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

Selected by the IRSN (Nuclear Authority) to check the safety of programs embedded in nuclear reactors

Only company selected in the Ockham Criteria from the SATE V exhibit

Chosen by the Linux Foundation to develop tools for security of Core Internet Infrastructure

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

© 2015 TrustInSoft
Sell guarantees on software used in sensitive systems
Pain Points In Cyber Security
It’s the right moment now!

Threats

1990
Virus

Worms

Botnet

2000

2010
Advanced Persistent Threats

2020
Mafia, Nation-states with huge resources

Solutions

Anti-Virus

Intrusion Detection Systems

Data leakage Prevention

Harden the software

TRUST IN SOFT
Standard market practice

Best Effort

but no guarantees
must check the two are in sync
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 »
Idea 2: let’s perform an algebraic proof.
\[(a+b)^2=(a+b)\times(a+b)\]
\[= a^2+ab + b^2+ba\]
\[=a^2+b^2+2ab\]
int max (int x, int y) {
    if (x>y) return x; else return y;
}
/*@ ensures \result \geq x && \
\result \geq y; 
ensures \result == x || \
\result == y; */

int max (int x, int y) {
    if (x>y) return x; else return y;
}
An example of application on real-world code

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.

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

This stack has a configuration proven to be without an Heartbleed-like flaw.
http://trust-in-soft.com/polarssl-verification-kit
### CWE list

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

<table>
<thead>
<tr>
<th>PolarSSL</th>
<th>Version 1.1.8 with patches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target architecture</td>
<td>IA-32</td>
</tr>
<tr>
<td>Endianness</td>
<td>Little endian</td>
</tr>
<tr>
<td>ABI</td>
<td>GCC/Linux IA-32</td>
</tr>
<tr>
<td>Provider</td>
<td>Offspark B.V. : <a href="https://polarssl.org/">https://polarssl.org/</a></td>
</tr>
<tr>
<td>Copyright holder</td>
<td>Brainspark B.V.</td>
</tr>
<tr>
<td>License</td>
<td>Dual licensing GPL and closed source commercial license</td>
</tr>
<tr>
<td>Pricing policy</td>
<td>Free for GPL version, see website¹ for details on other licenses.</td>
</tr>
</tbody>
</table>
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_MAXCONTENTLEN + 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 );
```
Server usage pattern

ssl_init → ssl_set_rng → ssl_set_cipher Suites → ssl_set_endpoint → ssl_set_bio → ssl_set_own_cert → ssl_set_session → ssl_handshake → ssl_write → End → ssl_read
C-implementation

```c
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;
      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

```c
#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.

```c
#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
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;
the two
are not in
sync!!!
Static Analysis Tool Exposition and the Ockham Soundness Criteria
What about Open source?
what about open source?
idea of the program

source code

binary code
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.

common open source base contributed by many different persons around the world
Process based vs. Product based
Core Infrastructure Initiative
Fortifying our future.
Tusting the crowd is nice

Formal Guarantees are definitive
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
...