Compiler-Assisted Automatic Error Detection

Seminar on Software-Based Fault-Tolerance

Jan Bessai





Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

MASK

How it work

AN-Encoding

Aspect-Oriented Implementation

SWIFT

How it works Scope

valuation

Methodology Studies and Results

Conclusion and

Summary

Feedback

Outline Motivation

Motivation

MASK

How it works

Scope

AN-Encoding

How it works

Aspect-Oriented Implementation

Scope

SWIFT

How it works

Scope

Evaluation

Methodology

Studies and Results

Conclusion and discussion

Summary

Outlook

Feedback

Compiler-Assisted Automatic Error Detection

Jan Bessai

otivation

. . . .

ASK

ope

N-Encoding

ow it works

spect-Oriented applementation

VVIFI

How it works

pe

aluation

ethodology udies and Results

Conclusion a

discussion

Summary

Feedback

Why use a Compiler?

Compiler-Assisted Automatic Error Detection

Jan Bessai

otivation

Motivation

MASK

How it work

AN-Encoding

How it works

Aspect-Oriented Implementation

SWIFT

SWIFT

How it works

ope

/aluation

lethodology

viethodology Studies and Results

Conclusion and

discussion Summary

Outlook

References

No need to change existing code:

- ▶ implement and test protection methods only once
- create variants by switching compile flags
- separate aspect of protection from program functionality

Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

Motivation

WASIC

How it works

AN-Encodin

How it works
Aspect-Oriented
Implementation

SWIFT

How it works

Methodology Studies and Results

Conclusion and

Summary Outlook

Feedbac

References

MASK

Example

Compiler-Assisted Automatic Error Detection

Jan Bessai

```
otivation
```

MASK

How it works

```
How it works
```

Aspect-Oriented Implementation

SWIFT

How it works

valuation

Evaluation Methodology

Methodology Studies and Results

Studies and Resul

Conclusion and discussion

discussion

Outlook

Feedback

eferences

```
1 mov ecx, 0 ; init counter
2 and ecx, 0xFFFE
3 again:
4   push ecx
5   call innerLoopFun ; with param ecx
6   add ecx, 2 ; increase counter by 2
7   and ecx, 0xFFFE
8   cmp ecx, 10 ; loop if ecx < 10
9   jl again</pre>
```

Observation:

- ▶ last bit of ecx is always zero
- ▶ invariant enforced by bitmask 0xFFFE

Explaination

Containment of errors

- static analysis to find invariants
 - e.g. Bitvector analysis on single bits (known from constant propagation)
- invariants enforced with bitmasks throughout code
- only bits really involved in computations can flip

Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

AACK

How it works

Scope

N-Encoding

ow it works spect-Oriented oplementation

SWIFT

How it works

alustion

Valuation Methodology

Methodology Studies and Results

Conclusion and

Summary Outlook

. .

What MASK can detect/prevent

- transient and permanent faults in masked bits
- flipped operands (if constraints are violated)
- ▶ invalid jump addresses (known from NX-Bit)

Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

MASK

How it works

AN-Encoding

How it works

Aspect-Oriented Implementation

S\//IFT

How it works

Scope

valuation

Methodology

Conclusion and

discussion

Outlook

Feedback

What MASK cannot do

- protect bits which are expected to change
- ensure application of bitmasks does not fail
- strength of protection depends on capabilities of the compiler

Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

1410114111011

How it works

Scope

AN-Encoding

How it works

Aspect-Oriented Implementation

SWIFT

How it works

aluation

Methodology Studies and Results

Conclusion and

Summary Outlook

Feedback

Implementation considerations

- tight coupling with compiler analysis code
- intimate knowledge of target architecture required
- consideration of side effects
 - e.g. if register indirect commands are involved

Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

.

How it works

Scope

AIN-Encoding

Aspect-Oriented

Scope

S\//IFT

SVVIFI

How it works

valuation

Nethodology

Studies and Result

Conclusion and

Summary

Feedback

Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

How it works

AN-Encoding

How it works

Aspect-Oriented Implementation

SWIFT

How it works

valuation

Methodology Studies and Results

Conclusion and

Summary

Feedback

References

AN-Encoding

Checking with Modulo

To protect the arithmetic operations + and *:

- ► choose a big constant *A*
- multiply unencoded values x, y with A:

$$A \cdot x + A \cdot y \mod A = A \cdot (x + y) \mod A = 0$$

 $(A \cdot x) \cdot (A \cdot y) \mod A = A^2 \cdot (x \cdot y) \mod A = 0$

Protect logic by replacement with arithmetic

$$\rightarrow$$
 $\neg x = 1 - x$, $x \land y = x \cdot y$, ...

Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

MARK

How it works

N-Encoding

How it works

Aspect-Oriented Implementation

SWIFT

ow it works

valuation

valuation

Methodology Studies and Results

Conclusion and discussion

Outlook

Properties

Hamming distance of valid words is spread:

- A = 127
- ▶ unencoded: 0000 0000 ^{1 Bit flipped} 0000 0001
- ► encoded: 0000 0000 ^{7 Bits flipped} 0111 1111
- \Rightarrow Datatypes have to grow by a factor of 4:

$$\underbrace{\underbrace{A \cdot x \cdot A \cdot y}^{\times 2}}_{\times 4}$$

Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

Motivation

MASK

How it works Scope

AN-Encoding

How it works

Aspect-Oriented Implementation

SWIFT

How it works

..........

Mathadalam

Studies and Results

discussion

Summary Outlook

Feedback

Choice of A

Mersenne primes $2^n - 1$ are optimal for A

- off by $|(2^n 1) 2^k| > 0$, if bit k flips
- multiplication shortcut: $x \cdot (2^n 1) = (x << n) x$
- no way to create A from factors

Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

Motivation

MASK

How it works Scope

AN-Encoding

How it works

Aspect-Oriented mplementation

SWIFT

How it works

Scope

/aluation

Methodology

Conclusion and

Summary

Outlook

Weaving in AN-Encoding

Implementation can be done using AspectC++

- provide a way to annotate values to protect
- aspect to change datatypes
- aspect to perform encoding
- available on GitHub: https://github.com/JanBessai/anencoding

Compiler-Assisted Automatic Error Detection

Ian Bessai

Aspect-Oriented Implementation

How it works

Transparent container-based annotation

Create a container Protected<T>:

- constructor takeing value of type T
- member value of type Encoded<T>::EncodedType
- arithmetic operations
- cast back to type T

Type encoding:

```
1 template <class T>
2 class Encoded {
3  public:
4   typedef T EncodedType;
5 };
```

Compiler-Assisted Automatic Error

Jan Bessai

Notivation

MARK

MASK

Scope

AN-Encoding

How it works

Aspect-Oriented Implementation

SWIFT

How it works

cope

valuation

Methodology

Studies and Results

discussion an

Summary Outlook

Feedback

Aspect-Oriented Implementation

How it works

Provide template specializations of Encoded<T>:

```
1template <>
2 class Encoded<char> {
   typedef int EncodedType;
4 };
6 template <>
7 class Encoded < int > {
   typedef mpz_class EncodedType;
9 };
```

Note: GNU Multiple Precision Arithmetic Library used for 128 bit integers (mpz_class)

Jan Bessai

```
1 aspect ANEncode {
    static const int A = 127:
3
    pointcut encode() =
      call("% protect < ... > (...)");
    advice encode() : after() {
      tjp->result()->value *= A;
q
    pointcut decode() =
10
      call("% Protected<...>::operator %()")
11
      && !negation()
12
    advice decode() : after() {
13
      *(tjp->result()) /= A;
14
    }
15
16
17 }:
```

Motivation

Mativation

MASK

low it works

AN-Encodin

How it work

Aspect-Oriented Implementation

....

VVIFI

How it works Scope

aluation

lethodology

Studies and Results

discussion discussion

Summary

Feedback

```
Compiler-Assisted
Automatic Error
Detection
```

Jan Bessai

```
pointcut addition() =
1
      call("% Protected<...>::operator+(...)");
2
   pointcut multiplication() =
3
      call("% Protected<...>::operator*(...)");
5
    . . .
   pointcut checkable() =
      addition() || multiplication() || ...;
7
    advice checkable() : after() {
      if (tjp->result()->value % A != 0) {
g
        cout << "Error detected!" << endl:
10
        cout << tip->result()->value << endl;</pre>
11
12
   }
13
14
    advice multiplication() : after() {
15
      tjp->result()->value /= A;
16
17
```

Motivation

Motivation

MANCK

MASK

Scope

N-Encodin

dan ta mada

Aspect-Oriented Implementation

cope

5VVIF I

Coops

aluation

aluation

Studies and Results

Conclusion and discussion

Summary Outlook

Feedback

What AN-Encoding can detect

Redundancy in encoding

- transient and permanent faults in arithmetic/logic operations (if modulo works..)
- extensions (ANB, ANBD) to find exchanged operands and faulty jumps:

$$A \cdot x + A \cdot y + B_x + B_y + D \mod A =$$
 $A \cdot (x + y) + B_x + B_y + D \mod A = B_x + B_y + D$
 $(A \cdot x) \cdot (A \cdot y) + B_x + B_y + D \mod A =$
 $A^2 \cdot (x \cdot y) + B_x + B_y + D \mod A = B_x + B_y + D$

- \triangleright B_x, B_y : operand identifiers
- ▶ D: identifier for expected source position

Compiler-Assisted Automatic Error

Jan Bessai

Motivation

MASK

How it works Scope

N-Encoding

ow it works spect-Oriented aplementation

Scope

SVVIF I

How it works Scope

Evaluation

Methodology Studies and Results

Conclusion and discussion

Outlook

Feedback

Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

....

