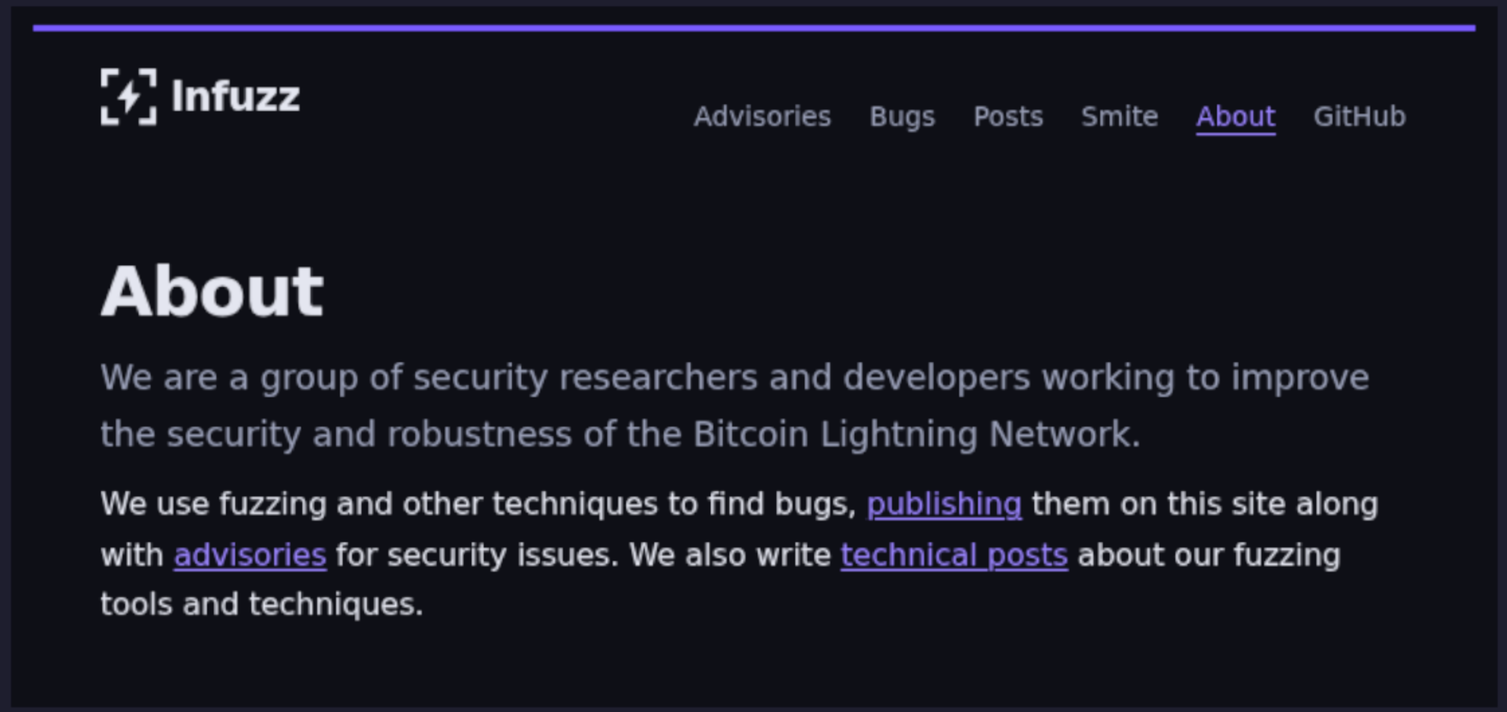
Now:



Fuzzing the Lightning Network.

