Testing TLS

Hubert Kario
Quality Engineer
24-10-2015

2014

redhat.

Heartbleed

OpenSSL CCS bug

gotofail

Certificate handling

CVE-2014-6321 in schannel a.k.a. Winshock

POODLE

2015

redhat.

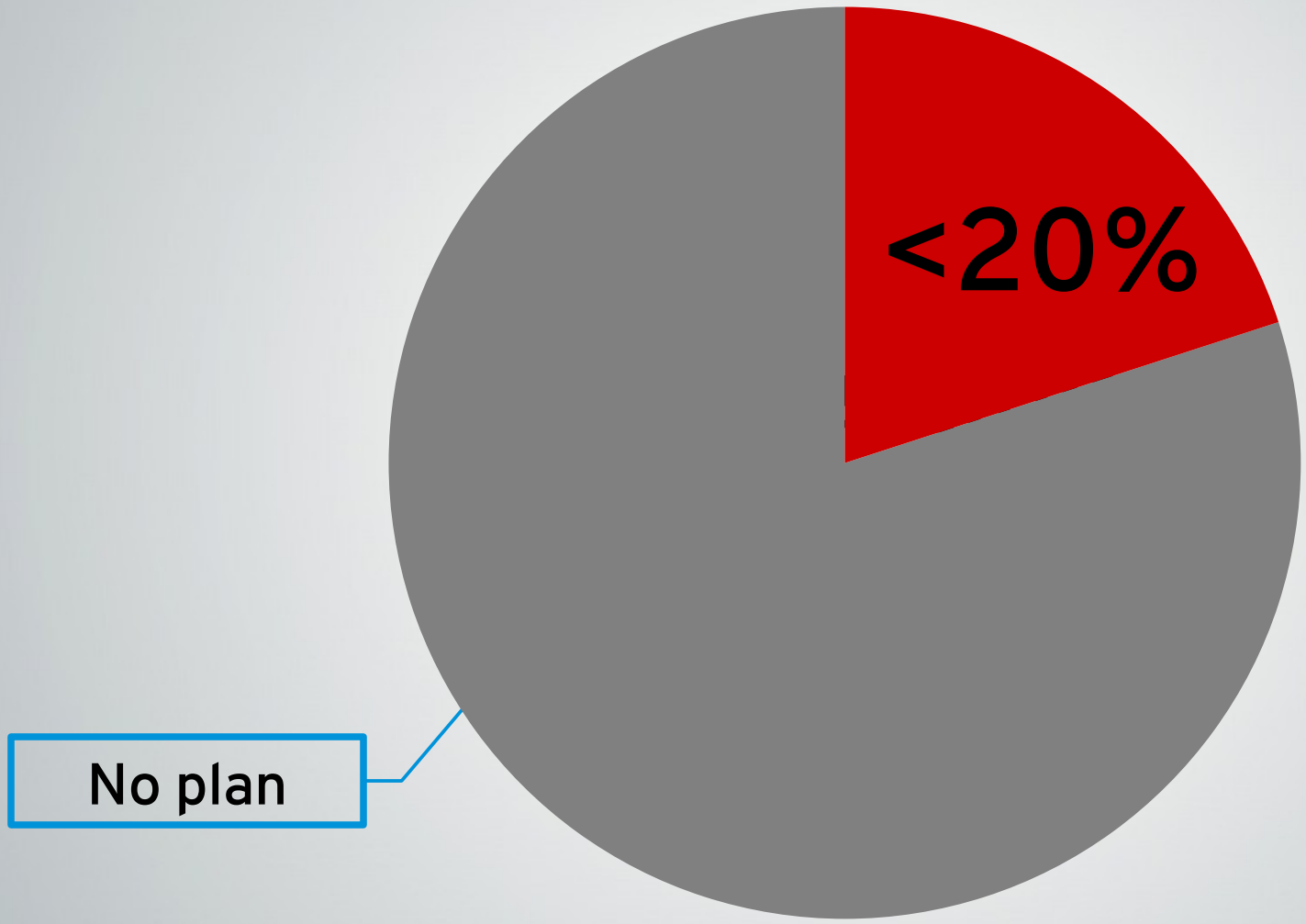
FREAK

LOGJAM

State of testing

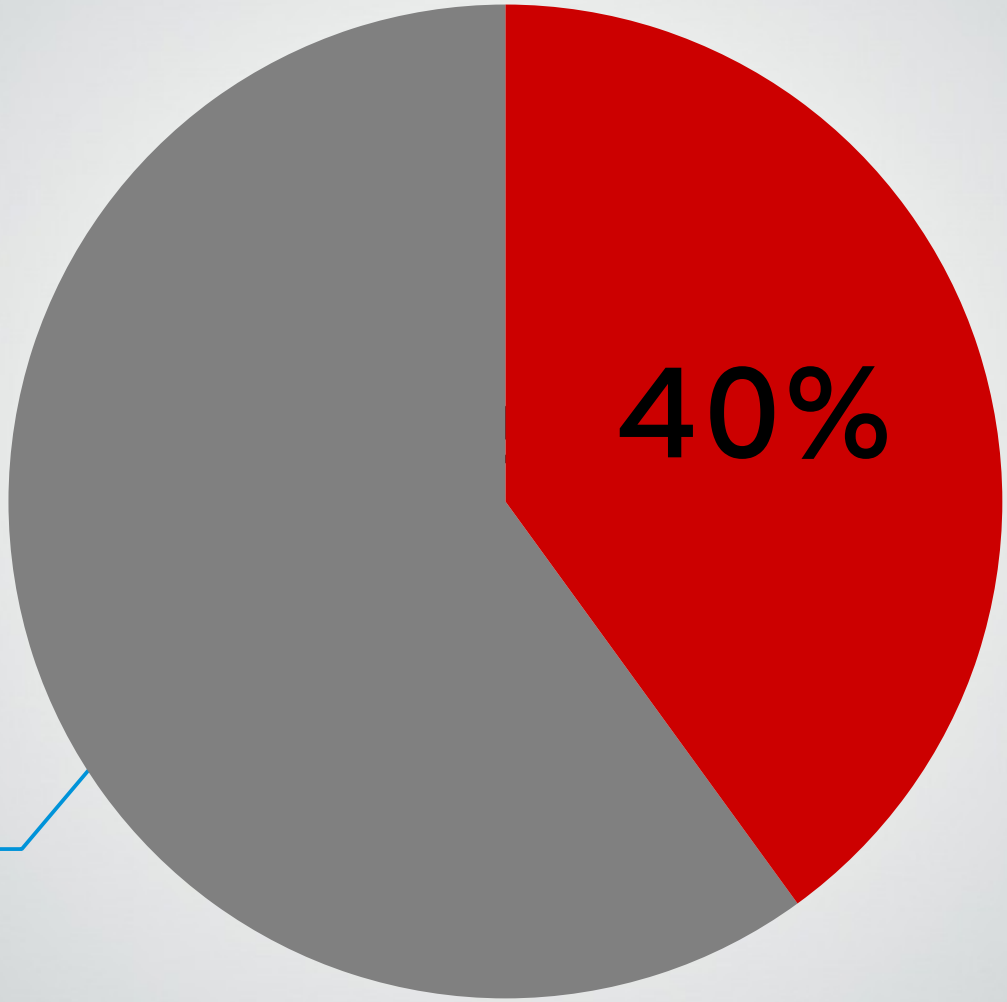
redhat.

