TRUST IN SOFT

Mathematical Guarantees Eliminate Software Risk

TrustInSoft
Updates about the Frama-C software publisher company

Benjamin Monate

co-founder & CTO

Frama-C Day 2016, June 17th, 2016.

Company Facts

- French startup created in 2013 as a Spin-off of CEA

NIST National Institute of Standards and Technology — Only company selected in the Ockham Criteria from the SATE V exhibit

THE LINUX FOUNDATION — Chosen by the Linux Foundation to develop tools for security of Core Internet Infrastructure

RSA Conference — Nominated as one the 10 most innovative companies in cybersecurity – RSA '15 Conference

FIC 2016 — Most Innovative SME, Special Jury prize 2016 at the Forum International de la Cybersécurité 2016 sponsored by Airbus Defence&Space

TrustInSoft Unique Value Proposal

provide guarantees on software used in sensitive systems

Current Customers

TrustInSoft works with the most demanding developers of sensitive software.

Since 2013

Aeronautics
DO-178C – ED-12C

Nuclear Reactors
IEC-60880 IEC-62138

Defense

Since 2014

Rail
EN-50128

Space

Telecom

Since 2015

Automotive
ISO 2626-2

Smart Factories

Cyber
CWE

Customer names are under strict NDAs – 50% in the US

ARM mbed can claim they have the first ever TLS/SSL stack without buffer overflows.

ARM mbed

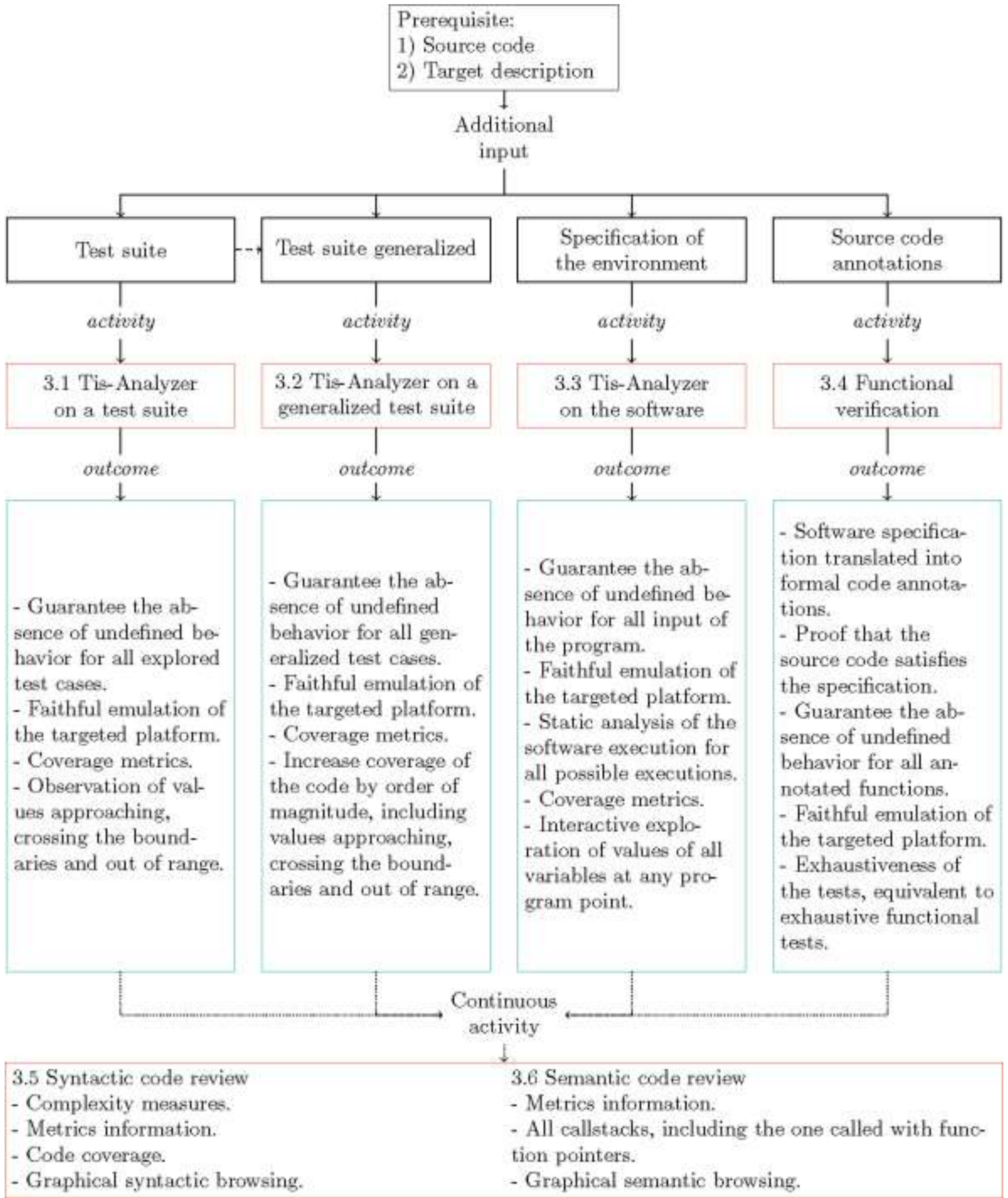
Using TrustInSoft Analyzer we have generated a report which tells how to compile, configure and deploy mbed TLS in a given perimeter in order to be immune from all attacks caused by CWE 119 to 127, 369, 415, 416, 457, 476, 562, 690.

