TRUST IN SOFT

Mathematical Guarantees Eliminate Software Risk

Using Formal Methods on Real-World Software

Fabrice Derepas

Co-founder & CEO

OSSIR, November 10th, 2015.

About TrustInSoft

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

IRSN	Selected by the IRSN (Nuclear Autority) to check the safety of programs embedded in nuclear reactors
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
RSAConference	Nominated as one the 10 most innovative companies in cybersecurity – RSA '15 Conference

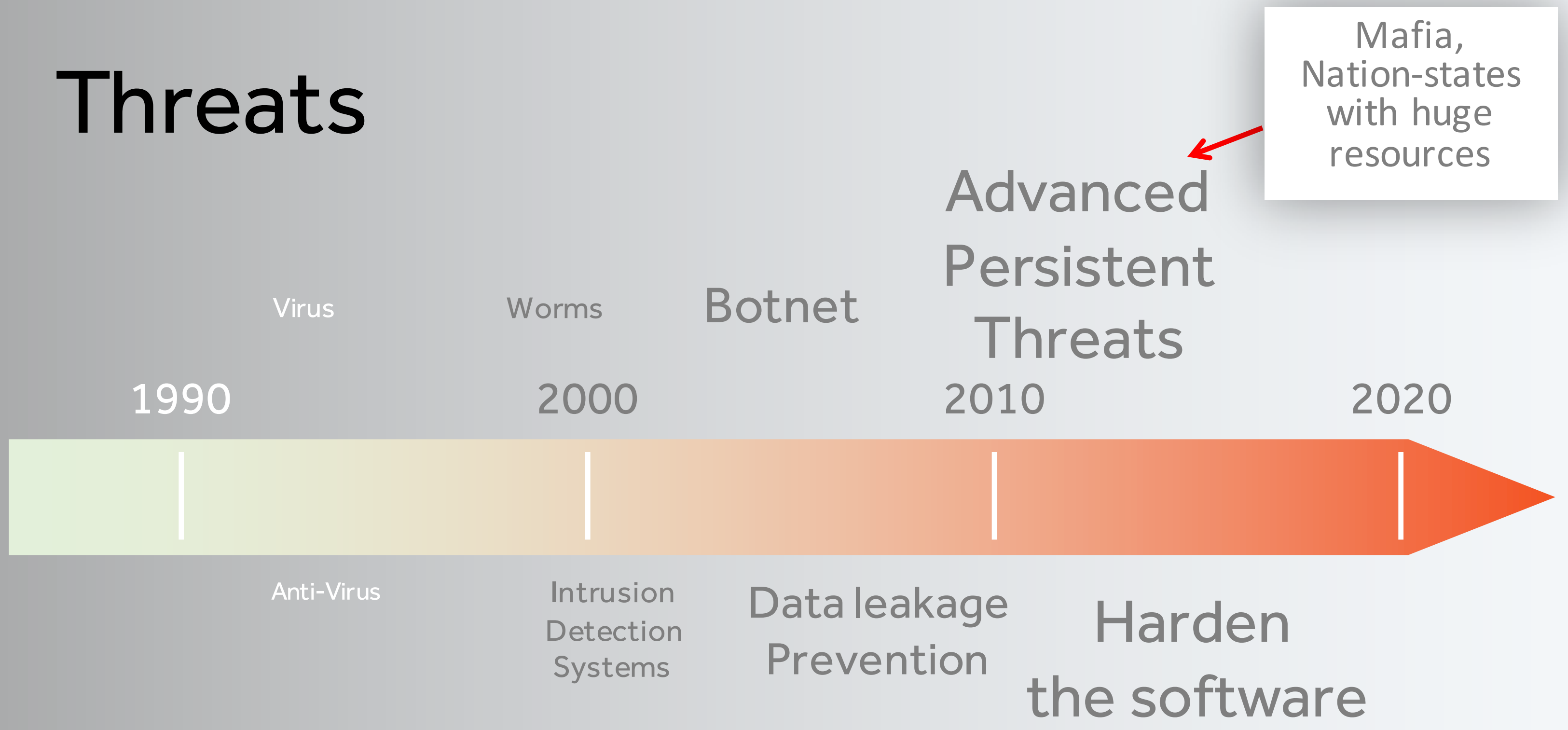
TrustInSoft Unique Value Proposal

Sell guarantees on software used in sensitive systems

Pain Points In Cyber Security

It's the right moment now!