OSS projects w/test plans



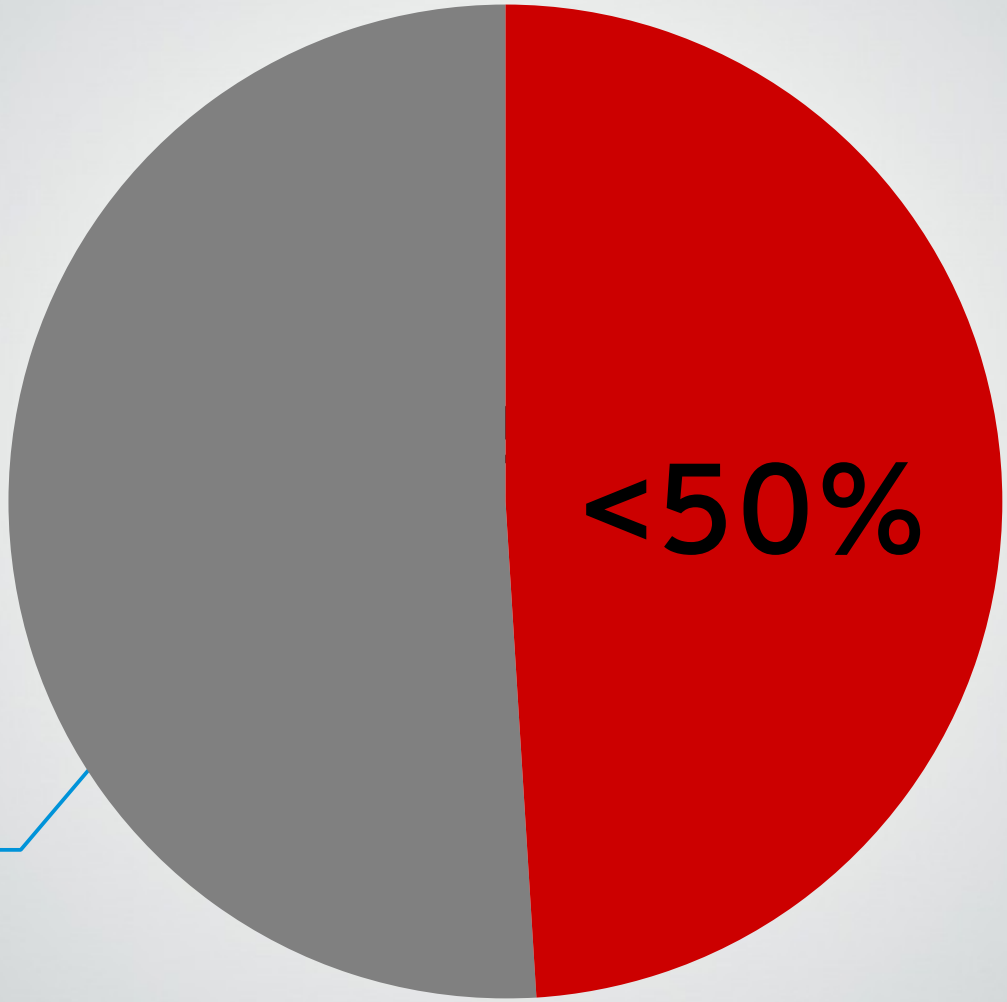
Source: Farooq & Quadri, 2011

OSS projects w/test tools



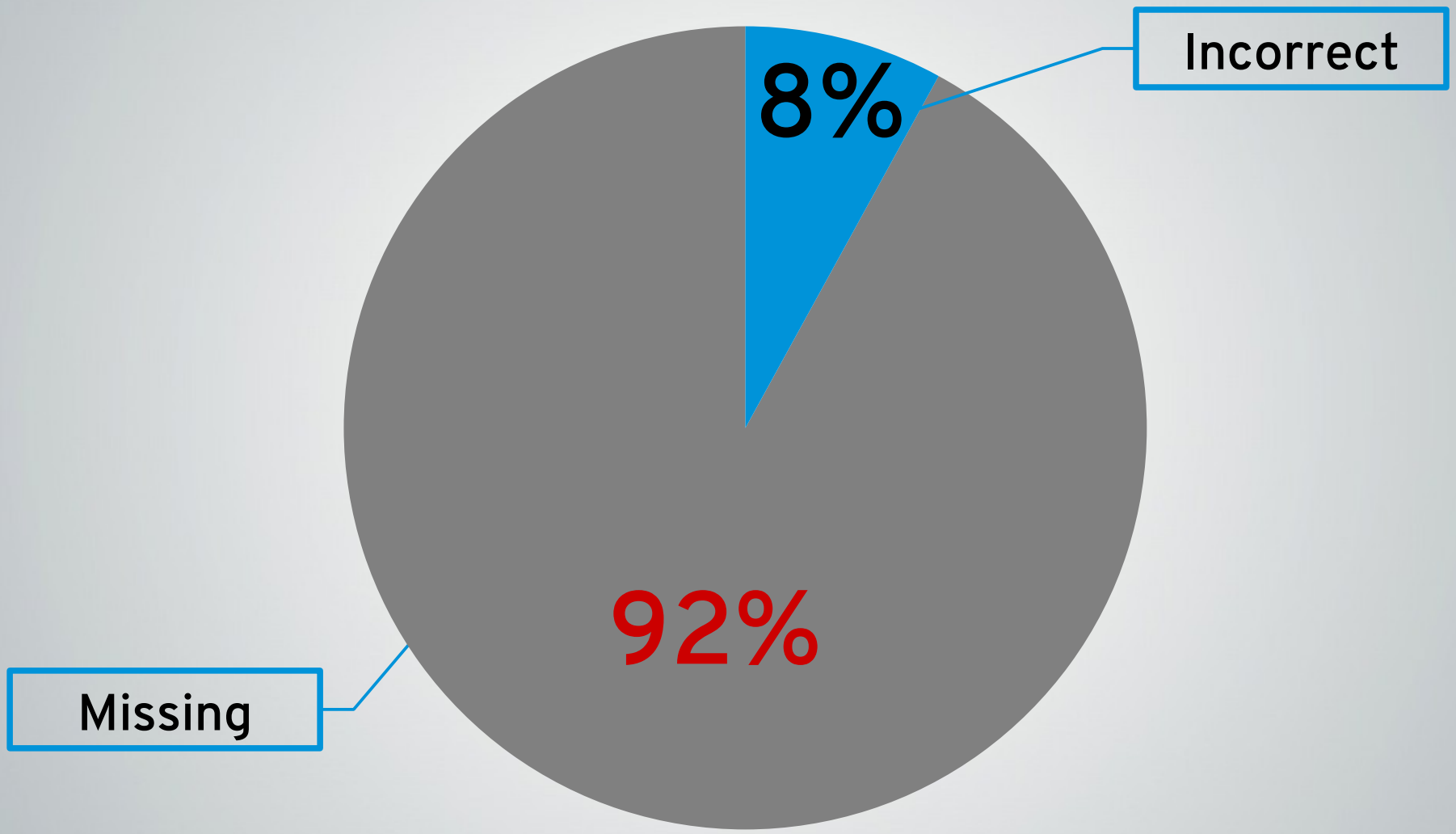
Source: Farooq & Quadri, 2011

Code coverage tools



