MINES Saint-Étienne

Une école de l'IMT

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

PROJET FINANCÉ PAR L'ANR - PROJECT FUNDED BY THE ANR

FROM RESEARCH TO INDUSTRY cea tech

INVIA

PROJET COFFI ANR-18-CES39-003

VERIFICATION OF THE PROGRAM'S CFI BASED ON A TRACE ENCODER

Anthony ZGHEIB

Journée thématique sur les attaques par injection de fautes

23 Septembre 2021

INSPIRING INNOVATION SINCE 1816

Table of Contents

Table of Contents

Introduction

Why is it necessary to guarantee the program's CFI?

- Software attacks : Buffer Overflow, ROP, Code Reutilization Attacks...
- Hardware attacks : Fault Injection...
- Example - Code pin verification :

```
      int counter = 3;
      void VerifyPin() {
1 →     if (counter > 0)
2 →       if (Cmp(userPIN,devicePIN))
             Accept();
          else
3 →          counter--;
      }
```

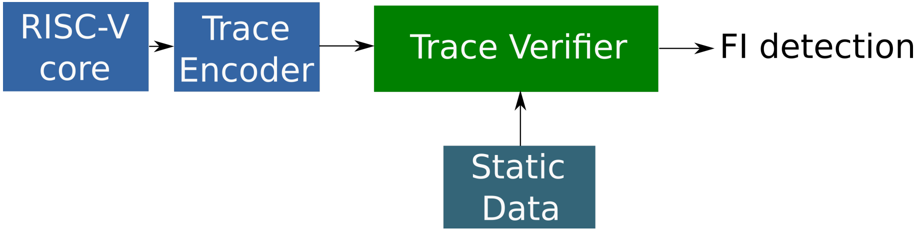
- **CFI Verification = Correct Program Execution.**

Table of Contents

How could we verify the program's CFI?

Methodology

- Get information about what is executed at the RISC-V core level => **Trace Encoder (TE)**
- Compare these data to static data obtained from a static analysis of the binary program => **Trace Verifier (TV)**
- Detect if a fault injection attack is made => TV's output

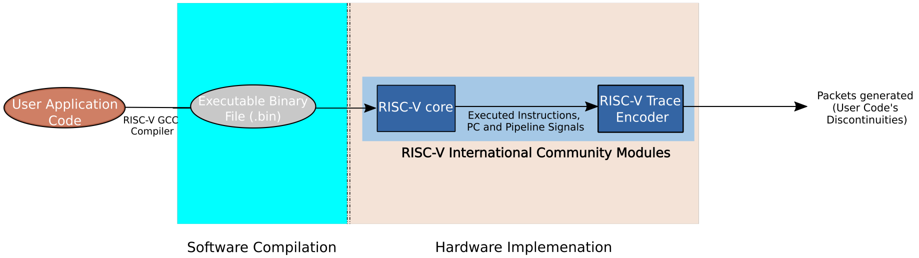


Trace Encoder

Definition

- Module designed by the RISC-V community.
- Overall objective: Compression of the program's execution path.
- Interpret the executed instructions from the RISC-V core.
- Report the discontinuities present in a program in the form of packets.
  - Instructions presenting an uninferable PC discontinuity.
  - Interruptions and exceptions...

Packet generation example

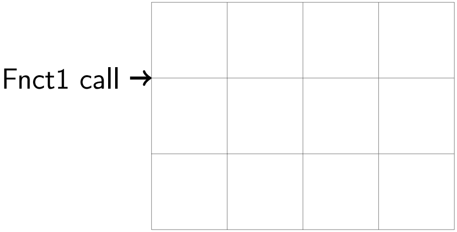

User Application Code
RISC-V GCC Compiler
Executable Binary File (.bin)
RISC-V core
Executed Instructions, PC and Pipeline Signals
RISC-V Trace Encoder
RISC-V International Community Modules
Packets generated (User Code's Discontinuities)
Software Compilation
Hardware Implemenation

Example of a packet - Use Case

- Having a function call where the program encounter n branches.
- A packet will be sent containing : the number of branches (n), the branch map (branch taken or not) and the return address ...

