Mind your Language(s)! A discussion about languages and security

Éric Jaeger & Olivier Levillain

LangSec Workshop @ IEEE SSP, 2014-05-18

ANSSI

ANSSI (French Network and Information Security Agency) has InfoSec (and no Intelligence) missions:

- detect and early react to cyber attacks
- prevent threats by supporting the development of trusted products and services
- provide reliable advice and support
- communicate on information security threats and the related means of protection

These missions concern:

- governmental entities
- companies
- the general public

Foreword

What this presentation is about

- the impact of the language on security properties is understudied
- it covers a broad spectrum of subjects

- since 2005, two studies: JavaSec and LaFoSec
- each time, our partners that did not at first share (or even understand) our concerns

- the followig examples do not aim at criticising particular languages
- no language was harmed during our work[1]

[1] They were already like that when we began.

Outline

Outline

[Java] Objection

Object encapsulation: a security mechanism?

Source (java/Introspect.java)

```
import java.lang.reflect.*;
class Secret { private int x = 42; }
public class Introspect {
  public static void main (String[] args) {
    try { Secret o = new Secret();
          Class c = o.getClass();
          Field f = c.getDeclaredField("x");
          f.setAccessible(true);
          System.out.println("x="+f.getInt(o));
        }
    catch (Exception e) { System.out.println(e); }
  }
}
```

- Some keyword may be confusing
- Even if possible, introspection can not easily be banned in practice

[OCaml] The danger of <

OCaml also has encapsulation mechanisms: modules

Source (ocaml/hsm.ml)

```
module type Crypto = sig val id:int end;;

module C : Crypto  =
struct
  let id=Random.self_init(); Random.int 8192
  let key=Random.self_init(); Random.int 8192
end;;
```

It is a sealed box, where `id` is visible, but not `key`

`C.id` returns `- : int = 2570`

`C.key` returns `Error: Unbound value C.key`

Yet this encapsulation is not robust, since the box can be compared on a weighing scale

Source (ocaml/hsmoracle.ml)

```
let rec oracle o1 o2 =
  let o = (o1 + o2)/2 in
  let module O = struct let id=C.id let key=o end in
  if (module O:Crypto)>(module C:Crypto)
  then oracle o1 o
  else (if (module O:Crypto)<(module C:Crypto)
        then oracle o o2
        else o);;

oracle 0 8192;;
```

Outline

[Shell] True, False, FILE_NOT_FOUND

How many value a boolean condition (e.g. `x=y`) can take?

Source (shell/login.sh)

```
#!/bin/bash
PIN=1234
echo -n "Please type your PIN code (4 digits): "
read -s PIN_TYPED; echo

if [ "$PIN" -ne "$PIN_TYPED" ]; then
    echo "Invalid PIN code."; exit 1
else
    echo "Authentication OK"; exit 0
fi
```

[Shell] True, False, FILE_NOT_FOUND

How many value a boolean condition (e.g. `x=y`) can take?

Source (shell/login.sh)

```
#!/bin/bash
PIN=1234
echo -n "Please type your PIN code (4 digits): "
read -s PIN_TYPED; echo

if [ "$PIN" -ne "$PIN_TYPED" ]; then
    echo "Invalid PIN code."; exit 1
else
    echo "Authentication OK"; exit 0
fi
```

In shell, the following excerpt shows a third option should be treated. A bad PIN will be rejected, but "foo" will be accepted

[C] True, False, FILE_NOT_FOUND

A recent vulnerability on GnuTLS may now sound familiar (March 2014, `lwn.net`)

> *But this bug is arguably much worse than Apple's, as it has allowed crafted certificates to evade validation check for all versions of GnuTLS ever released since that project got started in late 2000.[...]*
> *The* `check_if_ca` *function is supposed to return true (any non-zero value in C) or false (zero) depending on whether the issuer of the certificate is a certificate authority (CA). A true return should mean that the certificate passed muster and can be used further, but the bug meant that error returns were misinterpreted as certificate validations.*

[C] True, False, FILE_NOT_FOUND

A recent vulnerability on GnuTLS may now sound familiar (March 2014, `lwn.net`)