Threats



Virus

Worms

Botnet

Advanced Persistent Threats

Mafia, Nation-states with huge resources

1990

2000

2010

2020

Anti-Virus

Intrusion Detection Systems

Data leakage Prevention

Harden the software

TRUST IN SOFT

Solutions

Standard market practice

FORTIFY
An HP Company

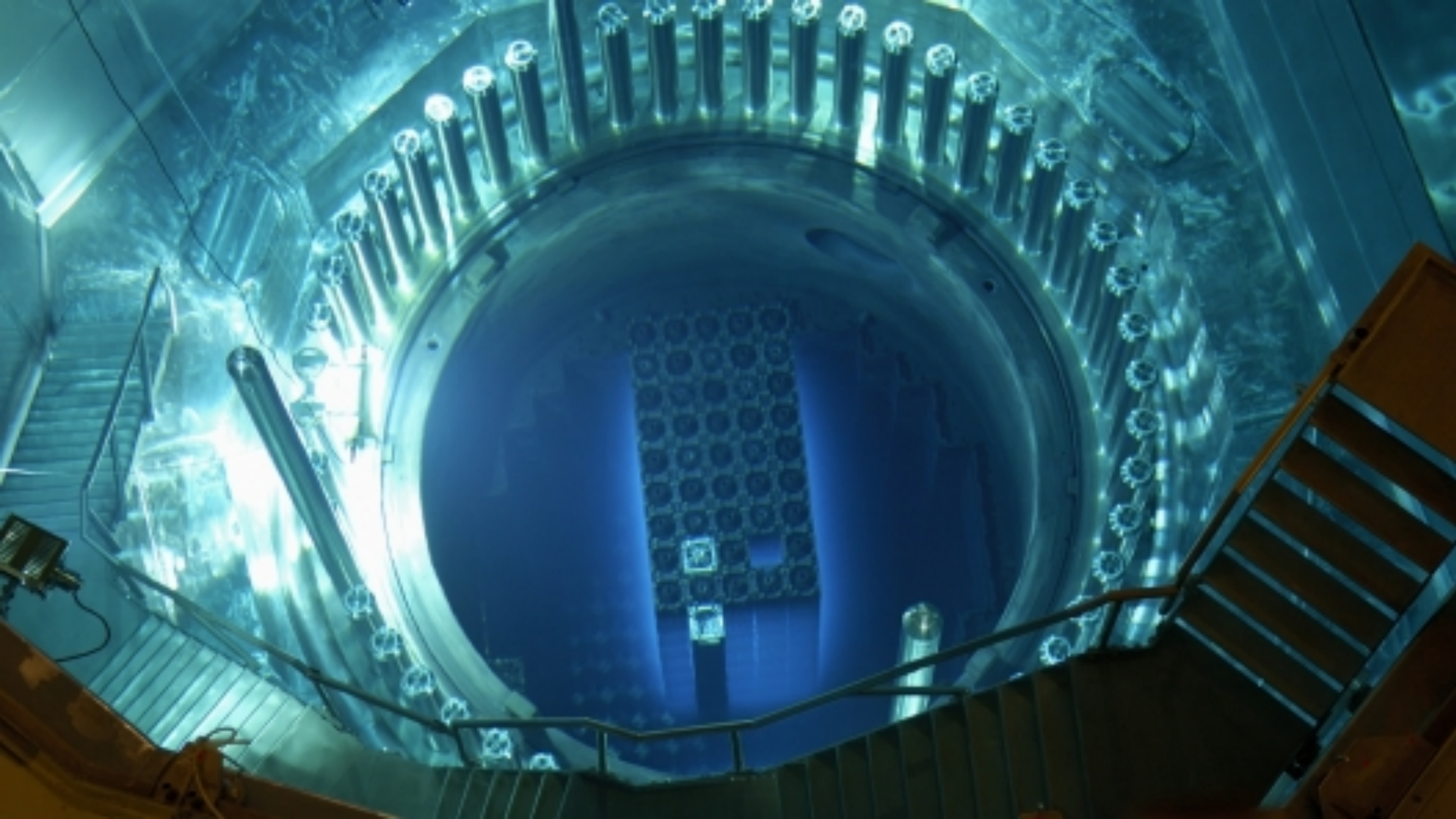
coverity
A Synopsys Company

klocwork
a Rogue Wave Company

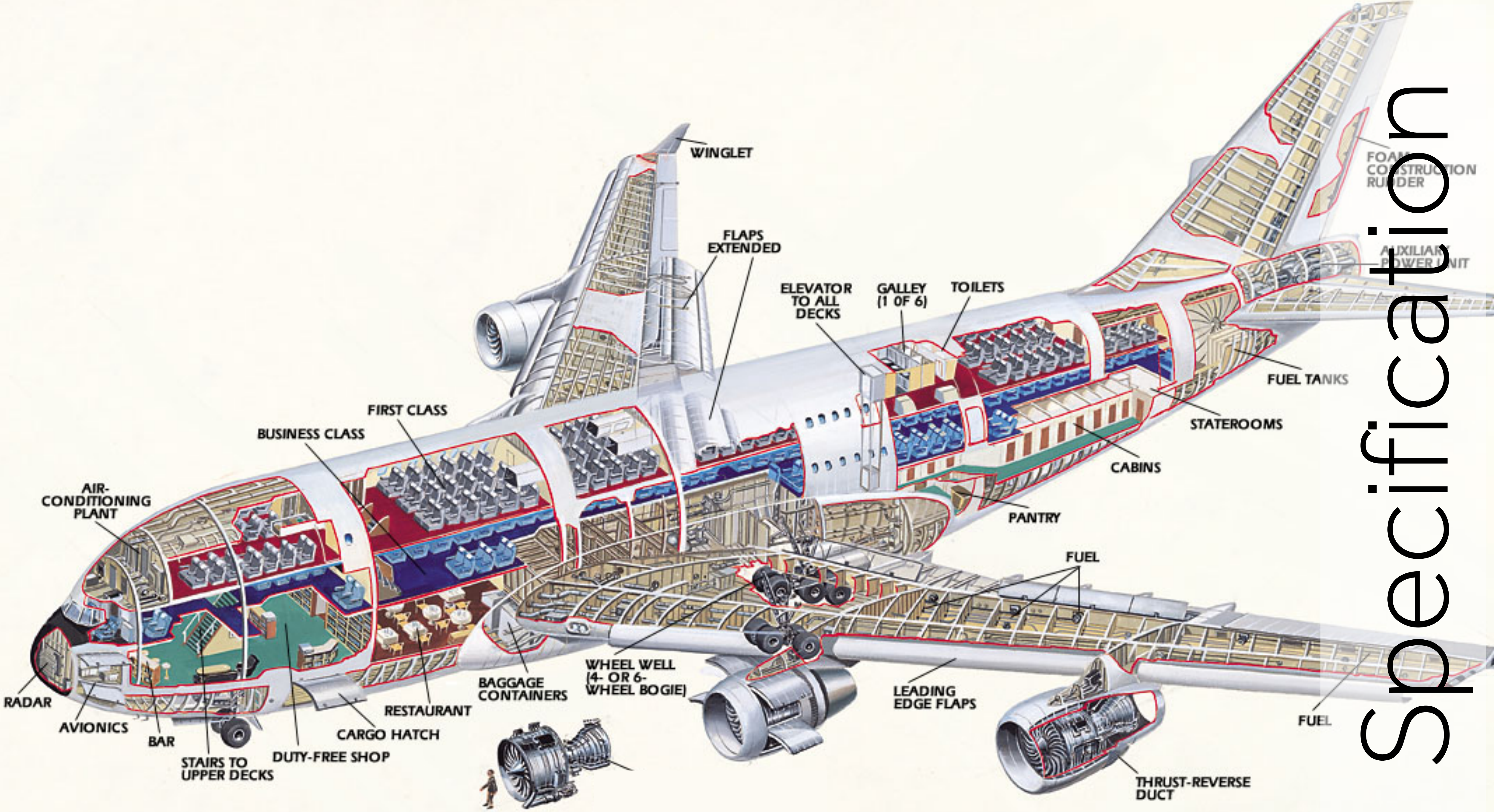
Best Effort

but no guarantees

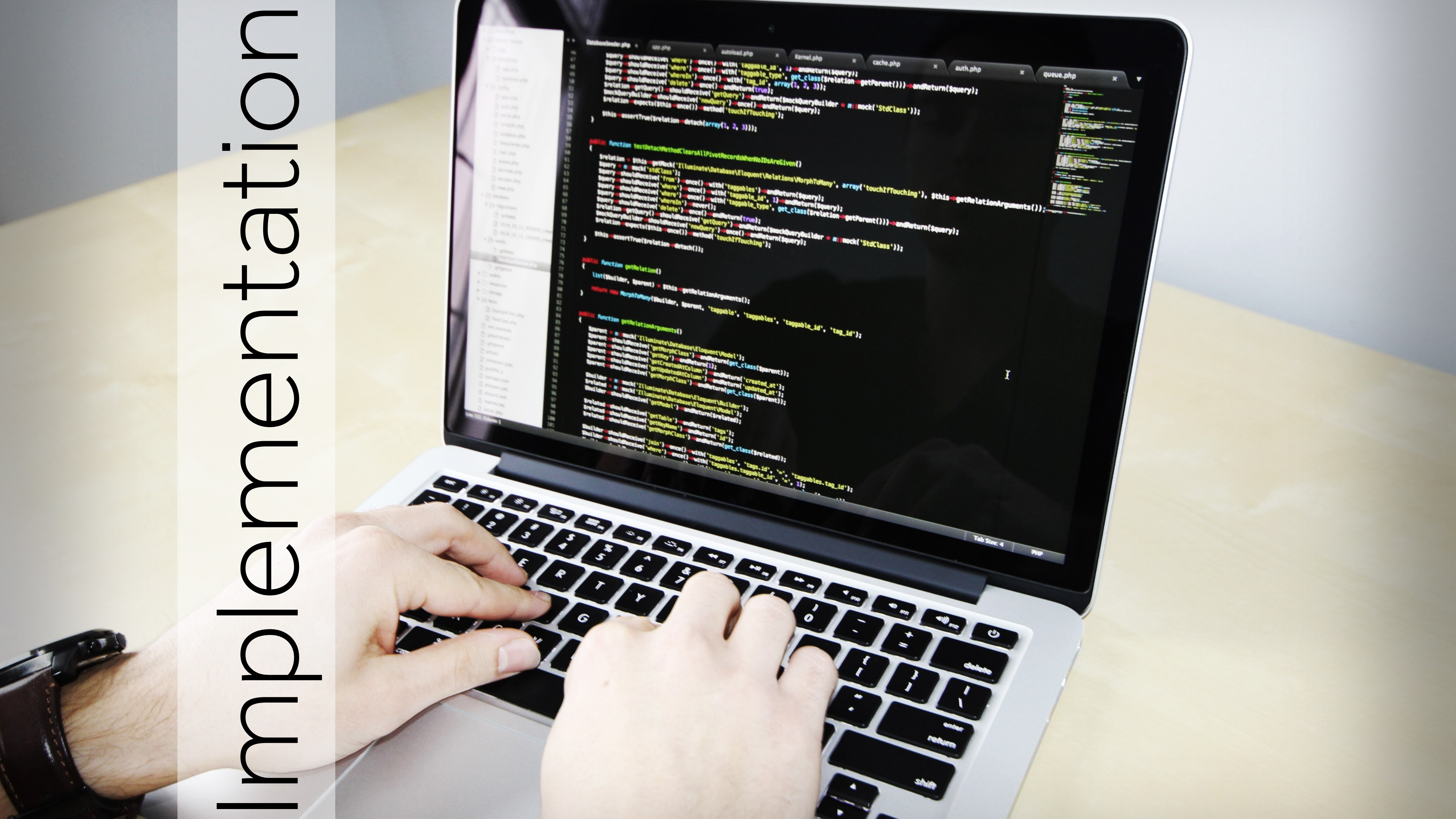
NIST
National Institute of Standards and Technology
U.S. Department of Commerce