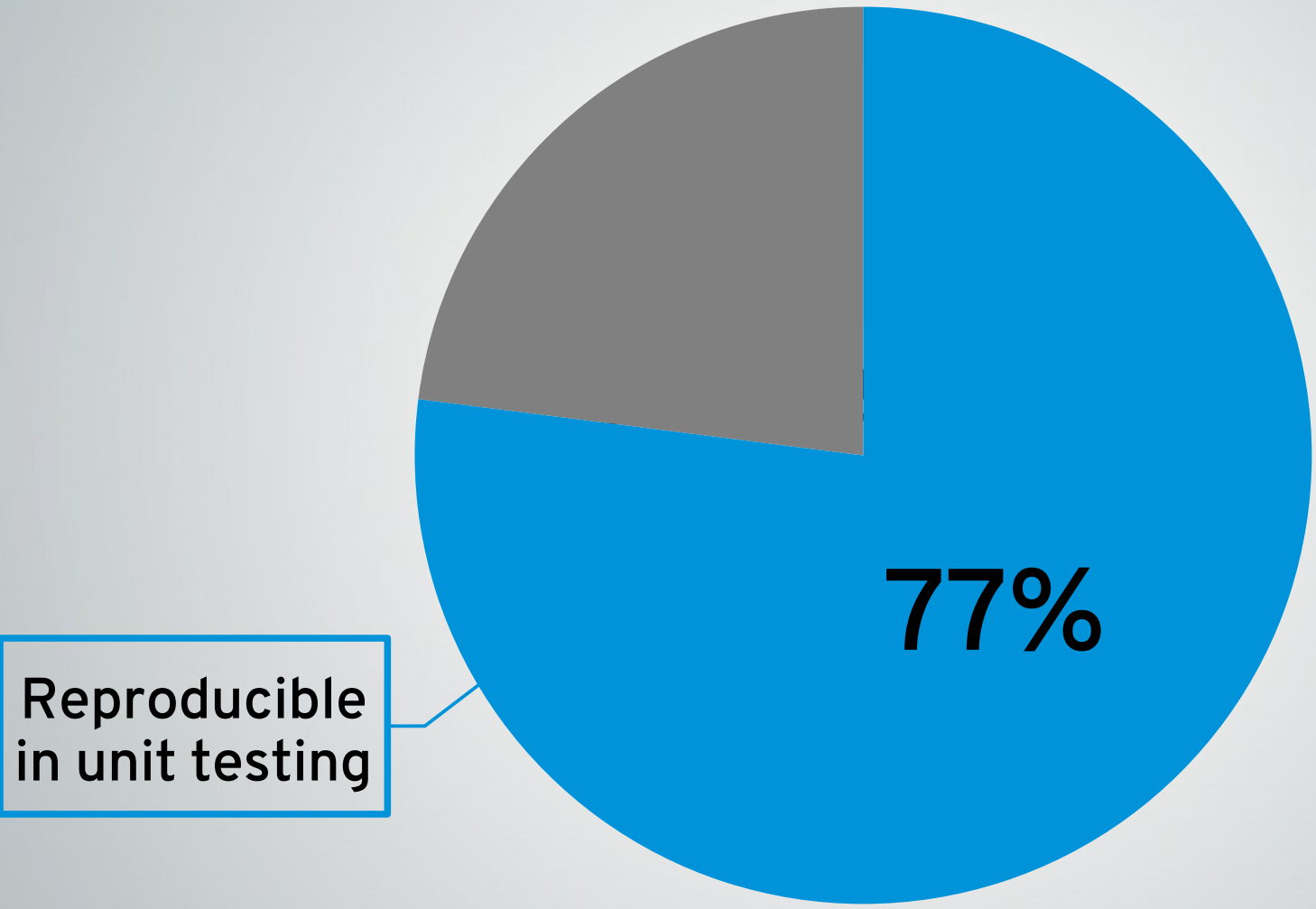
Source: Farooq & Quadri, 2011

Bad error handling



Source: Yan, Luo, Zhuang, Rodrigues, et al, 2014

Unit tests vs bugs

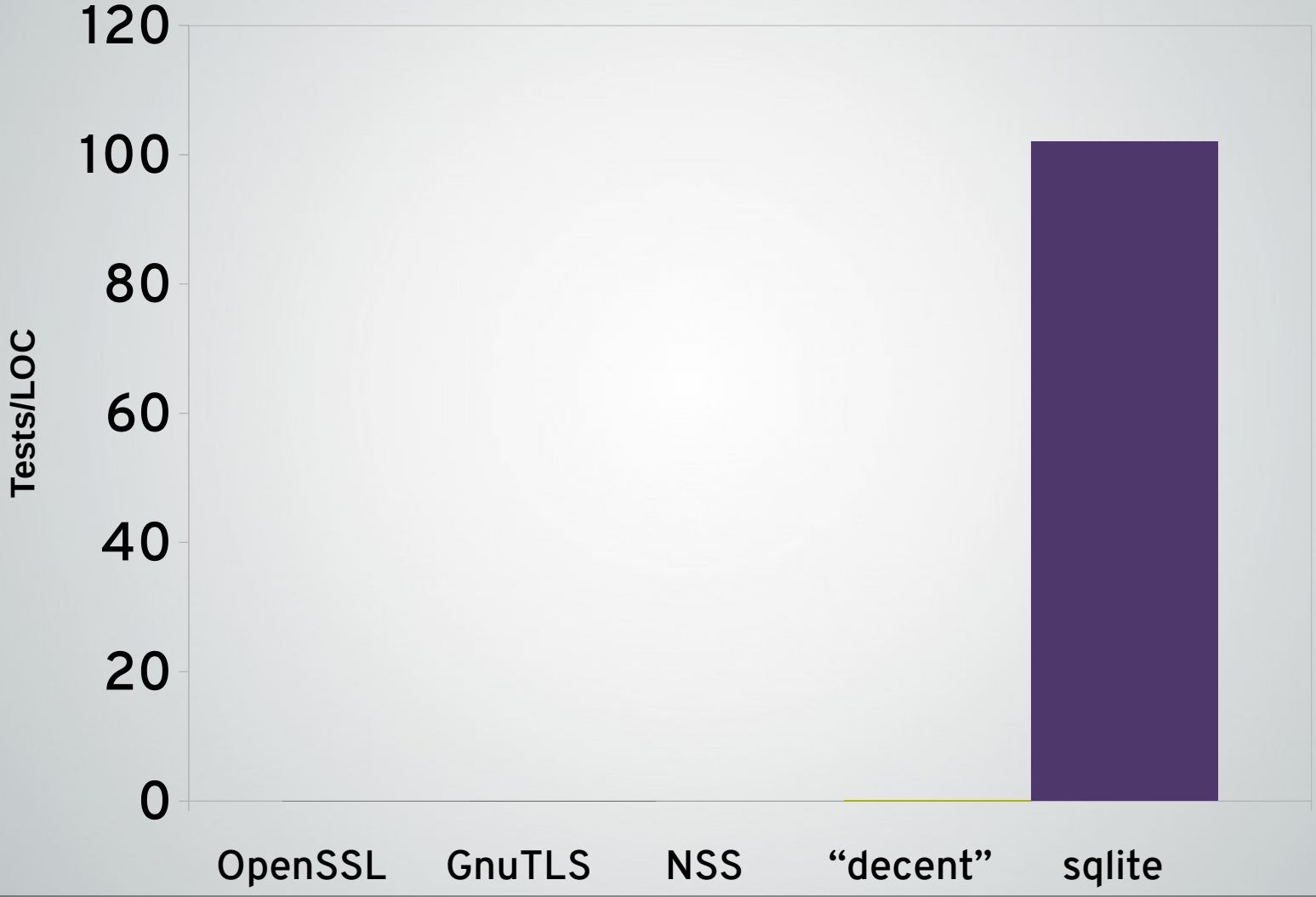


Source: Yan, Luo, Zhuang, Rodrigues, et al, 2014