> *But this bug is arguably much worse than Apple's, as it has allowed crafted certificates to evade validation check for all versions of GnuTLS ever released since that project got started in late 2000.[...]*
> *The* `check_if_ca` *function is supposed to return true (any non-zero value in C) or false (zero) depending on whether the issuer of the certificate is a certificate authority (CA). A true return should mean that the certificate passed muster and can be used further, but the bug meant that error returns were misinterpreted as certificate validations.*

The same flaw was pre-existant in OpenSSL... in 2008

[JavaScript] *Cast*astroph

Source (js/cast2.js)

```
if ('0'==0) print("'0'==0");
else print("'0'<>0");
if (0=='0.0') print("0=='0.0'");
else print("0<>'0.0'");
if ('0'=='0.0') print("'0'=='0.0'");
else print("'0'<>'0.0'");
```

[JavaScript] *Cast*astroph

Source (js/cast2.js)

```
if ('0'==0) print("'0'==0");
else print("'0'<>0");
if (0=='0.0') print("0=='0.0'");
else print("0<>'0.0'");
if ('0'=='0.0') print("'0'=='0.0'");
else print("'0'<>'0.0'");
```

`'0'==0`, `0=='0.0'` et `'0'<>'0.0'`

[JavaScript] *Cast*astroph

Source (js/cast2.js)

```
if ('0'==0) print("'0'==0");
else print("'0'<>0");
if (0=='0.0') print("0=='0.0'");
else print("0<>'0.0'");
if ('0'=='0.0') print("'0'=='0.0'");
else print("'0'<>'0.0'");
```

`'0'==0`, `0=='0.0'` et `'0'<>'0.0'`

Source (js/cast3.js)

```
a=1; b=2; c='Foo';
print(a+b+c); print(c+a+b); print(c+(a+b));
```

[JavaScript] *Cast*astroph

Source (js/cast2.js)

```
if ('0'==0) print("'0'==0");
else print("'0'<>0");
if (0=='0.0') print("0=='0.0'");
else print("0<>'0.0'");
if ('0'=='0.0') print("'0'=='0.0'");
else print("'0'<>'0.0'");
```

`'0'==0`, `0=='0.0'` et `'0'<>'0.0'`

Source (js/cast3.js)

```
a=1; b=2; c='Foo';
print(a+b+c); print(c+a+b); print(c+(a+b));
```

`3Foo`, `Foo12` and `Foo3`

Source (php/castincr.php)

```
$x="2d8"; print($x+1); print("\n");

$x="2d8"; print(++$x."\n"); print(++$x."\n"); print(++$x."\n");

if ("0xF9"=="249") { print("Equal\n"); }
else { print("Different\n"); }
```

Source (php/castincr.php)

```
$x="2d8"; print($x+1); print("\n");

$x="2d8"; print(++$x."\n"); print(++$x."\n"); print(++$x."\n");

if ("0xF9"=="249") { print("Equal\n"); }
else { print("Different\n"); }
```

The first line produces 3 (an int)

Source (php/castincr.php)

```
$x="2d8"; print($x+1); print("\n");

$x="2d8"; print(++$x."\n"); print(++$x."\n"); print(++$x."\n");

if ("0xF9"=="249") { print("Equal\n"); }
else { print("Different\n"); }
```

The first line produces `3` (an int)

The second displays `2d9` (string), `2e0` (string) then `3` (float).

Source (php/castincr.php)

```
$x="2d8"; print($x+1); print("\n");

$x="2d8"; print(++$x."\n"); print(++$x."\n"); print(++$x."\n");

if ("0xF9"=="249") { print("Equal\n"); }
else { print("Different\n"); }
```

The first line produces `3` (an int)

The second displays `2d9` (string), `2e0` (string) then `3` (float).