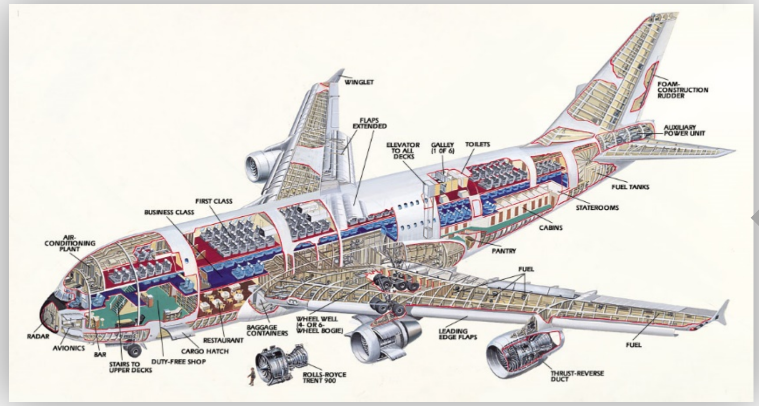
Specification

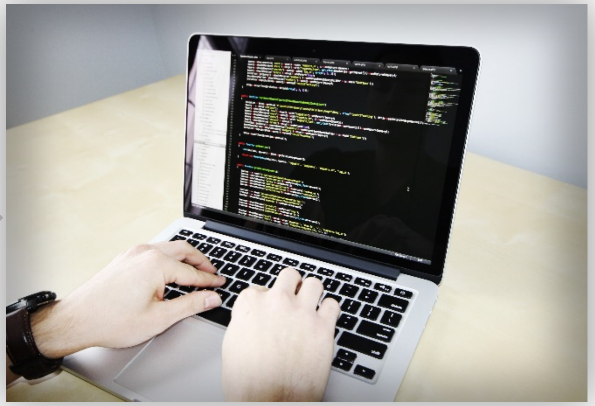


Implementation



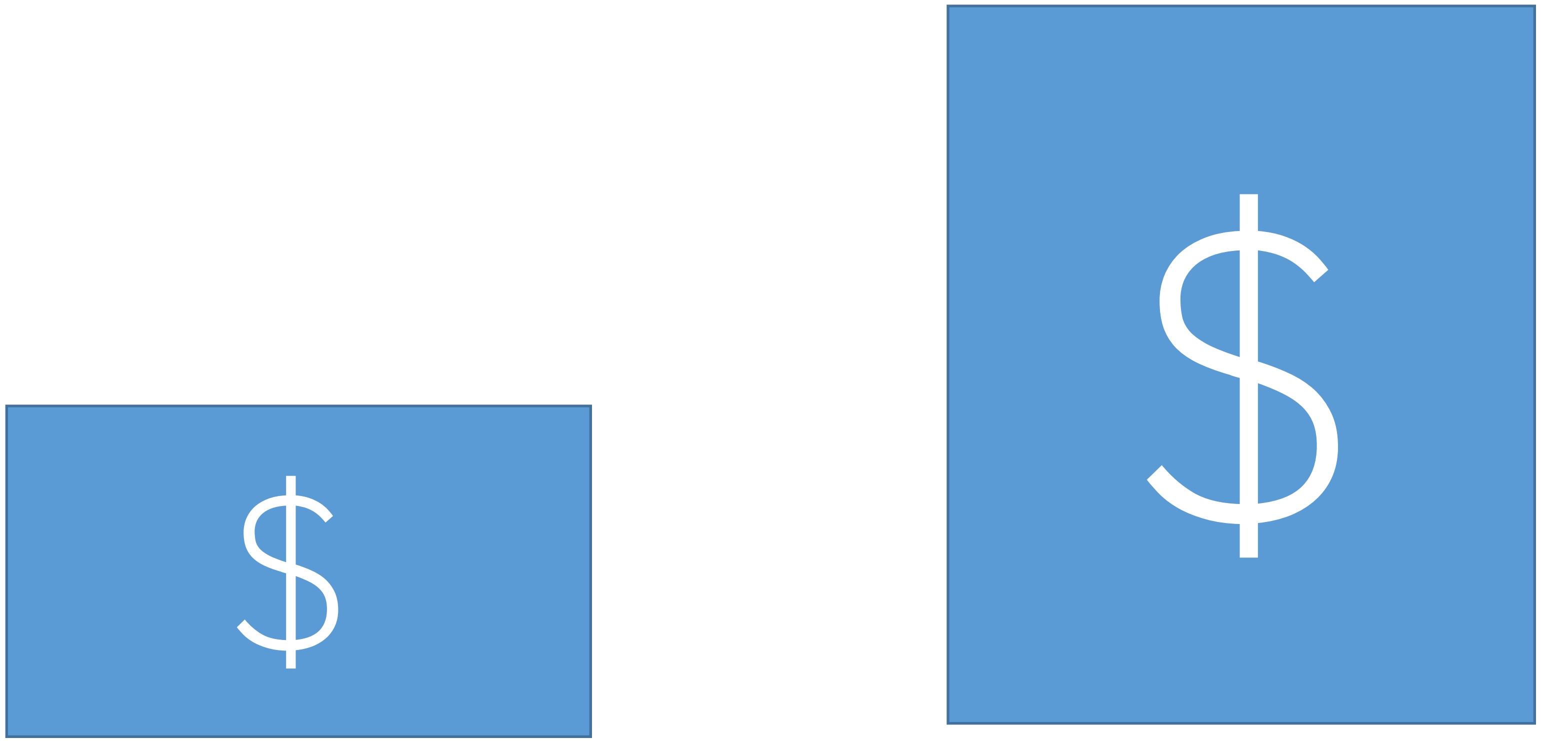
Specification





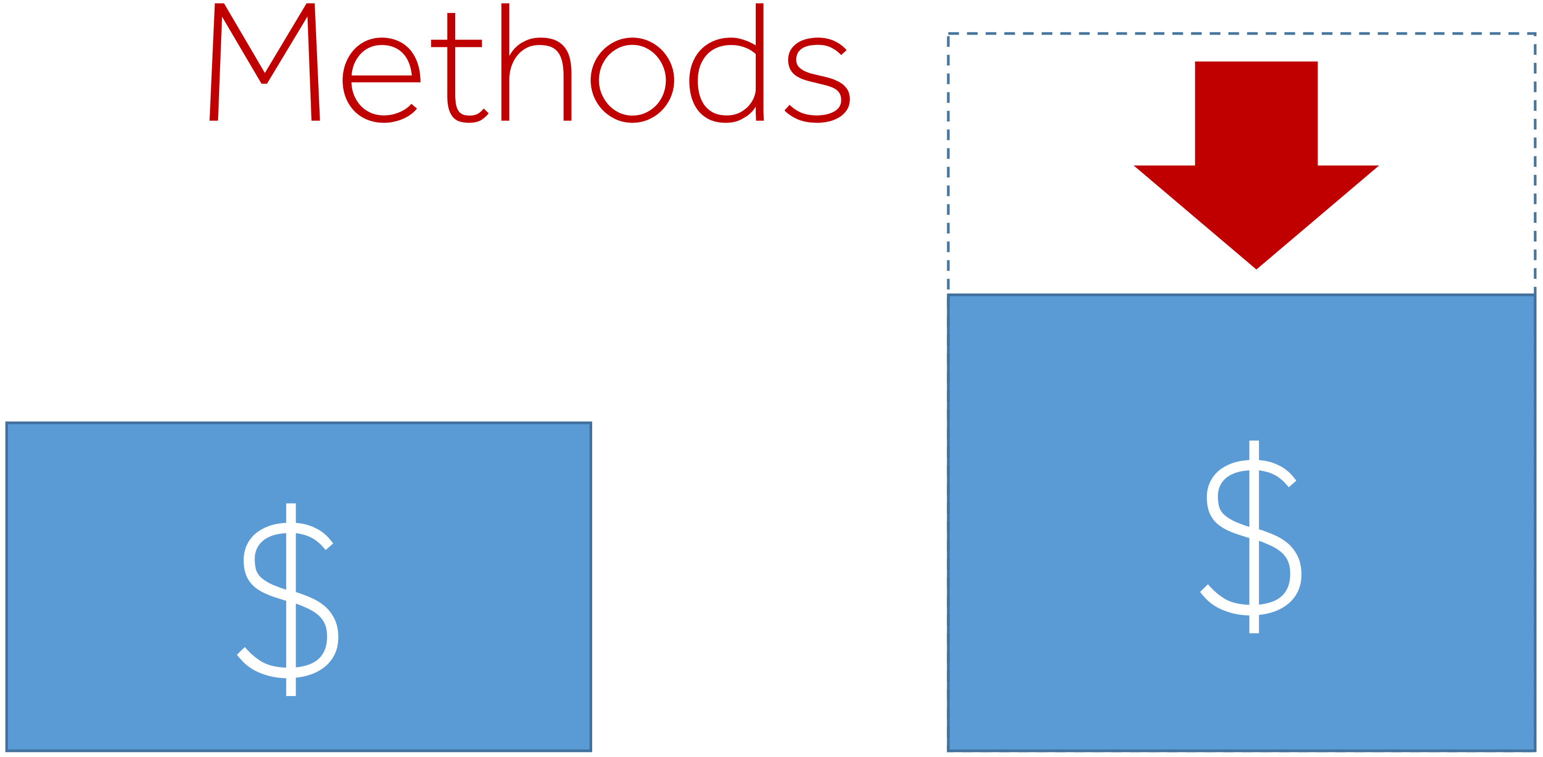
must check the two are in sync

Implementation

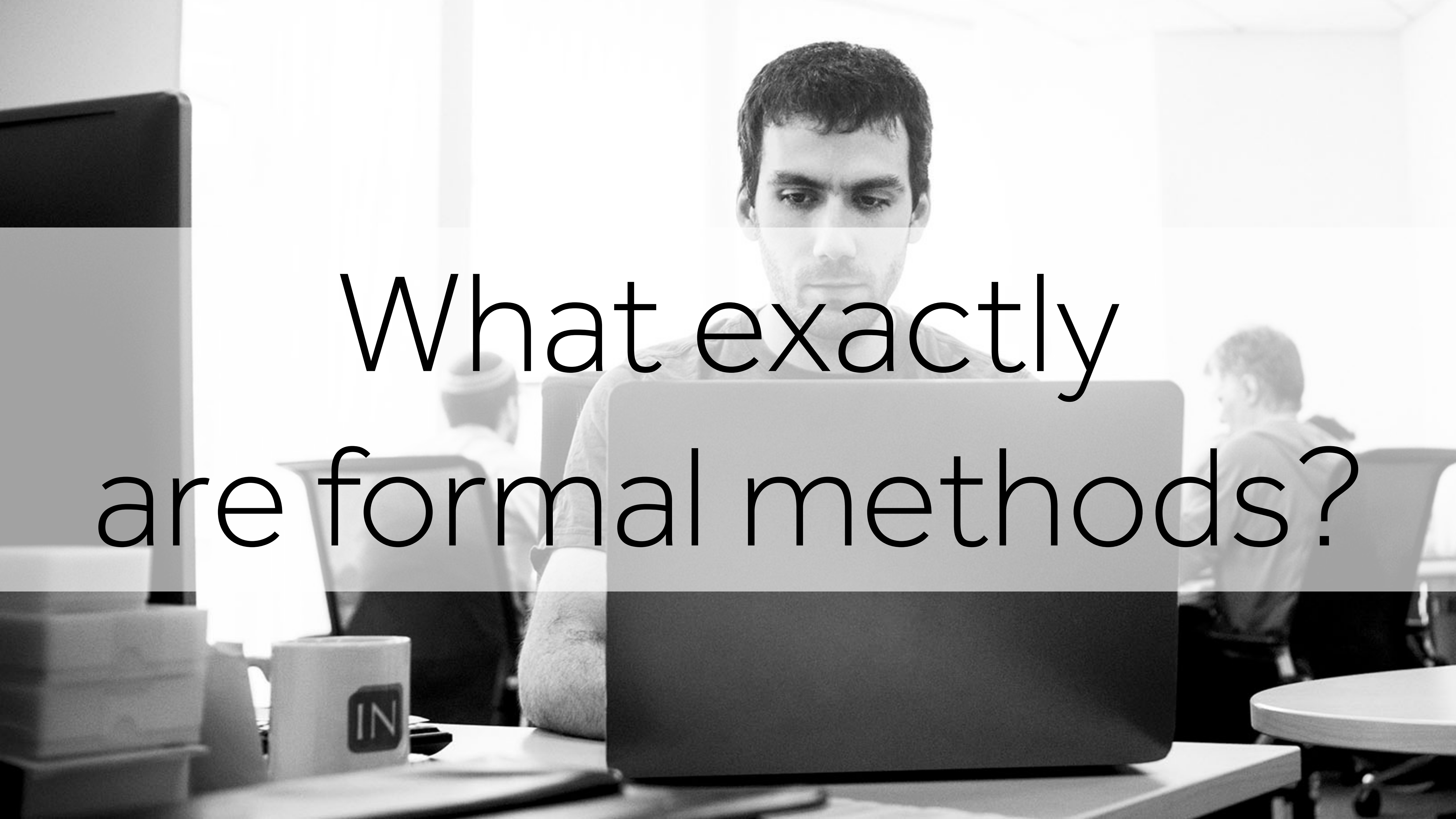


Conception Verification

Formal Methods



Conception Verification



What exactly are formal methods?

$$(a+b)^2=a^2+b^2+2ab$$

is this true?

$$(a+b)^2=a^2+b^2+2ab$$

Idea 1: let’s test for many values of « a » and « b »

$$(a+b)^2=a^2+b^2+2ab$$

Idea 2: let’s perform an algebraic proof

$$(a+b)^2=(a+b)x(a+b)$$

$$= a^2+ab + b^2+ba$$

$$=a^2+b^2+2ab$$

```
int max (int x, int y) {
  if (x>y) return x; else return y;
}
```

software

logic

```
/*@ ensures \result >= x &&
            \result >= y;
    ensures \result == x ||
            \result == y;
*/
```

```