https://lnfuzz.org/

39 bugs found | 4 advisories | 4 implementations





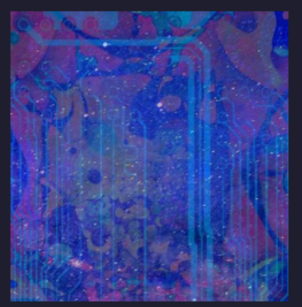
morehouse



NishantBansal2003



erickcestari



ekzyis

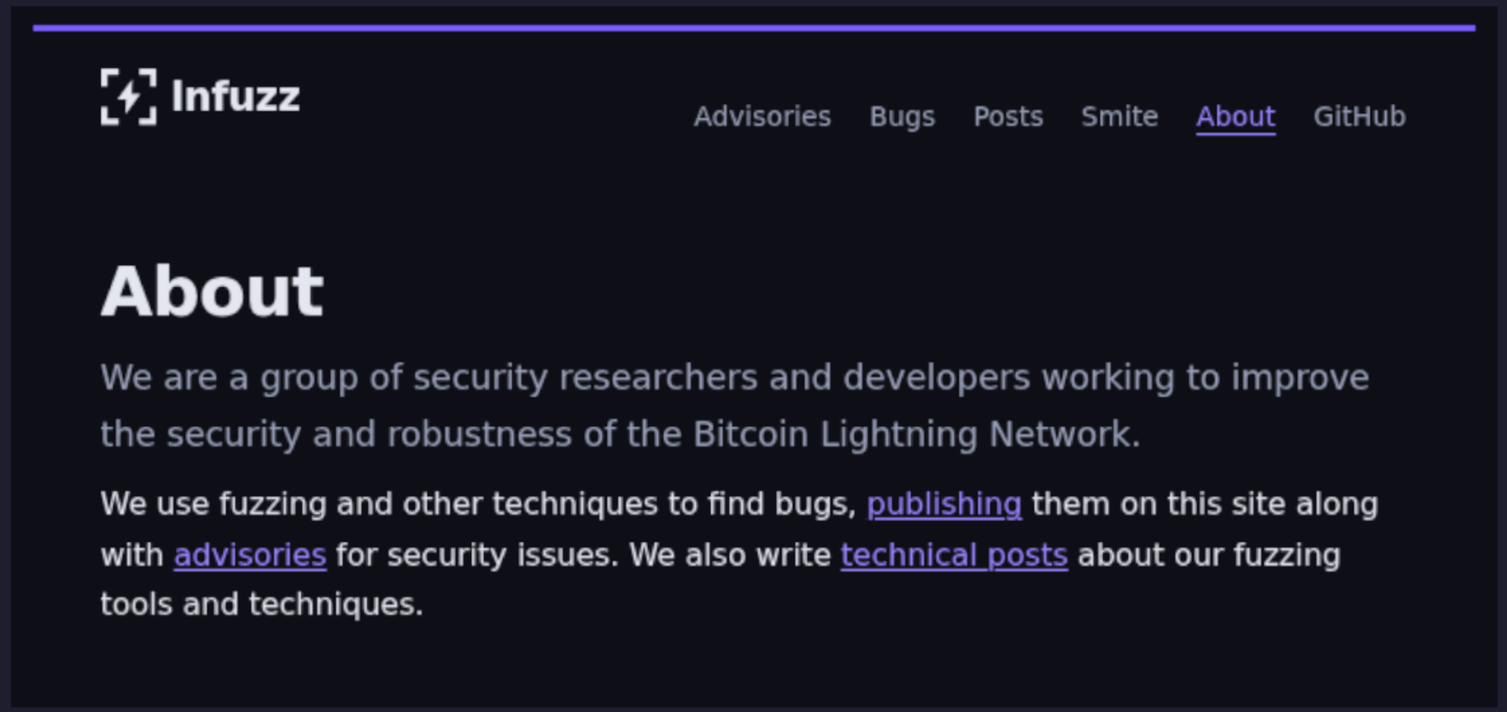


devvaansh



Ashish-Kumar-Dash

... and many more as vagabonds





morehouse



NishantBansal2003



erickcestari



ek



devvaansh



Ashish-Kumar-Dash

... and many more as vagabonds

Fuzzing?

- automated software testing technique
- explore input space with random data
- different types:
  - differential
  - coverage-guided
  - protocol-aware
  - snapshot
  - directed
  - ...

Fuzzing?

- automated software testing technique
- explore input space with random data
- different types:
  - differential
  - coverage-guided
  - protocol-aware
  - snapshot
  - directed
  - ...

Anyway~

Let's fuzz something!

Fuzzing!

```go
func NewOnionPacket(
    sessionKey *secp256k1.PrivateKey,
    associatedData []byte,
    hops []Hop,
) (*OnionPacket, error) {

    // ECDH with each hop to generate shared secrets
    // ...

    // build the onion with padding, hmac, encryption etc.
    // (and try not to cry)
    // ...

    packet := &OnionPacket{
        Version:      0x00,
        EphemeralKey: sessionKey.PubKey(),
        HopPayloads:  [1300]byte(hopPayload),
        HMAC:         hmac,
    }
    return packet, nil
}
```

Fuzzing!

```go
func FuzzBolt04_NewOnionPacket(f *testing.F) {
    sessionKey := hexBytes("4041424344454647484a4b4c4d4e4f404142434445464748494a4b4c4d4e4f")
    assocData := hexBytes("0102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20")
    pubKey := hexBytes("02eec7245d6b7d2ccb30380bfbe2a3648cd7a942653f5aa340edcea1f283686619")
    f.Add(sessionKey, assocData, pubKey, bytes.Repeat(hexBytes("08abcdef0123456789"), 4))

    f.Fuzz(func(t *testing.T, sessionKeyBytes, assocData, pubKeyBytes, data []byte) {
        // validate keys, construct hops from data to increase fuzz coverage
        // ...

        packet, err := NewOnionPacket(sessionKey, assocData, hops)

        // decide if output makes sense:
        //   * packet serialization matches input?
        //   * error expected?
        // ...
    })
}
```