The third prints `Equal`

This may lead to security concerns

Source (php/hash.php)

```
$s1='QNKCDZO'; $h1=md5($s1);
$s2='240610708'; $h2=md5($s2);
$s3='A169818202'; $h3=md5($s3);
$s4='aaaaaaaaaaaumdozb'; $h4=md5($s4);
$s5='badthingsrealmlavznik'; $h5=sha1($s5);

if ($h1==$h2) print("Collision\n");
if ($h2==$h3) print("Collision\n");
if ($h3==$h4) print("Collision\n");
if ($h4==$h5) print("Collision\n");
```

This may lead to security concerns

Source (php/hash.php)

```
$s1='QNKCDZO'; $h1=md5($s1);
$s2='240610708'; $h2=md5($s2);
$s3='A169818202'; $h3=md5($s3);
$s4='aaaaaaaaaaaumdozb'; $h4=md5($s4);
$s5='badthingsrealmlavznik'; $h5=sha1($s5);

if ($h1==$h2) print("Collision\n");
if ($h2==$h3) print("Collision\n");
if ($h3==$h4) print("Collision\n");
if ($h4==$h5) print("Collision\n");
```

`Collision` is printed 4 times, but we did not break MD5 nor SHA1

Outline

[C] Sauf exception. . .

We all know this example and what happens with `abs(x++)`

Source (c/strategy1-abs.c)

```
#define abs(X) (X)>=0?(X):(-X)
```

What will the following code do?

Source (c/strategy3.c)

```
int zero(int x) { return 0; }
int main(void) { int x=0; x=zero(1/x); return 0; }
```

[C] Sauf exception. . .

We all know this example and what happens with `abs(x++)`

Source (c/strategy1-abs.c)

```
#define abs(X) (X)>=0?(X):(-X)
```

What will the following code do?

Source (c/strategy3.c)

```
int zero(int x) { return 0; }
int main(void) { int x=0; x=zero(1/x); return 0; }
```

It depends on the optimisation level:

- `-O0` : `Floating point exception (core dumped)`
- `-O1` : normal termination

[OCaml] Mutatis mutandis

In OCaml, code is static and strings are mutable. What about strings appearing in code?

Source (ocaml/mutable.ml)

```
let check c =
  if c then "OK" else "KO";;

let f=check false in
  f.[0]<-'O'; f.[1]<-'K';;

check true;;
check false;;
```

[OCaml] Mutatis mutandis

In OCaml, code is static and strings are mutable. What about strings appearing in code?

Source (ocaml/mutable.ml)

```
let check c =
  if c then "OK" else "KO";;

let f=check false in
  f.[0]<-'O'; f.[1]<-'K';;

check true;;
check false;;
```

Both `check` calls return `"OK"`

[OCaml] Mutatis mutandis

In OCaml, code is static and strings are mutable. What about strings appearing in code?

Source (ocaml/mutable.ml)

```
let check c =
  if c then "OK" else "KO";;

let f=check false in
  f.[0]<-'O'; f.[1]<-'K';;

check true;;
check false;;
```

Both `check` calls return `"OK"`

Such mutable shared strings may be used to determine control flow, or to escape characters (`Char.escaped`)

[Python] Glocal variables

Python allows for comprehension lists, which is another syntax for a `map` application

Source (python/listcomp.py)

```
>>> l = [s+1 for s in [1,2,3]]
>>> l
[2, 3, 4]
```

Now, what is the value of `s`?

[Python] Glocal variables

Python allows for comprehension lists, which is another syntax for a `map` application

Source (python/listcomp.py)

```
>>> l = [s+1 for s in [1,2,3]]
>>> l
[2, 3, 4]
```

Now, what is the value of `s`?

Unless you use Python 3, `s` is `3`, whereas the `s` variable should have been local (bound).

Outline

[C] No comments ?

Syntax matter...

Source (c/comments2.c)

```
#include <stdio.h>

int main(void) {

// /!\ DO NOT REMOVE COMMENTS IN NEXT BLOCK /!\
/*****************************************
  const char status[]="Safe";
// /!\ SET TO SAFE ONLY FOR TESTS /!\
*****************************************/

// /!\ NEXT LINE REALLY IMPORTANT /!\
  const char status[]="Unsafe";
  printf("Status: %s\n",status);
}
```

- C trigrams
- UTF-8 characters / encoding allowed in JAVA

Outline

[C] Story of a real kernel bug

Source (c/badoptim.c)

```
struct tun_struct *tun = __tun_get(tfile);
struct sock *sk = tun->sk;
if (!tun)
     return POLLERR;
/* use *sk for write operations */
```

This particular undefined behavior led to an optimisation, which is now known as CVE-2009-1897.

