Boosting the BINSEC symbolic execution engine: A France CyberSecurity Challenge journey

Frédéric Recoules
Beginning of the journey

Former intern
(Binsec/Rel2)
Beginning of the journey

Former intern (Binsec/Rel2)

Former PhD student (Binsec/Rel)
Beginning of the journey

Former intern (Binsec/Rel2)

Former PhD student (Binsec/Rel)

Z80
Beginning of the journey

Former intern
(Binsec/Rel2)

Former PhD student
(Binsec/Rel)

Z80

France Cybersecurity Challenge 2022
France Cybersecurity Challenge

- Preselection for the European Cybersecurity Challenge
- crypto, reverse, pwn, web, forensics, hardware, side channel or fault attack, etc.
- From 5 April 2024, 14h to 14 April
An example of reverse engineering challenge
An example of reverse engineering challenge

> -=-=-=-= Very secure vault -=-=-=-=
> Please enter your very secure password:
> _
An example of reverse engineering challenge

• =-=-=-=-=- Very secure vault =-=-=-=-=-
• Please enter you very secure password:
• aaaaaaaa
An example of reverse engineering challenge

- Very secure vault ==========
- Please enter your very secure password:
- aaaaaaaaa
- Wrong password: authorities have been alerted!
An example of reverse engineering challenge

> ==-==-== Very secure vault ==-==-==
> Please enter your very secure password:
> _
An example of reverse engineering challenge

> ==-=-=-=-=-= Very secure vault ==-=-=-=-=-=
> Please enter very secure password:
> 346bc05be4ed8361a68a3d9748fc9b87de397e1
An example of reverse engineering challenge

> ====== Very secure vault =======
> Please enter your very secure password:
> 346bc605be4ed8361a68a3d9748fc9b87de397e1
> \o/ Access granted! \o/
> Here is your flag:
> ECSC{346bc605be4ed8361a68a3d9748fc9b87de397e1}
An example of reverse engineering challenge

```
\> =-=-=-=-=- Very secure vault =-=-=-=-=-
\> Please enter your very secure password:
\> 346bc605be4ed8361a68a3d9748fc9b87de397e1
\> \o/ Access granted! \o/
\> 
\> Here is your flag:
\> ECSC\{346bc605be4ed8361a68a3d9748fc9b87de397e1\}
```

```c
int check_char(int pos, char c) {
    return password[(pos + 10) % 40] == c;
}

int main(int argc, char **argv) {
    char c;
    int pos = 0;
    int res = 1;
    int sum = 0;

    while((c = getchar()) != EOF) {
        if (c == '\n') break;
        res &= check_char(pos, c);
        if (res) sum += c;
        pos += 1;
    }
    if (res == 1 && sum == 2827) success();
    else failure();
    return 0;
}
```
This talk in a nutshell

Goal
Introduce some recent improvements of the BINSEC symbolic execution engine through the prism of CTF challenges

Highlights
- Symbolic execution and its limitations
- Improvements
  - Efficient use of SMT solvers
  - Straightforward path merging
  - JIT specialization of the interpreter
BINSEC in a nutshell (since 2012)

BINSEC

Symbolic engine

Generic IR

Vulnerability Assessment

- Fault injection
- Side channel attack
- Attacker model

Bug finding

- Advanced fuzzing
- Test case generation

Reverse Engineering

- Capture The Flag
- Deobfuscation
- Decompilation

Supply chain

Security critical components

- Fault injection
- Side channel attack
- Attacker model

Malware comprehension

- Capture The Flag
- Deobfuscation
- Decompilation

Bitwuzla

56k LOC

OCaml

The annual meeting of the GT MFS 2024

BINSEC in a nutshell (since 2012)

