Secure Calling Convention with Uninitialized Capabilities

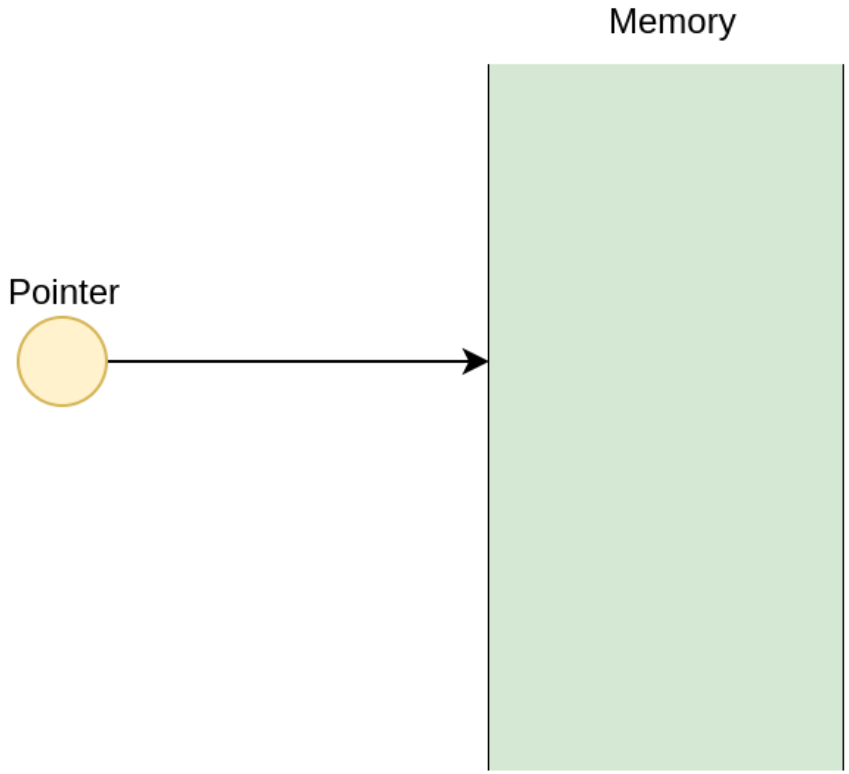
Sander Huyghebaert
Promotor: Prof. Dr. Dominique Devriese
Supervisor: Thomas Van Strydonck
Supervisor: Dr. Steven Keuchel

