OpenSSL CONFERENCE

New Insights into the Formal Verification of AWS-LC, a Fork of OpenSSL

Nicky Mouha, KeyCryptic

Overview

13.10.2026

- Focus
  - Topic: mlkem-native/mldsa-native
  - Claimed proof: no memory/integer overflow
  - This talk: proof issues

Overview

13.10.2026

- Focus
  - Topic: mlkem-native/mldsa-native
  - Claimed proof: no memory/integer overflow
  - This talk: proof issues
- Insights
  - Implicit buffer bounds
  - Loop termination
  - `forall()` does not loop over all indices 🆕

Overview

13.10.2026

- Focus
  - Topic: mlkem-native/mldsa-native
  - Claimed proof: no memory/integer overflow
  - This talk: proof issues
- Insights
  - Implicit buffer bounds
  - Loop termination
  - `forall()` does not loop over all indices 🆕
- Warning
  - Issues acknowledged
  - No vulnerabilities found!

Undefined Behavior

13.10.2026

OpenSSL CONFERENCE

- Undefined Behavior in C
  - Violation of C standard rules
  - Can have unpredictable behavior
  - And potentially introduce vulnerability!

Undefined Behavior

13.10.2026

OpenSSL CONFERENCE

- Undefined Behavior in C
  - Violation of C standard rules
  - Can have unpredictable behavior
  - And potentially introduce vulnerability!

- Examples of Underfined Behavior
  - Array index out of bounds (buffer overflow)
  - Signed int overflow (**not**: unsigned overflow)
  - Infinite loop (non-constant controlling expr)
  - ...

SHA-3 Buffer Overflow

OpenSSL CONFERENCE

SHA-3 Buffer Overflow (Mouha and Celi, CT-RSA 2023) → code execution!

In `XKCP/lib/high/Keccak/KeccakSponge.inc`:

Addition overflow:

```
if (partialBlock + instance->byteIOIndex > rateInBytes) {
  partialBlock = rateInBytes - instance->byteIOIndex;
}
```

SHA-3 Buffer Overflow

OpenSSL CONFERENCE

SHA-3 Buffer Overflow (Mouha and Celi, CT-RSA 2023) → code execution!

In `XKCP/lib/high/Keccak/KeccakSponge.inc`:

Addition overflow:

```
if (partialBlock + instance->byteIOIndex > rateInBytes) {
  partialBlock = rateInBytes - instance->byteIOIndex;
}
```

No addition overflow:

```
if (partialBlock > rateInBytes - instance->byteIOIndex) {
  partialBlock = rateInBytes - instance->byteIOIndex;
}
```

SHA-3 Buffer Overflow

OpenSSL CONFERENCE

SHA-3 Buffer Overflow (Mouha and Celi, CT-RSA 2023) → code execution!

In `XKCP/lib/high/Keccak/KeccakSponge.inc`:

Addition overflow:

```
if (partialBlock + instance->byteIOIndex > rateInBytes) {
  partialBlock = rateInBytes - instance->byteIOIndex;
}
```

No addition overflow:

```
if (partialBlock > rateInBytes - instance->byteIOIndex) {
  partialBlock = rateInBytes - instance->byteIOIndex;
}
```

SHA-3 Buffer Overflow

OpenSSL CONFERENCE

SHA-3 Buffer Overflow (Mouha and Celi, CT-RSA 2023) → code execution!

In `XKCP/lib/high/Keccak/KeccakSponge.inc`:

Addition overflow:

```
if (partialBlock + instance->byteIOIndex > rateInBytes) {
  partialBlock = rateInBytes - instance->byteIOIndex;
}
```

No addition overflow:

```
if (partialBlock > rateInBytes - instance->byteIOIndex) {
  partialBlock = rateInBytes - instance->byteIOIndex;
}
```

SHA-3 Buffer Overflow

OpenSSL CONFERENCE

SHA-3 Buffer Overflow (Mouha and Celi, CT-RSA 2023) → code execution!