Fuzzing!

```
go test -fuzz=FuzzBolt04 ./lightning/bolt04
```

```
──────────────────────────────────── [finished with error] ────────────────────────────────────
```

```
fuzz: elapsed: 0s, gathering baseline coverage: 0/2 completed
failure while testing seed corpus entry: FuzzBolt04_NewOnionPacket/seed#1
fuzz: elapsed: 0s, gathering baseline coverage: 0/2 completed
--- FAIL: FuzzBolt04_NewOnionPacket (0.04s)
    --- FAIL: FuzzBolt04_NewOnionPacket (0.02s)
        bolt04_fuzz_test.go:39: panic recovered: runtime error: slice bounds out of
range [1335:1300]

FAIL
exit status 1
FAIL    github.com/ekzyis/lnpilot/lightning/bolt04    0.056s
```

Fuzzing!

```
  func NewOnionPacket(
    associatedData []byte,
    hops []Hop,
  ) (*OnionPacket, error) {
+   var total int
+   for _, hop := range hops {
+     total += hop.OnionPayloadLength()
+     if total > OnionPacketPayloadSize {
+       return fmt.Errorf(
+         "total onion payload length %d exceeds packet size %d",
+         total, OnionPacketPayloadSize,
+       )
+     }
+   }

    // ECDH with each hop to generate shared secrets
```

Fuzzing!

```
go test -fuzz=FuzzBolt04 -fuzztime=5s ./lightning/bolt04
```

```
──────────────────────────────────── [finished] ────────────────────────────────────

fuzz: elapsed: 0s, gathering baseline coverage: 0/2 completed
fuzz: elapsed: 0s, gathering baseline coverage: 2/2 completed, now fuzzing with 8
workers
fuzz: elapsed: 3s, execs: 62199 (20733/sec), new interesting: 13 (total: 15)
fuzz: elapsed: 6s, execs: 89821 (9207/sec), new interesting: 26 (total: 28)
fuzz: elapsed: 6s, execs: 89821 (0/sec), new interesting: 26 (total: 28)
PASS
ok      github.com/ekzyis/lnpilot/lightning/bolt04    6.023s
```

Fuzzing!

```
$ go test -run FuzzBolt04 -coverprofile cover.out ./lightning/bolt04
```

```
$ go tool cover -func cover.out
lightning/bolt04/bolt04.go:44:        Serialize               100.0%
lightning/bolt04/bolt04.go:52:        OnionPayloadLength      100.0%
lightning/bolt04/bolt04.go:56:        validateHops            100.0%
lightning/bolt04/bolt04.go:67:        NewOnionPacket          100.0%
lightning/bolt04/bolt04.go:156:       generateFiller          100.0%
lightning/bolt04/bolt04.go:177:       _chacha20                87.5%
lightning/bolt04/bolt04.go:191:       deriveKey               100.0%
lightning/bolt04/bolt04.go:198:       shiftRight              100.0%
lightning/bolt04/bolt04.go:202:       _hmac                   100.0%
lightning/bolt04/bolt04.go:209:       xor                      87.5%
lightning/bolt04/bolt04.go:223:       computeBlindingFactor   100.0%
total:                                (statements)             97.7%
```

Fuzzing!

```
$ go test -run FuzzBolt04 -coverprofile cover.out ./lightning/bolt04
          ^^^^^^^^^^^^^^^
          coverage determined by seeds, not mutations during fuzzing!
$ go tool cover -func cover.out

lightning/bolt04/bolt04.go:44:        Serialize               100.0%
lightning/bolt04/bolt04.go:52:        OnionPayloadLength      100.0%
lightning/bolt04/bolt04.go:56:        validateHops            100.0%
lightning/bolt04/bolt04.go:67:        NewOnionPacket          100.0%
lightning/bolt04/bolt04.go:156:       generateFiller          100.0%
lightning/bolt04/bolt04.go:177:       _chacha20                87.5%
lightning/bolt04/bolt04.go:191:       deriveKey               100.0%
lightning/bolt04/bolt04.go:198:       shiftRight              100.0%
lightning/bolt04/bolt04.go:202:       _hmac                   100.0%
lightning/bolt04/bolt04.go:209:       xor                      87.5%
lightning/bolt04/bolt04.go:223:       computeBlindingFactor   100.0%
total:                                (statements)             97.7%
```

Caveat

Coverage does not tell you if the fuzzer can reach these statements *with mutations*