This stack has a configuration proven to be without an Heartbleed-like flaw.

You can download such a report here: http://trust-in-soft.com/polarssl-verification-kit

Progressive methodologies

- From enhanced testing to full functional verification
- Adapt level of verification to each customer
- Incremental guarantees



- Guarantee the absence of undefined behavior for all explored test cases.
- Faithful emulation of the targeted platform.
- Coverage metrics.
- Observation of values approaching, crossing the boundaries and out of range.

- Guarantee the absence of undefined behavior for all generalized test cases.
- Faithful emulation of the targeted platform.
- Coverage metrics.
- Increase coverage of the code by order of magnitude, including values approaching, crossing the boundaries and out of range.

- Guarantee the absence of undefined behavior for all input of the program.
- Faithful emulation of the targeted platform.
- Static analysis of the software execution for all possible executions.
- Coverage metrics.
- Interactive exploration of values of all variables at any program point.

- Software specification translated into formal code annotations.
- Proof that the source code satisfies the specification.
- Guarantee the absence of undefined behavior for all annotated functions.
- Faithful emulation of the targeted platform.
- Exhaustiveness of the tests, equivalent to exhaustive functional tests.

3.5 Syntactic code review
- Complexity measures.
- Metrics information.
- Code coverage.
- Graphical syntactic browsing.

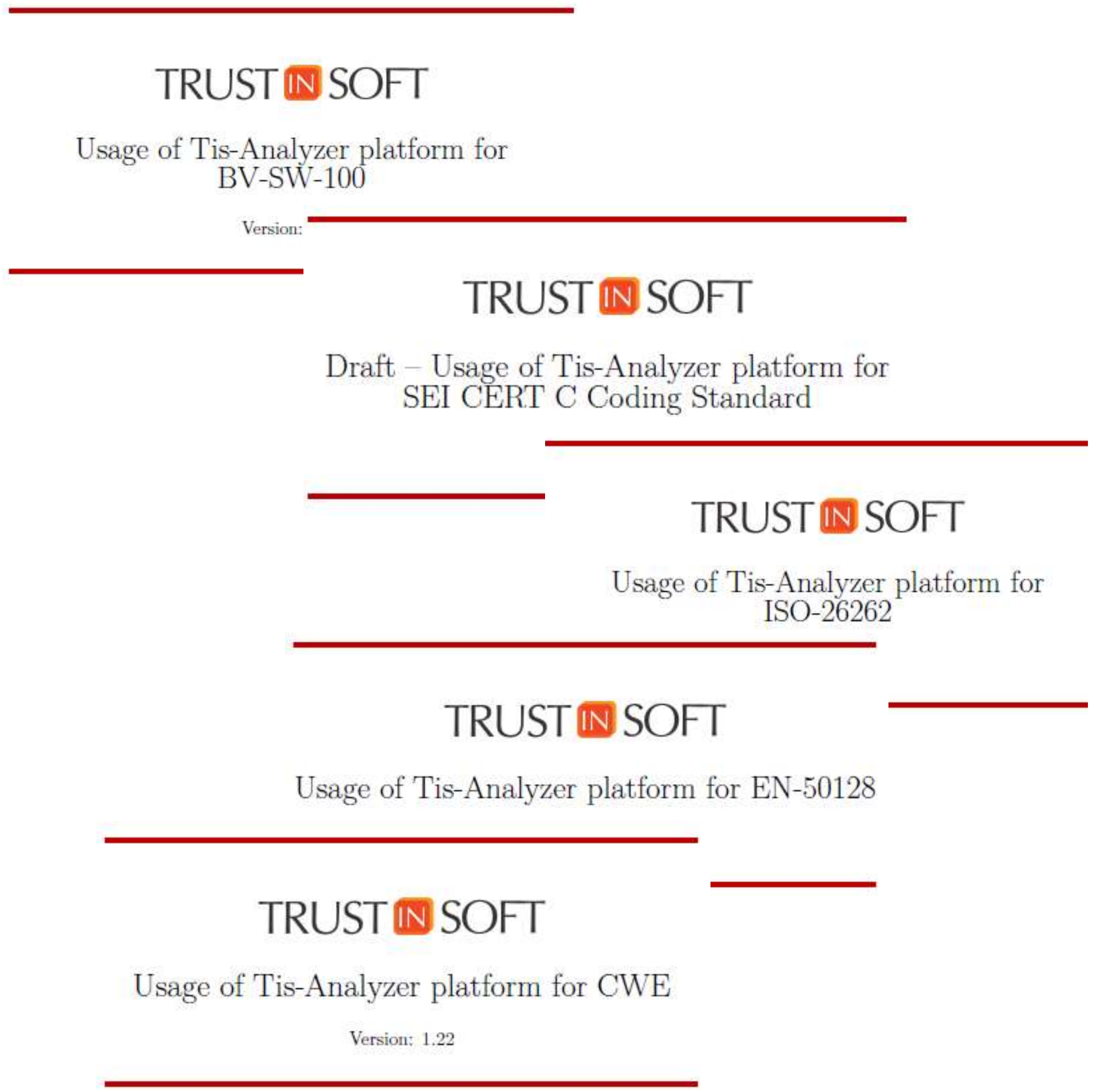
3.6 Semantic code review
- Metrics information.
- All callstacks, including the one called with function pointers.
- Graphical semantic browsing.

Standard compliance

Documents for common software verification standards

- ISO 26262,
- EN50128,
- DO178B/C,
- BV-SW-100,
- SEI CERT C

- Preparation phase: for each verification activity, state the support level customer can expect from TIS-Analyzer
- Production phase: propose specific tool adaptations to match the customer process



TRUST IN SOFT

Usage of Tis-Analyzer platform for BV-SW-100

Version:

TRUST IN SOFT

Draft – Usage of Tis-Analyzer platform for SEI CERT C Coding Standard

TRUST IN SOFT

Usage of Tis-Analyzer platform for ISO-26262

TRUST IN SOFT

Usage of Tis-Analyzer platform for EN-50128

TRUST IN SOFT

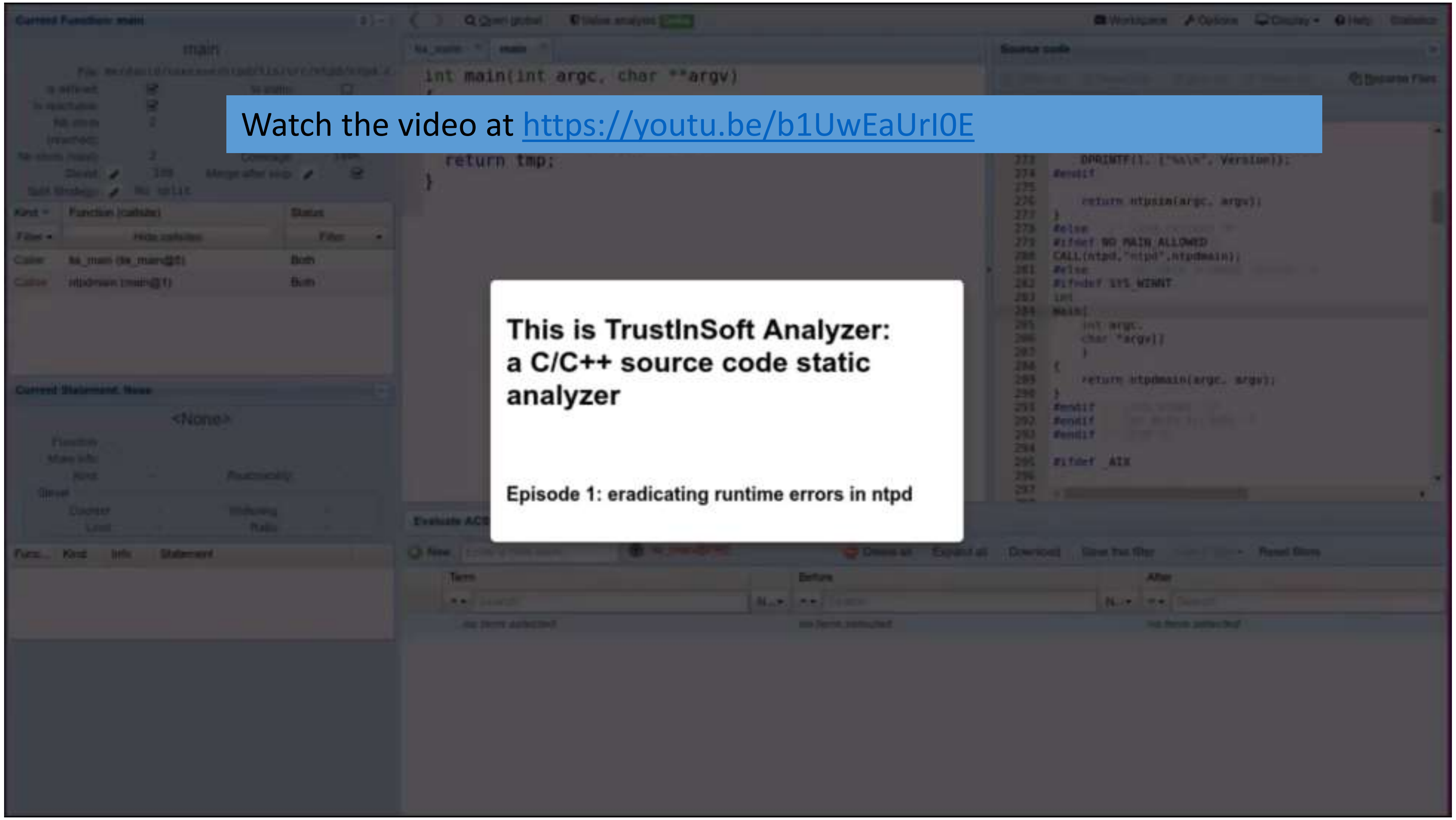
Usage of Tis-Analyzer platform for CWE

Version: 1.22

Training sessions: 1 to 5 days for engineers

- Introduction to formal method for software analysis
- Why undefined behaviors matter in C11
- TrustInSoft Analyzer for Safety Standard Compliance
- Testing Software with a perfect source semantics: TIS-Interpreter
- Eradicating undefined behaviors in existing C applications
- Developing secured applications in C with TIS-Analyzer

A glance at TrustInSoft Analyzer User Interface



Watch the video at https://youtu.be/b1UwEaUrI0E

This is TrustInSoft Analyzer: a C/C++ source code static analyzer

Episode 1: eradicating runtime errors in ntpd

Innovation at TrustInSoft

- Global Dataflow Analysis

  Seeding a random number generator

- Side Channel Attacks Analysis

  Constant time or memory access

- Strict aliasing analysis

  Typed memory model

Innovation at TrustInSoft

- Precision improvement for derived analysis
    - Value analysis result stored as a graph
- Analysis of large Open Source Software
    - Communication stack, compression libraries, image libraries
- Tis-Interpreter
    - Combination with state-of-the-art fuzzers
    - Open source at https://github.com/TrustInSoft/tis-interpreter

Innovation at TrustInSoft

TrustInSoft Analyzer for C++

Derived from CEA STANCE prototype

All C++11 language including lambdas

C++11 STL support

C++14 support

Support for ACSL++

Innovation at TrustInSoft: research projects

ANR ANASTASEC – Cyber-security in avionics

FUI INGOPCS – Secured Industrial Communication Stack

DGE S3P – Smart Safe and Secure Platform for IoT

FUI SecureOCaml – Securing OCaml programs

RAPID AUROCHS – Code Analysis of Cryptographic Stack

ANR VOCAL – Certified OCaml libraries

Any questions?