VUB VRIJE UNIVERSITEIT BRUSSEL

OUTLINE

1. Introduction
2. Uninitialized Capabilities
3. Secure Calling Convention
4. Evaluation
5. Future Work
6. Conclusions

POINTERS



- No Bounds
- No Permissions

CALLING CONVENTION WITH POINTERS

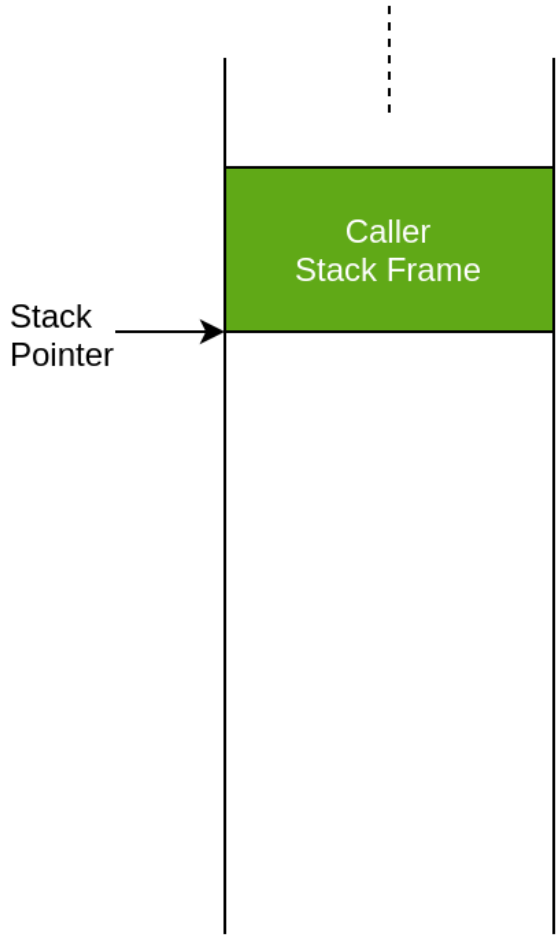


Figure: Stack at some point of a program

CALLING CONVENTION WITH POINTERS

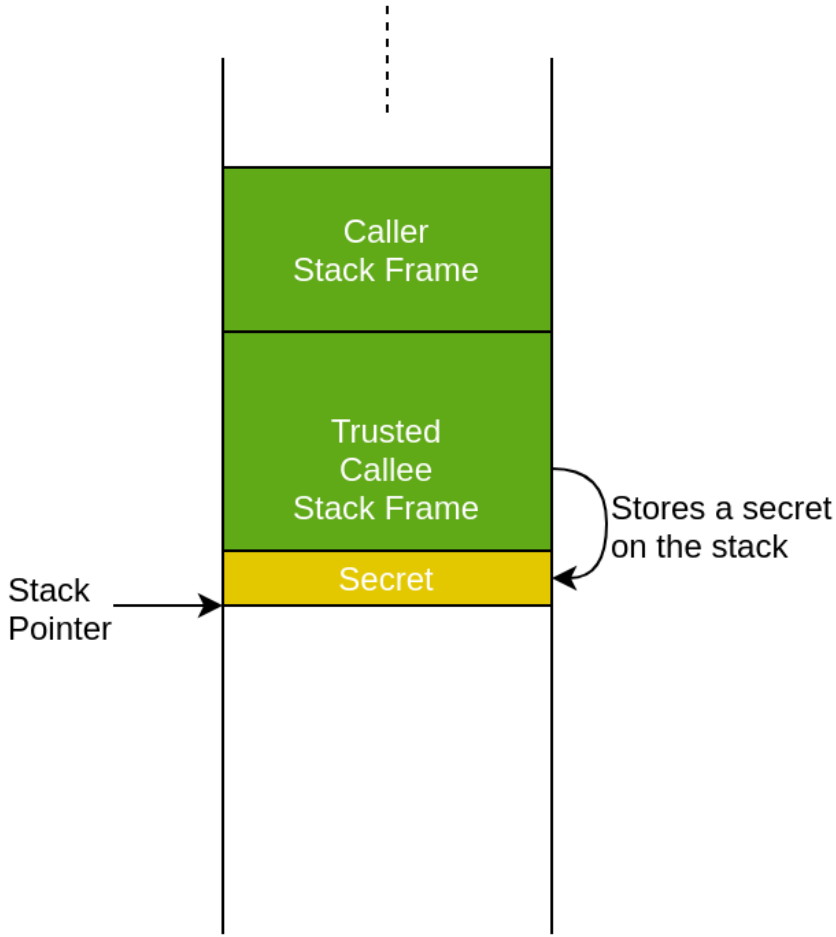


Figure: Stack after calling a function

CALLING CONVENTION WITH POINTERS

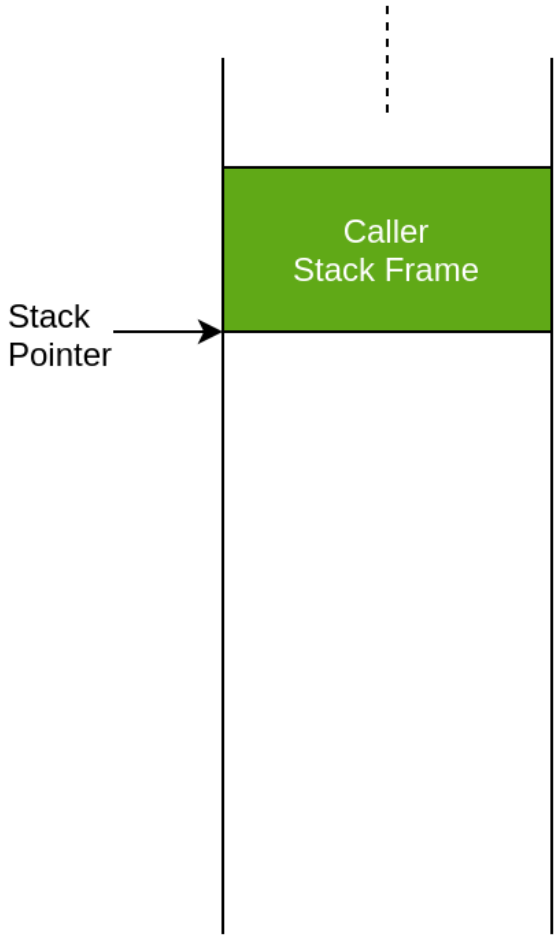


Figure: Callee returns

CALLING CONVENTION WITH POINTERS

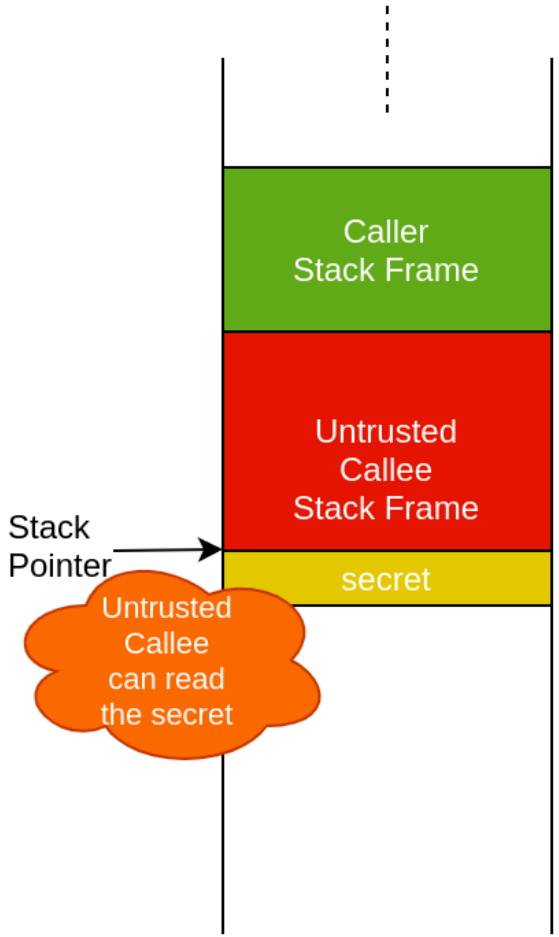


Figure: Caller invokes untrusted function

CAPABILITY MACHINES

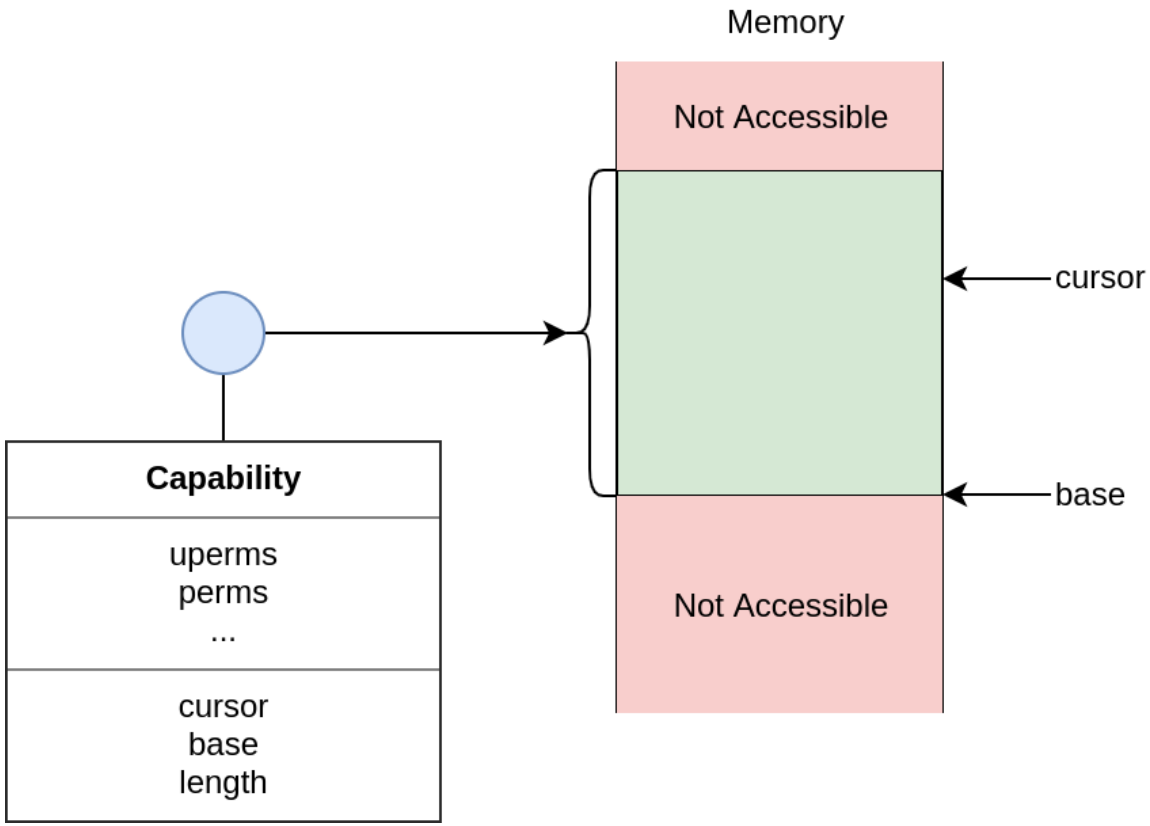


Figure: Capability for a region of memory

- Bounds
- Permissions

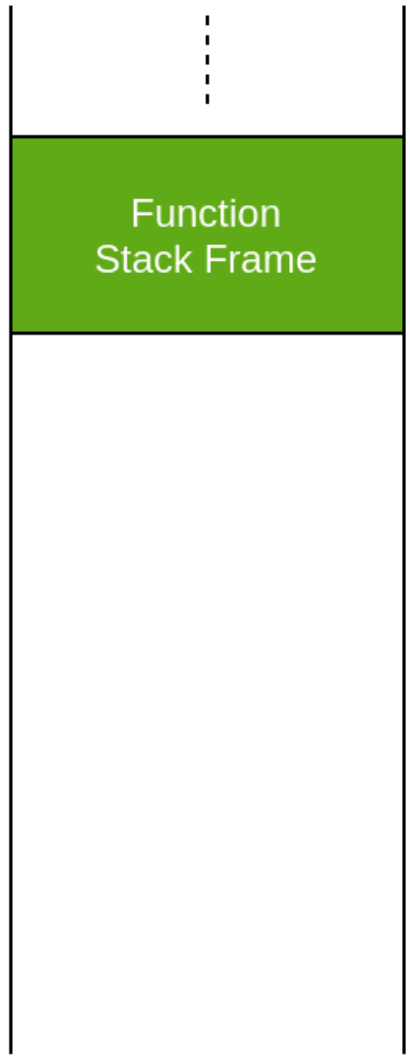
CHERI

- Capability extension to Instruction Set Architectures
- Hardware implementation
  - Capability instructions
- Backwards compatibility
- Software stack
  - CLang/LLVM, CHERIBSD, QEMU Emulator

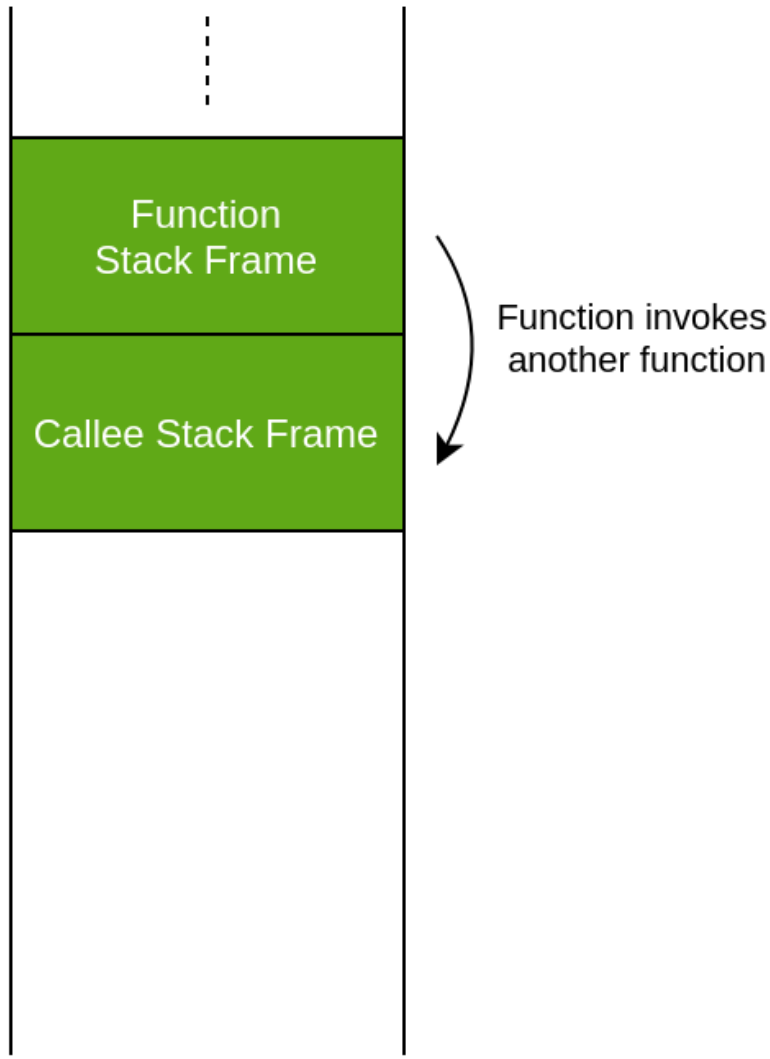
RELATED WORK

- Secure Calling Convention with Local Capabilities
  - Skorstengaard et al. (2018)
  - Enforces Well-Bracketed Control Flow (WBCF) and Local State Encapsulation (LSE)
  - Clearing Requirement
- Secure Calling Convention With Linear Capabilities
  - Skorstengaard et al. (2019)
  - Difficult to implement in hardware