int max (int x, int y) {
  if (x>y) return x; else return y;
}
```

software

An example of application on real-world code

ARM mbed First ever SSL stack *guaranteed* ***without buffer overflows.***

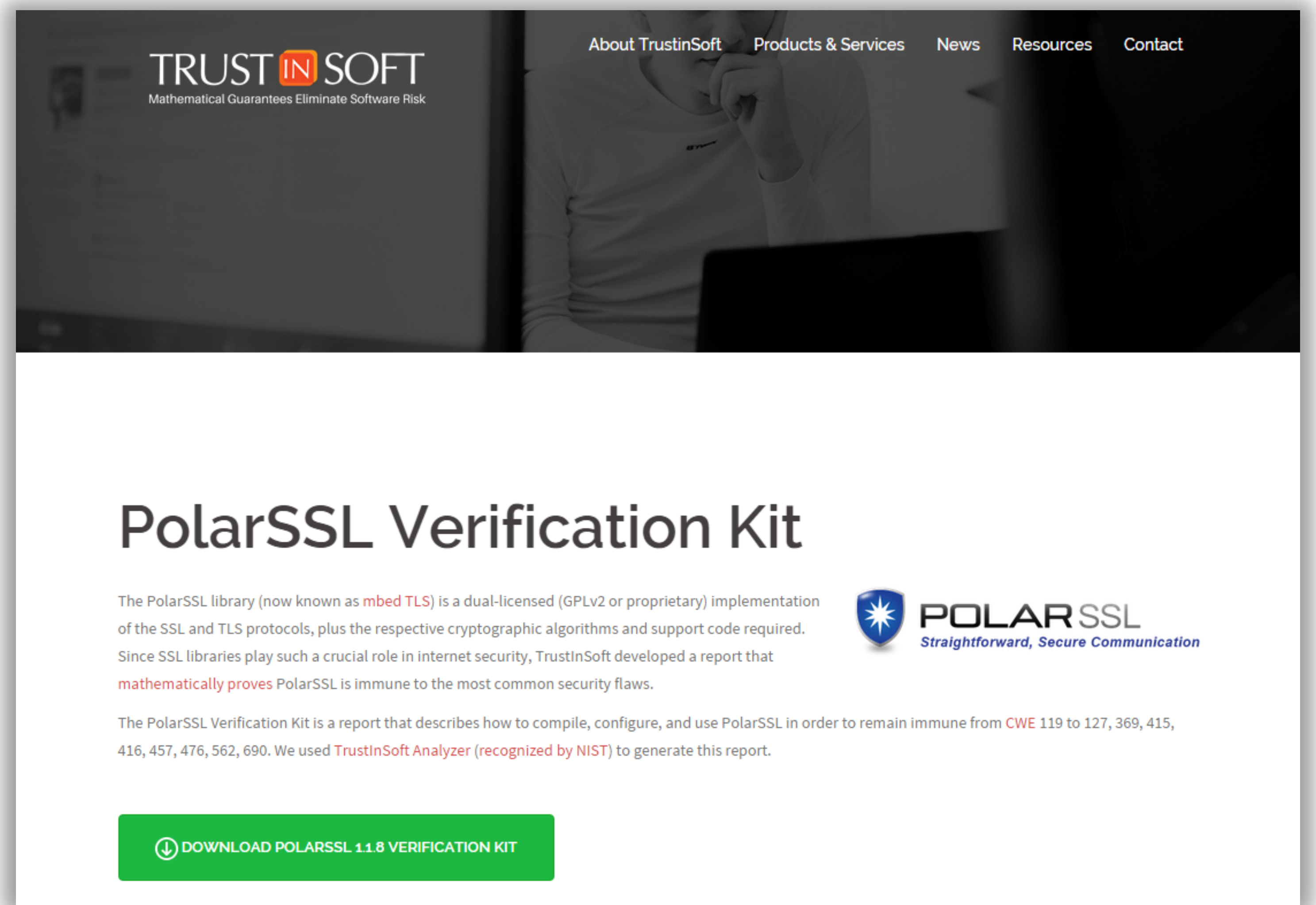
Using **TrustInSoft Analyzer** we have generated a report which tells how to **compile**, **configure** and **deploy** mbed TLS in a given perimeter in order to be immune from all attacks caused by CWE 119 to 127, 369, 415, 416, 457, 476, 562, 690.

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

In this case the specification is "the stack will never crash".

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

http://trust-in-soft.com/polarssl-verification-kit


TRUST IN SOFT
Mathematical Guarantees Eliminate Software Risk
About TrustinSoft
Products & Services
News
Resources
Contact
POLAR SSL
Straightforward, Secure Communication