- Fault injection
- Side channel attack
- Attacker model

Supply chain

Security critical components

Malware comprehension

- Capture The Flag
- Deobfuscation
- Decompilation

Symbolic engine

Generic IR
Symbolic execution in a nutshell

int main () {
    int x = input();
    int y = input();
    int z = 2 * y;
    if (z == x) {
        if (x > y + 10)
            failure;
    }
    success;
}

\[\sigma := \emptyset\]
\[\phi := \top\]

\[x = \text{input}()\]
\[y = \text{input}()\]
\[z = 2 \times y\]

\[\sigma := \{ x \rightarrow \alpha, y \rightarrow \beta, z \rightarrow 2\beta \}\]

\[\phi := \top \land 2\beta \neq \alpha\]

\[x > y + 10\]

\[\phi := \top \land 2\beta = \alpha\]

\[\phi := \top \land 2\beta = \alpha \land \alpha \leq \beta + 10\]

\[\phi := \top \land 2\beta = \alpha \land \alpha > \beta + 10\]

\[\text{UNDER APPROX.}\]
- Correct
- Incomplete
(maybe $k$-complete)
Theoretical and practical limits
Theoretical and practical limits

⚠ Search space (# paths)

⚠ Constraint solving (query cost)
The annual meeting of the GT MFS 2024

Theoretical and practical limits

⚠ Search space
(# paths)

⚠ Exploration speed

⚠ Constraint solving
(# queries)

⚠ Constraint solving
(query cost)
How to help competitors

Applicability 1
32 / 64 bits architectures, static / shared / self-modifying code, etc.

Usability 2
easy to install, easy to configure, meaningful result, etc.

Speed 3
trial-and-error, should be faster than doing it by hand!
Sneak peak over usability progresses

From salt

[disasm]
decode-replacement =
    <fgets> ->
        0: nondet_assume ((asize<64>),
            (0<64> < asize<64>) && (asize<64> <= rsi<64>)));
        goto 1
    1: i<64> := 0<64>; goto 2
    2: if (i<64> <=s (asize<64> - 1<64>)) goto 3 else goto 6
    3: @[rdi<64> + i<64>, 1] := stdin[stdin_p<64>, 1]; goto 4
    4: stdin_p<64> := stdin_p<64> + 1<64>; goto 5
    5: i<64> := i<64> + 1<64>; goto 2
    6: rax<64> := rdi<64>; goto 7
    7: rsp<64> := rsp<64> + 8<64>; goto 8
    8: goto @(rsp<64> - 8<64>, 8) // ret
    __stack_chk_fail" ->
        0: assert (0<1>); goto 0

To sugar

starting from core

stdin_p<64> := 0
replace <fgets> {ptr, size, _} by
    asize<64> := nondet
    assume 0 < asize <= size
    for i<64> in 0 to asize - 1 do
        @[ptr + i] := stdin[stdin_p]
        stdin_p := stdin_p + 1
    end
    return ptr
end

reach <puts> then print c string stdin
halt at @(rsp, 8)
abort at __stack_chk_fail

The annual meeting of the GT MFS 2024
10
Sneak peak over usability progresses

From salt

```plaintext
from salt

[disasm]

decode-replacement =

<fgets> ->

0: nondet_assume ((asize<64>),
    ((0<64> < asize<64>) && (asize<64> <= rsi<64>)));
    goto 1
1: i<64> := 0<64>; goto 2
2: if (i<64> <=s (asize<64> - 1<64>)) goto 3 else goto 6
3: rdi<64> + i<64>, 1] := stdin[stdin_p<64>, 1]; goto 4
4: stdin_p<64> := stdin_p<64> + 1<64>; goto 5
5: i<64> := i<64> + 1<64>; goto 2
6: rax<64> := rdi<64>; goto 7
7: rsp<64> := rsp<64> + 8<64>; goto 8
8: goto @[rsp<64> - 8<64>, 8] // ret
<__stack_chk_fail> ->
0: assert (0<1>); goto 0
```

To sugar

Starting from core

```plaintext

stdin_p<64> := 0
replace <fgets> {ptr, size, _} by

asize<64> := nondet
assume 0 < asize <= size
for i<64> in 0 to asize - 1 do
  @[ptr + i] := stdin[stdin_p]
  stdin_p := stdin_p + 1
end
return ptr

reach <puts> then print c string stdin
halt at @[rsp, 8]
abort at <__stack_chk_fail>
```