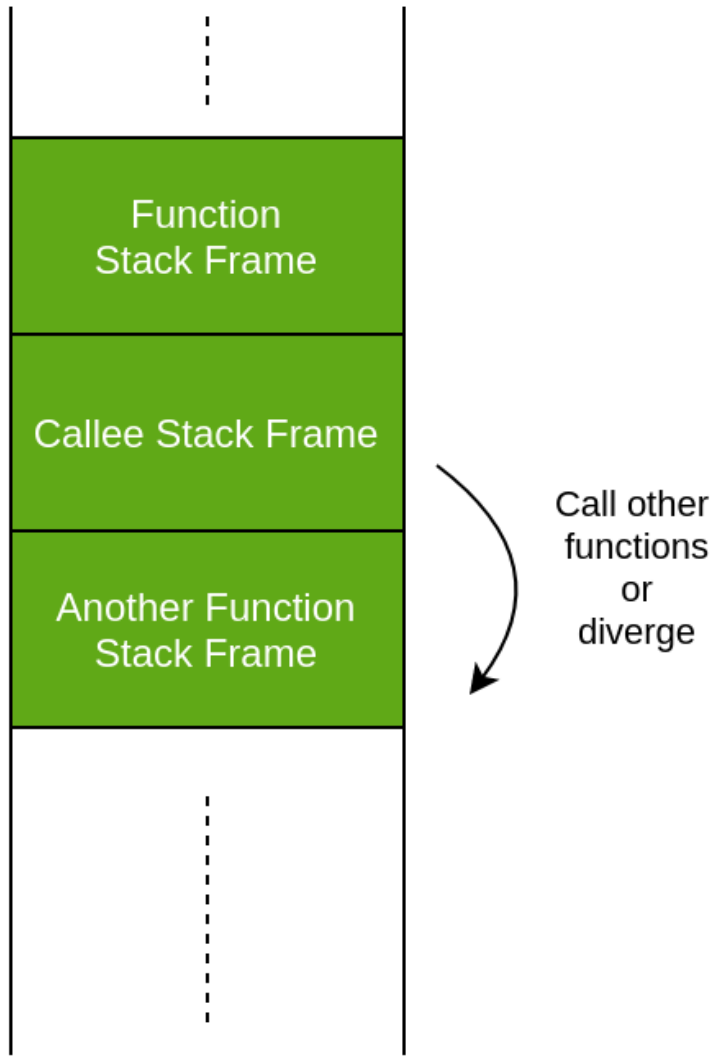
WELL-BRACKETED CONTROL FLOW



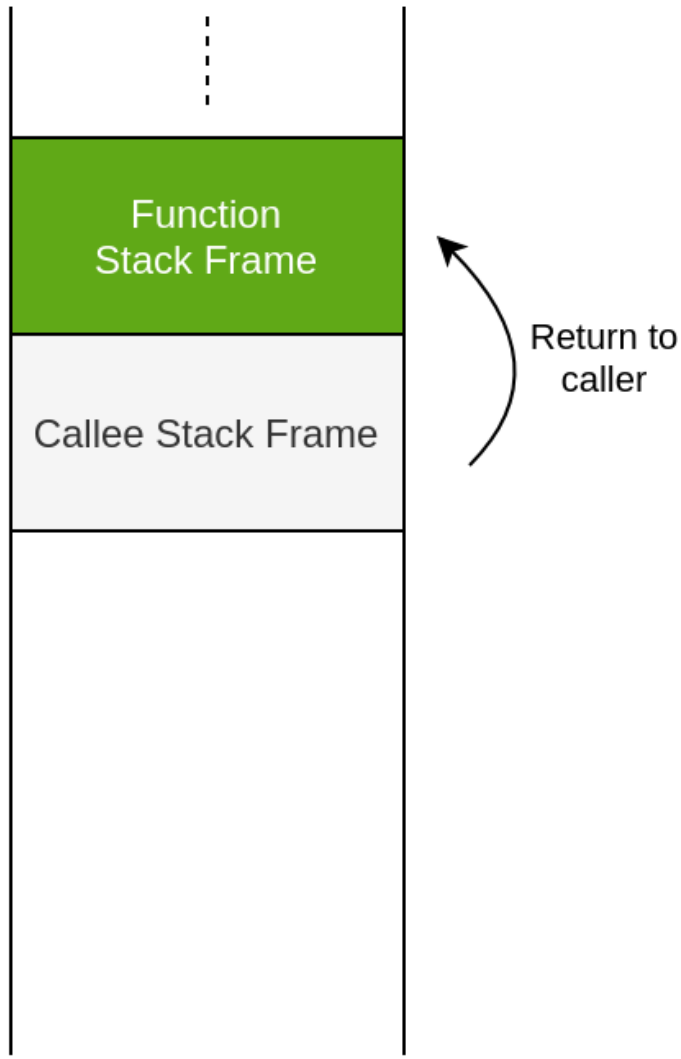
WELL-BRACKETED CONTROL FLOW



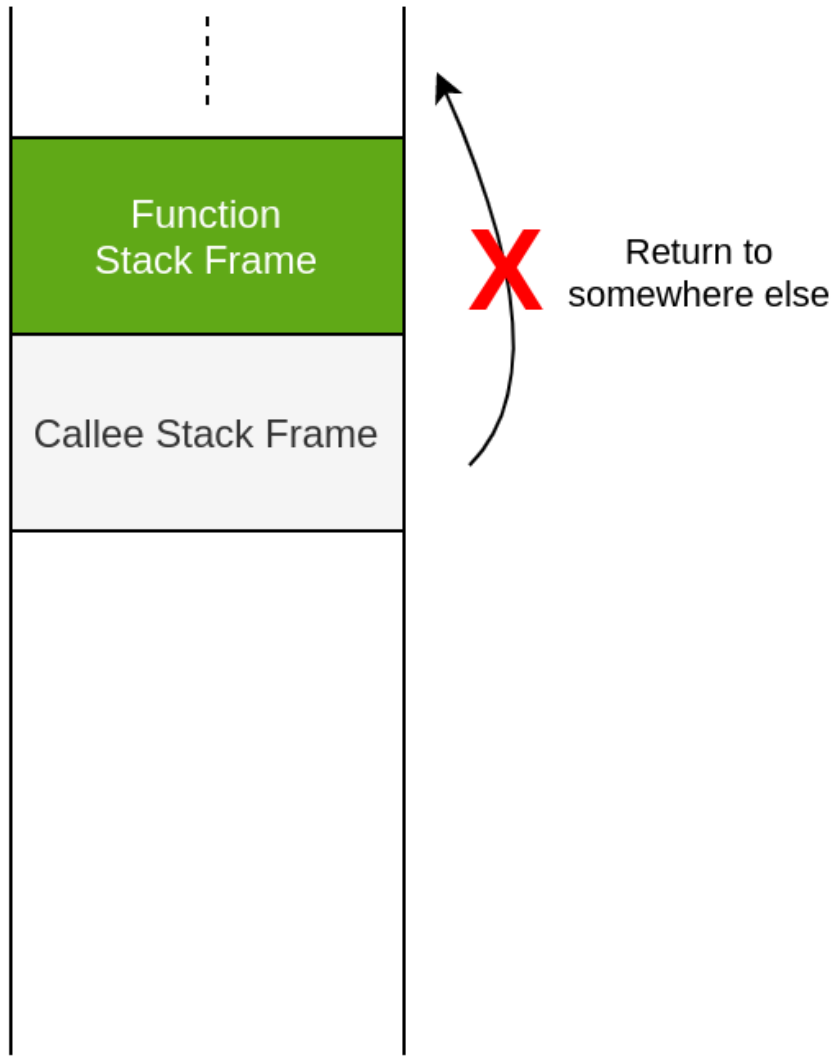
WELL-BRACKETED CONTROL FLOW



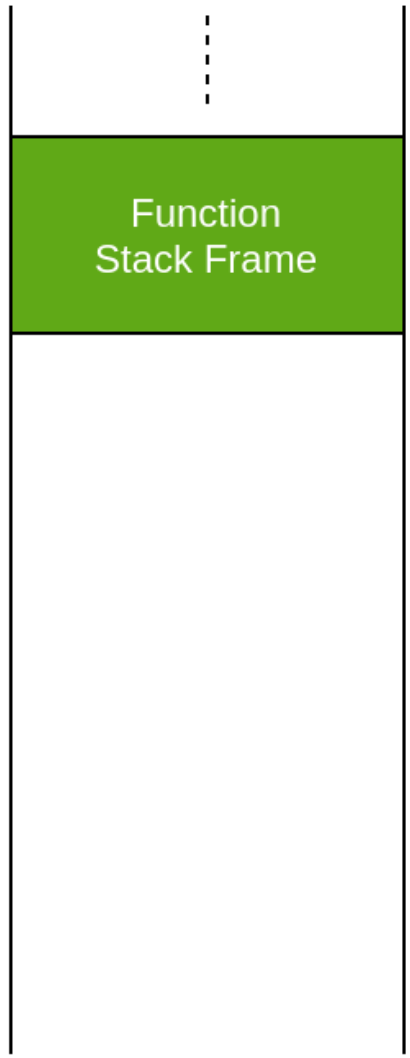
WELL-BRACKETED CONTROL FLOW



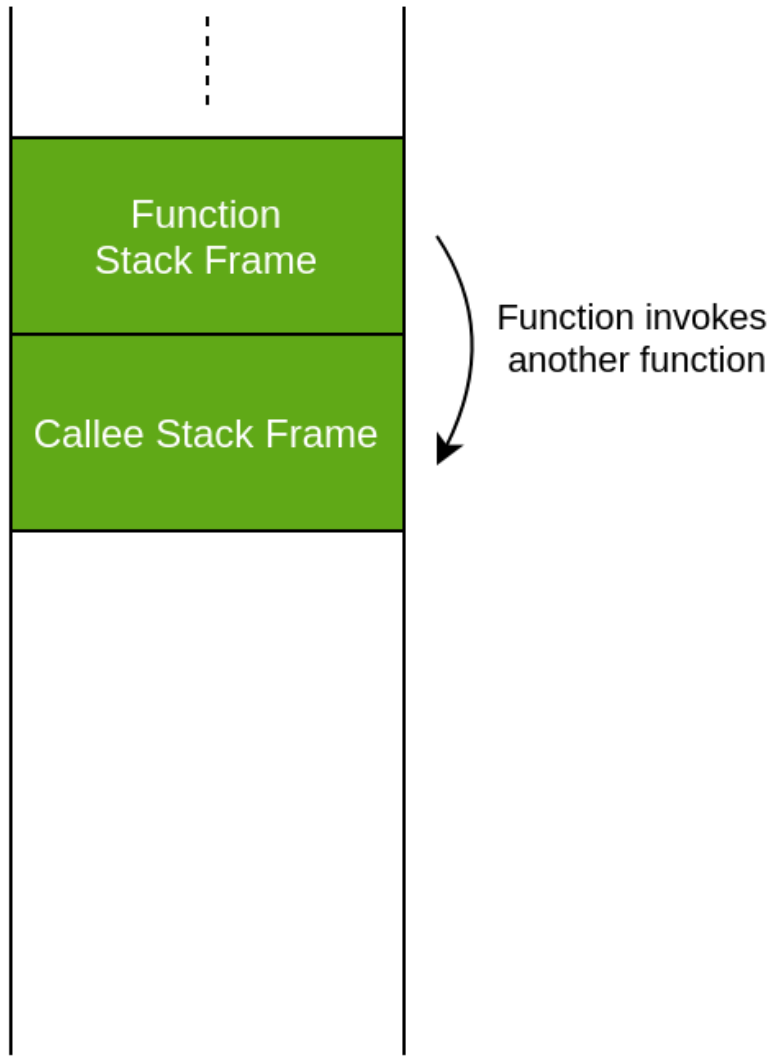
WELL-BRACKETED CONTROL FLOW