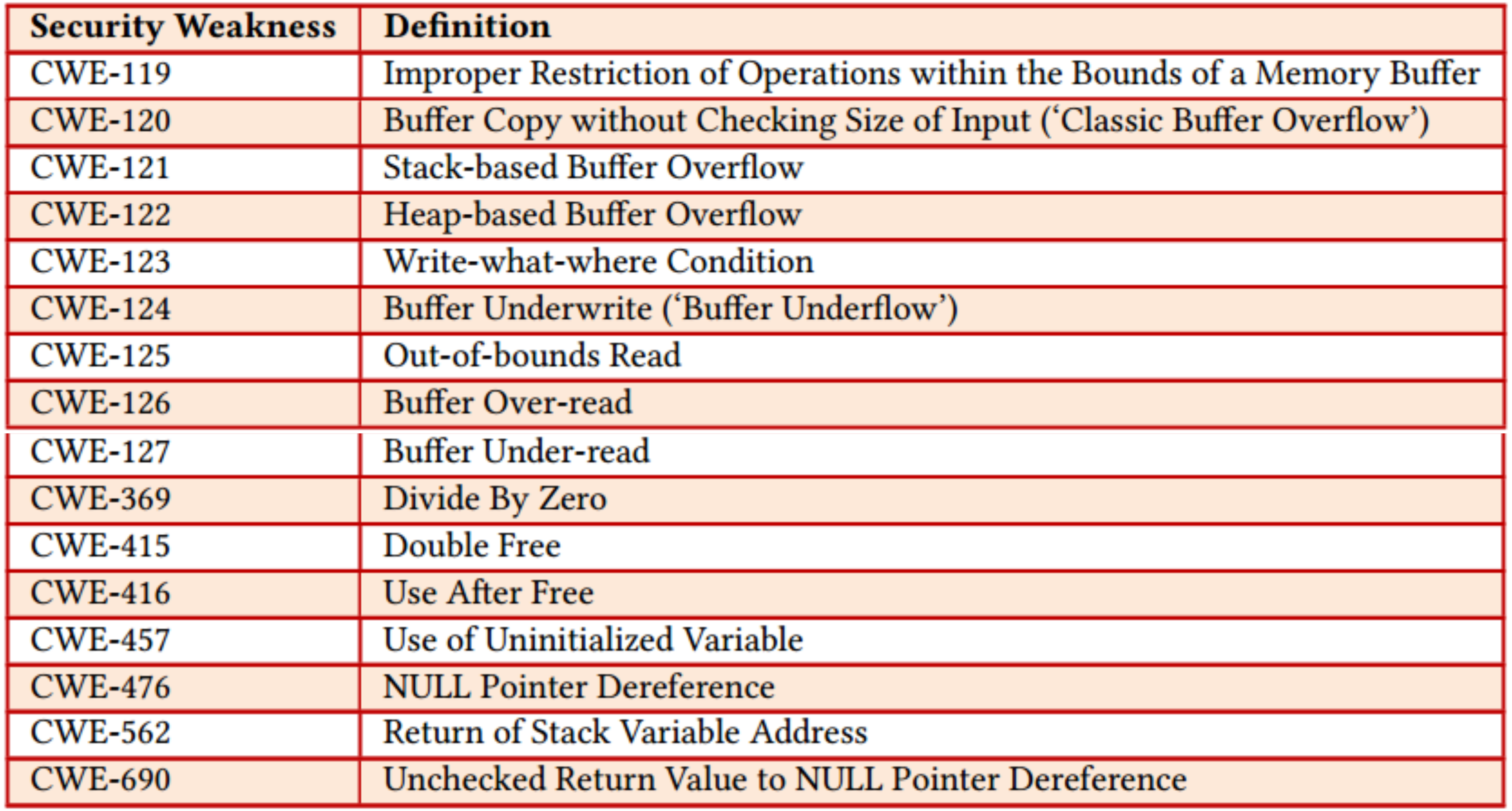
PolarSSL Verification Kit

The PolarSSL library (now known as mbed TLS) is a dual-licensed (GPLv2 or proprietary) implementation of the SSL and TLS protocols, plus the respective cryptographic algorithms and support code required. Since SSL libraries play such a crucial role in internet security, TrustInSoft developed a report that mathematically proves PolarSSL is immune to the most common security flaws.

The PolarSSL Verification Kit is a report that describes how to compile, configure, and use PolarSSL in order to remain immune from CWE 119 to 127, 369, 415, 416, 457, 476, 562, 690. We used TrustInSoft Analyzer (recognized by NIST) to generate this report.

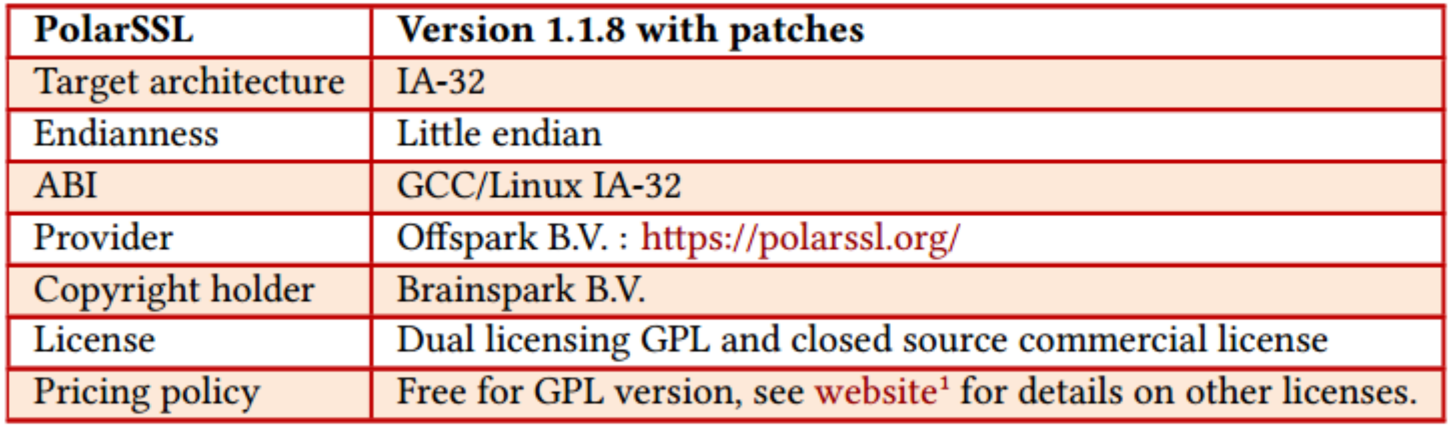
DOWNLOAD POLARSSL 1.1.8 VERIFICATION KIT

CWE list



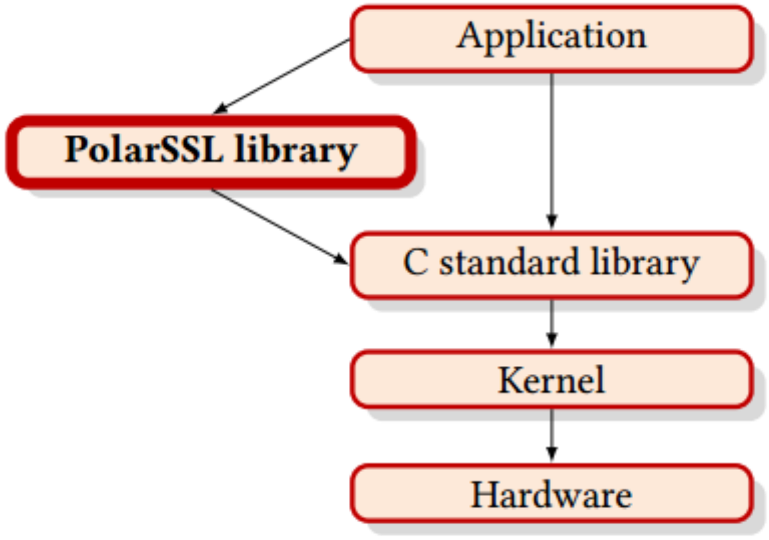
Security Weakness	Definition
CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer
CWE-120	Buffer Copy without Checking Size of Input (‘Classic Buffer Overflow’)
CWE-121	Stack-based Buffer Overflow
CWE-122	Heap-based Buffer Overflow
CWE-123	Write-what-where Condition
CWE-124	Buffer Underwrite (‘Buffer Underflow’)
CWE-125	Out-of-bounds Read
CWE-126	Buffer Over-read
CWE-127	Buffer Under-read
CWE-369	Divide By Zero
CWE-415	Double Free
CWE-416	Use After Free
CWE-457	Use of Uninitialized Variable
CWE-476	NULL Pointer Dereference
CWE-562	Return of Stack Variable Address
CWE-690	Unchecked Return Value to NULL Pointer Dereference

Target architecture

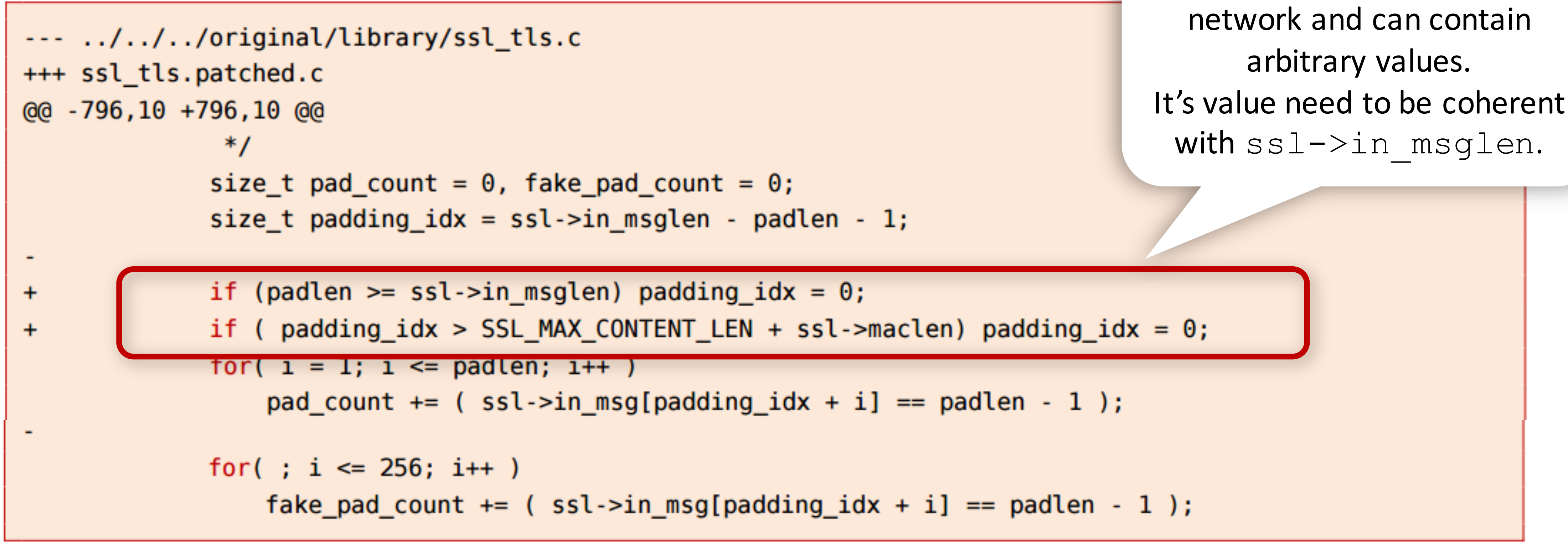


PolarSSL	**Version 1.1.8 with patches**
Target architecture	IA-32
Endianness	Little endian
ABI	GCC/Linux IA-32
Provider	Offspark B.V. : https://polarssl.org/
Copyright holder	Brainspark B.V.
License	Dual licensing GPL and closed source commercial license
Pricing policy	Free for GPL version, see website[1] for details on other licenses.