How it works

AN-Encoding

How it works
Aspect-Oriented
Implementation

SWIFT

How it works

Scono

Evaluation

Methodology
Studies and Results

Conclusion and

Summary

Feedbac

References

SWIFT

Example

Compiler-Assisted Automatic Error Detection

Ian Bessai

How it works

Scope

```
1push eax ; safe eax
2 add eax, ebx; compute
3 push eax; safe result
4 mov eax, [esp + 4]; restore eax
sadd eax, ebx; compute again
6 push ebx; safe ebx
7 mov ebx, [esp + 4]; load previous result
8 cmp eax, ebx ; compare results
9 jeq ok
10 call errorHandler; on mismatch
11 ok: ...
```

Explaination

Detection Jan Bessai

Compiler-Assisted

Automatic Error

Notivation

riotivation

How it works

Scope

AN-Encoding

How it works Aspect-Oriented

Scone

SWIFT

How it works

cope

...........

valuation

Methodology

Conclusion and

Summary

Outlook

eferences

Most intuitive form of error detection

repeat operations, compare results

Errors can even be repaired:

compute three times (or more) and use most frequent result

What SWIFT can(not) prevent

Redundancy of operations:

- protects from transient faults only
- applicable for (almost) every operation
 - ▶ if no side effects are present
- no way to deal with faulty jumps

Compiler-Assisted Automatic Error Detection

Ian Bessai

How it works

Scope

Implementation considerations

- intimate knowledge of target architecture is required (as with MASK)
- performance not only decreased by factor of two
 - ► additional register pressure
 - commands to safe and restore redundant results
 - code to compare (and vote)

Compiler-Assisted Automatic Error

Jan Bessai

Motivation

.

MASK

Scope Scope

AN-Encoding

AIN-Elicouling

spect-Oriented

Scope

SWIFT

How it works

Scope

valuation

Methodology Studies and Result

Conclusion and

Summary

Outlook

Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

....

How it works

Scope

AN-Encoding

How it works Aspect-Oriented Implementation

S\//IFT

Evaluation

How it works

Scope

Methodology

Studies and Results

Studies and Results

Conclusion and

Summary

Outlook

Feedbac

Things to consider

Compiler-Assisted Automatic Error Detection

Jan Bessai

lotivation

IVIASIN

How it work

AN-Encoding

How it works

Aspect-Oriented mplementation

Scope

SWIFT

How it works

ow it works

valuation

Evaluation

Methodology

Canalusian and

discussion a

Summary

Feedback

eferences

Empirical evaluation based on 3 decisions:

- programs under test
- fault model (which kinds of errors?)
- performance measure for
 - errors prevented
 - overhead introduced

Next the two most complete studies are presented

Study by Schiffel, Schmitt, Süßkraut and Fetzer

Compiler-Assisted Automatic Error Detection

Jan Bessai

/lotivation

motivatio

How it wor

A.A. -

AN-Encoding

Aspect-Oriented

Scope

SWIFT

low it wor

valuation

Methodology

Studies and Results

Conclusion and discussion

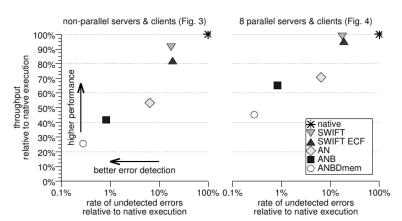
Summary Outlook

References

"Software-Implemented Hardware Error Detection: Costs and Gains" [2]:

- compared SWIFT and AN-Encoding
- ▶ tested 6 standard programs (e.g. md5, primes)
- considered faulty operands, lost operations, permanent and burst faults
- analyzed cases of Silent Data Corruption (SDC)
- measured overhead in a client-server scenario as network throughput
 - included parallel execution

Study by Schiffel, Schmitt, Süßkraut and Fetzer (Results)



- EFC: Ehanced Controflow Checking (redundant jump targets)
- ► ANBDmem: D as memory operation counter

Compiler-Assisted Automatic Error Detection

Jan Bessai

/lotivation

VCK

How it w

N-Encoding

How it works
Aspect-Oriented
mplementation
Scope

WIFT

low it works

Evaluation Methodology

Studies and Results
Conclusion an

Summary Outlook

. .