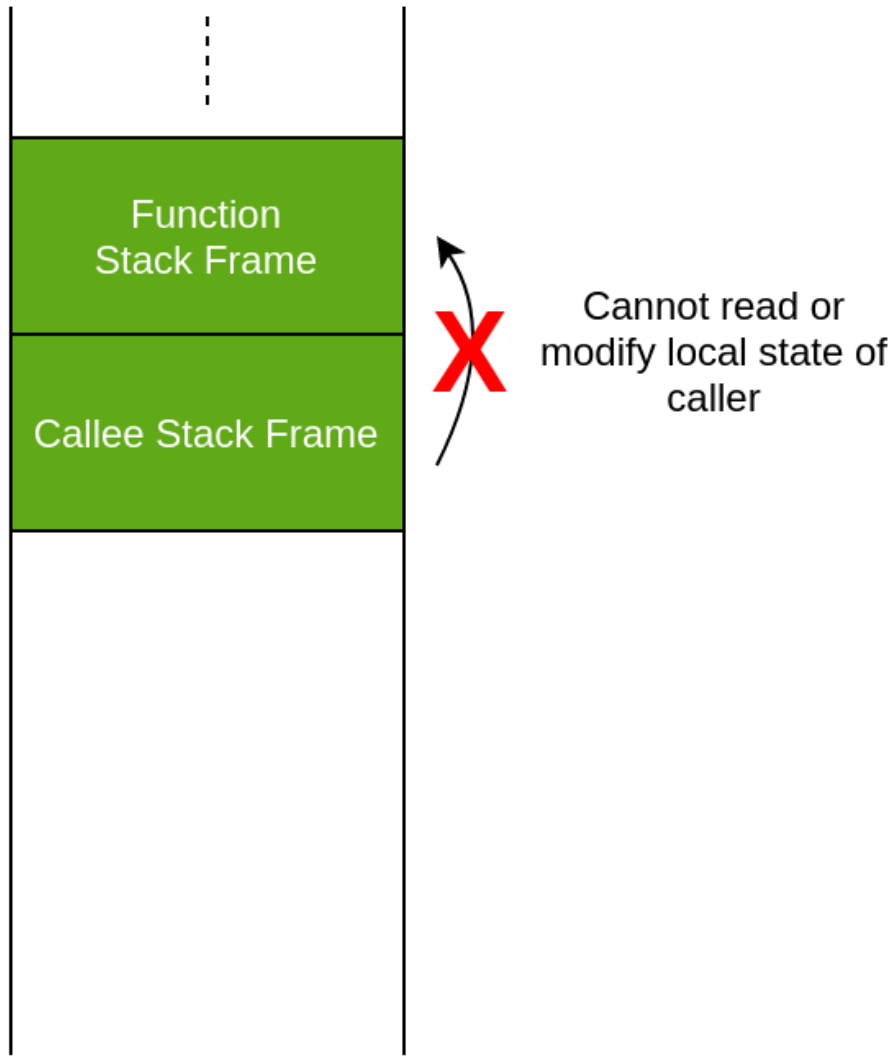
LOCAL STATE ENCAPSULATION



LOCAL STATE ENCAPSULATION



LOCAL STATE ENCAPSULATION



POPL21 PAPER

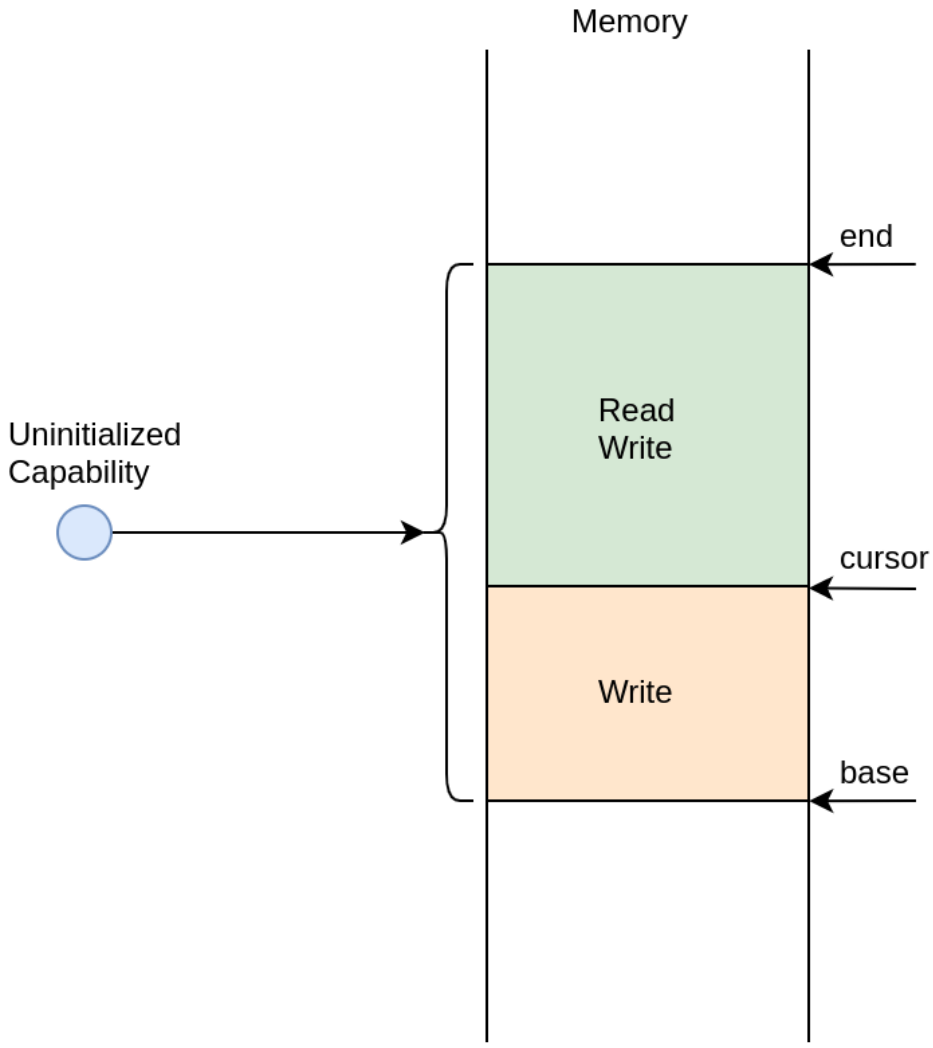
- "Efficient and Provable Local Capability Revocation using Uninitialized Capabilities"
- Collaboration between VUB and Aarhus University (Denmark)
- Thesis Results Part of Paper