[sse]

enabled = true
directives = <puts> reach; <main + 144> cut
**Sneak peak over usability progresses**

From salt

```plaintext
[kernel]
entrypoint = main

[disasm]
decode-replacement =

<fgets> ->

0: nondet_assume ((asize<64>),
  ((0<64> < asize<64>) && (asize<64> <= rsi<64>)));
  goto 1
1: i<64> := 0<64>; goto 2
2: if (i<64> <=s (asize<64> - 1<64>)) goto 3 else goto 6
3: &[rdi<64> + i<64>, 1] := stdin[stdin_p<64>, 1]; goto 4
4: stdin_p<64> := stdin_p<64> + 1<64>; goto 5
5: i<64> := i<64> + 1<64>; goto 2
6: rax<64> := rdi<64>; goto 7
7: rsp<64> := rsp<64> + 8<64>; goto 8
8: goto &[rsp<64> - 8<64>, 8] // ret
<__stack_chk_fail> ->
0: assert (0<1>); goto 0

[sse]
enabled = true
directives = <puts> reach; <main + 144> cut
```

To sugar

**starting from core**

```plaintext
stdin_p<64> := 0
replace <fgets> {ptr, size, _} by
  asize<64> := nondet
  assume 0 < asize <= size
for i<64> in 0 to asize - 1 do
  &[ptr + i] := stdin[stdin_p]
  stdin_p := stdin_p + 1
end
return ptr
end

reach <puts> then print c string stdin
halt at &[rsp, 8]
abort at <__stack_chk_fail>
```

**CORE DUMP**
Dynamic shared libraries handling
Complex initialization

**GOAL & ACTION**
Format values as needed
Sneak peak over usability progresses

From salt

```plaintext
[disasm]
decode-replacement =

<fgets> ->

0: nondet Assume ((asize<64>),
    ((0<64) < asize<64>) && (asize<64> <= rsi<64>)));

goto 1
1: i<64> := 0<64>; goto 2
2: if (i<64> <= s (asize<64> - 1<64>)) goto 3 else goto 6
3: @[rdi<64> + i<64>, 1] := stdin[stdin_p<64>, 1]; goto 4
4: stdin_p<64> := stdin_p<64> + 1<64>; goto 5
5: i<64> := i<64> + 1<64>; goto 2
6: rax<64> := rdi<64>; goto 7
7: rsp<64> := rsp<64> + 8<64>; goto 8
8: goto @[rsp<64> - 8<64>, 8] // ret

<sse]

directives = <puts> reach; <main + 144> cut
```

To sugar

```
starting from core

stdin_p<64> := 0
replace <fgets> {ptr, size, _} by
asize<64> := nondet
assume 0 < asize <= size
for i<64> in 0 to asize - 1 do
    @[ptr + i<64>] := stdin(stdin_p)
    stdin_p := stdin_p + 1
end

return ptr

reach <puts> then print c string stdin

halt at @[rsp, 8]
abort at __stack_chk_fail
```

CORE DUMP
Dynamic shared libraries handling
Complex initialization

SYNTACTIC SUGAR
Optional size
Syntactic bloc (if, for, while)
Automatic argument & return

GOAL & ACTION
Format values as needed

The annual meeting of the GT MFS 2024

The annual meeting of the GT MFS 2024
Avoid calling the SMT solver
Avoid calling the SMT solver
Avoid calling the SMT solver

SMT SOLVER
Generate a new model
Avoid calling the SMT solver

SMT SOLVER
Generate a new model
Avoid calling the SMT solver

SMT SOLVER
Generate a new model
Avoid calling the SMT solver

01 SMT SOLVER
Generate a new model

02 UNDER APPROX.
Along a path
Propagate the previous model
Halve the solver queries
Avoid calling the SMT solver

01 SMT SOLVER
Generate a new model

02 UNDER APPROX.
Along a path
Propagate the previous model
Halve the solver queries
Avoid calling the SMT solver

01 **SMT SOLVER**
Generate a new model

02 **UNDER APPROX.**
Along a path
Propagate the previous model
Halve the solver queries