In `XKCP/lib/high/Keccak/KeccakSponge.inc`:

Addition overflow:

```
if (partialBlock + instance->byteIOIndex > rateInBytes) {
  partialBlock = rateInBytes - instance->byteIOIndex;
}
```

No addition overflow:

```
if (partialBlock > rateInBytes - instance->byteIOIndex) {
  partialBlock = rateInBytes - instance->byteIOIndex;
}
```

mldsa-native: Implicit Buffer Bounds (1/3)

OpenSSL CONFERENCE

In mldsa-native/mldsa/src/fips202/fips202.c:

```
while (pos + inlen >= r)
__loop__(
  assigns(pos, i, in, inlen,
    memory_slice(s, sizeof(uint64_t) *  MLD_KECCAK_LANES))
  invariant(inlen <= loop_entry(inlen))
  invariant(pos <= r)
  invariant(in == loop_entry(in) + (loop_entry(inlen) - inlen)))
{
  mld_keccakf1600_xor_bytes(s, in, pos, r - pos);
  inlen -= r - pos;
  in += r - pos;
  mld_keccakf1600_permute(s);
  pos = 0;
}
```

mldsa-native: Implicit Buffer Bounds (1/3)

OpenSSL CONFERENCE

In `mldsa-native/mldsa/src/fips202/fips202.c`:

```
while (pos + inlen >= r)
__loop__(
  assigns(pos, i, in, inlen,
    memory_slice(s, sizeof(uint64_t) *  MLD_KECCAK_LANES))
  invariant(inlen <= loop_entry(inlen))
  invariant(pos <= r)
  invariant(in == loop_entry(in) + (loop_entry(inlen) - inlen)))
{
  mld_keccakf1600_xor_bytes(s, in, pos, r - pos);
  inlen -= r - pos;
  in += r - pos;
  mld_keccakf1600_permute(s);
  pos = 0;
}
```

Addition overflow in `pos + len`? Function definition?

mldsa-native: Implicit Buffer Bounds (2/3)

OpenSSL CONFERENCE

In `mldsa-native/mldsa/src/fips202/fips202.c`:

```
static unsigned int keccak_absorb(uint64_t s[MLD_KECCAK_LANES],
                                  unsigned int pos, unsigned int r,
                                  const uint8_t *in, size_t inlen)
__contract__(

  requires(r < sizeof(uint64_t) * MLD_KECCAK_LANES)
  requires(pos <= r)
  requires(memory_no_alias(s, sizeof(uint64_t) * MLD_KECCAK_LANES))
  requires(memory_no_alias(in, inlen))
  assigns(memory_slice(s, sizeof(uint64_t) * MLD_KECCAK_LANES))
  ensures(return_value < r))
{
```

mldsa-native: Implicit Buffer Bounds (2/3)

OpenSSL CONFERENCE

In `mldsa-native/mldsa/src/fips202/fips202.c`:

```
static unsigned int keccak_absorb(uint64_t s[MLD_KECCAK_LANES],
                                  unsigned int pos, unsigned int r,
                                  const uint8_t *in, size_t inlen)
__contract__(

  requires(r < sizeof(uint64_t) * MLD_KECCAK_LANES)
  requires(pos <= r)
  requires(memory_no_alias(s, sizeof(uint64_t) * MLD_KECCAK_LANES))
  requires(memory_no_alias(in, inlen))
  assigns(memory_slice(s, sizeof(uint64_t) * MLD_KECCAK_LANES))
  ensures(return_value < r))
{
```

Why addition overflow not detected by CBMC?

mldsa-native: Implicit Buffer Bounds (3/3)

OpenSSL CONFERENCE

In `mldsa-native/mldsa/src/fips202/fips202.c`:

```
static unsigned int keccak_absorb(uint64_t s[MLD_KECCAK_LANES],
                                  unsigned int pos, unsigned int r,
                                  const uint8_t *in, size_t inlen)
__contract__(
  requires(inlen <= MLD_MAX_BUFFER_SIZE)
  requires(r < sizeof(uint64_t) * MLD_KECCAK_LANES)
  requires(pos <= r)
  requires(memory_no_alias(s, sizeof(uint64_t) * MLD_KECCAK_LANES))
  requires(memory_no_alias(in, inlen))
  assigns(memory_slice(s, sizeof(uint64_t) * MLD_KECCAK_LANES))
  ensures(return_value < r))
{
```

Now: implicit CBMC assumption added! Necessary?