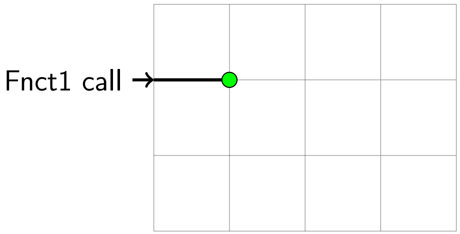
Packet generation example

```
int fnct1(int a, int b, int c) {
        if (a=1) {
                ...
                if(b=2) {
                        ...
                } else {
                        ...
                }
                end if;
        } else {
                ...
        }
        if(c=a+b) {
                ...
        } else {
                ...
        }
return a+b+c;
}

int main() {

int a = 1;
int b = 3;
int c = 4;

c=fnct1(a,b,c);

return 0;
}
```



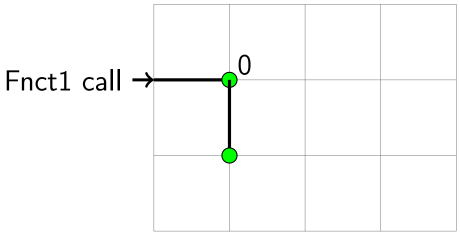
Packet generation example

```
  int fnct1(int a, int b, int c) {
1 →     if (a=1) {
                ...
                if(b=2) {
                        ...
                } else {
                        ...
                }
                end if;
        } else {
                ...
        }
        if(c=a+b) {
                ...
        } else {
                ...
        }
  return a+b+c;
  }

  int main() {

  int a = 1;
  int b = 3;
  int c = 4;

  c=fnct1(a,b,c);

  return 0;
  }
```



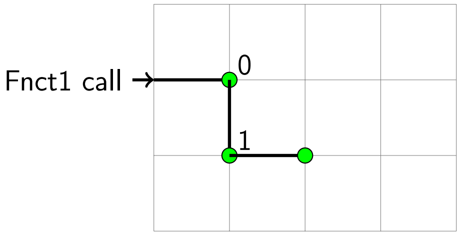
Packet generation example

```
    int fnct1(int a, int b, int c) {
1 →         if (a=1) {
                    ...
2 →                 if(b=2) {
                            ...
                    } else {
                            ...
                    }
                    end if;
            } else {
                    ...
            }
            if(c=a+b) {
                    ...
            } else {
                    ...
            }
    return a+b+c;
    }

    int main() {

    int a = 1;
    int b = 3;
    int c = 4;

    c=fnct1(a,b,c);

    return 0;
    }
```

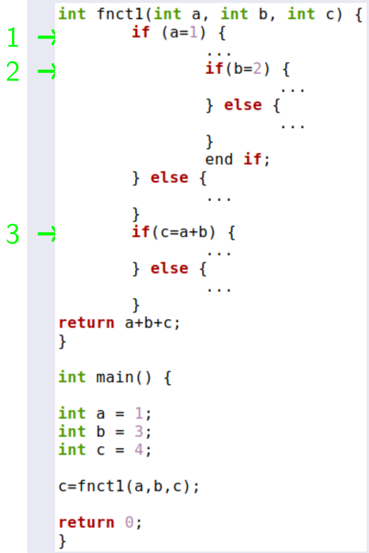


Packet generation example

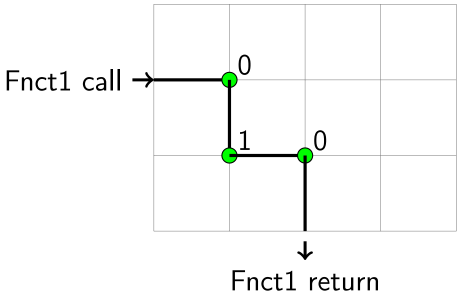
```
    int fnct1(int a, int b, int c) {
1 →         if (a=1) {
                    ...
2 →                 if(b=2) {
                            ...
                    } else {
                            ...
                    }
                    end if;
            } else {
                    ...
            }
3 →         if(c=a+b) {
                    ...
            } else {
                    ...
            }
    return a+b+c;
    }

    int main() {

    int a = 1;
    int b = 3;
    int c = 4;

    c=fnct1(a,b,c);

    return 0;
    }
```