OUTLINE

1. Introduction
2. Uninitialized Capabilities
3. Secure Calling Convention
4. Evaluation
5. Future Work
6. Conclusions

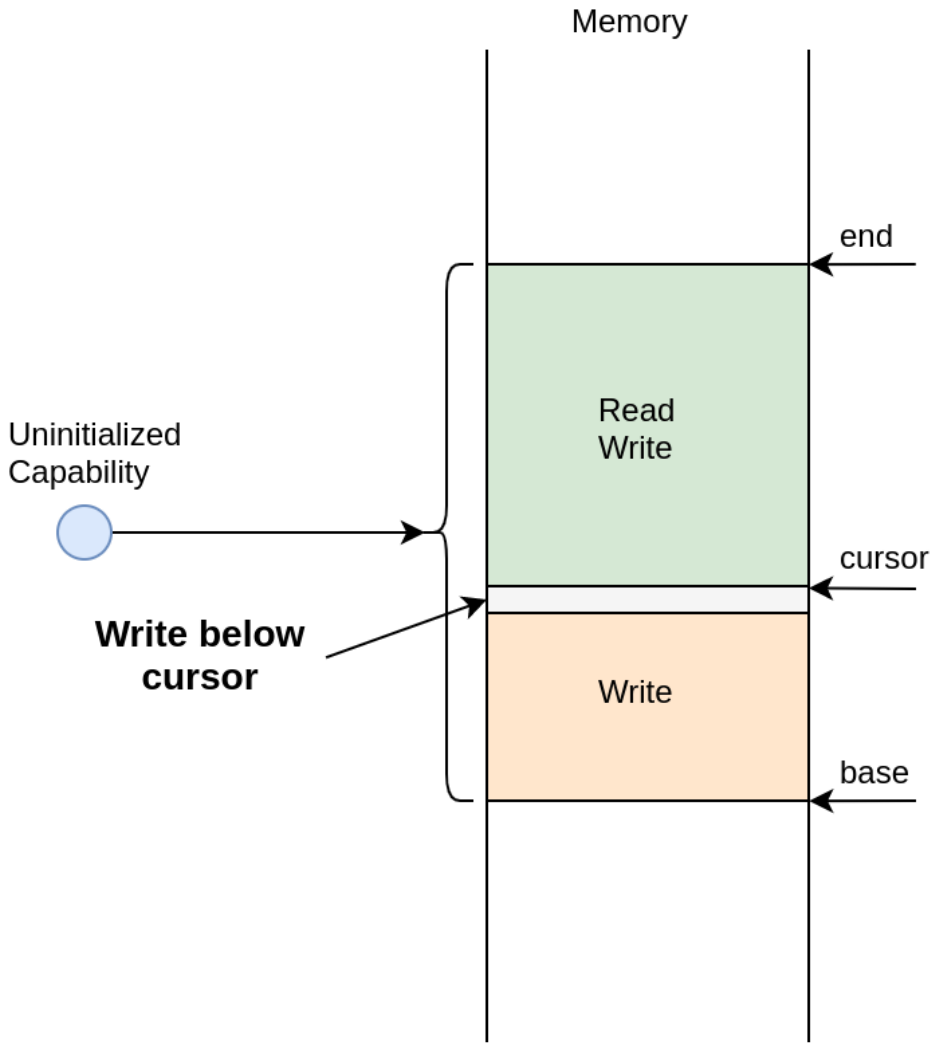
UNINITIALIZED CAPABILITIES

CONCEPT



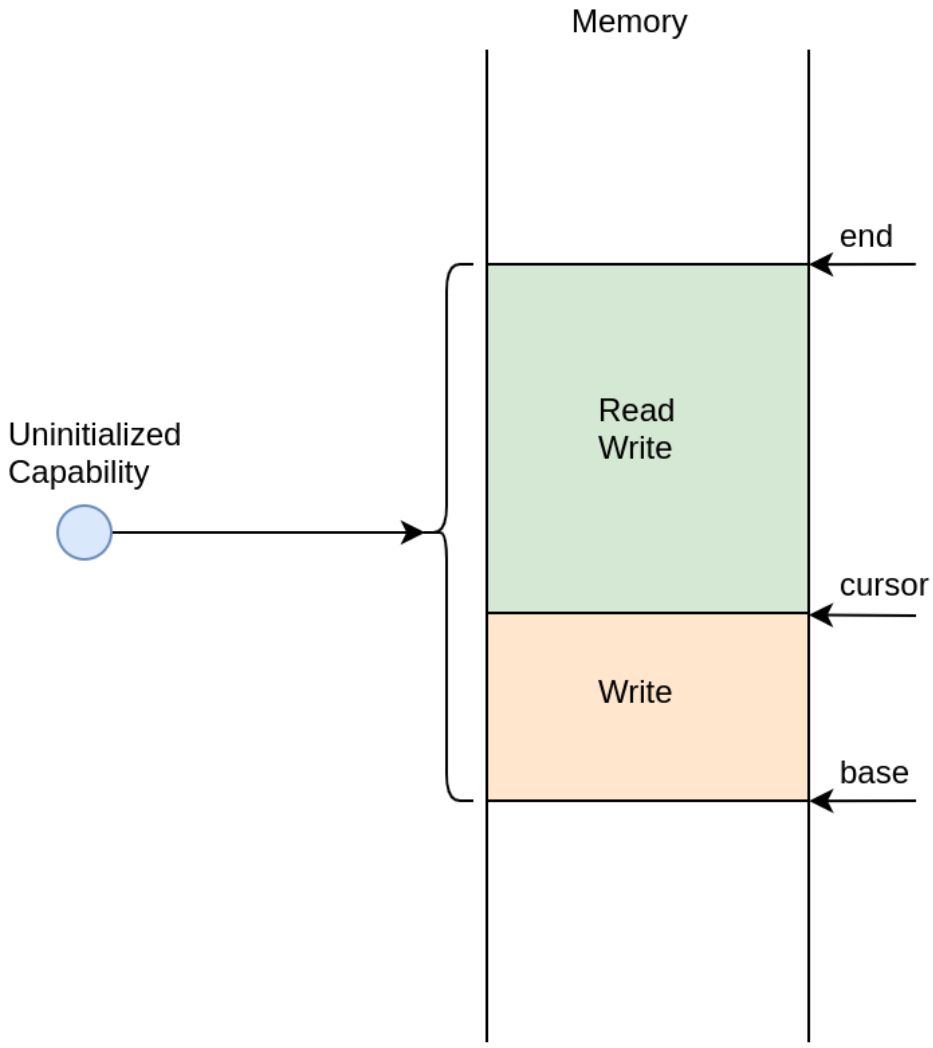
UNINITIALIZED CAPABILITIES

CONCEPT



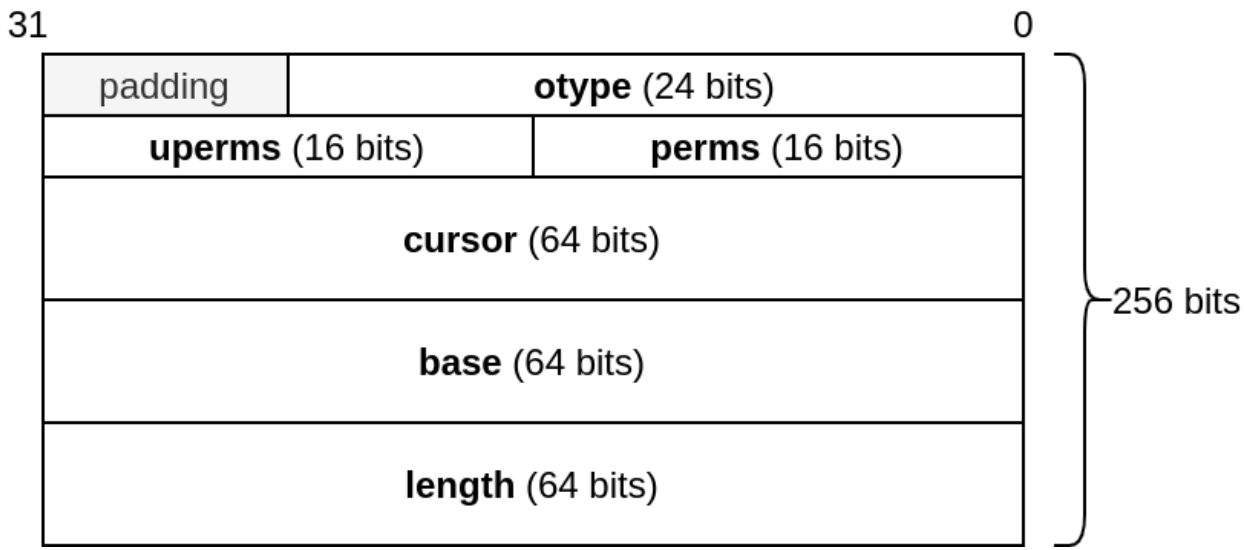
UNINITIALIZED CAPABILITIES

CONCEPT



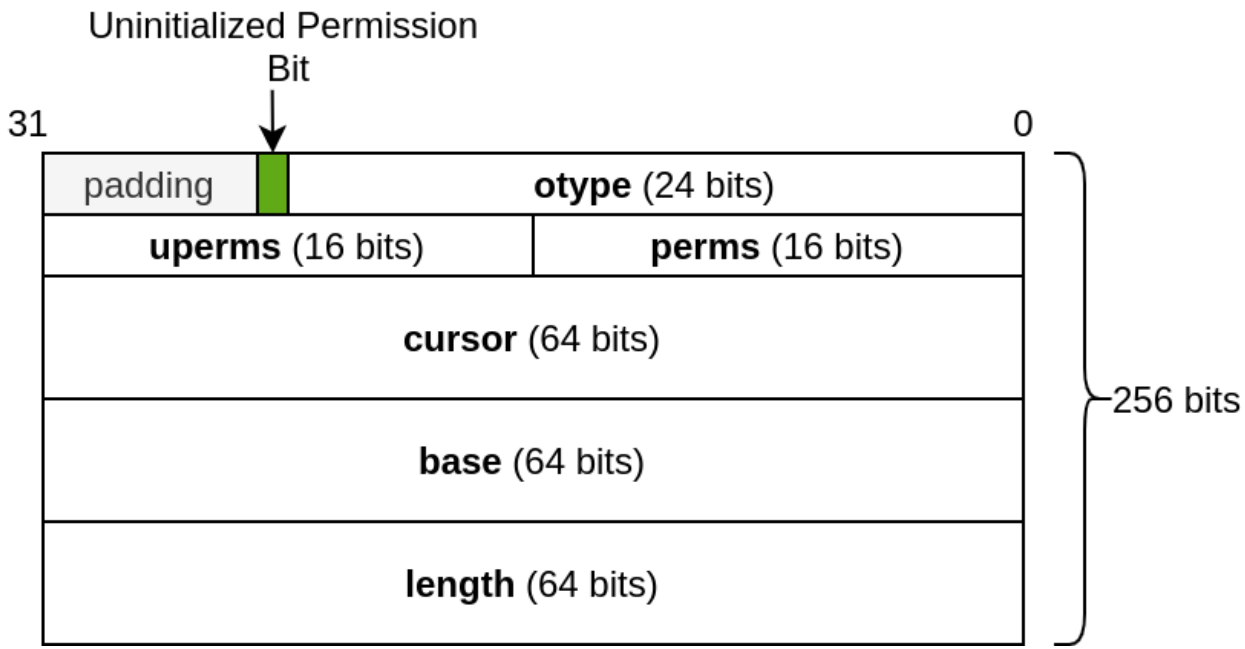
UNINITIALIZED CAPABILITIES

IMPLEMENTATION OVERVIEW: PERMISSION BIT



UNINITIALIZED CAPABILITIES

IMPLEMENTATION OVERVIEW: PERMISSION BIT



UNINITIALIZED CAPABILITIES

IMPLEMENTATION OVERVIEW: INSTRUCTION MODIFICATIONS

- Load Instructions
  - Uninitialized capabilities cannot load if ***address*** $<$ ***cursor***