Study by Reis, Chung and August

Compiler-Assisted Automatic Error Detection

Jan Bessai

/lotivation

MACK

How it work

AN-Encoding

How it works Aspect-Oriented

Scope

WIFT

low it works

valuation

Methodology

Studies and Results

Conclusion and discussion

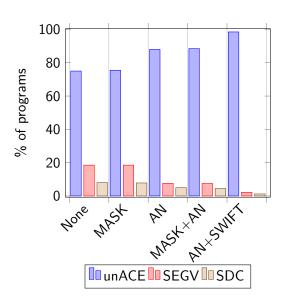
Outlook

References

"Automatic Instruction-Level Software-Only Recovery" [1]:

- compared SWIFT, Trump (AN-Encoding + unencoded copy for recovery) and MASK
- ▶ tested 21 different programs
- considered Single Event Upset (SEU) faults involving 1 flipped Bit only
- analyzed SDC, Segfaults (SEGV) and errors in data unnecessary for architecutally correct execution (unACE)
- measured overhead in execution time

Study by Reis, Chung and August (Results)



Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

.

How it works

AN-Encoding

low it works spect-Oriented mplementation

SWIFT

How it works

aluation

lethodology

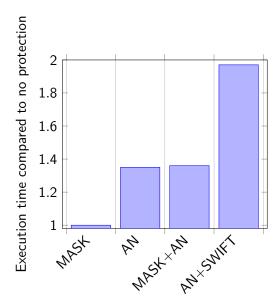
Studies and Results

Conclusion discussion

Outlook

Deferences

Study by Reis, Chung and August (Results)



Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

MASK

Scope Scope

N-Encoding

How it works Aspect-Oriented Implementation

WIFT

How it works

aluation

Methodology Studies and Results

Conclusion ar

discussion

Outlook

Conclusion and discussion

Compiler-Assisted Automatic Error

Jan Bessai

Motivation

Motivation

MASK

How it works Scope

AN-Encoding

How it works
Aspect-Oriented
Implementation

SWIFT

How it works Scope

Methodology Studies and Results

Conclusion and

discussio Summary

Outlook

Feedbac

Methods seen

Compiler-Assisted Automatic Error Detection

Jan Bessai

lotivation

low it works

...

How it works

Aspect-Oriented Implementation

S\//IET

SVVIFI

How it works

..........

valuation

Methodology

Conclusion and

discussion Summary

Outlook

- MASK: Static analysis for containment of errors
- ► AN-Encoding: Value level redundancy by multiplication
 - ► Fine grain Aspect-Oriented implementation
- SWIFT: Operation level redundancy by repetition

Evaluation results

- compilers can reduce error rates up to 90%
- runtime degrades by a factor of about 2
- no method can eliminate all errors
- all methods add code which can fail

Compiler-Assisted Automatic Error Detection

Ian Bessai

How it works

Summary

Further research topics

- standardized production grade compilers
- better control over which parts of a program need protection
- Multi Strategy Beta Reduction (see seminar paper)
- standardized benchmarks

Compiler-Assisted Automatic Error Detection

Jan Bessai

Motivation

MASK

How it works

AN-Encoding

ow it works spect-Oriented aplementation

SWIFT

How it works

/aluation

Methodology

Conclusion and

Summary Outlook

Feedback

Feedback or Questions?

Compiler-Assisted Automatic Error Detection

Jan Bessai

/lotivation

MACK

How it works

AN-Encoding

How it works Aspect-Oriented Implementation

SWIFT How it works

Scope

valuation

Methodology Studies and Results

Conclusion and

Summary Outlook

Feedback

References

► Thank you :) !

Compiler-Assisted Automatic Error

Jan Bessai

Motivation

MASK

How it work

AN-Encoding

How it works Aspect-Oriented Implementation

SWIFT

How it works

valuation

Methodology Studios and Populte

Conclusion and

Summary Outlook

- Reis, G.A., Chang, J., August, D.I.: Automatic Instruction-level Software-Only Recovery. IEEE Micro 27, 36-47 (2007), http://doi.ieeecomputersociety. org/10.1109/MM.2007.4
- [2] Schiffel, U., Schmitt, A., Süßkraut, M., Fetzer, C.: Software-Implemented Hardware Error Detection: Costs and Gains. In: The Third International Conference on Dependability (DEPEND 2010). pp. 51–57. IEEE Computer Society, Los Alamitos, CA, USA (2010)