Eternal War in Memory

Systematization of Knowledge

László Szekeres, Mathias Payer, Tao Wei, Dawn Song

Stony Brook University
University of California, Berkeley
Peking University
Problem

• C/C++ is unsafe
• Everybody runs C/C++ code
• They surely have exploitable vulnerabilities
Overview

• What are the attacks?
• What are the deployed protections?
• What are the *not* deployed protections?
• Why aren’t they deployed?
Attack model
Classic stack smashing attack

- Make pointer out-of-bounds
- Make pointer dangling

- Use pointer to write
- Use pointer to read

- Modify a code pointer...
  - ... to target code address

- Use pointer by indir. call/jmp
- Use pointer by ret instruction

- Exec. gadgets or functions
- Execute injected code

- Control-flow hijack
Use-after-free exploits

- Make pointer out-of-bounds
- Use pointer to write
- Use pointer to read
- Modify a code pointer...
  - ... to target code address
    - Use pointer by indir. call/jmp
    - Use pointer by ret instruction
      - Exec. gadgets or functions
      - Execute injected code
        - Control-flow hijack
Corrupting newer and newer pointers

- Make pointer out-of-bounds
- Make pointer dangling

- Use pointer to write
- Use pointer to read

Modify a code pointer...
... to target code address

- Use pointer by indir. call/jmp
- Use pointer by ret instruction

- Exec. gadgets or functions
- Execute injected code

Control-flow hijack
Modifying the code itself

- Make pointer out-of-bounds
- Make pointer dangling

- Use pointer to write
- Use pointer to read

- Modify a code pointer...
  - ... to target code address

- Use pointer by indir. call/jmp
- Use pointer by ret instruction

- Exec. gadgets or functions
- Execute injected code

- Control-flow hijack

Modify a data pointer

- Modify code...
  - ... to attacker specified code

- Code corruption
Code integrity

- Make pointer out-of-bounds
- Make pointer dangling
- Use pointer to write
- Use pointer to read
- Modify a code pointer...
  ... to target code address
  Use pointer by indir. call/jmp
  Use pointer by ret instruction
  Exec. gadgets or functions
  Execute injected code
  Control-flow hijack
- Code corruption
- Modify a data pointer
  ... to attacker specified code
- Read-only code pages
Non-executable data

- Make pointer out-of-bounds
- Make pointer dangling
- Use pointer to write
- Use pointer to read
- Modify a code pointer...
  ... to target code address
  - Use pointer by indir. call/jmp
  - Use pointer by ret instruction
- Exec. gadgets or functions
- Execute injected code

Control-flow hijack

Modify a data pointer

Read-only code pages

- ... to attacker specified code

Code corruption
Non-executable data

Modify a data pointer

Read-only code pages
... to attacker specified code

Make pointer out-of-bounds
Make pointer dangling

Use pointer to write
Use pointer to read

Modify code pointer...
... to target code address

Use pointer by indir. call/jmp
Use pointer by ret instruction

Exec. gadgets or functions

Non-exec. data pages

Code corruption

Control-flow hijack
Return-oriented programming

- Make pointer out-of-bounds
- Make pointer dangling
- Use pointer to write
- Use pointer to read
- Modify a code pointer...
- ... to target code address
- Use pointer by indir. call/jmp
- Use pointer by ret instruction
- Exec. gadgets or functions
- Non-exec. data pages
- Control-flow hijack