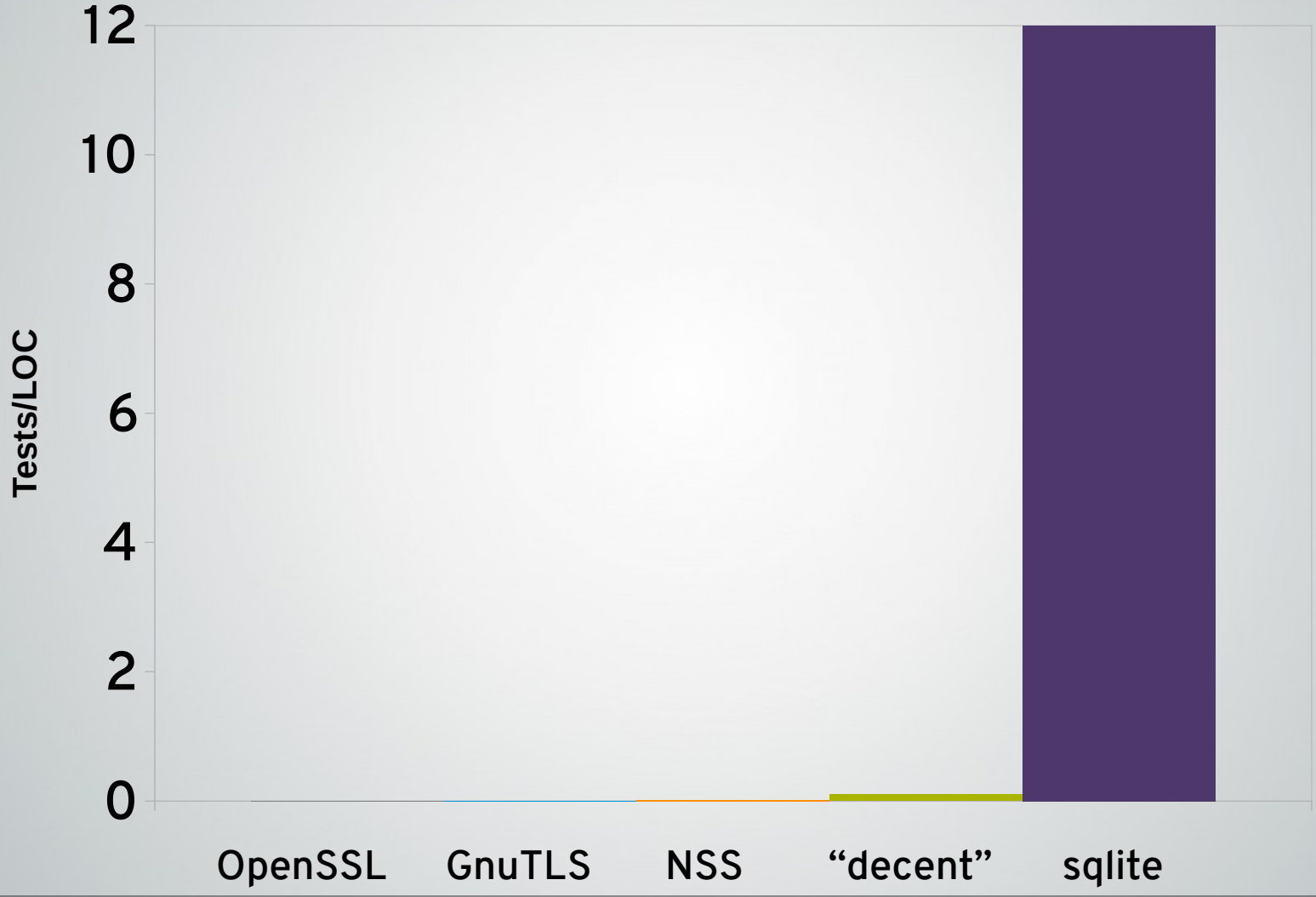
OSS TLS libraries

	OpenSSL	NSS	GnuTLS
Framework			
N° tests	100-200	>7000	100-200
Negative tests			

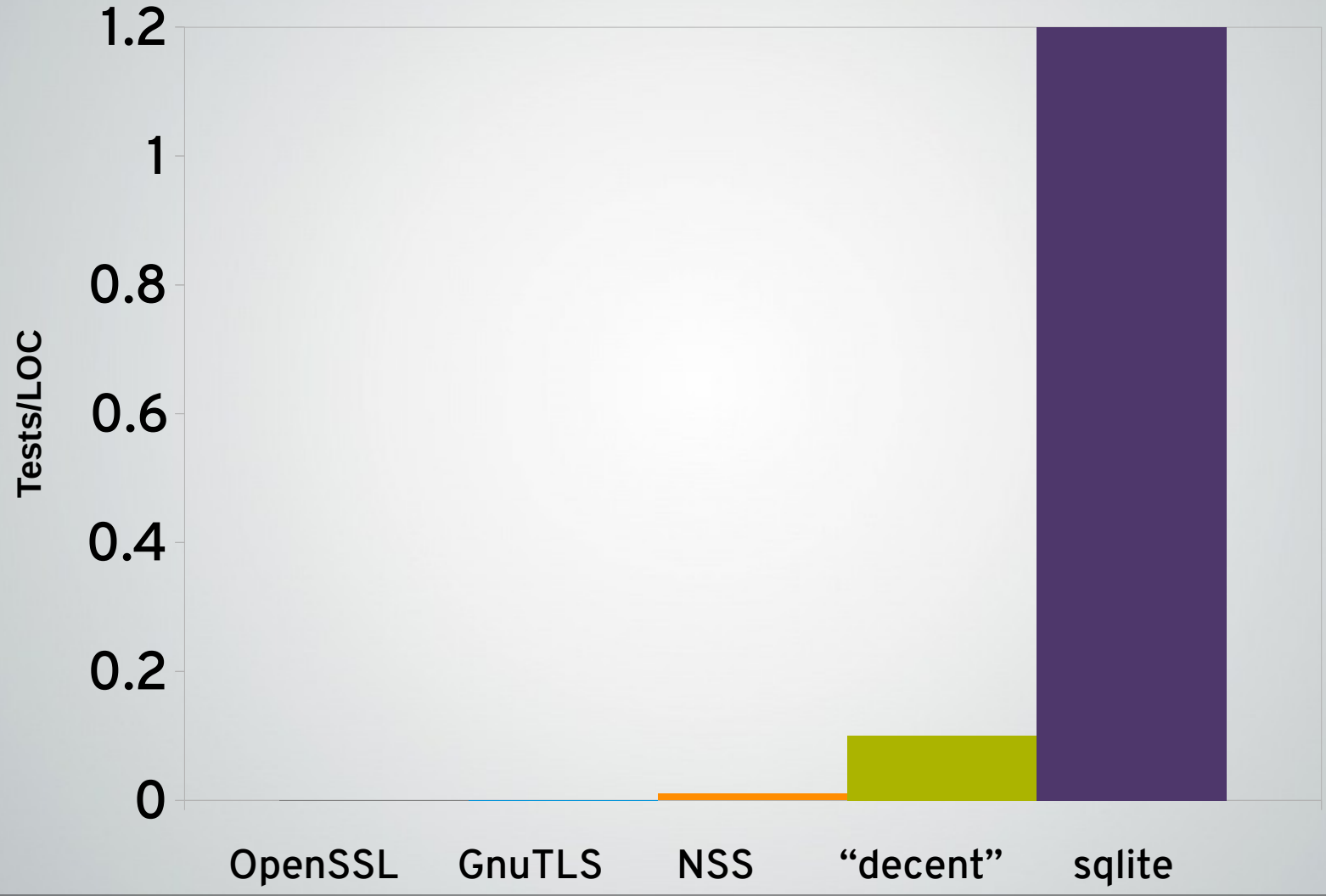
Test coverage



Test coverage



Test coverage



Why is that?