Packet generation example



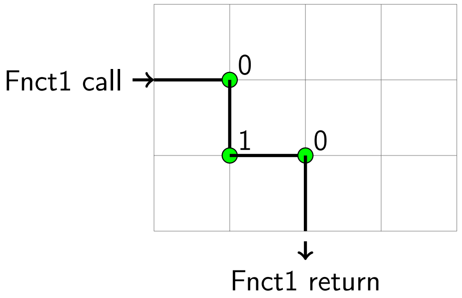
```
int fnct1(int a, int b, int c) {
        if (a=1) {
                ...
                if(b=2) {
                        ...
                } else {
                        ...
                }
                end if;
        } else {
                ...
        }
        if(c=a+b) {
                ...
        } else {
                ...
        }
return a+b+c;
}

int main() {

int a = 1;
int b = 3;
int c = 4;

c=fnct1(a,b,c);

return 0;
}
```



Packet generation example

1 →
2 →
3 →

```
int fnct1(int a, int b, int c) {
        if (a=1) {
                ...
                if(b=2) {
                        ...
                } else {
                        ...
                }
                end if;
        } else {
                ...
        }
        if(c=a+b) {
                ...
        } else {
                ...
        }
return a+b+c;
}

int main() {

int a = 1;
int b = 3;
int c = 4;

c=fnct1(a,b,c);

return 0;
}
```



Example (Generated packet)

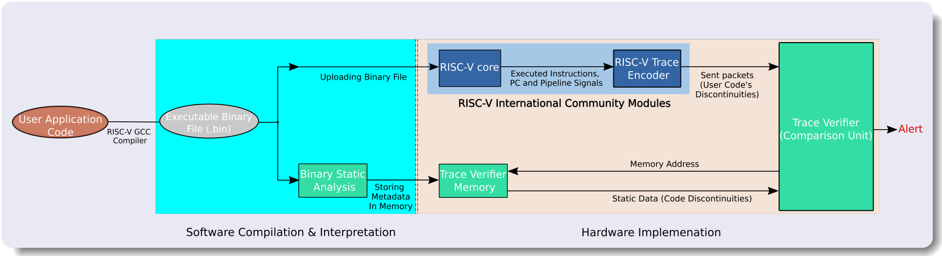
- Number of branches : 3
- Branch Map : 010
- Return address

Trace Verifier

- It's verification system based on the TE sent packets.
- Compare the TE packets to static data issued from a static analysis of the binary program.
    - A static analysis is made after the compilation process.
    - Branch, jump, call and return instructions with their addresses are stored in a memory.
- A flag is raised when a fault is detected.