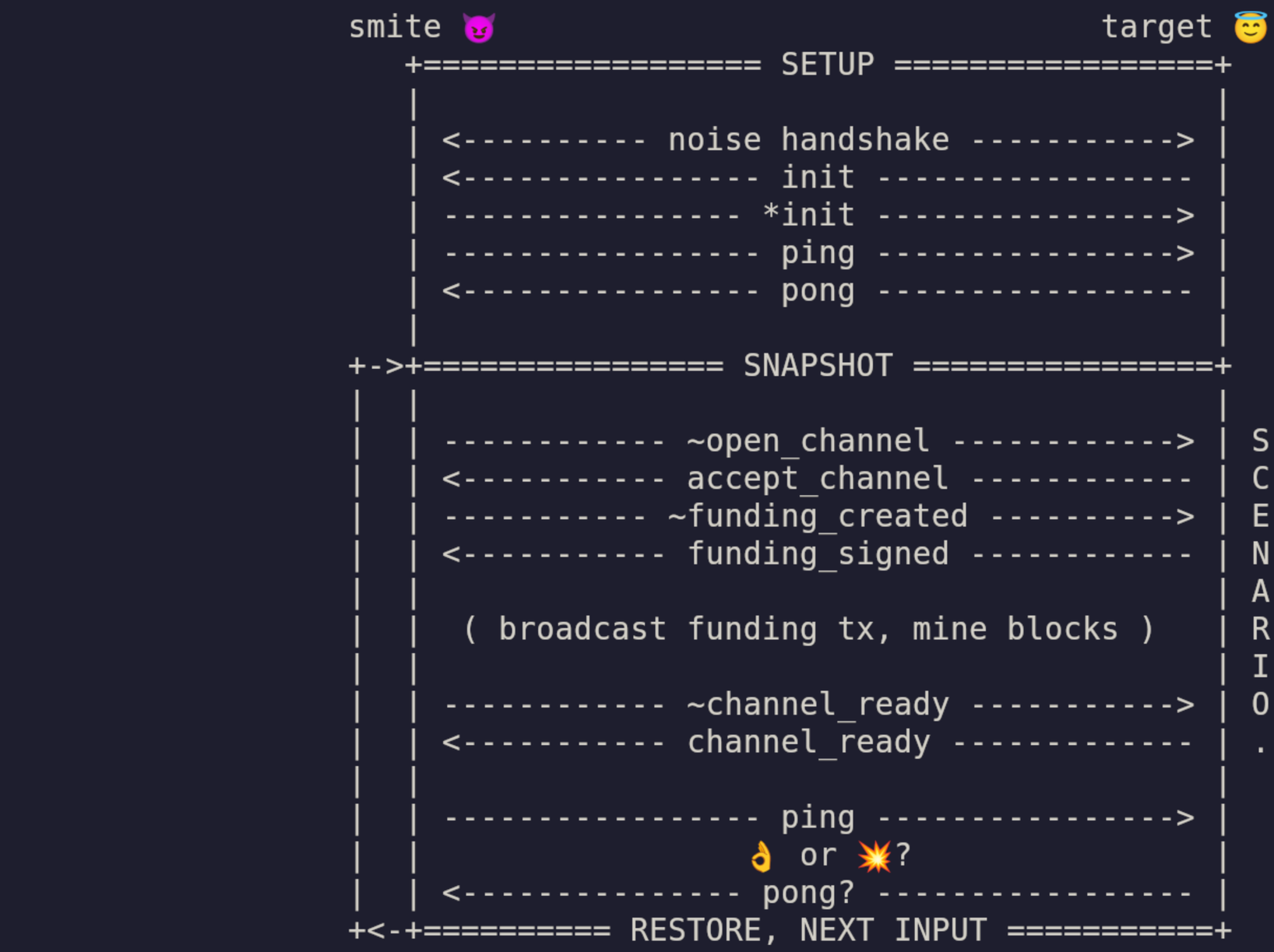
Why *snapshot* fuzzing?

Snapshot fuzzing?

> One obvious challenge that needs to be addressed is **non-determinism**.
>
> Fuzzamoto uses full-system snapshot fuzzing to take a **snapshot of all its state** (memory, CPU state, devices, ...)

– Niklas Gögge, *Fuzzamoto: Introduction*, https://brink.dev/blog/2026/01/07/fuzzamoto-introduction/

Snapshot fuzzing!