Trusted Computing Base



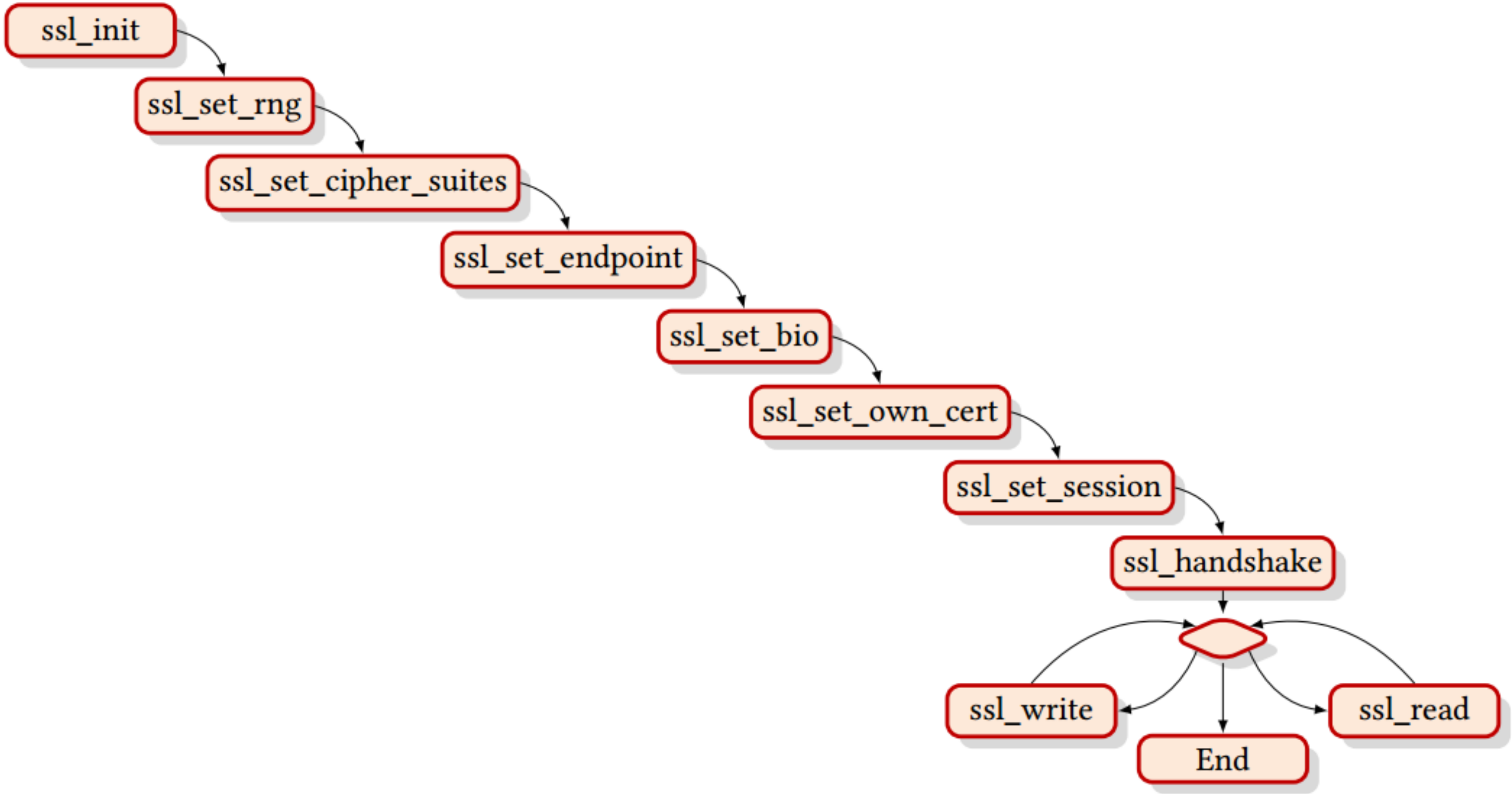
Example of an applied patch



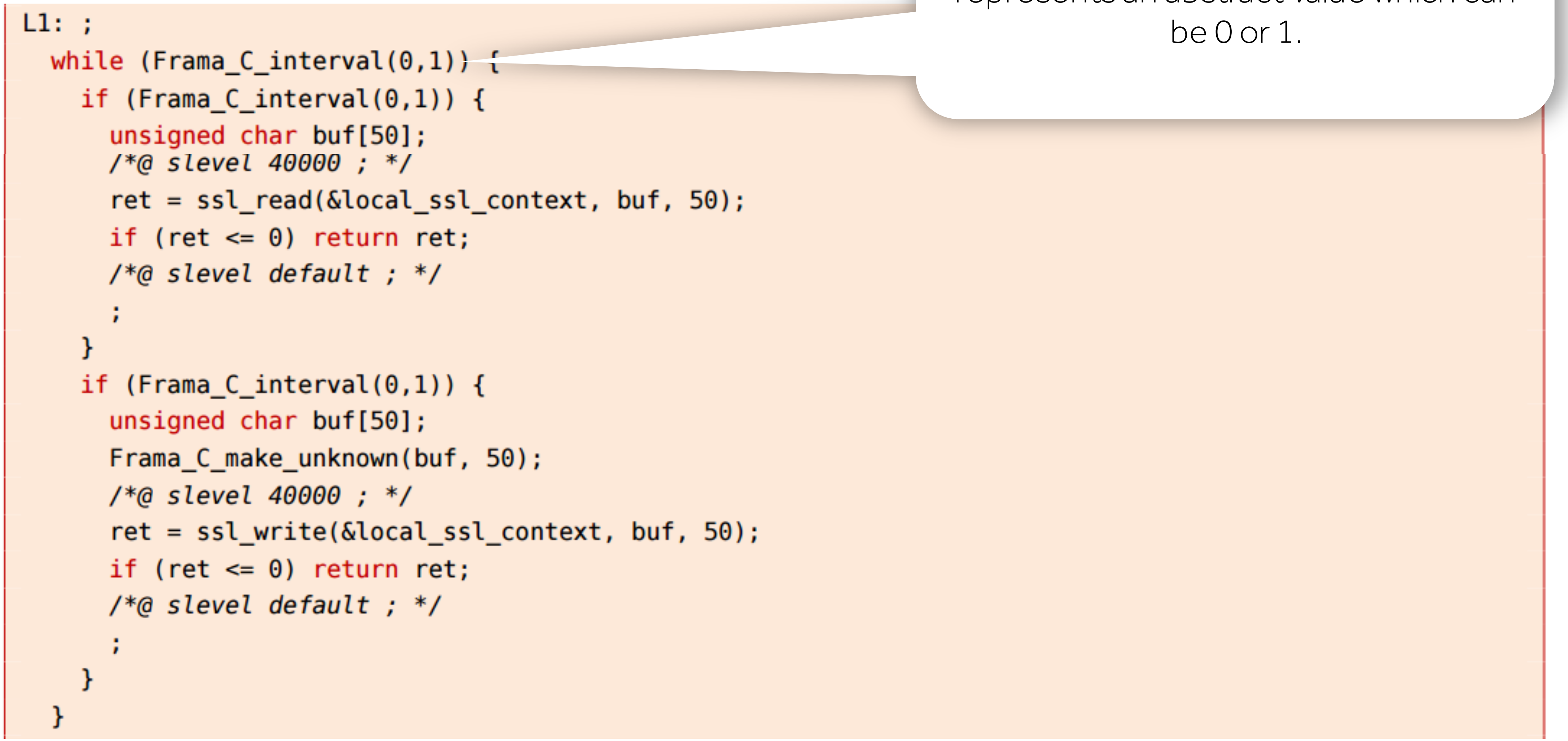
```
--- ../../../original/library/ssl_tls.c
+++ ssl_tls.patched.c
@@ -796,10 +796,10 @@
              */
             size_t pad_count = 0, fake_pad_count = 0;
             size_t padding_idx = ssl->in_msglen - padlen - 1;
-
+            if (padlen >= ssl->in_msglen) padding_idx = 0;
+            if ( padding_idx > SSL_MAX_CONTENT_LEN + ssl->maclen) padding_idx = 0;
             for( i = 1; i <= padlen; i++ )
                 pad_count += ( ssl->in_msg[padding_idx + i] == padlen - 1 );
-
             for( ; i <= 256; i++ )
                 fake_pad_count += ( ssl->in_msg[padding_idx + i] == padlen - 1 );
```

`padlen` is read from the network and can contain arbitrary values.
It's value need to be coherent with `ssl->in_msglen`.