benjamin.monate@trust-in-soft.com

TRUST IN SOFT
Mathematical Guarantees Eliminate Software Risk

222 avenue du Maine 75014, Paris, France
Suite 231, 2415 Third Street, San Francisco, CA 94107

BN_consttime_swap from OpenSSL: constant time

```
void BN_consttime_swap(BN_ULONG condition, BIGNUM *a,
BIGNUM *b, int nwords) {
    BN_ULONG t;
    int i;

    bn_wcheck_size(a, nwords);
    bn_wcheck_size(b, nwords);

    assert(a != b);
    assert((condition & (condition - 1)) == 0);
    assert(sizeof(BN_ULONG) >= sizeof(int));

    condition = ((condition - 1) >> (BN_BITS2 - 1)) - 1;

    t = (a->top ^ b->top) & condition;
    a->top ^= t;
    b->top ^= t;

#define BN_CONSTTIME_SWAP(ind) \
    do { \
        t = (a->d[ind] ^ b->d[ind]) & condition; \
        a->d[ind] ^= t; \
        b->d[ind] ^= t; \
    } while (0)
```

```
    switch (nwords) {
    default:
        for (i = 10; i < nwords; i++)
            BN_CONSTTIME_SWAP(i);
        /* Fallthrough */
    case 10:
        BN_CONSTTIME_SWAP(9);    /* Fallthrough */
    case 9:
        BN_CONSTTIME_SWAP(8);    /* Fallthrough */
    case 8:
        BN_CONSTTIME_SWA
    case 7:
        BN_CONSTTIME_SWA
    case 6:
        BN_CONSTTIME_SWA
    case 5:
        BN_CONSTTIME_SWAP(4);    /* Fallthrough */
    case 4:
        BN_CONSTTIME_SWAP(3);    /* Fallthrough */
    case 3:
        BN_CONSTTIME_SWAP(2);    /* Fallthrough */
    case 2:
        BN_CONSTTIME_SWAP(1);    /* Fallthrough */
    case 1:
        BN_CONSTTIME_SWAP(0);
    }
#undef BN_CONSTTIME_SWAP
}
```

We automatically confirm there are no problems in the `BN_consttime_swap` function

Full source code: https://github.com/openssl/openssl/blob/d59c7c81e3850dc667d61047850c3b6936eb5fca/crypto/bn/bn_lib.c#L808

s2n_verify_cbc from S2N: constant time?

```
int s2n_verify_cbc(struct s2n_connection *conn,
                   struct s2n_hmac_state *hmac,
                   struct s2n_blob *decrypted)
{

    // . . . .

    int cutoff = check - padding_length;
    for (int i = 0, j = decrypted->size - 1 - check;
            i < check && j < decrypted->size; i++, j++) {
        uint8_t mask = ~(0xff << ((i >= cutoff) * 8));
        mismatches |= (decrypted->data[j] ^ padding_length) & mask;
    }
    if (mismatches) {
        S2N_ERROR(S2N_ERR_CBC_VERIFY);
    }
    return 0;
}
```

This function has been written to compute if there is a padding problem in the variable `mismatches` and to return this value.

When not in debug mode the S2N_ERROR function returns simply the error code. A possible solution
`return (!!mismatches) * S2N_ERR_CBC_VERIFY;`

Error detected

Full source code: https://github.com/awslabs/s2n/blob/e5cadd53a85ac27976d5bdb7ba1501945675c5de/tls/s2n_cbc.c#L48

Detected undefined behaviors

- Division by zero
- Memory Accesses
  - Memory access
  - Index out of bound
    - Memory problem
    - Overflow in array accesses
- Valid string
- Invalid shift
- Pointer comparison
- Differing blocks
- Overflow
- Float to integer
- Not separated
- Overlap
- Dangling and Uninitialized pointers
  - Initialization
  - Dangling
- Is nan or infinite
- Function type matches

Other possible analyses

- Exhaustive detection of all undefined behaviors of the program
- Functional dependency analyses
- Control and data flow analyses
- Shared variable detection
- Race condition detection
- Memory leak
- Proof of functional properties
- Command line and graphical interface

Dedicated all-included support

- All technical questions answered
- Tool Expertise and Customization
- Review of your integration processes and normative requirements