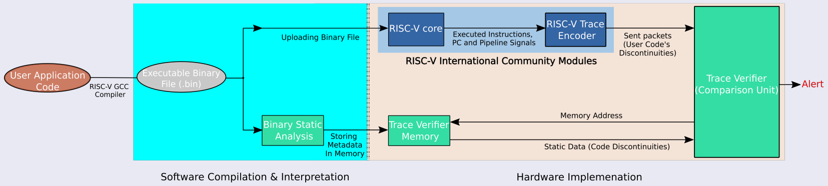
Table of Contents

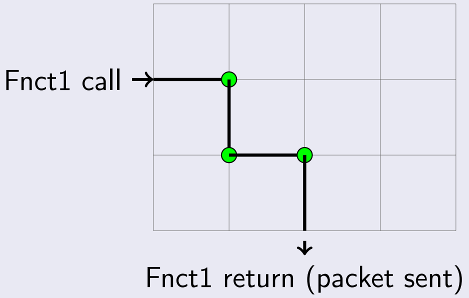
Exploited solution - Architecture



User Application Code
RISC-V GCC Compiler
Executable Binary File (.bin)
Uploading Binary File
Binary Static Analysis
Storing Metadata In Memory
RISC-V core
Executed Instructions, PC and Pipeline Signals
RISC-V Trace Encoder
RISC-V International Community Modules
Sent packets (User Code's Discontinuities)
Trace Verifier (Comparison Unit)
Alert
Memory Address
Trace Verifier Memory
Static Data (Code Discontinuities)
Software Compilation & Interpretation
Hardware Implemenation

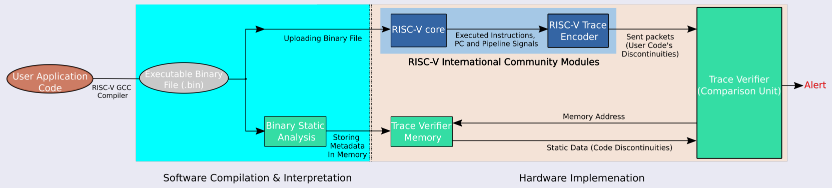
First approach - ASIC RISC-V Model


User Application Code
RISC-V GCC Compiler
Executable Binary File (.bin)
Uploading Binary File
Binary Static Analysis
Storing Metadata In Memory
RISC-V core
Executed Instructions, PC and Pipeline Signals
RISC-V Trace Encoder
RISC-V International Community Modules
Sent packets (User Code's Discontinuities)
Trace Verifier (Comparison Unit)
Alert
Memory Address
Trace Verifier Memory
Static Data (Code Discontinuities)
Software Compilation & Interpretation
Hardware Implemenation

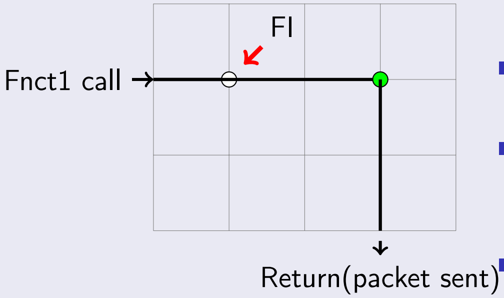


- Verification process starts when a packet is sent.
- Navigation through the static data and constitution of the path followed by the program.
- Comparison process.

First approach - ASIC RISC-V Model

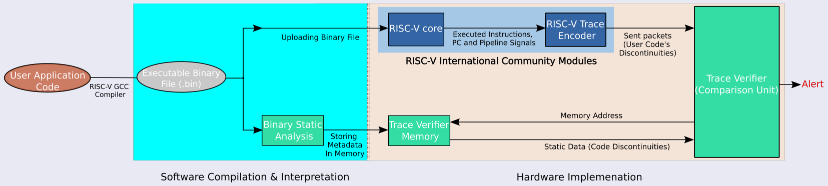


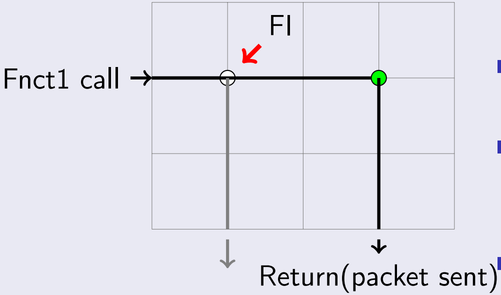
User Application Code
RISC-V GCC Compiler
Executable Binary File (.bin)
Uploading Binary File
Binary Static Analysis
Storing Metadata In Memory
RISC-V core
Executed Instructions, PC and Pipeline Signals
RISC-V Trace Encoder
RISC-V International Community Modules
Sent packets (User Code's Discontinuities)
Trace Verifier (Comparison Unit)
Alert
Memory Address
Trace Verifier Memory
Static Data (Code Discontinuities)
Software Compilation & Interpretation
Hardware Implemenation



- Verification process starts when a packet is sent.
- Navigation through the static data and constitution of the path followed by the program.
- Comparison process.