Modify a data pointer

Read-only code pages

... to attacker specified code

Code corruption
Return integrity

Make pointer out-of-bounds
Make pointer dangling

Use pointer to write
Use pointer to read

Modify a code pointer...
... to target code address

Use pointer by indir. call/jmp
Use pointer by ret instruction

Exec. gadgets or functions

Read-only code pages

Modify a data pointer
... to attacker specified code

Code corruption

Non-exec. data pages

Control-flow hijack
Return integrity

Make pointer out-of-bounds
Make pointer dangling

Use pointer to write
Use pointer to read

Modify a code pointer...
... to target code address

Use pointer by indir. call/jmp

Stack canaries

Exec. gadgets or functions
Non-exec. data pages

Control-flow hijack

Read-only code pages
... to attacker specified code

Modify a data pointer

Code corruption
Hijacking indirect calls and jumps

- Make pointer out-of-bounds
- Make pointer dangling

- Use pointer to write
- Use pointer to read

Modify a code pointer...
... to target code address

- Use pointer by indir. call/jmp
- Use pointer by ret instruction

Modify a data pointer
... to attacker specified code

- Exec. gadgets or functions
- Stack canaries
- Non-exec. data pages

Control-flow hijack

Read-only code pages

Code corruption
Address space randomization

- Make pointer out-of-bounds
- Make pointer dangling
- Use pointer to write
- Use pointer to read
- Modify a code pointer...
- ... to target code address
- Modify a data pointer...
- ... to attacker specified code
- Use pointer by indir. call/jmp
- Use pointer by ret instruction
- Exec. gadgets or functions
- Exec. gadgets or functions
- Use pointer by indir. call/jmp
- Control-flow hijack
- Stack canaries
- Non-exec. data pages

- Code corruption
- Read-only code pages
Address space randomization

- Make pointer out-of-bounds
  - Use pointer to write
  - Use pointer to read
  - Modify a code pointer...
  - ASLR
    - Use pointer by indir. call/jmp
    - Exec. gadgets or functions
    - Non-exec. data pages
    - Stack canaries
    - Control-flow hijack

- Modify a data pointer
  - Read-only code pages
    - ... to attacker specified code

- Code corruption
Data-only attack

- Make pointer out-of-bounds
- Make pointer dangling

- Use pointer to write
- Use pointer to read

- Modify a code pointer...
  - ... to target code address
  - Use pointer by indir. call/jmp
  - Exec. gadgets or functions

- Modify a data pointer...
  - Use corrupted data variable

- Modify data...
  - ... to attacker specified value

- Read-only code pages
  - ... to attacker specified code

- Non-exec. data pages
  - Control-flow hijack

- ASLR

- Stack canaries

- Data-only attack
Information leakage

- Make pointer out-of-bounds
- Make pointer dangling
- Use pointer to write
- Use pointer to read
- Modify a code pointer...
  - Use pointer by indir. call/jmp
  - Exec. gadgets or functions
  - Interpret the leaked value
- Modify a data pointer...
  - Use corrupted data variable
  - Use pointer by ret instruction
- Read-only code pages
  - ... to attacker specified code
  - ... to attacker specified value
- Modify data...
- Output data
  - ... to target code address
- Code corruption
- Control-flow hijack
- Data-only attack
- Information leak

- ASLR
- Stack canaries
- Non-exec. data pages
Bypassing ASLR with user scripting

- Make pointer out-of-bounds
- Make pointer dangling
- Use pointer to write
- Use pointer to read

Modify a data pointer
- Read-only code pages
  - ... to attacker specified code
- Modify a code pointer...
  - ... to target code address
  - Use pointer by indir. call/jmp
    - Exec. gadgets or functions
      - Control-flow hijack
      - Code corruption

- Modify data...
  - ... to attacker specified value
  - Use corrupted data variable
    - Non-exec. data pages
    - Stack canaries
      - Interpret the leaked value
      - Output data

- Modify code...
  - Data-only attack
  - Information leak
Bypassing stack cookies

Make pointer out-of-bounds
Make pointer dangling

Use pointer to write
Use pointer to read

Modify a code pointer...
... to target code address

Use pointer by indir. call/jmp
Use pointer by ret instruction

Exec. gadgets or functions

Non-exec. data pages

Code corruption

Modify a data pointer
... to attacker specified code

Read-only code pages

Modify data...
... to attacker specified value

Exec. gadgets or functions

Control-flow hijack

Modify data...
... to attacker specified value

Output data

Interpret the leaked value

Modify data...
... to attacker specified value

Interpret the leaked value

Use pointer by ret instruction

Use corrupted data variable

Read-only code pages

Non-exec. data pages

Control-flow hijack

Data-only attack

Information leak
Problems due to JIT compilation

1. Make pointer out-of-bounds
2. Make pointer dangling
3. Use pointer to write
4. Use pointer to read

- Modify a data pointer
  - ... to attacker specified code

- Modify code...
  - ... to attacker specified code

- Modify a code pointer...
  - ... to target code address

- Use pointer by indir. call/jmp
- Use pointer by ret instruction
- Execute gadgets or functions
- Execute injected code