[Java] Serial killer

Source (java/Deserial.java)

```
import java.io.*;
class Friend { } // Unlikely to be dangerous!
class Deserial {
  public static void main (String[] args)
    throws FileNotFoundException, IOException,
           ClassNotFoundException {
    FileInputStream fis = new FileInputStream("friend");
    ObjectInputStream ois = new ObjectInputStream(fis);
    Friend f=(Friend)ois.readObject();
    System.out.println("Hello world");
  }
}
```

[Java] Serial killer

Source (java/Deserial.java)

```
import java.io.*;
class Friend { } // Unlikely to be dangerous!
class Deserial {
  public static void main (String[] args)
    throws FileNotFoundException, IOException,
           ClassNotFoundException {
    FileInputStream fis = new FileInputStream("friend");
    ObjectInputStream ois = new ObjectInputStream(fis);
    Friend f=(Friend)ois.readObject();
    System.out.println("Hello world");
  }
}
```

At runtime, we may read `Bad things happen!` since the serialised file contained an object of a different class. The *cast* might fail, but it might be too late

[Java] Serial killer

Source (java/Deserial.java)

```
import java.io.*;
class Friend { } // Unlikely to be dangerous!
class Deserial {
  public static void main (String[] args)
    throws FileNotFoundException, IOException,
           ClassNotFoundException {
    FileInputStream fis = new FileInputStream("friend");
    ObjectInputStream ois = new ObjectInputStream(fis);
    Friend f=(Friend)ois.readObject();
    System.out.println("Hello world");
  }
}
```

At runtime, we may read `Bad things happen!` since the serialised file contained an object of a different class. The *cast* might fail, but it might be too late

Not controlling which code is run may be dangerous (CVE-2008-5353)

Some concerns about memory management

When dealing with interpreted languages and Garbage collectors

- what `chmod -x` does?
- can the memory pages be marked as non-exeutable?
- how can we really enforce $W \wedge X$?
- what does a JIT compiler change?

- how can I be sure a data is not spread by a *mark and copy* strategy?
- can I have guarantees on a key lifetime?
- can I zeroise it in some way?

Outline

[Java] Clone Wars

The official Java specification about `Object.clone()`

> *The general intent is that, for any object x, the expression:* `x.clone()!= x` *will be true, and that the expression:* `x.clone().getClass()== x.getClass()` *will be true, but these are not absolute requirements. While it is typically the case that:* `x.clone().equals(x)` *will be true, this is not an absolute requirement.*

Serialisation specifications (`writeObject` and `readObject` functions) are also worth reading

[C] Specs and checks

In "*The C programming language (Second edition)*", by *B. W. Kernighan & D. M. Ritchie*

> *The direction of truncation for* `/` *and the sign of the result for* `%` *are machine-dependent for negative operands, as is the action taken on overflow or underflow.*

How would you check that a compiler complies to this non-deterministic specification?

[C] Specs and checks

In "*The C programming language (Second edition)*", by *B. W. Kernighan & D. M. Ritchie*

> *The direction of truncation for* `/` *and the sign of the result for* `%` *are machine-dependent for negative operands, as is the action taken on overflow or underflow.*

How would you check that a compiler complies to this non-deterministic specification?

Would your check reject a compiler changing its mind at each division, which would lead to `1/-2==1/-2` being false. This is an instance of the Refinement Paradox

Outline

Lessons learned

- Programming languages can impact software security
- There is room for improvement in them
- We could benefit from more research and tools

- Writing secure software requires a broad vision in many aspects of computer science
- Teaching should take more those aspects into account

Questions?

Thank you for your attention

olivier.levillain@ssi.gouv.fr