Libraries and bad data

Invisible bugs

Fuzzy testing

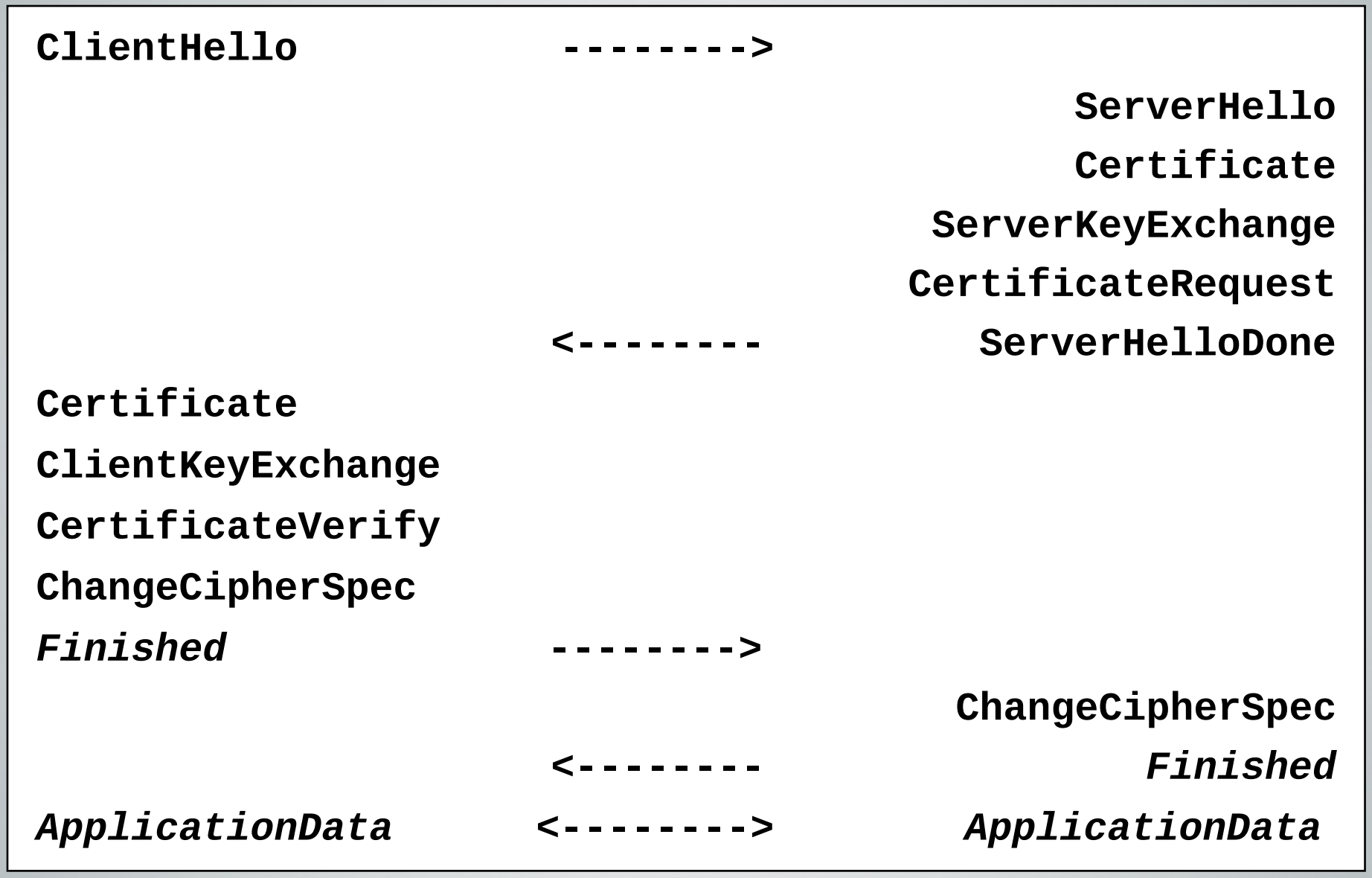
Compatibility fears

Fears of untested code

Fixing the problem

Duplication of effort

Full TLS handshake



Existing fuzzers

TLS testing (and fuzzing)

Timing information

Tlsfuzzer

(and tlslite-ng)