- Modify data...
  - ... to attacker specified value

- Use corrupted data variable

- Output data
  - Interpret the leaked value

- Code corruption
- Control-flow hijack
- Data-only attack
- Information leak

Information leak
Make pointer out-of-bounds
Make pointer dangling
Use pointer to write
Use pointer to read

Modify a data pointer
Modify code...
... to attacker specified code

Modify a code pointer...
... to target code address

Use pointer by indir. call/jmp
Use pointer by ret instruction
Execute gadgets or functions
Execute injected code

Modify data...
... to attacker specified value

Use corrupted data variable

Output data
Interpret the leaked value
# Deployed protections

<table>
<thead>
<tr>
<th>Hijack protection</th>
<th>Policy</th>
<th>Technique</th>
<th>Weakness</th>
<th>Perf.</th>
<th>Comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>W⊕R</td>
<td>Page flags</td>
<td>JIT</td>
<td>1x</td>
<td>Good</td>
<td></td>
</tr>
<tr>
<td>Return integrity</td>
<td>Stack cookies</td>
<td>Direct overwrite</td>
<td>1x</td>
<td>Good</td>
<td></td>
</tr>
<tr>
<td>Address space rand.</td>
<td>ASLR</td>
<td>Info-leak.</td>
<td>1.1x</td>
<td>Good</td>
<td></td>
</tr>
</tbody>
</table>
Proposed solutions
Control-flow integrity

Make pointer out-of-bounds
Make pointer dangling

Use pointer to write
Use pointer to read

Modify a code pointer...
... to target code address

Use pointer by indir. call/jmp
Use pointer by ret instruction

Exec. gadgets or functions
Execute injected code

Code corruption

Modify data...
... to attacker specified value
Use corrupted data variable

Modify data...
... to attacker specified value
Interpret the leaked value

Output data

Data-only attack
Information leak

Control-flow hijack
Control-flow integrity

- Make pointer out-of-bounds
- Make pointer dangling
- Use pointer to write
- Use pointer to read
- Modify a code pointer...
- ... to target code address
- Modify a data pointer...
- ... to attacker specified code
- Modify code...
- Modify data...
- Output data
- Interpret the leaked value
- Use pointer by indir, call/jmp
- Use pointer by ret instruction
- Use corrupted data variable
- Exec. gadgets or functions
- Execute injected code
- Control-flow hijack
- Code corruption
- Data-only attack
- Information leak
p = &f
jmp p

f() {
    ...
}

q = &g
jmp q

g() {
    ...
}
p = &f
jmp p

f() {
    ...
}

q = &g
jmp q

g() {
    ...
}
p = &f
jmp p
f() {
  ...
}

q = &g
jmp q
g() {
  ...
}
check

p = &f
jmp p

if (...) 
q = &f
else
q = &g
jmp q

cffi

f() {
...
}
g() {
...
}
Over-approximation problem

```c
f() {
...
}

if (...) {
    q = &f
} else {
    q = &g
}

g() {
...
}
```
Over-approximation problem

```c
p = &f
jmp p

if (...) {
  q = &f
} else {
  q = &g
}
jmp q

printf() {
  ...
}

system() {
  ...
}
```
Modularity problem

```c
ID
printf() {
    ...
}

ID
system() {
    ...
}
```
## CFI

<table>
<thead>
<tr>
<th>Hijack protection</th>
<th>Policy</th>
<th>Technique</th>
<th>Weakness</th>
<th>Perf.</th>
<th>Comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>W⊕R</td>
<td>Page flags</td>
<td>JIT</td>
<td>1x</td>
<td>Good</td>
<td></td>
</tr>
<tr>
<td>Return integrity</td>
<td>Stack cookies</td>
<td>Direct overwrite</td>
<td>1x</td>
<td>Good</td>
<td></td>
</tr>
<tr>
<td>Address space rand.</td>
<td>ASLR</td>
<td>Info-leak.</td>
<td>1.1x</td>
<td>Good</td>
<td></td>
</tr>
<tr>
<td>Control-flow integ.</td>
<td>CFI</td>
<td>Over-approx.</td>
<td>1.4x</td>
<td>Libraries</td>
<td></td>
</tr>
</tbody>
</table>
Memory safety