First approach - ASIC RISC-V Model


User Application Code
RISC-V GCC Compiler
Executable Binary File (.bin)
Uploading Binary File
Binary Static Analysis
Storing Metadata In Memory
RISC-V core
Executed Instructions, PC and Pipeline Signals
RISC-V Trace Encoder
RISC-V International Community Modules
Sent packets (User Code's Discontinuities)
Trace Verifier (Comparison Unit)
Alert
Memory Address
Trace Verifier Memory
Static Data (Code Discontinuities)
Software Compilation & Interpretation
Hardware Implemenation



- Verification process starts when a packet is sent.
- Navigation through the static data and constitution of the path followed by the program.
- Comparison process.

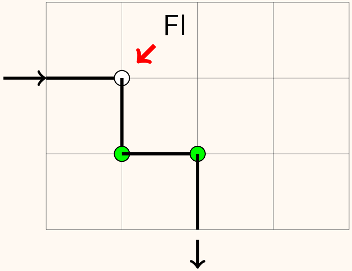
First approach - FI detection

Fault coverage

- Changing the return address of a function.
- Instruction skip on a function call.

Particular cases

- Instruction skip on branch instructions.
- It depends on the branch number and return address.



Limitations

- Corruption of a branch instruction (funct3, branch address ...).

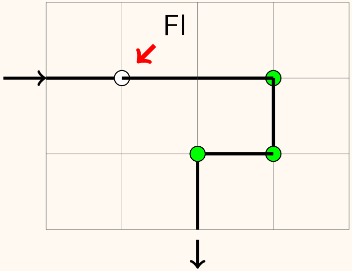
First approach - FI detection

Fault coverage

- Changing the return address of a function.
- Instruction skip on a function call.

Particular cases

- Instruction skip on branch instructions.
- It depends on the branch number and return address.



Limitations

- Corruption of a branch instruction (funct3, branch address ...).

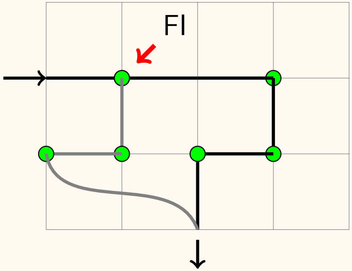
First approach - FI detection

Fault coverage

- Changing the return address of a function.
- Instruction skip on a function call.

Particular cases

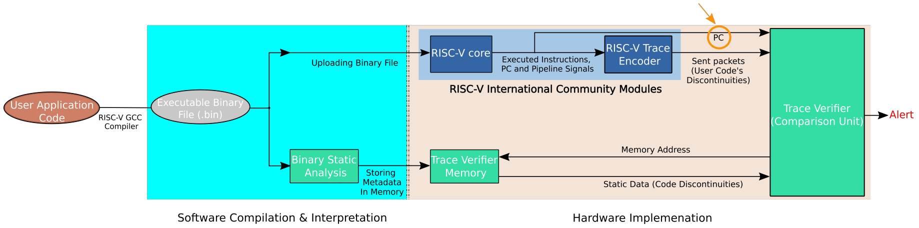
- Instruction skip on branch instructions.
- It depends on the branch number and return address.



Limitations

- Corruption of a branch instruction (funct3, branch address ...).