03 **OVER APPROX.**
Along a path
Discard invalid constraints
Souk : the diamond loop

- **Explosion**
  All combinations are explored even if they can no longer lead to the goal.
  \[ 2^n \]

- **Early Exits**
  Paths end as soon as they can no longer lead to the goal.
  \[ n+1 \]

- **Merging**
  Paths are merged when possible – reasoning is harder but paths are under control.
  \[ 2 \]

Quick merging from timeout (\( \geq 2^{71} \) paths) to 30s
Licorne : a difficulty that is not virtual

01 Shared library
Binary is not self-contained?
_allow starting from a process snapshot! (coredump)

02 Self-modifying code
QEMU performs just in time compilation
⚠ write in memory can change code!

03 Billion of instructions
Emulate an emulator?
⚡ need to be twice as performant!

UNICORN
multi-architecture CPU emulator framework (based on QEMU)
License in a nutshell
Licorne in a nutshell

The annual meeting of the GT MFS 2024
Licorne in a nutshell

999 859 832 instructions

The annual meeting of the GT MFS 2024
Licorne in a nutshell

\[ V = \begin{bmatrix} \vdots & \vdots & \vdots \\ V_1 & V_2 & V_3 \\ \vdots & \vdots & \vdots \end{bmatrix} \times 31 \]

\[ V = \begin{bmatrix} 232124671203872083221 & 0 & 0 & 0 \\ 0 & 3797207351213602299 & 0 & 0 \\ 0 & 0 & 979707166530166755 & 0 \\ 0 & 0 & 0 & 1401754654689990061 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2193733461346243843 \\ 0 & 0 & 0 & 0 & 0 & 2193733461346243843 \end{bmatrix} \times V + \]

\[ 202353593412028096306 \]

\[ 5265659239627236637 \]

\[ -2371665477955254209 \]

\[ 4567941066487989871 \]

\[ 901707264336013363 \]

\[ -111389708376275154 \]

\[ -164329868519723812 \]

999 859 832 instructions 0.6

RESOLUTION ✔

~3h

The annual meeting of the GT MFS 2024
Licorne in a nutshell

\[ V = \begin{bmatrix} 2312467120372082310 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3797207561223029999 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1810716657301067755 & 0 & 0 \\ 0 & 0 & 0 & 0 & 133934645497920617 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \times V + \begin{bmatrix} 2029355904220986306 \\ 5265569239827266537 \\ -2371964777755742509 \\ 4578741035647358871 \\ 9017076404336013363 \\ -113389768378751504 \\ -164329868519723812 \end{bmatrix}

\[ V = \begin{bmatrix} 999 & 859 & 832 \end{bmatrix}

0.6

\[ \text{RESOLUTION } \checkmark \]

\[ \sim 3h \]

0.8

\[ \text{RESOLUTION } \checkmark \]

10m24

999 859 832 instructions

EXE

The annual meeting of the GT MFS 2024
Licorne in a nutshell

\[ V = \begin{bmatrix} \text{GAME OVER} \end{bmatrix} \times 31 \]

\[ V = \begin{bmatrix} 23212467120387208331 0 0 0 0 0 0 0 0 0 \text{GAME OVER} \end{bmatrix} \times 31 \]

999 859 832 instructions

0.6

RESOLUTION

0.8

RESOLUTION

~3h

10m24

The annual meeting of the GT MFS 2024
Licorne in a nutshell

The annual meeting of the GT MFS 2024

999 859 832 instructions

0.6

0.8

RESOLUTION ✓

RESOLUTION ✓

~3h

10m24

≥ 99%
Licorne in a nutshell

\[ V = 2312467120387208281 \times V = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \]

\[ \times 31 \]

GAME OVER

\[ V = 20293598142209636 \times V = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \]

\[ \times 31 \]

\[ V = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \]

\[ \times V + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \]

\[ \begin{pmatrix} 999 859 832 \text{ instructions} \end{pmatrix} \]

\[ \begin{pmatrix} 0.6 \end{pmatrix} \]

\[ \begin{pmatrix} 0.8 \end{pmatrix} \]

\[ \begin{pmatrix} 0.8 + \text{JIT} \end{pmatrix} \]

\[ \begin{pmatrix} \sim 3h \end{pmatrix} \]

\[ \begin{pmatrix} 10m24 \end{pmatrix} \]

\[ \begin{pmatrix} 7m15 \end{pmatrix} \]

\[ \geq 99\% \]

The annual meeting of the GT MFS 2024
Specialization of an interpreter

**SWITCH**

01.