- Make pointer out-of-bounds
- Make pointer dangling

- Use pointer to write
- Use pointer to read

Modify a data pointer
Modify code...
... to attacker specified code

Modify a code pointer...
... to target code address

Use pointer by indir. call/jmp
Use pointer by ret instruction

Exec. gadgets or functions
Execute injected code

Code corruption
Control-flow hijack

Modify data...
... to attacker specified value
Use corrupted data variable

Interpret the leaked value

Modify a code pointer...
... to attacker specified code

Modify data...
... to attacker specified value
Use corrupted data variable

Data-only attack
Information leak

Information leak
Memory safety

**Pointer metadata tracking & checking**

- Modify a data pointer
  - ... to attacker specified code

- Modify code...
  - ... to attacker specified code

- Modify a code pointer...
  - ... to target code address
    - Use pointer by indir. call/jmp
    - Exec. gadgets or functions
      - Execute injected code
    - Use pointer by ret instruction
      - Interpret the leaked value

- Modify data...
  - ... to attacker specified value
    - Use corrupted data variable
    - Interpret the leaked value

- Output data
  - Interpret the leaked value

- Control-flow hijack

- Code corruption

- Data-only attack

- Information leak
# SoftBounds+CETS

<table>
<thead>
<tr>
<th>Policy</th>
<th>Technique</th>
<th>Weakness</th>
<th>Perf.</th>
<th>Comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hijack protection</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W⊕R</td>
<td>Page flags</td>
<td>JIT</td>
<td>1x</td>
<td>Good</td>
</tr>
<tr>
<td>Return integrity</td>
<td>Stack cookies</td>
<td>Direct overwrite</td>
<td>1x</td>
<td>Good</td>
</tr>
<tr>
<td>Address space rand.</td>
<td>ASLR</td>
<td>Info-leak.</td>
<td>1.1x</td>
<td>Good</td>
</tr>
<tr>
<td>Control-flow integrity</td>
<td>CFI</td>
<td>Over-approx.</td>
<td>1.4x</td>
<td>Libraries</td>
</tr>
<tr>
<td><strong>Generic protection</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory safety</td>
<td>SB+CETS</td>
<td>None</td>
<td>2-4x</td>
<td>Good</td>
</tr>
</tbody>
</table>
Data integrity

- Make pointer out-of-bounds
- Make pointer dangling
- Use pointer to write
- Use pointer to read

Modify a data pointer
- ... to attacker specified code

Modify code...
- ... to attacker specified code

Modify a code pointer...
- ... to target code address

Modify data...
- ... to attacker specified value

Output data
- Interpret the leaked value

Use pointer by indir. call/jmp
- Exec. gadgets or functions
- Execute injected code

Use pointer by ret instruction

Use corrupted data variable
- Data-only attack

Code corruption
- Control-flow hijack

Information leak
Data integrity

- Make pointer out-of-bounds
- Make pointer dangling
- Use pointer to write
- Use pointer to read

Object metadata tracking & checking

- Modify a data pointer
- Modify code...
- Modify code pointer...
- Modify data...

- ... to attacker specified code
- ... to target code address
- ... to attacker specified value
- Interpret the leaked value

- Use pointer by indir. call/jmp
- Use pointer by ret instruction
- Use corrupted data variable
- Exec. gadgets or functions
- Execute injected code

- Code corruption
- Control-flow hijack
- Data-only attack
- Information leak

Output data
Valgrind / ASAN

\[ p = \text{alloc}(5) \]
\[ p[0] = v \]