Second approach - Adding the PC - RISC-V FPGA Model



User Application Code
RISC-V GCC Compiler
Executable Binary File (.bin)
Uploading Binary File
Binary Static Analysis
Storing Metadata In Memory
RISC-V core
Executed Instructions, PC and Pipeline Signals
RISC-V Trace Encoder
PC
Sent packets (User Code's Discontinuities)
RISC-V International Community Modules
Trace Verifier (Comparison Unit)
Alert
Memory Address
Trace Verifier Memory
Static Data (Code Discontinuities)
Software Compilation & Interpretation
Hardware Implemenation

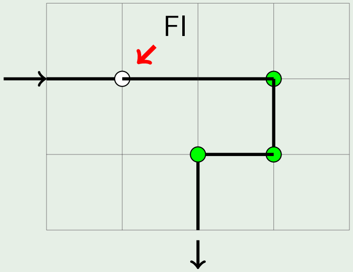
Features

- We pull the PC and connect it to the TV.
- Calculation is made before receiving the packet.
- Verification process is faster compared to the first approach.

Second approach - FI detection

Fault coverage

- Changing the return address of a function.
- Instruction skip on a function call and **branch instruction**.
  - Total number of branches=4.



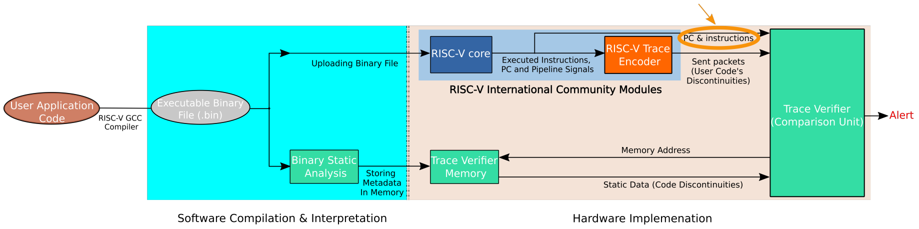
Particular cases

- Changing the branch address in case it was taken : beq a4,a5,**180**.

Limitations

- Corruption of a branch instruction (funct3, branch address ...).

Third approach - Redefining the TE RISC-V standard



User Application Code
RISC-V GCC Compiler
Executable Binary File (.bin)
Uploading Binary File
Binary Static Analysis
Storing Metadata In Memory
RISC-V core
Executed Instructions, PC and Pipeline Signals
RISC-V Trace Encoder
RISC-V International Community Modules
PC & instructions
Sent packets (User Code's Discontinuities)
Trace Verifier (Comparison Unit)
Alert
Memory Address
Trace Verifier Memory
Static Data (Code Discontinuities)
Software Compilation & Interpretation
Hardware Implemenation

Features

- PC and executed instructions are pulled and connected to the TV.
- Adjustment to the TE RISC-V standard by defining the qualified instructions (jump, branch, return...).
- Packet is sent after each qualified instruction.

Third approach - FI detection

Fault coverage

- Changing the return address of a call function.
- Instruction skip on a function call and branch instruction.
- Their corruption / substitution with other instructions.

Table of Contents

Results - Comparison between the 3 approaches

- Each approach covers a specific number of threats :

Approach	SFC	RAC	SBI	CDI	L
TV-ASIC	✓	✓	(X)	X	- -
TV-FPGA-PC	✓	✓	✓	X	-
TE-TV-CFI	✓	✓	✓	✓	+

- SFC : Skip on function calls.
- RAC : Return address change.
- SBI : Skip on branch instructions.
- CDI : Corruption of a discontinuity instruction.
- L : Latency.

Conclusion - Comparison between the 3 approaches

- Hardware Area Overhead (in terms of slices) :

Approach	TE	TV	**Total**
TV-ASIC	241	360	601
TV-FPGA-PC	241	641	882
TE-TV-CFI	95	575	670

- RISC-V IBEX : 635 slices.

Table of Contents

Perspectives

- Definition of a new packet for the basic blocks' verification, by adjusting the TE's standard.
- Verification of the correct instructions execution in the processor pipeline (cf. COFFI Project).

Thank you for your attention