  - CL[BHWD][U], CLC
- Instructions that modify the cursor
  - Only store right below cursor can modify the cursor of an uninitialized capability
  - CSetOffset, CIncOffset, CSetAddr, CAndAddr

UNINITIALIZED CAPABILITIES

IMPLEMENTATION OVERVIEW: NEW INSTRUCTIONS

- Uninitialized Permission Bit
  - Get, Set and Drop
- Uninitialized Store Instructions
- Shrink a Capability

OUTLINE

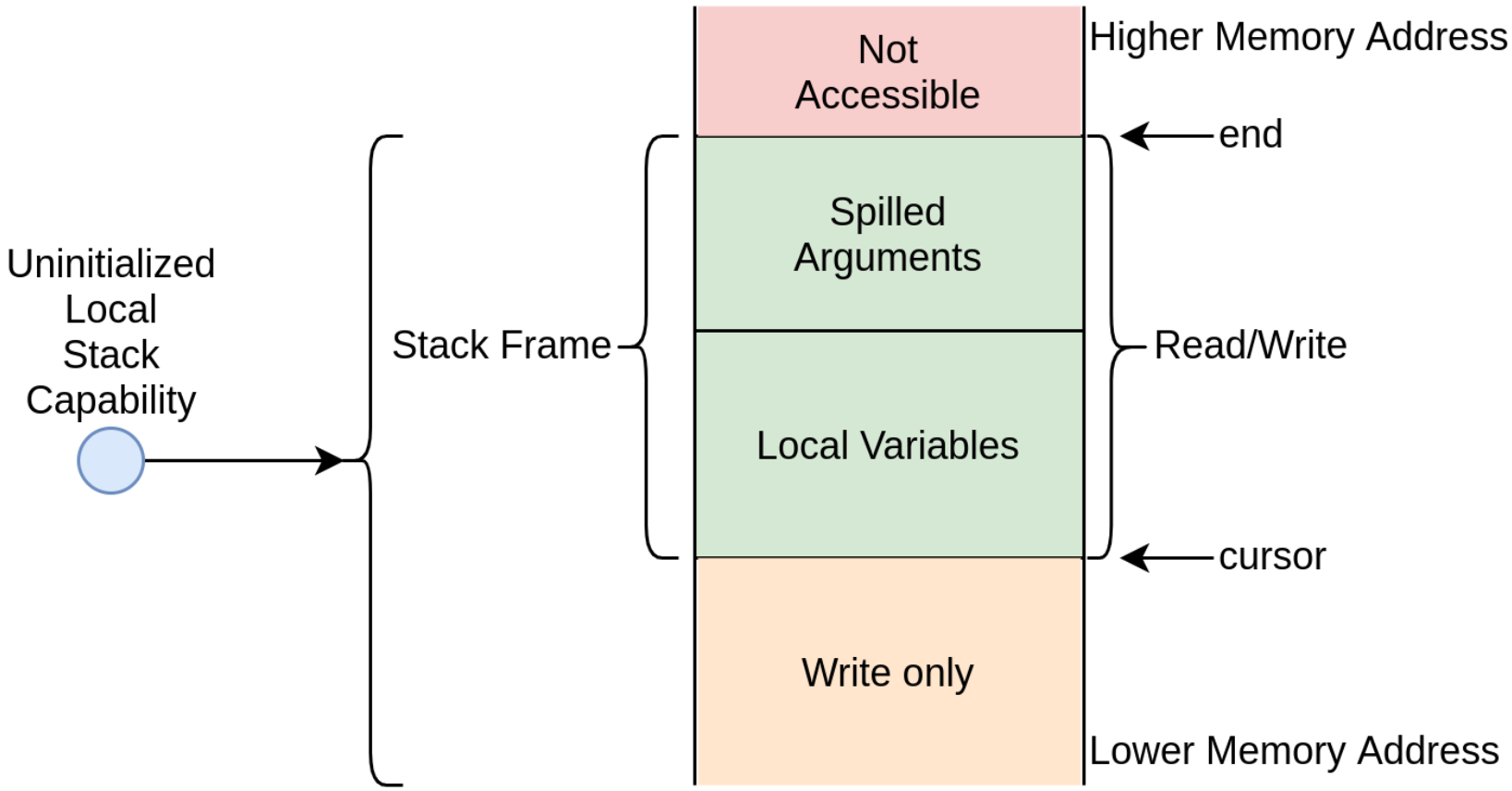
1. Introduction
2. Uninitialized Capabilities
3. Secure Calling Convention
4. Evaluation
5. Future Work
6. Conclusions

SECURE CALLING CONVENTION

- Based on Calling Convention with Local Capabilities
  - Skorstengaard et al. (2018)
- Enforces Well-Bracketed Control Flow (WBCF)
- Enforces Local State Encapsulation