* echoes their features, minus gossip_queries, gossip_queries_ex, option_dual_fund and option_provide_storage to avoid noise in snapshot

~ built from the fuzz input

Snapshot fuzzing!

smite: mutators

tweak

```
push_msat             1_000_000 -> u64::MAX
dust_limit_satoshis         546 -> 16_777_216
to_self_delay               144 -> u16::MAX
feerate_per_kw            2_500 -> 0
```

swap

```
open_channel
  push_msat -> funding_satoshis
```

reorder

```
broadcast tx --\ /-- mine blocks
mine blocks  --/ \-- broadcast tx
```

Snapshot fuzzing!

smite: mutators

drop

```
      funding_signed
 -    ( broadcast tx )
      ( mine blocks )
      channel_ready
```

insert

```
       A  open_channel
      A  accept_channel
 +    B  node_announcement
 +    B  channel_update
      A  funding_created
```

Snapshot fuzzing!

smite: mutators

splice *(planned)*

```
      A  open_channel
      A  accept_channel
                              node_announcement  B
                                 channel_update  B
      ------------------------------------------------
      A  open_channel
                              node_announcement  B
      A  accept_channel
                                 channel_update  B
```

Snapshot fuzzing!

smite: generators

```
FundingFlowGenerator:

     >  open_channel
     <  accept_channel
     >  funding_created
     <  funding_signed
        ( mine tx )
     <> channel_ready

NodeAnnouncementGenerator:

     >  node_announcement

ChannelAnnouncementGenerator:

        ( mine 2-of-2 P2WSH funding tx )
     >  channel_announcement
```

Challenges

x86_64 != x86_64

smite-nyx-sys Compilation Errors on ARM64 macOS #47

Closed as not planned

kartikangiras opened on Apr 13

Summary

The `smite-nyx-sys` crate fails to compile on ARM64 Apple Darwin (aarch64-apple-darwin) due to:

1. Missing `kAFL_hypercall` function declarations in C code
2. Invalid inline assembly constraints incompatible with ARM64 architecture

morehouse on May 6 — Collaborator

x86_64 is required for Nyx, so we don't want to even try supporting ARM64.

morehouse closed this as not planned on May 6

Challenges

x86_64 != x86_64

However, not all x86_64 are created equal:

```
 INFO  [bitcoind] Waiting for bitcoind to be ready...
 INFO  [bitcoind] bitcoind is ready
 INFO  [ldk] Starting ldk-node-wrapper...
 ...
 DEBUG [process] ldk-node-wrapper: exited with signal: 4 (SIGILL)
 DEBUG [process] bitcoind: exited with exit status: 0
```

SIGILL: Invalid instruction

Challenges

x86_64 != x86_64

Issue:

Compiling on newer CPUs may emit instructions that the QEMU-Nyx guest does not support

```
Architecture:                         x86_64
CPU op-mode(s):                       32-bit, 64-bit
Byte Order:                           Little Endian
CPU(s):                               8
Vendor ID:                            GenuineIntel
Model name:                           11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz
Flags:
  3dnowprefetch abm acpi adx aes aperfmperf apic arat arch_capabilities arch_perfmon art avx avx2 avx512_bitalg avx512bw avx512cd avx512dq avx512f avx512ifma avx512vbmi
avx512_vbmi2 avx512vl avx512_vnni avx512_vp2intersect avx512_vpopcntdq bmi1 bmi2 bts cat_l2 cdp_l2 clflush clflushopt clwb cmov constant_tsc cpuid cpuid_fault cx16 cx8 de
ds_cpl dtes64 dtherm dts epb ept ept_ad erms est f16c flexpriority flush_l1d fma fpu fsgsbase fsrm fxsr gfni ht hwp hwp_act_window hwp_epp hwp_notify hwp_pkg_req ibpb ibrs
ibrs_enhanced ibt ida intel_pt invpcid lahf_lm lm mca mce md_clear mmx monitor movbe movdir64b movdiri msr mtrr nonstop_tsc nopl nx ospke pae pat pbe pcid pclmulqdq pdcm
pdpe1gb pebs pge pku pln pni popcnt pse pse36 pts rdpid rdrand rdseed rdt_a rdtscp rep_good sdbg sep sha_ni smap smep split_lock_detect ss ssbd sse sse2 sse4_1 sse4_2 ssse3
stibp syscall tm tm2 tpr_shadow tsc tsc_adjust tsc_deadline_timer tsc_known_freq umip user_shstk vaes vme vmx vnmi vpclmulqdq vpid x2apic xgetbv1 xsave xsavec xsaveopt xsaves
xtopology xtpr
```

💥 with avx512* flags

vs.