Loop Merging (Loop Fusion)

OpenSSL CONFERENCE

ISO/IEC JTC1/SC22 WG14/N1528

```
for (p = q; p != 0; p = p->next) {
  ++count;
}

for (p = q; p != 0; p = p->next) {
  ++count2;
}
```

Loop Merging (Loop Fusion)

OpenSSL CONFERENCE

ISO/IEC JTC1/SC22 WG14/N1528

```
for (p = q; p != 0; p = p->next) {
  ++count;
}

for (p = q; p != 0; p = p->next) {
  ++count2;
}
```

```
for (p = q; p != 0; p = p->next) {
  ++count;
  ++count2;
}
```

Loop Merging (Loop Fusion)

OpenSSL CONFERENCE

ISO/IEC JTC1/SC22 WG14/N1528

```
for (p = q; p != 0; p = p->next) {
  ++count;
}

for (p = q; p != 0; p = p->next) {
  ++count2;
}
```

```
for (p = q; p != 0; p = p->next) {
  ++count;
  ++count2;
}
```

Infinite loop: undefined behavior in C11 → loop merging allowed!

mldsa-native: Loop Termination (1/2)

OpenSSL CONFERENCE

In mldsa-native/mldsa/src/fips202/fips202.c:

```
while (inlen >= r - pos)
__loop__(
  assigns(pos, in, inlen,
    memory_slice(s, sizeof(uint64_t) *  MLD_KECCAK_LANES))
  invariant(inlen <= loop_entry(inlen))

  invariant(pos < r)
  invariant(in == loop_entry(in) + (loop_entry(inlen) - inlen)))
{
  mld_keccakf1600_xor_bytes(s, in, pos, r - pos);
  inlen -= r - pos;
  in += r - pos;
  mld_keccakf1600_permute(s);
  pos = 0;
}
```

What if `pos == r`? Infinite loop!

mldsa-native: Loop Termination (2/2)

OpenSSL CONFERENCE

In mldsa-native/mldsa/src/fips202/fips202.c:

```
while (inlen >= r - pos)
__loop__(
  assigns(pos, in, inlen,
    memory_slice(s, sizeof(uint64_t) *  MLD_KECCAK_LANES))
  invariant(inlen <= loop_entry(inlen))
  invariant(r > 0)
  invariant(pos < r)
  invariant(in == loop_entry(in) + (loop_entry(inlen) - inlen)))
{
  mld_keccakf1600_xor_bytes(s, in, pos, r - pos);
  inlen -= r - pos;
  in += r - pos;
  mld_keccakf1600_permute(s);
  pos = 0;
}
```

Now: Updated bound on r! Necessary?

mldsa-native: forall() is not “for all”

OpenSSL CONFERENCE

In mldsa-native/mldsa/src/cbmc.h:

```
#define forall(qvar, qvar_lb, qvar_ub, predicate)                \
  __CPROVER_forall                                               \
  {                                                              \
    unsigned qvar;                                               \
    ((qvar_lb) <= (qvar) && (qvar) < (qvar_ub)) ==> (predicate)   \
  }
```

Problem: `unsigned` is 32 bits on 64-bit platforms

Impact: unsound proofs and implementations inconsistent with documentation

Example: `mld_ct_memcmp()` is not an „ordinary memcmp()“ (documentation), compares only first 2^{32} values

Conclusion

13.10.2026

- mlkem-native/mldsa-native
  - Claimed proof: no memory/integer overflow
- This talk: proof issues
  - Implicit buffer bounds
  - Loop termination
  - `forall()` does not loop over all indices
- No vulnerabilities found!

Conclusion

13.10.2026

- mlkem-native/mldsa-native
  - Claimed proof: no memory/integer overflow
- This talk: proof issues
  - Implicit buffer bounds
  - Loop termination
  - `forall()` does not loop over all indices
- No vulnerabilities found!
- Questions?