**LANGUAGE**

**PRIMITIVES**

XCHG

ADD

The annual meeting of the GT MFS 2024
Specialization of an interpreter

**Switch**
- **01**
  - **Xchg RAX, RDX**
- **02**
  - **Add RAX, RDX**

**Language**
- **Xchg**
  - X, Y
- **Add**
  - X, Y

**Primitives**
- Up
- Down
- Add

**Partial Specialization**
- **01+02**
  - **Xchg RAX, RDX**
  - **Add RAX, RDX**

---

The annual meeting of the GT MFS 2024
Specialization of an interpreter
Capture the Flag (CTF) challenges evolve

- Challenges
- Automation
- Countermeasure
- Experiments
- New Automation

FRANCE CYBERSECURITY CHALLENGE

The annual meeting of the GT MFS 2024
Conclusion

MOUTHON RANKED HIGH
Mainly because it is pretty good at CTF challenges, but I like to believe it is also a little bit because of BINSEC -_(-艸-)_/—
Conclusion

01 MOUTHON RANKED HIGH
Mainly because it is pretty good at CTF challenges, but I like to believe it is also a little bit because of BINSEC ^_^/

02 BINSEC GETS BETTER
New development and refactoring end up benefitting other parts of the platform (e.g. CCS 2023)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Binsec/Rel</th>
<th>Binsec/Rel2</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES-CBC-bearssl (BS)</td>
<td>16.77</td>
<td>0.31</td>
<td>x 54</td>
</tr>
<tr>
<td>AES-GCM-bearssl (BS)</td>
<td>53.32</td>
<td>0.48</td>
<td>x 111</td>
</tr>
<tr>
<td>PolyChacha-bearssl (CT)</td>
<td>9.72</td>
<td>0.18</td>
<td>x 53</td>
</tr>
<tr>
<td>PolyChacha-mbedtls</td>
<td>18.62</td>
<td>0.49</td>
<td>x 38</td>
</tr>
<tr>
<td>PolyChacha-openssl (EVP)</td>
<td>0.77</td>
<td>0.09</td>
<td>x 8</td>
</tr>
<tr>
<td>Chacha20-openssl</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

01 MOUTHON RANKED HIGH
Mainly because it is pretty good at CTF challenges, but I like to believe it is also a little bit because of BINSEC -\(_(ツ)_/\"

02 BINSEC GETS BETTER
New development and refactoring end up benefitting other parts of the platform (e.g. CCS 2023)

JOBS
WE ARE HIRING

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Binsec/Rel</th>
<th>Binsec/Rel2</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES-CBC-bearssl (BS)</td>
<td>16.77</td>
<td>0.31</td>
<td>x 54</td>
</tr>
<tr>
<td>AES-GCM-bearssl (BS)</td>
<td>53.32</td>
<td>0.48</td>
<td>x 111</td>
</tr>
<tr>
<td>PolyChacha-bearssl (CT)</td>
<td>9.72</td>
<td>0.18</td>
<td>x 53</td>
</tr>
<tr>
<td>PolyChacha-mbedtls</td>
<td>18.62</td>
<td>0.49</td>
<td>x 38</td>
</tr>
<tr>
<td>PolyChacha-openssl (EVP)</td>
<td>💩</td>
<td>21.55</td>
<td>x 163+</td>
</tr>
<tr>
<td>Chacha20-openssl</td>
<td>0.77</td>
<td>0.09</td>
<td>x 8</td>
</tr>
</tbody>
</table>

The annual meeting of the GT MFS 2024