✅ without avx512* flags

```
Architecture:                         x86_64
CPU op-mode(s):                       32-bit, 64-bit
Byte Order:                           Little Endian
CPU(s):                               20
Vendor ID:                            GenuineIntel
Model name:                           13th Gen Intel(R) Core(TM) i5-13600K
Flags:
  3dnowprefetch abm acpi adx aes aperfmperf apic arat arch_capabilities arch_lbr arch_perfmon art avx avx2 avx_vnni bmi1 bmi2 bts clflush clflushopt clwb cmov constant_tsc
cpuid cpuid_fault cx16 cx8 de ds_cpl dtes64 dtherm dts epb ept ept_ad erms est f16c flexpriority flush_l1d fma fpu fsgsbase fsrm fxsr gfni hfi ht hwp hwp_act_window hwp_epp
hwp_notify hwp_pkg_req ibpb ibrs ibrs_enhanced ibt ida intel_pt invpcid lahf_lm lm mca mce md_clear mmx monitor movbe movdir64b movdiri msr mtrr nonstop_tsc nopl nx ospke pae
pat pbe pcid pclmulqdq pconfig pdcm pdpe1gb pebs pge pku pln pni popcnt pse pse36 pts rdpid rdrand rdseed rdtscp rep_good sdbg sep serialize sha_ni smap smep smx
split_lock_detect ss ssbd sse sse2 sse4_1 sse4_2 ssse3 stibp syscall tm tm2 tpr_shadow tsc tsc_adjust tsc_deadline_timer tsc_known_freq umip user_shstk vaes vme vmx vnmi
vpclmulqdq vpid waitpkg x2apic xgetbv1 xsave xsavec xsaveopt xsaves xtopology xtpr
```

Challenges

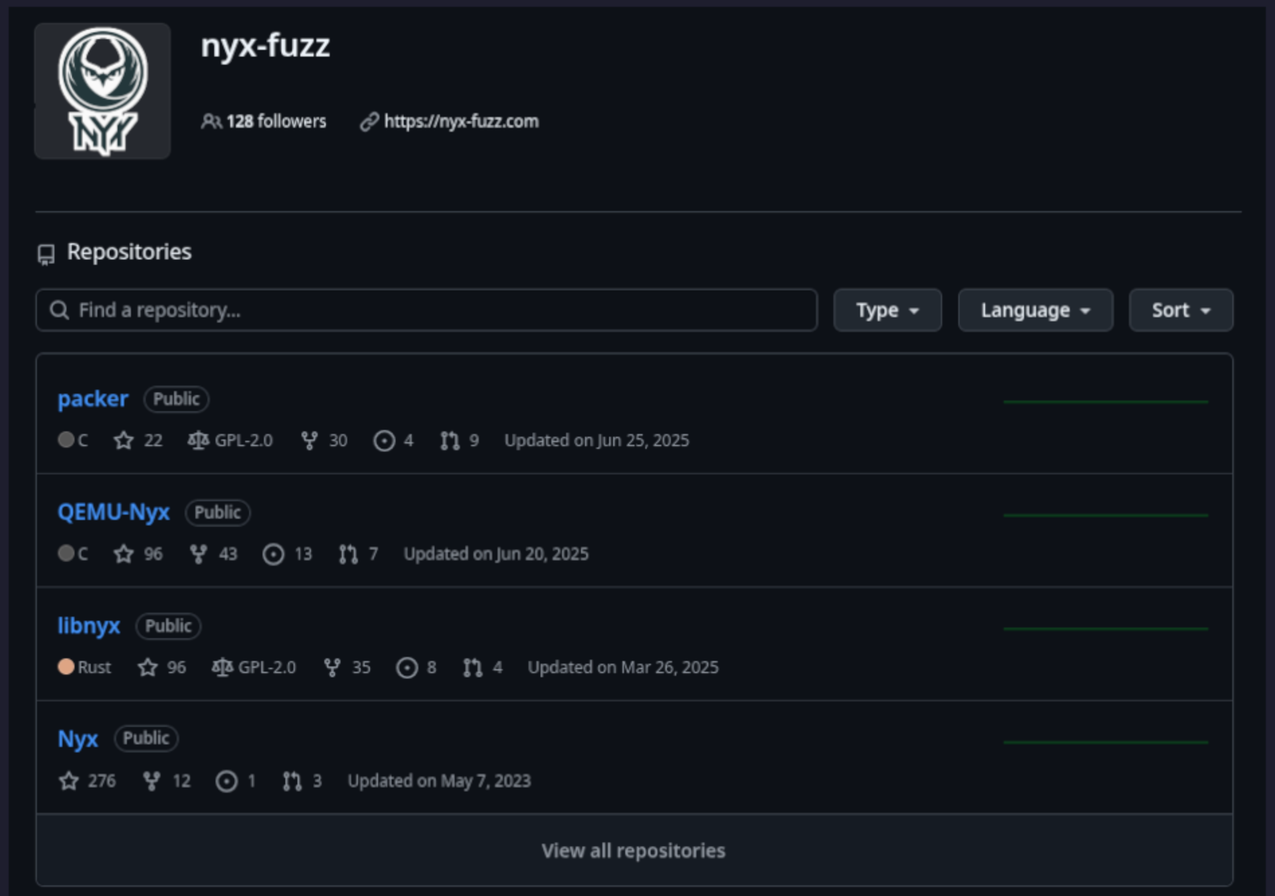
x86_64 != x86_64

The fix:

```
   COPY workloads/ldk/Cargo.toml workloads/ldk/Cargo.lock ./
   COPY workloads/ldk/src/ src/
   ENV AFL_NO_CFG_FUZZING=1
 + # Capping codegen to x86-64-v3 matches the VM's ceiling and keeps the build
 + # reproducible across all host CPUs.
 + ENV RUSTFLAGS="-C target-cpu=x86-64-v3"
   RUN cargo afl build --release
```

Challenges

unmaintained dependencies



Challenges

target quirks

- slow chain sync: poll vs zmq
- architecture:
  - JIT compiler
  - how to compile with instrumentation etc.