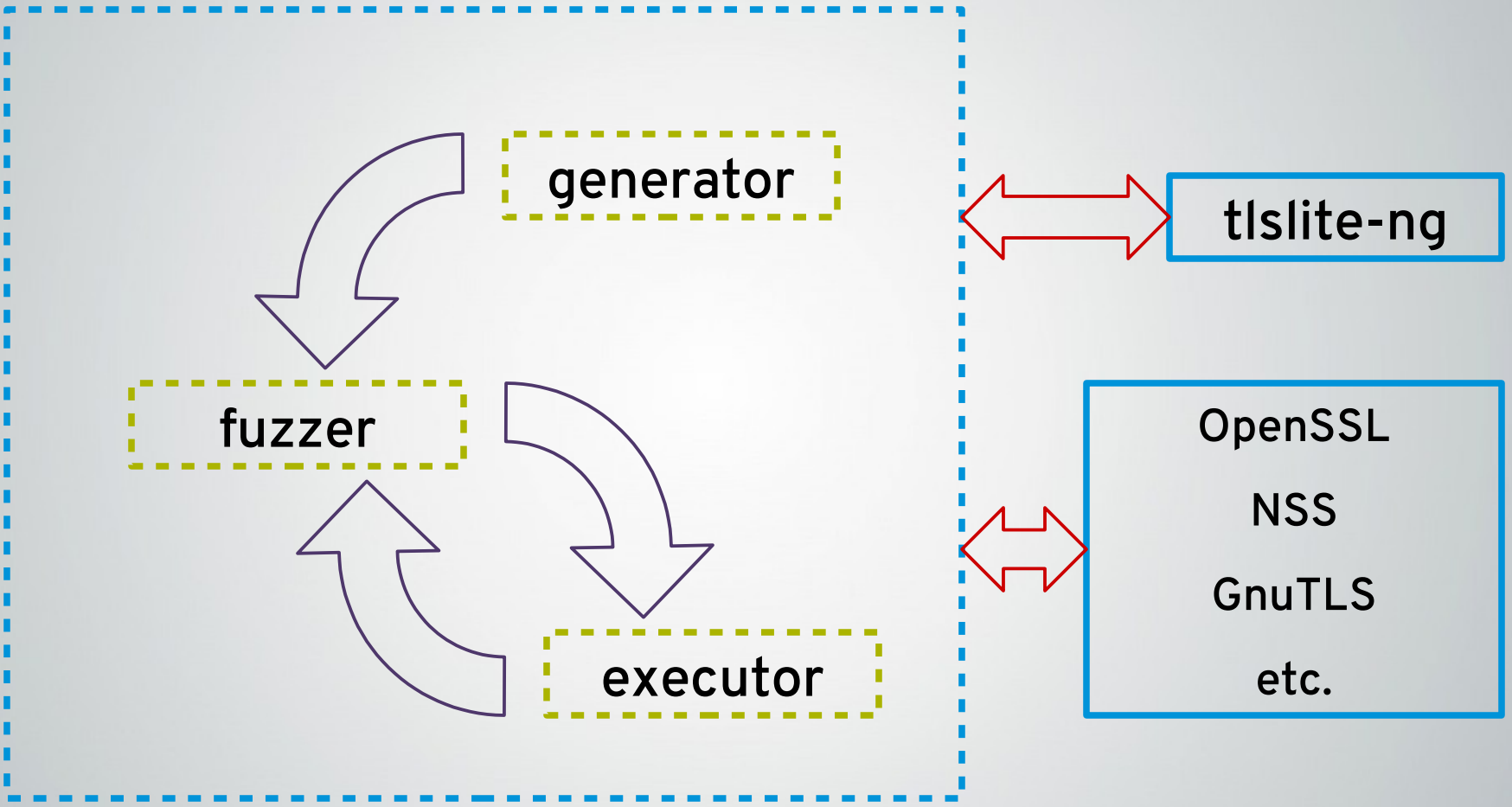
redhat

Use cases

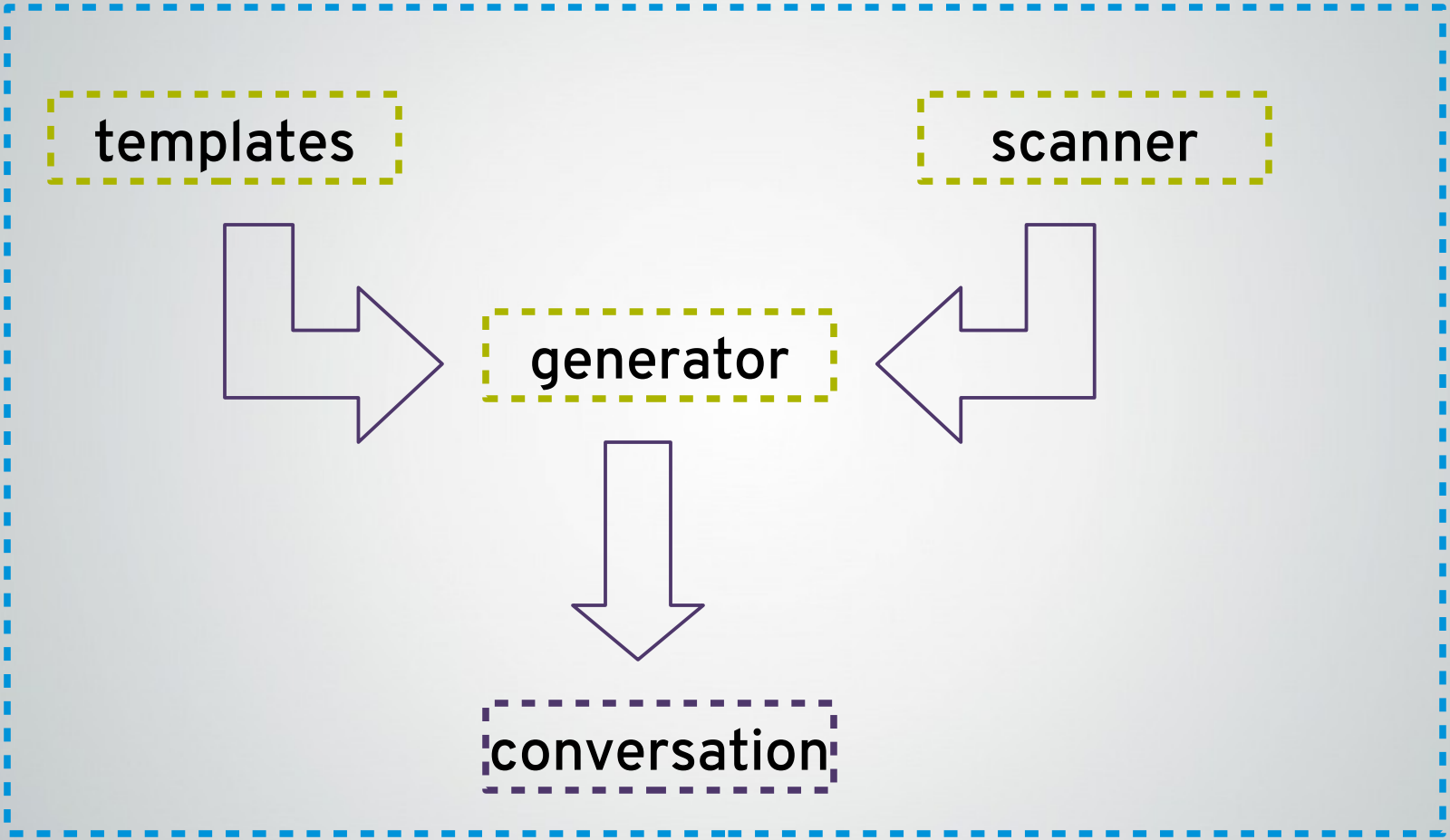
1. **Manual run (setup)**

2. **Automated run**

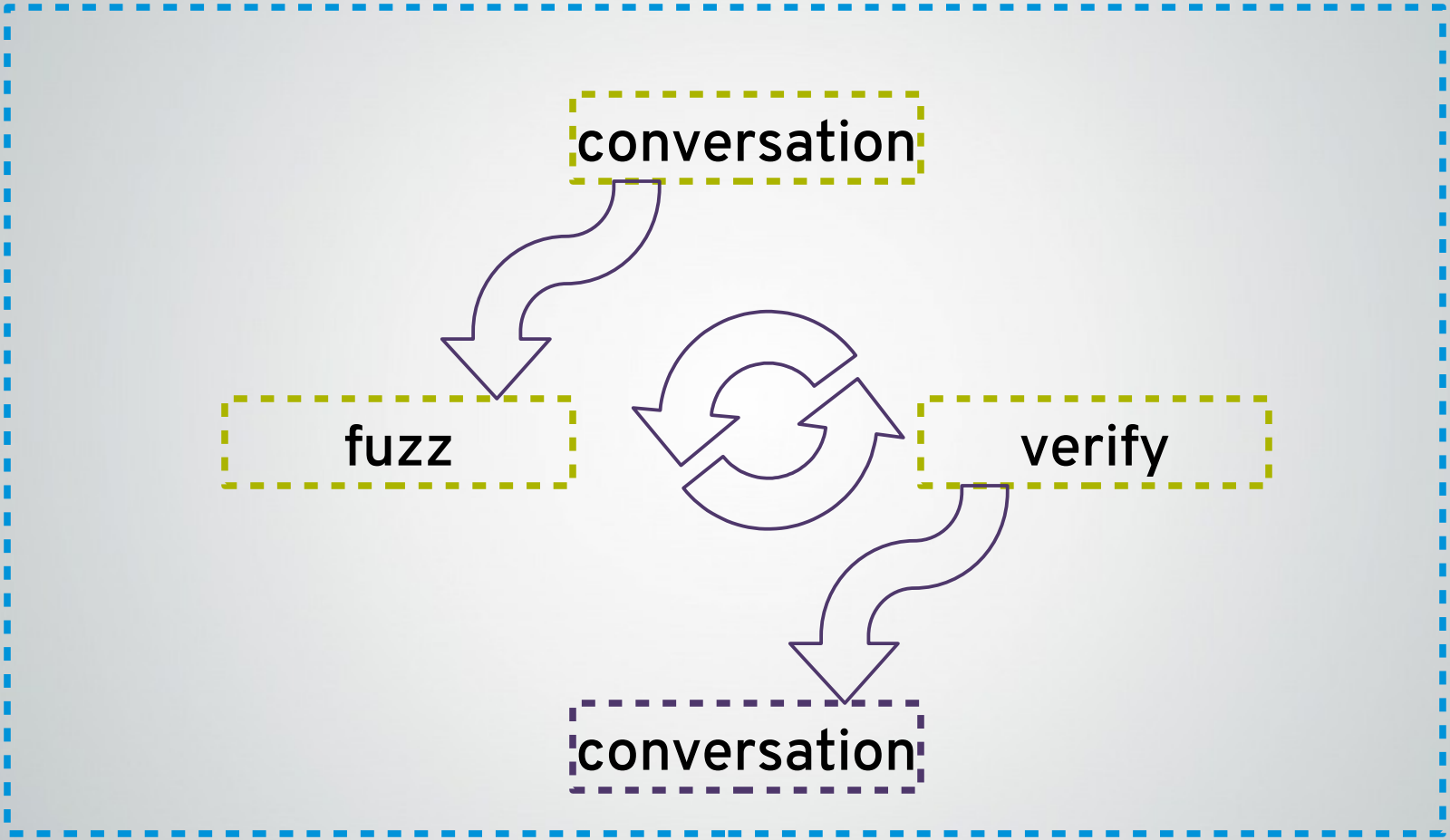
Architecture



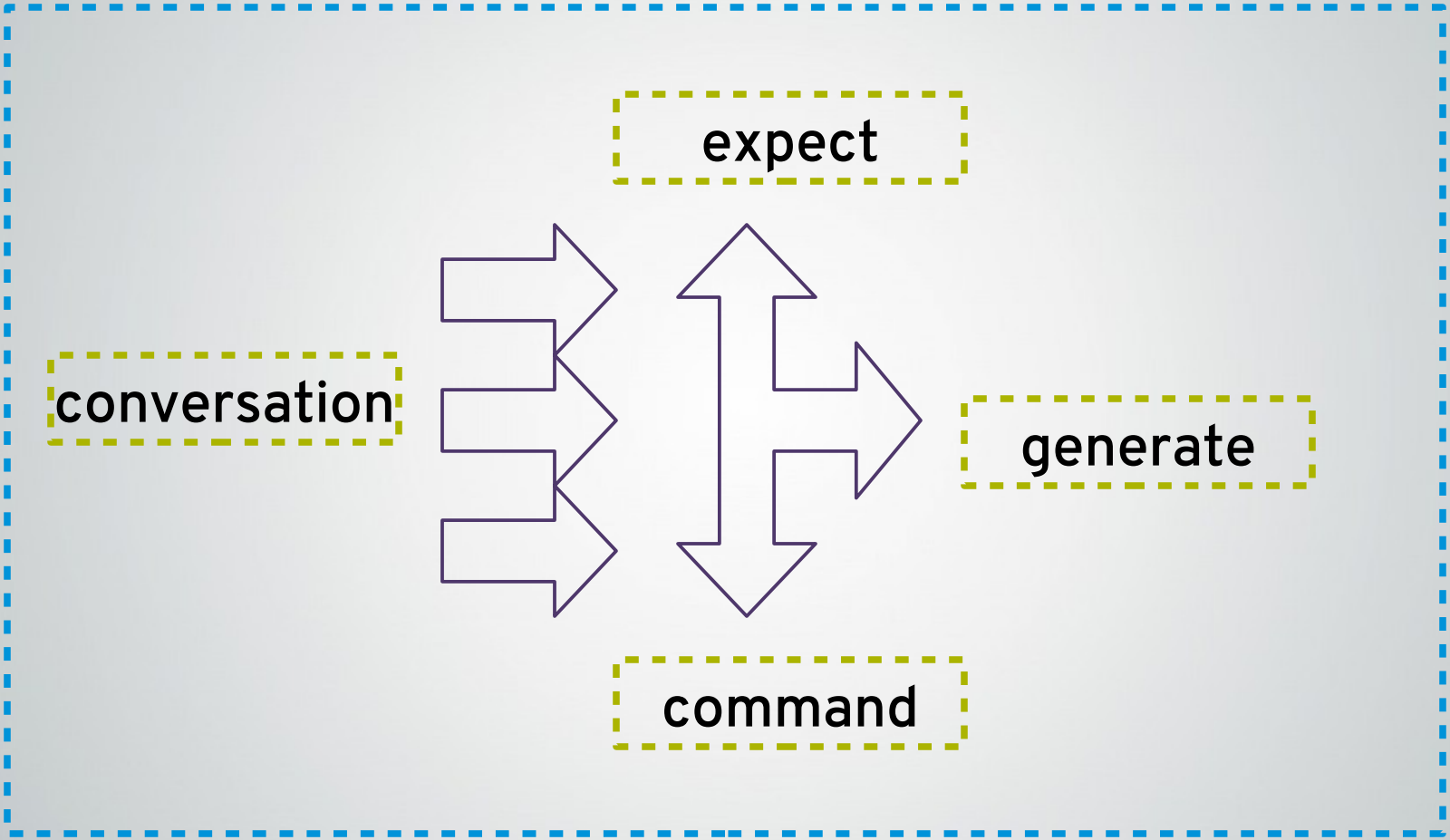
Generator architecture



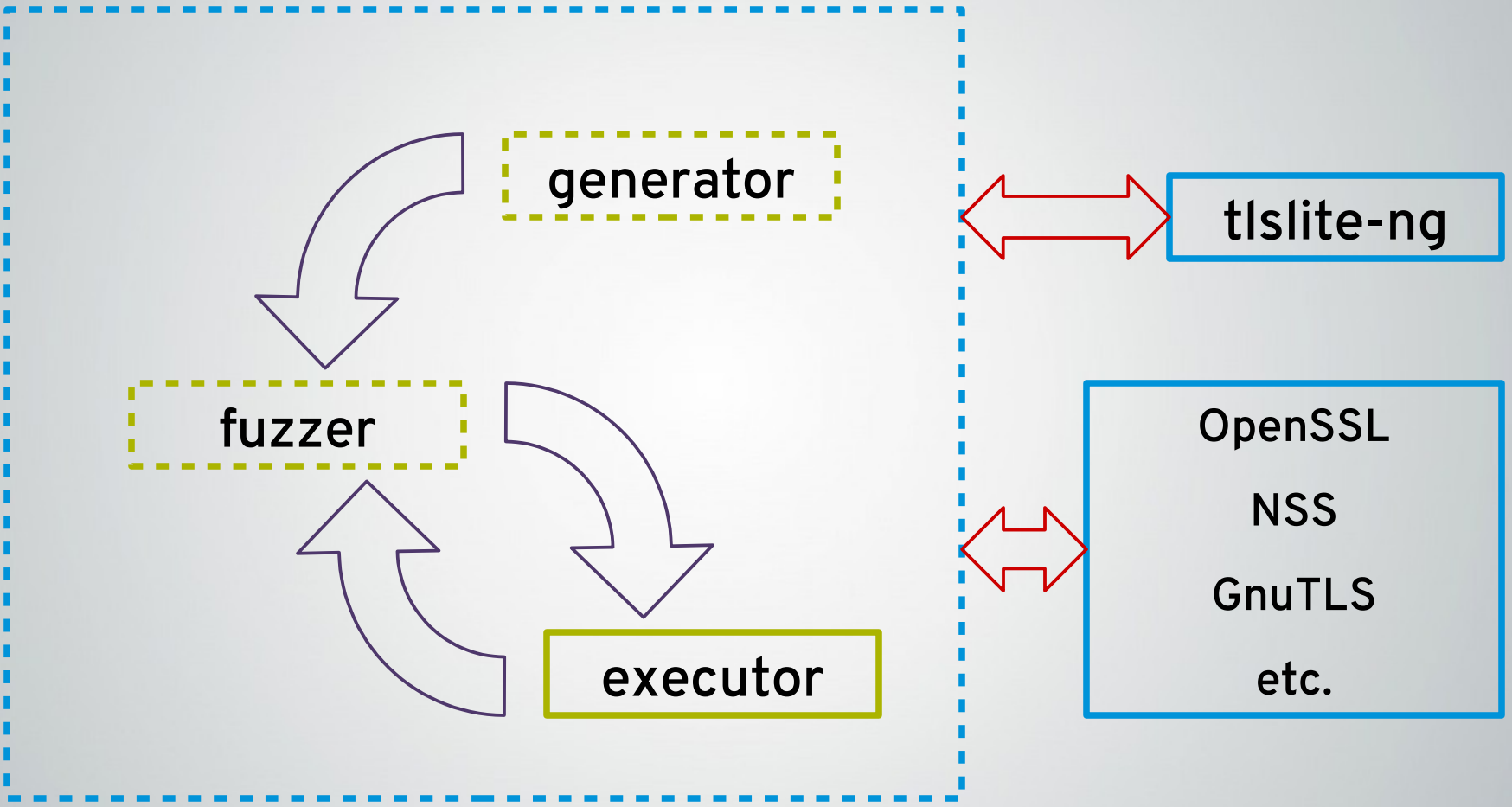
Fuzzer architecture



Runner architecture



Architecture



Correct run

```
$ openssl s_server -key /tmp/localhost.key -cert /tmp/localhost.crt
-www >/dev/null 2>&1
$ PYTHONPATH=. python scripts/test-interleaved-application-data-and-
fragmented-handshakes-in-renegotiation.py
Application data inside Finished...
OK
Application data inside Client Key Exchange...
OK
Application data inside Client Hello...
OK
Test end
successful: 3
failed: 0
```

Failing run

```
$ openssl s_server -key /tmp/localhost.key -cert /tmp/localhost.crt
-www >/dev/null 2>&1
$ PYTHONPATH=. python scripts/test-interleaved-application-data-and-
fragmented-handshakes-in-renegotiation.py
(...snip...)
Application data inside Client Hello...
Error encountered while processing node
<tlsfuzzer.expect.ExpectServerHello object at 0x7f0ac61d3310> with
last message being: <tlslite.messages.Message object at
0x7f0ac5f36a50>

(...snip...)
AssertionError: Unexpected message from peer: Alert(fatal,
unexpected_message)

Test end
successful: 1
failed: 2
```

Example test case

```
conversation = Connect("localhost", 4433)
node = conversation
ciphers = [CipherSuite.TLS_RSA_WITH_AES_128_CBC_SHA]
node = node.add_child(ClientHelloGenerator(ciphers))
node = node.add_child(ExpectServerHello())
node = node.add_child(ExpectCertificate())
node = node.add_child(ExpectServerHelloDone())
node = node.add_child(ClientKeyExchangeGenerator())
node = node.add_child(ChangeCipherSpecGenerator())
node = node.add_child(FinishedGenerator())
node = node.add_child(ExpectChangeCipherSpec())
node = node.add_child(ExpectFinished())
node = node.add_child(ApplicationDataGenerator(
                      bytearray(b"hello server!\n")))
node = node.add_child(AlertGenerator(
             AlertLevel.warning,
             AlertDescription.close_notify))
node = node.add_child(ExpectAlert())
node.next_sibling = ExpectClose()
```

