Anxiety: a dynamic symbolic execution framework

Alexander Gerasimov
Senior Researcher
agerasimov@ispras.ru
Dynamic symbolic execution

- **Test coverage generation & debugging**
  - 1975 - SELECT (LISP)
  - 1976 – EFFIGY (PL/I)
  - 2004 – CUTE, DART (C)

- **Defects detection**
  - 2006 – EXE (C)
  - 2008 – BitBlaze (Executable, System-wide)
  - 2008 – SAGE (Executable)
  - 2008 – KLEE (LLVM)
  - 2010 – Avalanche (Executable)
  - 2011 – S²E (Executable, System-wide)
  - 2012 – Mayhem (Executable)
Limitations and requirements

Avalanche tool

- Based on Valgrind DBI Framework (Linux)
- Heavily rely on CVC format

Develop a more flexible framework for dynamic symbolic execution

Requirements:

- User-mode programs analysis
- Support Windows and Linux programs
- Support different SMT-solvers
- Support modular structure for solving different tasks
Anxiety framework

DynamoRIO
PIN
DynInst
Valgrind

Windows and Linux

Linux only

input data

Tracer

combined trace

trace splitter

dangerous operations
new path constraint

input data and metric

new input data

input data generator

path constraint

Coverage analysis

defects and input data

Solvers

CVC4
MathSAT
STP
Z3
## Analysis results

<table>
<thead>
<tr>
<th>Program</th>
<th>Reason</th>
<th>Step</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>vde_l3</td>
<td>SEGV</td>
<td>2</td>
<td>Incorrect argument which starts with ‘-’</td>
</tr>
<tr>
<td>roarfilt</td>
<td>SEGV</td>
<td>49</td>
<td>Argument ‘-R’</td>
</tr>
<tr>
<td>umax_pp</td>
<td>SEGV</td>
<td>444</td>
<td>Argument ‘-[xywhlt] [^ ]+’</td>
</tr>
<tr>
<td>pnmhistmap</td>
<td>SEGV</td>
<td>20</td>
<td>File with content ‘50 36 37 50 30 50 39 32 49 00’</td>
</tr>
<tr>
<td>jasper</td>
<td>ABRT</td>
<td>4</td>
<td>File with content ‘4d 49 46 0a a3 0a’ and arguments are ‘-f &lt;file&gt; -t mif –T [type]’</td>
</tr>
<tr>
<td>fdtdump</td>
<td>SEGV</td>
<td>31</td>
<td>File with incorrect content and length more than 8 bytes</td>
</tr>
<tr>
<td>faad</td>
<td>SEGV</td>
<td>6</td>
<td>File with content ‘41 44 49 46 00 00 01 00 00 00 00 60’</td>
</tr>
<tr>
<td>nettle-hash</td>
<td>ABRT</td>
<td>2</td>
<td>Incorrect argument starts with ‘-’ and ‘--’</td>
</tr>
<tr>
<td>instat</td>
<td>FPE</td>
<td>179</td>
<td>Argument ‘-i 0’</td>
</tr>
</tbody>
</table>
Goals

- Anxiety: a framework for dynamic symbolic execution
  - Supports user-mode programs dynamic symbolic execution
  - Supports analysis for Windows and Linux
  - Supports different SMT-solvers
  - Modular structure of tool allows adoption to different tasks
Limitations of approach

Under-taintedness due to indirect dependencies

def char convtab = {'a', 'b', ... , 'a', 'b', ...};

def char tolower(const char ch)
{
    // Symbolic index, untainted buffer
    return convtab[ch - 'a']; // *(convtab + ch - 'a')
}

Limitations

Under-taintedness due to indirect dependencies

Node* create(const Token &token) {
    if (token == 'select') {
        // External data
        return new SelectNode(); // Internal data
    } else if (...)
    ...
    return NULL; // Internal data
}
Limitations

Under-taintedness due to indirect dependencies

```c
switch(i) {
    // Symbolic i
    case 0: doZero(); break;
    case 1: doOne();
    case 2: doTwo(); break;
    default: doDefault();
}
```

```asm
mov ax, i
cmp ax, 2
jg DEFAULT
cmp ax, 0
jl DEFAULT
jmp table[ax]
data table
    ZERO
    ONE
    TWO
end data
ZERO: call doZero
jmp END
ONE: call doOne
TWO: call doTwo
jmp END
DEFAULT: call
doDefault
END:
```

Under-tainted jump
Limitations

Limited set of security predicates

- Divide by Zero
Limitations

Analysis path number explosion
Future research

- Under-/over-taintedness of program dataflow
- More security predicates (BoF, Memory Leak, …)
- Analysis methods combination
Research & Development team

- Mikhail Ermakov
- Sergey Vartanov
- Leonid Kruglov
- Alexander Novikov
- Daniil Kutz
- Seryozha Asryan
- Alexander Gerasimov
Thank you!