Advisories

Eclair: feature bits parsing DoS

ID: LNF-2026-0001 | severity: medium

reported: Mar 5, 2026 | fixed: Mar 17, 2026 | disclosed: Sep 24, 2026

affected: v0.13.1 and earlier | patched: v0.14.0



🐌

6 passes over every bit
-> ~3 objects per bit
=> ~300MB, 300ms / msg

- `node_announcement` and `channel_announcement` carry feature vectors too

Advisories

Eclair: zlib decompression bomb during decoding of channel queries

ID: LNF-2026-0002 | severity: medium

reported: Mar 6, 2026 | fixed: Mar 11, 2026 | disclosed: Sep 24, 2026

affected: v0.13.1 and earlier | patched: v0.14.0

---

https://github.com/lightning/bolts/pull/981 (2022):

```
   Encoding types:
   0: uncompressed array of `short_channel_id` types, in ascending order.
 - 1: compressed array of `short_channel_id` types, in ascending order, with zlib
 + 1: Previously used for zlib compression, this encoding MUST NOT be used.
```

Advisories

Eclair: zlib decompression bomb during decoding of channel queries



- `reply_channel_range` can also be used as a carrier

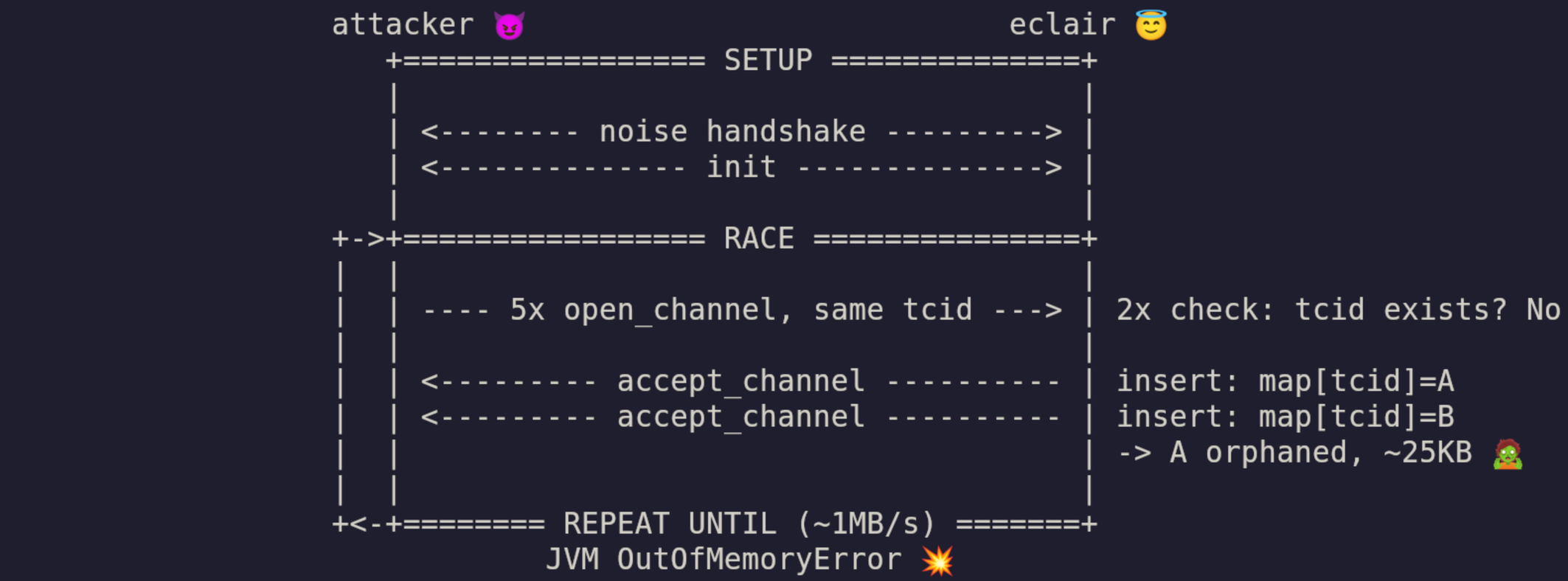
Advisories

Eclair: orphaned channel actors via TOCTOU race

ID: LNF-2026-0003 | severity: low