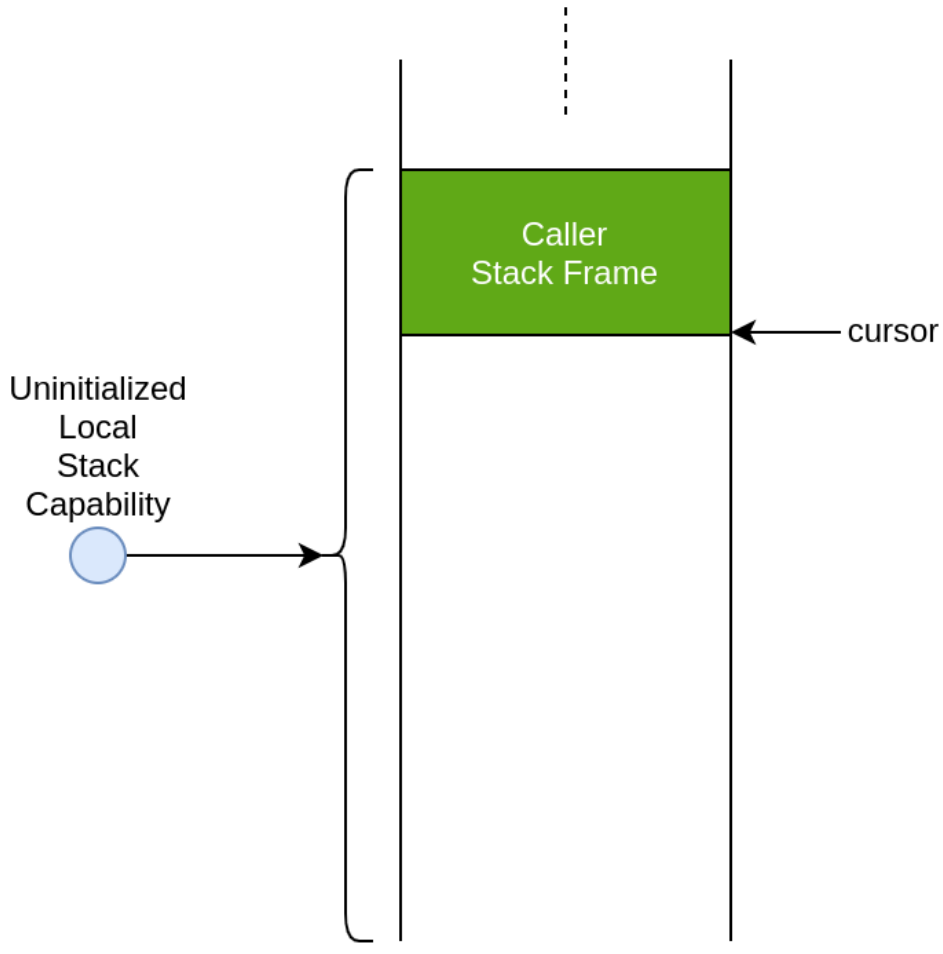
SECURE CALLING CONVENTION

STACK



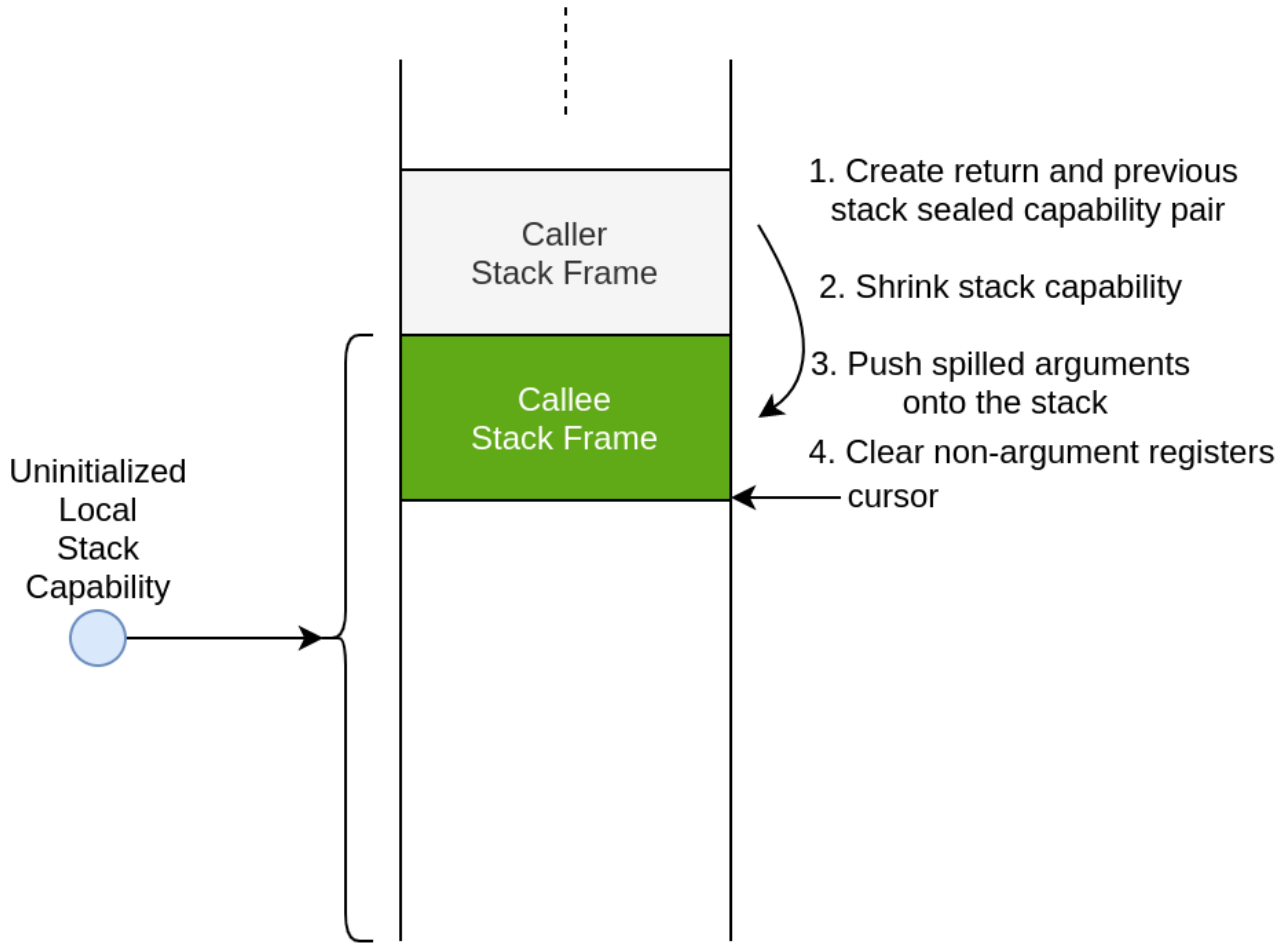
SECURE CALLING CONVENTION

INITIAL STACK



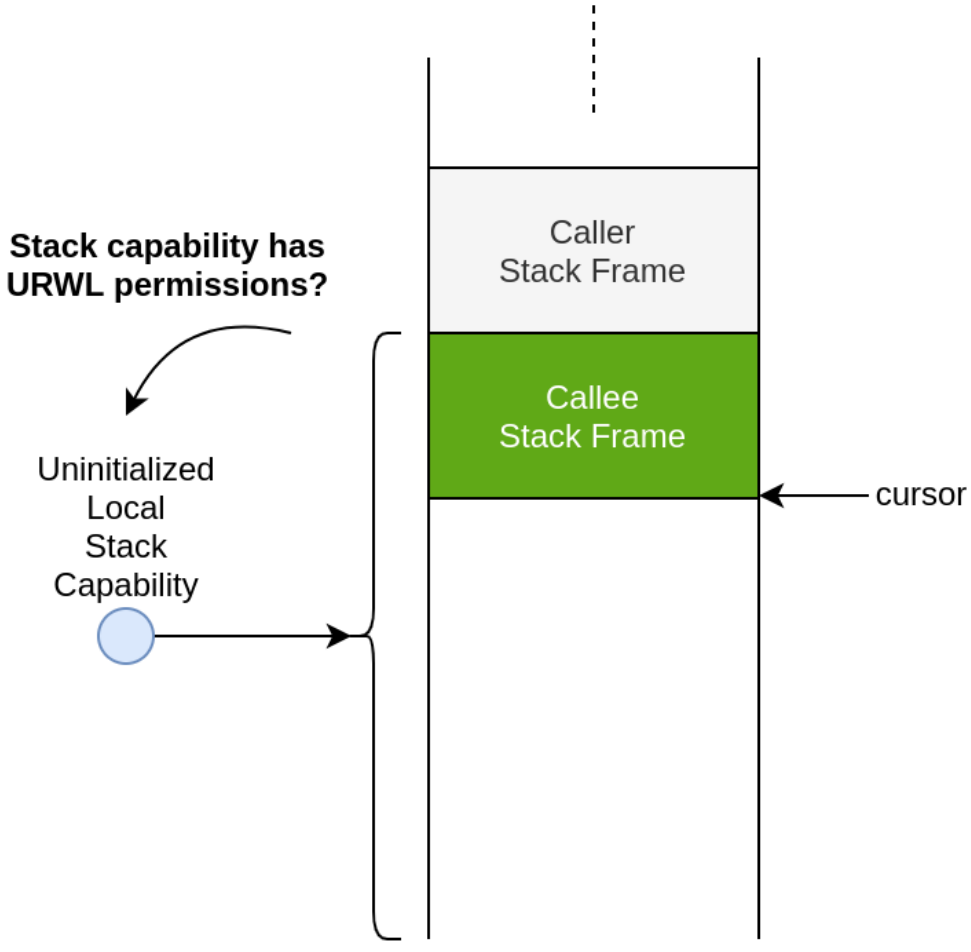
SECURE CALLING CONVENTION

FUNCTION INVOCATION