Server usage pattern



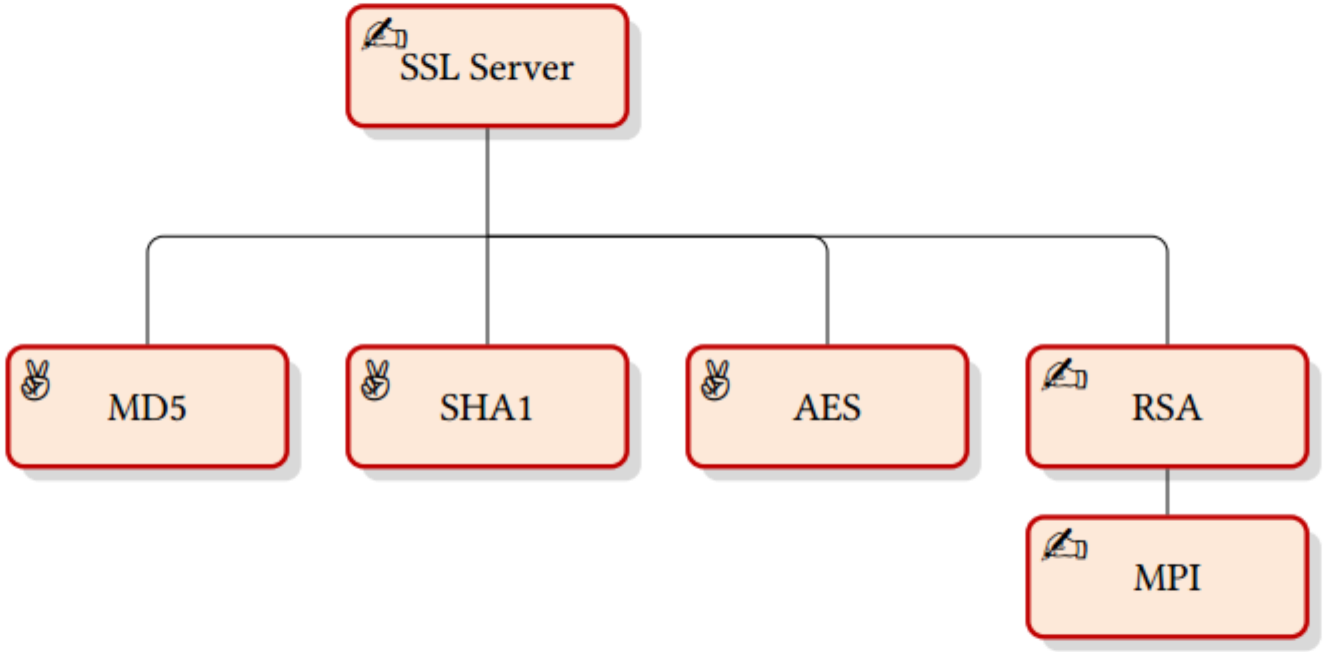
C-implementation



```
L1: ;
  while (Frama_C_interval(0,1)) {
    if (Frama_C_interval(0,1)) {
      unsigned char buf[50];
      /*@ slevel 40000 ; */
      ret = ssl_read(&local_ssl_context, buf, 50);
      if (ret <= 0) return ret;
      /*@ slevel default ; */
      ;
    }
    if (Frama_C_interval(0,1)) {
      unsigned char buf[50];
      Frama_C_make_unknown(buf, 50);
      /*@ slevel 40000 ; */
      ret = ssl_write(&local_ssl_context, buf, 50);
      if (ret <= 0) return ret;
      /*@ slevel default ; */
      ;
    }
  }
```

`Frama_C_interval (0,1)` represents an abstract value which can be 0 or 1.

Verification architecture



✍ SSL Server

✌ MD5

✌ SHA1

✌ AES

✍ RSA

✍ MPI

✌ formal trust: security property formally verified.
✍ semi-formal trust: everything reviewed.

Virtual machine example

This is the code of a virtual machine which computes 2^4

```
#define ARRAY_SIZE 11
unsigned char mem[ARRAY_SIZE]= \
  {80,7,5,5,3,5,3,5,4,11,2};
#define NEXT \
  if (pos<ARRAY_SIZE-1) ++pos;
break;

int main () {
  unsigned int A=0,B=0,pos=0;
  pos=0;
  while (1) {
    switch (mem[pos] & 7) {
    // add
    case 0: A+=mem[pos]>>3; NEXT;
    // substract
    case 1: A-=mem[pos]>>3; NEXT;
    // load
    case 2: A=mem[B]; NEXT;
    //store
    case 3: mem[B]=A; NEXT;
    // exit
    case 4: return A;
    // load and add
    case 5: if (B<ARRAY_SIZE)
        A=A+mem[B]; NEXT;
    // goto A
    case 6: if (A<ARRAY_SIZE)
        pos=A; break;
    // swap A and B
    case 7: {int tmp=B;B=A;A=tmp;}
        NEXT;
} } }
```

value of B not checked

All virtual machines with memory size of 11

```
#define ARRAY_SIZE 11
unsigned char mem[ARRAY_SIZE] = {80,7,5,5,3,5,3,5,4,11,2};
#define NEXT if (pos<ARRAY_SIZE-1) ++pos;\
  break;

int main () {
  unsigned int A=0,B=0,pos=0;
  while (1) {
    // . . .
```

Here is the program
for a given state of the virtual
machine
This program has no error

```
#define ARRAY_SIZE 11
unsigned char mem[ARRAY_SIZE];
#define NEXT \
  if (pos<ARRAY_SIZE-1) ++pos; break;

int main () {
  unsigned int A=0,B=0,pos=0;
  for (pos=0;pos<ARRAY_SIZE;++pos) mem[pos]=Frama_C_interval(0, 255);
  pos=0;
  while (1) {
    // . . .
```

TrustInSoft Analyzer
tests all possible virtual machine
of size 11.
256^{11} tests.
In a single run.

Symbolic value: all integers
between 0 and 255

Sectors using these techniques

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
- IT
  CWE

Customer names are under strict NDAs

Coq

Ocaml

CEA

logic

INRIA

Why

AltErgo

Frama-C

Airbus

Areva

software

EDF

IRSN

Dassault

Renault

Why You Should Care

Two possible approaches

- Detect threats
- Reduce attack surface

there is no anti-virus
in the airplane

```
// declare a table of size 100
int table[100];
// assign cell 101 with value
// from network
table[101]=43;
```

Specification

Specification

```
// declare a table of size 100
int table[100];
// assign cell 101 with value
// from network
table[101]=43;
```

the two are not in sync!!!

Implementation

NIST

National Institute of
Standards and Technology
U.S. Department of Commerce

Static Analysis Tool Exposition and the Ockham Soundness Criteria

What about Open source?

what about open source?

AUTOMOTIVE GRADE LINUX

Open Automotive Alliance

OpenXC

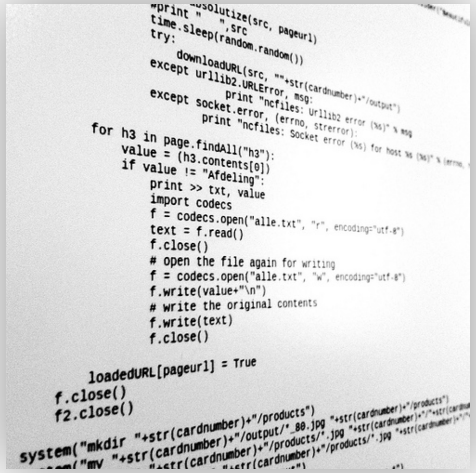
LOCAL MOTORS

GENIVI

idea
of the
program

source
code

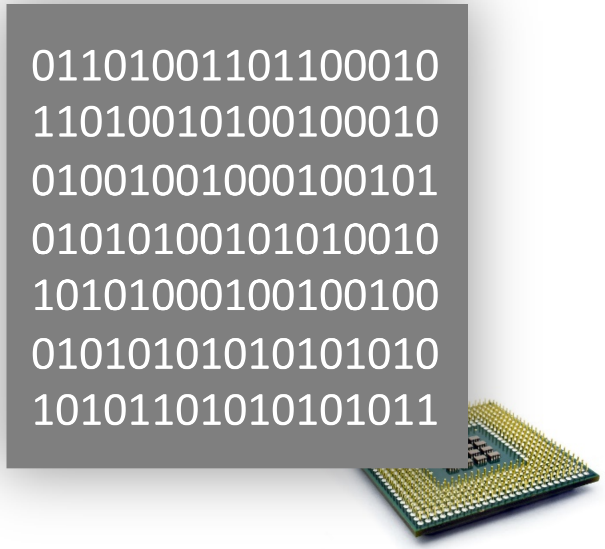
binary
code



```
#print "   ",src
time.sleep(random.random())
try:
    downloadURL(src, ""+str(cardnumber)+"/output")
except urllib2.URLError, msg:
        print "ncfiles: Urllib2 error (%s)" % msg
except socket.error, (errno, strerror):
      print "ncfiles: Socket error (%s) for host %s (%s)" % (errno,
for h3 in page.findAll("h3"):
    value = (h3.contents[0])
    if value != "Afdeling":
        print >> txt, value
        import codecs
        f = codecs.open("alle.txt", "r", encoding="utf-8")
        text = f.read()
        f.close()
        # open the file again for writing
        f = codecs.open("alle.txt", "w", encoding="utf-8")
        f.write(value+"\n")
        # write the original contents
        f.write(text)
        f.close()

  loadedURL[pageurl] = True
f.close()
f2.close()

system("mkdir "+str(cardnumber)+"/products")
        "+str(cardnumber)+"/output/*_80.jpg "+str(cardnumber)+"/products")
        "+str(cardnumber)+"/products/*.jpg "+str(cardnumber)+"/"+str(card
        "+str(cardnumber)+"/products/*.jpg "+str(cardnumber)+"/
```