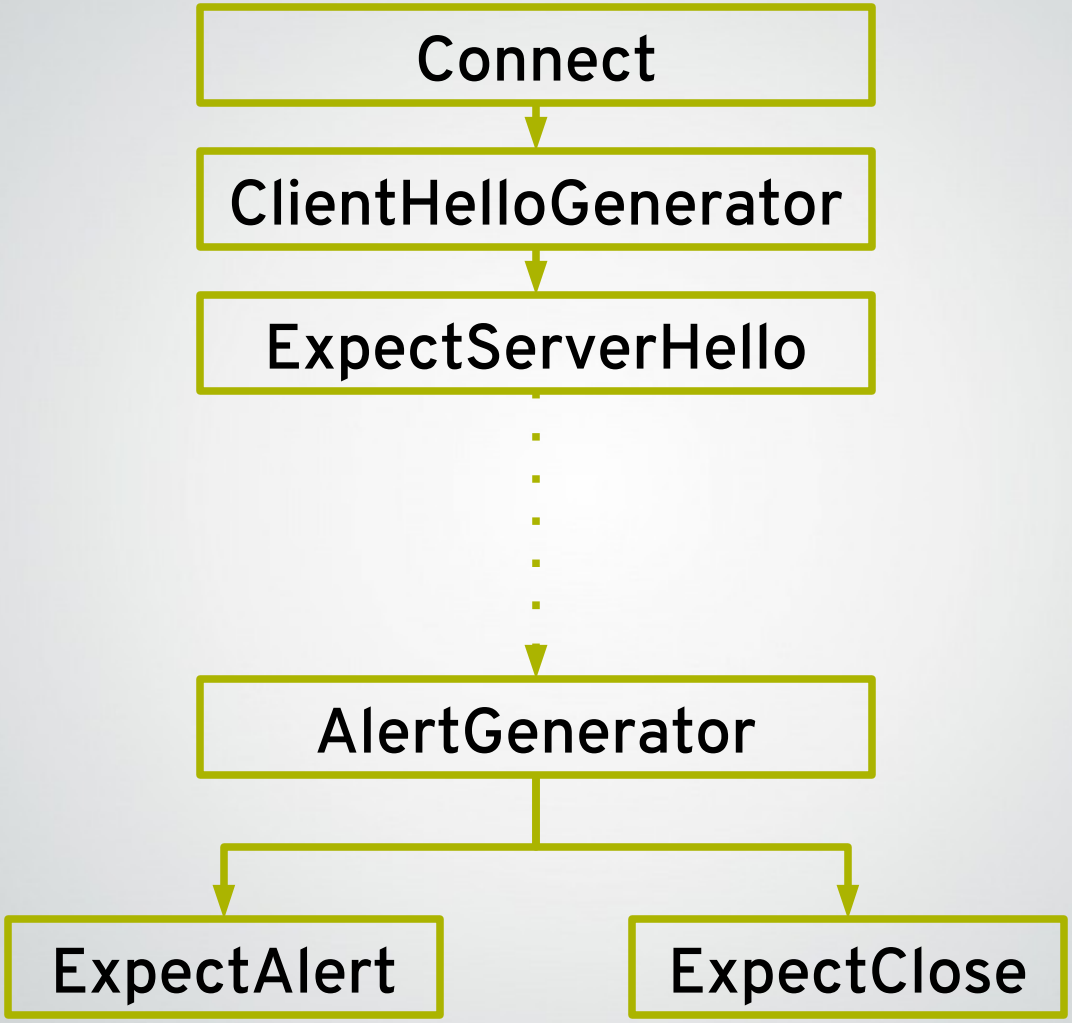

redhat.

Example test case

```
conversation = Connect("localhost", 4433)
node = conversation
ciphers = [CipherSuite.TLS_RSA_WITH_AES_128_CBC_SHA]
node = node.add_child(ClientHelloGenerator(ciphers))
node = node.add_child(ExpectServerHello())
node = node.add_child(ExpectCertificate())
node = node.add_child(ExpectServerHelloDone())
node = node.add_child(ClientKeyExchangeGenerator())
node = node.add_child(ChangeCipherSpecGenerator())
node = node.add_child(FinishedGenerator())
node = node.add_child(ExpectChangeCipherSpec())
node = node.add_child(ExpectFinished())
node = node.add_child(ApplicationDataGenerator(
                      bytearray(b"hello server!\n")))
node = node.add_child(AlertGenerator(
             AlertLevel.warning,
             AlertDescription.close_notify))
node = node.add_child(ExpectAlert())
node.next_sibling = ExpectClose()
```


redhat.

Decision tree



Invalid extension test case

```
conversation = Connect("localhost", 4433)
node = conversation
ciphers = [CipherSuite.TLS_RSA_WITH_AES_128_CBC_SHA]
ext = {0 : # server_name extension ID
       lambda _: TLSExtension().create(0, bytearray(b'\xff'*4))}
node = node.add_child(ClientHelloGenerator(ciphers, extensions=ext))
node = node.add_child(ExpectAlert(AlertLevel.fatal,
                                  AlertDescription.decode_error))
alert_node = node
node = node.add_child(ExpectCose())

alert_node.next_sibling = ExpectClose()
```


redhat.

Handshake message format

	Byte + 0	Byte + 1	Byte + 3	Byte + 4
Bytes 0..4	Message type	Message length		
Bytes 5..8	Version		Random (32 bytes)	
...	Session_ID length	Session_ID (0-32 bytes)		

Truncated message test case

```
conversation = Connect("localhost", 4433)
node = conversation
ciphers = [CipherSuite.TLS_RSA_WITH_AES_128_CBC_SHA]
node = node.add_child(truncate_handshake(
                                        ClientHelloGenerator(ciphers),
                                        1))
node = node.add_child(ExpectAlert(AlertLevel.fatal,
                                  AlertDescription.decode_error))
alert_node = node
node = node.add_child(ExpectCose())

alert_node.next_sibling = ExpectClose()
```

Padded message test case

```
conversation = Connect("localhost", 4433)
node = conversation
ciphers = [CipherSuite.TLS_RSA_WITH_AES_128_CBC_SHA]
node = node.add_child(pad_handshake(ClientHelloGenerator(ciphers),
                                    pad=bytearay(b'\xff\xff'))
node = node.add_child(ExpectAlert(AlertLevel.fatal,
                                  AlertDescription.decode_error))
alert_node = node
node = node.add_child(ExpectCose())

alert_node.next_sibling = ExpectClose()
```


redhat.

Features

- SSLv3, TLSv1.0, TLSv1.1 and TLSv1.2
- AES-CBC, AES-GCM, 3DES, RC4 and NULL ciphers
- MD5, SHA1, SHA256 and SHA384 HMAC
- RSA, SRP, SRP_RSA, DHE and DH_anon key exchange
- Encrypt-then-MAC
- TACK certificate pinning
- Client certificates
- Secure renegotiation
- TLS_FALLBACK_SCSV
- Next Protocol Negotiation
- ChaCha20/Poly1305 (soon™)
- ECDHE (soon™)

Missing stuff

- Drafts of TLSv1.3
- Extended master secret
- PSK key exchange
- ALPN
- AES-CCM
- CAMELLIA (CBC and GCM)
- ECDSA, DSA certificates
- Drafts of Curve25519
- Raw keys, GPG keys
- Heartbeat protocol
- Kerberos

Missing stuff

- **Test cases!**

Results

Contributing

- https://github.com/tomato42/tlsfuzzer
- https://github.com/tomato42/tlslite-ng

- GPLv2 for tlsfuzzer
- LGPLv2 for tlslite-ng

- Tags review request and help wanted

Questions?

Contact: hkario@redhat.com

Project: https://github.com/tomato42/tlsfuzzer

redhat