\[ \text{int } q[3] \]
\[ v = q[1] \]
p = alloc(5)
p[0] = v

int q[3]
v = q[1]
<table>
<thead>
<tr>
<th>Policy</th>
<th>Technique</th>
<th>Weakness</th>
<th>Perf.</th>
<th>Comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hijack protection</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W⊕R</td>
<td>Page flags</td>
<td>JIT</td>
<td>1x</td>
<td>Good</td>
</tr>
<tr>
<td>Return integrity</td>
<td>Stack cookies</td>
<td>Direct overwrite</td>
<td>1x</td>
<td>Good</td>
</tr>
<tr>
<td>Address space rand.</td>
<td>ASLR</td>
<td>Info-leak.</td>
<td>1.1x</td>
<td>Good</td>
</tr>
<tr>
<td>Control-flow integrity</td>
<td>CFI</td>
<td>Over-approx.</td>
<td>1.4x</td>
<td>Libraries</td>
</tr>
<tr>
<td><strong>Generic protection</strong></td>
<td>SB+CETS</td>
<td>None</td>
<td>2-4x</td>
<td>Good</td>
</tr>
<tr>
<td>Memory safety</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Data integrity</strong></td>
<td>WIT</td>
<td>Over-approx., Use-after-frees, Invalid reads, Sub-objects</td>
<td>1.2x</td>
<td>Libraries</td>
</tr>
</tbody>
</table>
**Data space randomization**

- **Make pointer out-of-bounds**
- **Make pointer dangling**

- **Use pointer to write**
- **Use pointer to read**

- **Modify a data pointer**
  - ... to attacker specified code

- **Modify code ...**
  - ... to attacker specified code

- **Modify a code pointer...**
  - ... to target code address

- **Modify data ...**
  - ... to attacker specified value

- **Output data**
  - Interpret the leaked value

- **Use pointer by indir. call/jmp**
  - Use pointer by ret instruction

- **Exec. gadgets or functions**
  - Execute injected code

- **Code corruption**
- **Control-flow hijack**
- **Data-only attack**
- **Information leak**
Data space randomization

- Make pointer out-of-bounds
- Make pointer dangling

- Use pointer to write
- Use pointer to read

- Modify a code pointer...
- Modify data...
- Output data

Modify a data pointer
Modify code ...
... to attacker specified code

Use pointer by indir. call/jmp
Use pointer by ret instruction
Use corrupted data variable

Exec. gadgets or functions
Execute injected code

Code corruption
Control-flow hijack
Data-only attack
Information leak

DSR (memory encryption)
\( v = v \oplus K \)

\( p = \text{alloc}(5) \)

\( p[0] = v \)

\( v = v \oplus K \)

\( \text{int } q[3] \)

\( v = q[1] \)
\( v = v \oplus K \)

\( p = \text{alloc}(5) \)

\( p[0] = v \)

\( v = v \oplus K \)

\( \text{int } q[3] \)

\( v = q[1] \)

\( v = v \oplus K \)
<table>
<thead>
<tr>
<th>Policy</th>
<th>Technique</th>
<th>Weakness</th>
<th>Perf.</th>
<th>Comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hijack protection</td>
<td>W⊕R</td>
<td>Page flags</td>
<td>JIT</td>
<td>1x</td>
</tr>
<tr>
<td>Return integrity</td>
<td>Stack cookies</td>
<td>Direct overwrite</td>
<td>JIT</td>
<td>1x</td>
</tr>
<tr>
<td>Address space rand.</td>
<td>ASLR</td>
<td>Info-leak.</td>
<td>1.1x</td>
<td>Good</td>
</tr>
<tr>
<td>Control-flow integrity</td>
<td>CFI</td>
<td>Over-approx.</td>
<td>1.4x</td>
<td>Libraries</td>
</tr>
<tr>
<td>Generic protection</td>
<td>Memory safety</td>
<td>SB+CETS</td>
<td>None</td>
<td>2-4x</td>
</tr>
<tr>
<td>Data integrity</td>
<td>WIT</td>
<td>Over-approx., Info-leak</td>
<td>1.2x</td>
<td>Libraries</td>
</tr>
<tr>
<td>Data space rand.</td>
<td>DSR</td>
<td>Over-approx., Info-leak</td>
<td>1.3x</td>
<td>Libraries</td>
</tr>
</tbody>
</table>
Data-flow integrity