01101001101100010
11010010100100010
01001001000100101
01010100101010010
10101000100100100
01010101010101010
10101101010101011

Photo credit: Lightspring/Shutterstock

free

as in freedom

free

as in free beer

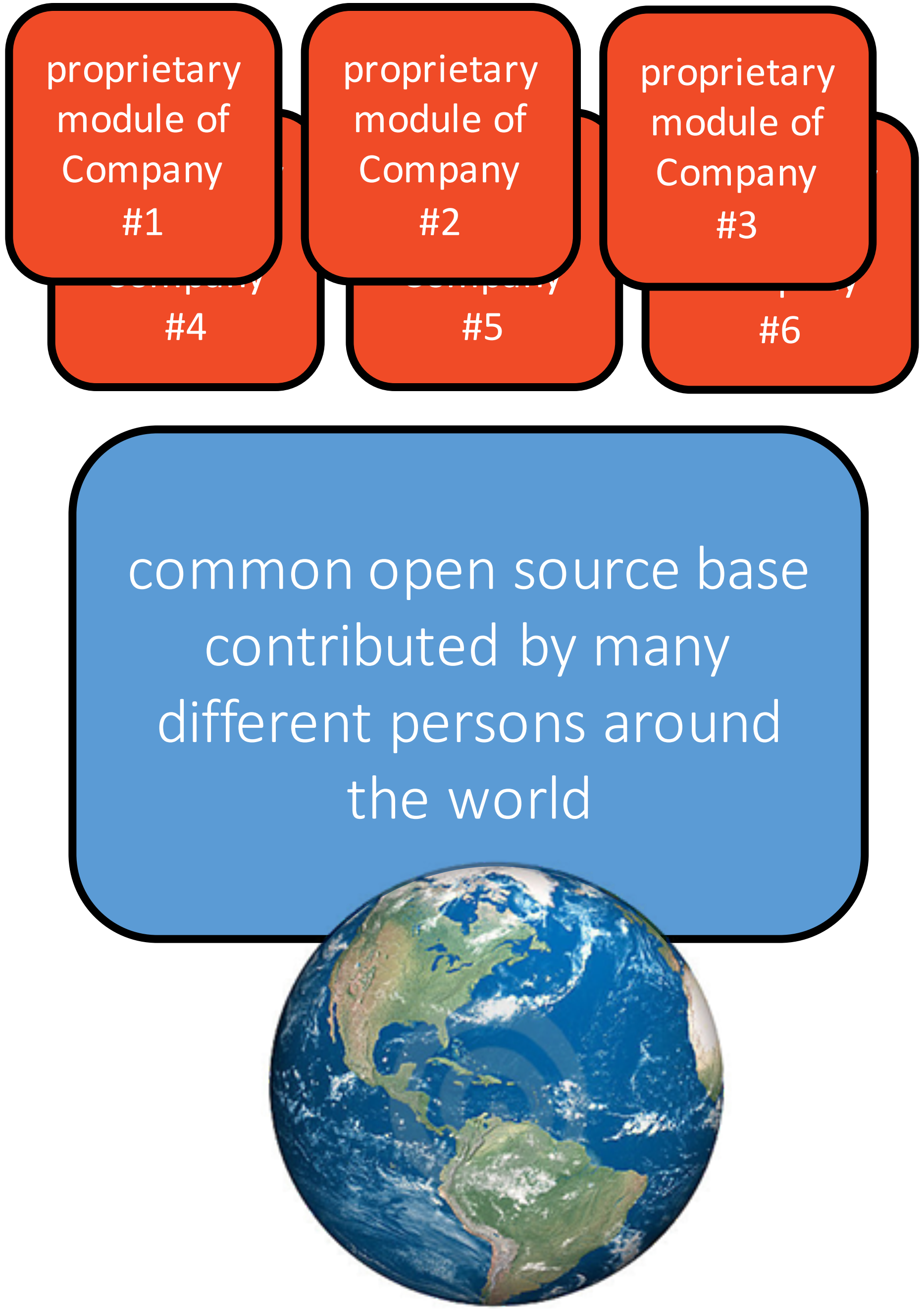
Example of Bosch Free and Open Source Software for GM

GM Cadillac, Chevrolet, GMC, Buick and Opel MY16 HMI Module (SW 15.1A025*)
Color Connected Navigation Head Unit 5.8'' for Chevrolet City Express
GM Cadillac, Chevrolet, GMC, Buick and Opel MY15 HMI Module (SW 14.0F105*)
GM Cadillac, Chevrolet, GMC, Buick and Opel MY15 HMI Module (SW 14.1F013*)
GM Cadillac, Chevrolet, GMC, Buick and Opel MY14 HMI Module (SW 12.6N185*)
GM Cadillac, Chevrolet, GMC, Buick and Opel MY14 HMI Module (SW 12.6N155*)
GM Cadillac, Chevrolet, GMC, Buick and Opel MY14 HMI Module (SW 12.6N146.3*)
GM Cadillac, Chevrolet, GMC, Buick and Opel MY14 HMI Module (SW 12.6N106* to 12.6N109*)
GM Cadillac, Chevrolet, GMC, Buick and Opel MY14 HMI Module (SW 12.6N096*, 12.6N098*)
GM Cadillac, Chevrolet, GMC, Buick and Opel MY14 HMI Module (SW 12.6N057.2* and 12.7N015* to 12.7N025*)
GM Cadillac, Chevrolet, GMC MY13 HMI Module with MY14 SW (SW 12.5Exxx* later than 12.5E040*)
GM Cadillac MY13 HMI Module (SW 12.2Sxxx*): XTS, ATS (Region North America)
GM Cadillac MY13 HMI Module (SW 12.3Sxxx*)
GM Cadillac MY13 HMI Module (SW 12.4Exxx* and 12.5Exxx* up to 12.5E022*): XTS, ATS, SRX (Region China)

Download the source from: http://oss.bosch-cm.com/gm.html

Why open source?

- There are many reasons for using open source software
- One of them is to reducing the costs of widely used on and contributed software components by sharing the development costs.



proprietary module of Company #1

proprietary module of Company #2

proprietary module of Company #3

#4

#5

#6

common open source base contributed by many different persons around the world

Process based
vs.
Product based

CORE INFRASTRUCTURE INITIATIVE

Core Infrastructure Initiative
Fortifying our future.

THE LINUX FOUNDATION

ADAM SHOSTACK
Technologist, Entrepreneur, Author, and Game Designer

ALAN COX
Longtime Linux Kernel Developer

BEN LAURIE
Senior Member of Security Team at Google

BRUCE SCHNEIER
Security Technologist and Author

DAN KAMINSKY
Security Researcher

DAN MEREDITH
Director of Open Technology Fund, Radio Free Asia

EDUARD KAREL DE JONG
Security and Privacy Expert

ERIC SEARS
Program Officer for Human Rights and International Justice

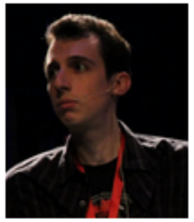
GREG KROAH-HARTMAN
Fellow, Linux Foundation

MATT GREEN
Assistant Research Professor at Johns Hopkins University

MICHAEL HOWARD
Senior Principal Cybersecurity Architect at Microsoft

ROBERT SEACORD
Secure Coding Technical Manager, CERT Division of Carnegie Mellon University's Software Engineering Institute (SEI)

TED TS'O
Staff Engineer at Google


TOM RITTER
Practice Director at NCC Group's Cryptography Services

amazon web services
Bloomberg
CISCO
DELL
facebook
FUJITSU
Google
HITACHI Inspire the Next
hp
HUAWEI
IBM
intel
Microsoft
NEC
NetApp
QUALCOMM
rackspace
salesforce
vmware

Trusting the crowd is nice

Formal Guarantees are definitive

TRUST IN SOFT

Mathematical Guarantees Eliminate Software Risk

contact@trust-in-soft.com

Suite 231
2415 Third Street,
San Francisco
USA

222 av. du Maine
75014 Paris
France

Example of eradicated weaknesses

Standard vulnerabilities:

- Buffer overflow, invalid pointer usage, Division by zero, non initialized memory read, dangling pointer, arithmetic overflow, NaN in a float computation, overflow in float to integer conversion,

Other vulnerabilities:

- CWE-078: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
- CWE-306: Missing Authentication for Critical Function
- CWE-798: Use of Hard-coded Credentials CWE-311: Missing Encryption of Sensitive Data CWE-807: Reliance on Untrusted Inputs in a Security Decision
- CWE-250: Execution with Unnecessary Privileges CWE-022: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
- CWE-863: Incorrect Authorization
- CWE-676: Use of Potentially Dangerous Function

CWE-732: Incorrect Permission Assignment for Critical Resource

CWE-327: Use of a Broken or Risky Cryptographic Algorithm

CWE-307: Improper Restriction of Excessive Authentication Attempts

CWE-134: Uncontrolled Format String

CWE-759: Use of a One-Way Hash without a Salt CWE-770: Allocation of Resources Without Limits or Throttling

CWE-754: Improper Check for Unusual or Exceptional Conditions

CWE-838: Inappropriate Encoding for Output Context

CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')

CWE-841: Improper Enforcement of Behavioral Workflow

CWE-772: Missing Release of Resource after Effective Lifetime

CWE-209: Information Exposure Through an Error Message

…