reported: July 15, 2026 | fixed: July 17, 2026 | disclosed: Sep 30, 2026

affected: v0.14.0 and earlier | patched: v0.14.1



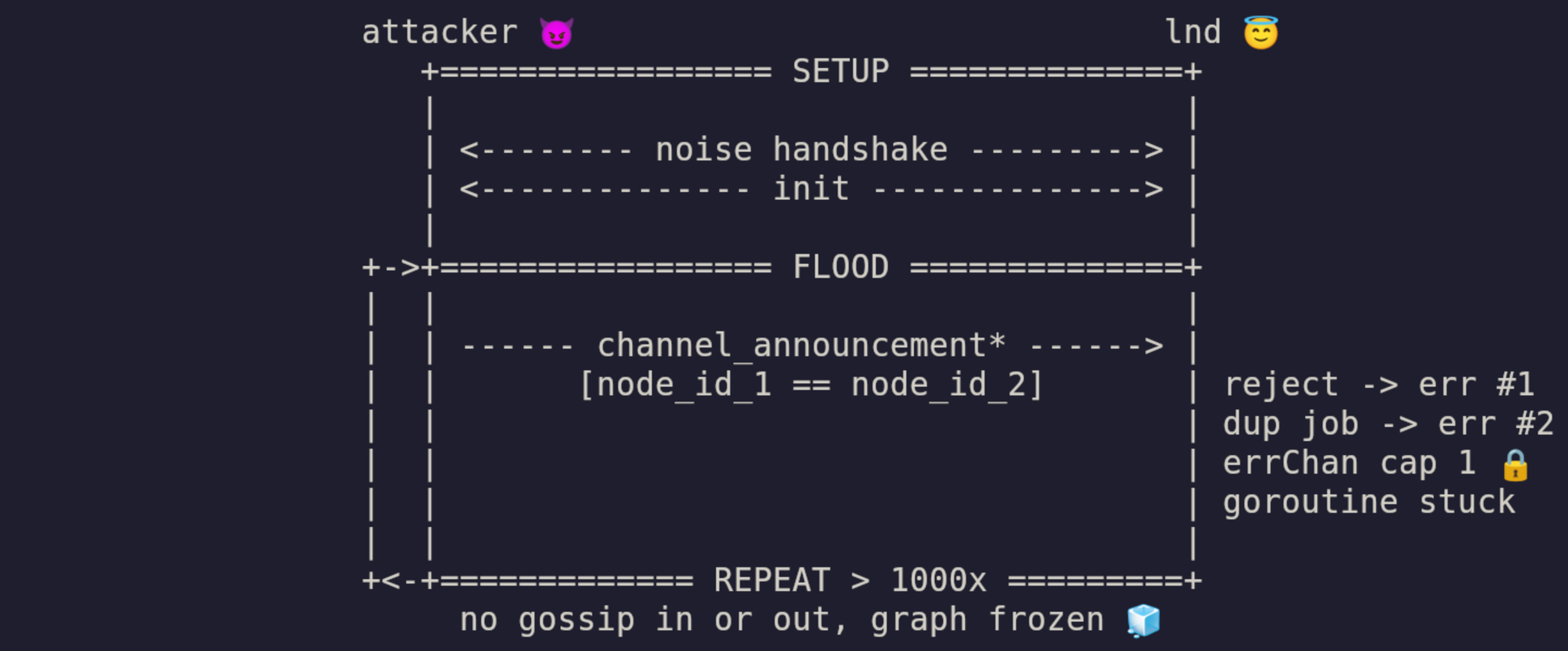
Advisories

LND: Gossip Stall via Malformed channel_announcement Flood

severity: T3 (low)

reported: Feb 2, 2026 | fixed: Feb 12, 2026 | disclosed: Sep 21, 2026

affected: 0.19.0-beta–0.20.0-beta | patched: 0.20.1-beta



- `channel_announcement` with two validation errors

Advisories

LND: Gossip Stall via Malformed channel_announcement Flood

The fix:

```
 - errChan := make(chan error, 1)
 + // Buffer up to two messages on errChan since up to two messages may be
 + // written and not all callers of this function actually read from
 + // errChan. Without this buffer goroutines end up blocking on writes to
 + // errChan, which prevents the gossiper from shutting down cleanly.
 + //
 + // TODO(ziggie): Redesign this once the actor model pattern becomes
 + // available. See https://github.com/lightningnetwork/lnd/pull/9820.
 + errChan := make(chan error, 2)
```

Plans

- experiment with Bedrock from Niklas Gögge of Nyx
- reproduce past vulnerabilities
- more scenarios, including initiated by the target
- my focus: channel close flow

The End

THANK YOU!!

Questions??