Make pointer out-of-bounds
Make pointer dangling

Use pointer to write
Use pointer to read

Modify a data pointer
Modify code ...
... to attacker specified code

Modify a code pointer...
... to target code address

Use pointer by indir. call/jmp
Use pointer by ret instruction

Exec. gadgets or functions
Execute injected code

Control-flow hijack
Code corruption

Modify data ...
... to attacker specified value

Use corrupted data variable

Output data
Interpret the leaked value

Data-only attack
Information leak

Make pointer out-of-bounds
Make pointer dangling

Use pointer to write
Use pointer to read

Modify a data pointer
Modify code ...
... to attacker specified code

Modify a code pointer...
... to target code address

Use pointer by indir. call/jmp
Use pointer by ret instruction

Exec. gadgets or functions
Execute injected code

Control-flow hijack
Code corruption

Modify data ...
... to attacker specified value

Use corrupted data variable

Output data
Interpret the leaked value

Data-only attack
Information leak
Data-flow integrity

- Make pointer out-of-bounds
- Make pointer dangling
- Use pointer to write
- Use pointer to read

- Modify a data pointer
- Modify code ...
  - ... to attacker specified code
- Modify a code pointer...
  - ... to target code address
- Modify data ...
  - ... to attacker specified value
- Output data
  - Interpret the leaked value

Use pointer by:
- indir, call/jmp
- ret instruction
- Use corrupted data variable

- Use pointer by:
- DFI
  - Exec. gadgets or functions
  - Execute injected code

- Exec. gadgets or functions
- Execute injected code

- Code corruption
- Control-flow hijack
- Data-only attack
- Information leak
<table>
<thead>
<tr>
<th>Policy</th>
<th>Technique</th>
<th>Weakness</th>
<th>Perf.</th>
<th>Comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>W⊕R</td>
<td>Page flags</td>
<td>JIT</td>
<td>1x</td>
<td>Good</td>
</tr>
<tr>
<td>Return integrity</td>
<td>Stack cookies</td>
<td>Direct overwrite</td>
<td>1x</td>
<td>Good</td>
</tr>
<tr>
<td>Address space rand.</td>
<td>ASLR</td>
<td>Info-leak.</td>
<td>1.1x</td>
<td>Good</td>
</tr>
<tr>
<td>Control-flow integrity</td>
<td>CFI</td>
<td>Over-approx.</td>
<td>1.4x</td>
<td>Libraries</td>
</tr>
<tr>
<td>Memory safety</td>
<td>SB+CETS</td>
<td>None</td>
<td>2-4x</td>
<td>Good</td>
</tr>
<tr>
<td>Data integrity</td>
<td>WIT</td>
<td>Over-approx.,...</td>
<td>1.2x</td>
<td>Libraries</td>
</tr>
<tr>
<td>Data space rand.</td>
<td>DSR</td>
<td>Over-approx.,...</td>
<td>1.3x</td>
<td>Libraries</td>
</tr>
<tr>
<td>Data-flow integrity</td>
<td>DFI</td>
<td>Over-approx.</td>
<td>2-3x</td>
<td>Libraries</td>
</tr>
<tr>
<td>Policy</td>
<td>Technique</td>
<td>Weakness</td>
<td>Perf.</td>
<td>Comp.</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------</td>
<td>---------------------------</td>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>Hijack protection</td>
<td>W⊕R</td>
<td>Page flags</td>
<td>JIT</td>
<td>1x</td>
</tr>
<tr>
<td>Return integrity</td>
<td>Stack cookies</td>
<td>Direct overwrite</td>
<td>1x</td>
<td>Good</td>
</tr>
<tr>
<td>Address space rand.</td>
<td>ASLR</td>
<td>Info-leak.</td>
<td>1.1x</td>
<td>Good</td>
</tr>
<tr>
<td>Control-flow integ.</td>
<td>CFI</td>
<td>Over-approx.</td>
<td>1.4x</td>
<td>Libraries</td>
</tr>
<tr>
<td>Memory safety</td>
<td>SB+CETS</td>
<td>None</td>
<td>2-4x</td>
<td>Good</td>
</tr>
<tr>
<td>Data integrity</td>
<td>WIT</td>
<td>Over-approx.,...</td>
<td>1.2x</td>
<td>Libraries</td>
</tr>
<tr>
<td>Data space rand.</td>
<td>DSR</td>
<td>Over-approx.,...</td>
<td>1.3x</td>
<td>Libraries</td>
</tr>
<tr>
<td>Data-flow integrity</td>
<td>DFI</td>
<td>Over-approx.</td>
<td>2-3x</td>
<td>Libraries</td>
</tr>
</tbody>
</table>
Questions