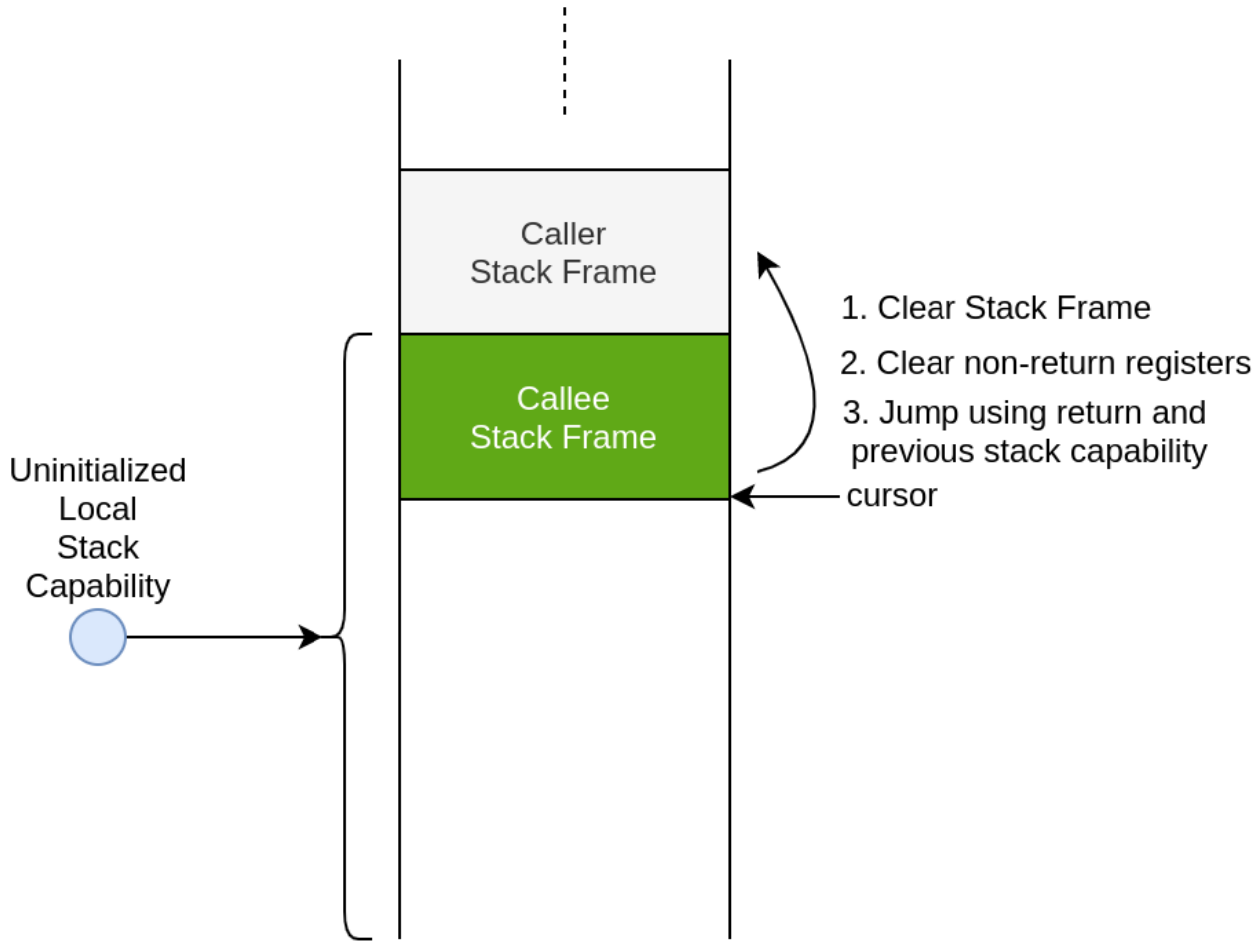
SECURE CALLING CONVENTION

FUNCTION PROLOGUE



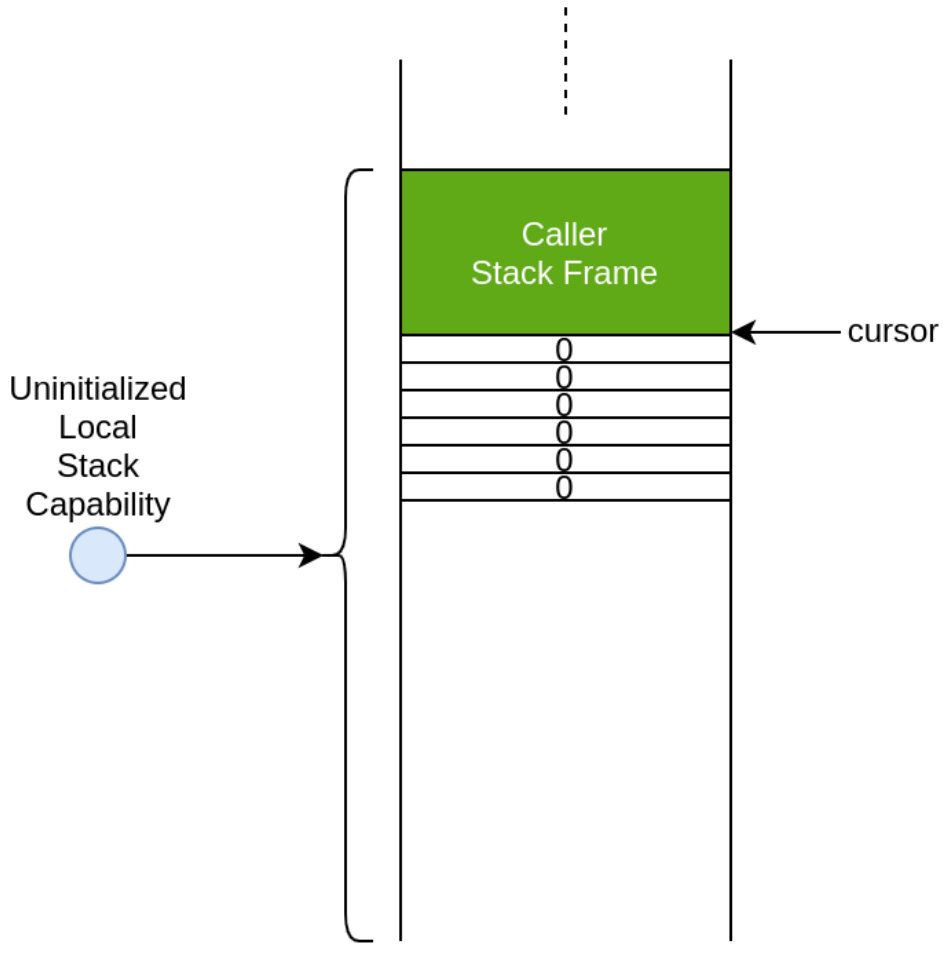
SECURE CALLING CONVENTION

FUNCTION EPILOGUE



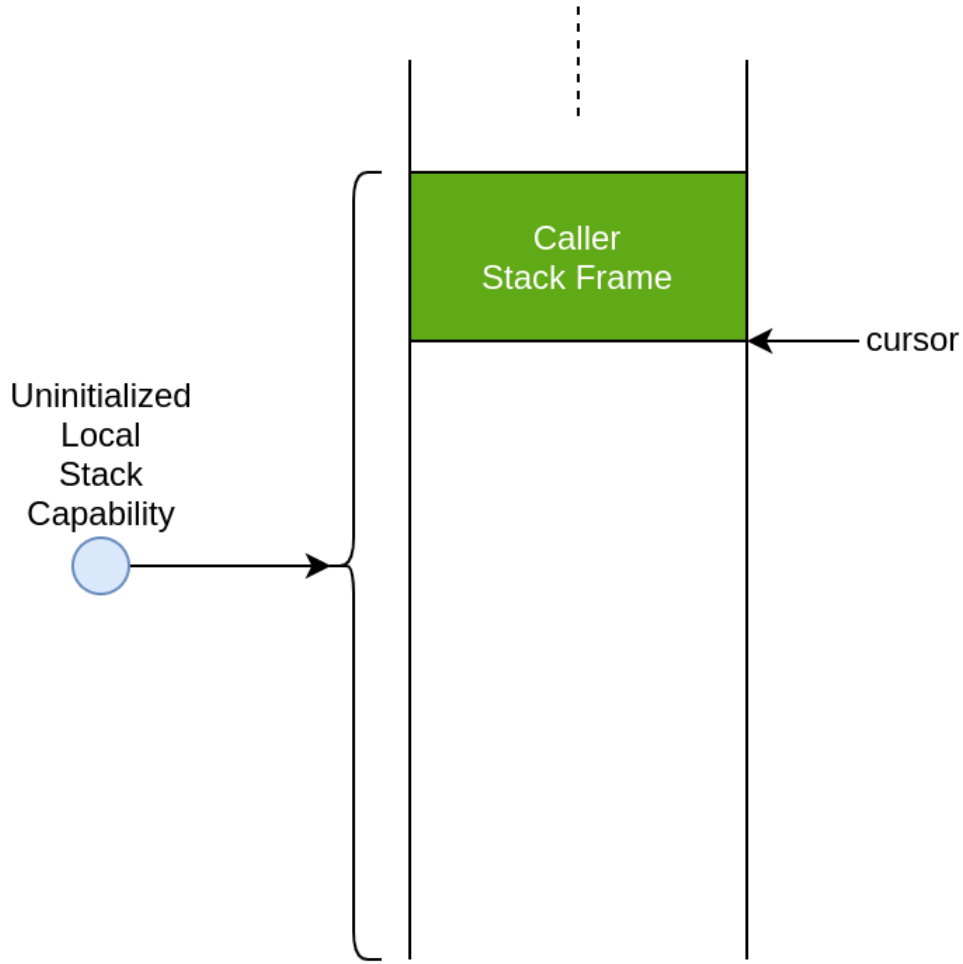
SECURE CALLING CONVENTION

FUNCTION EPILOGUE



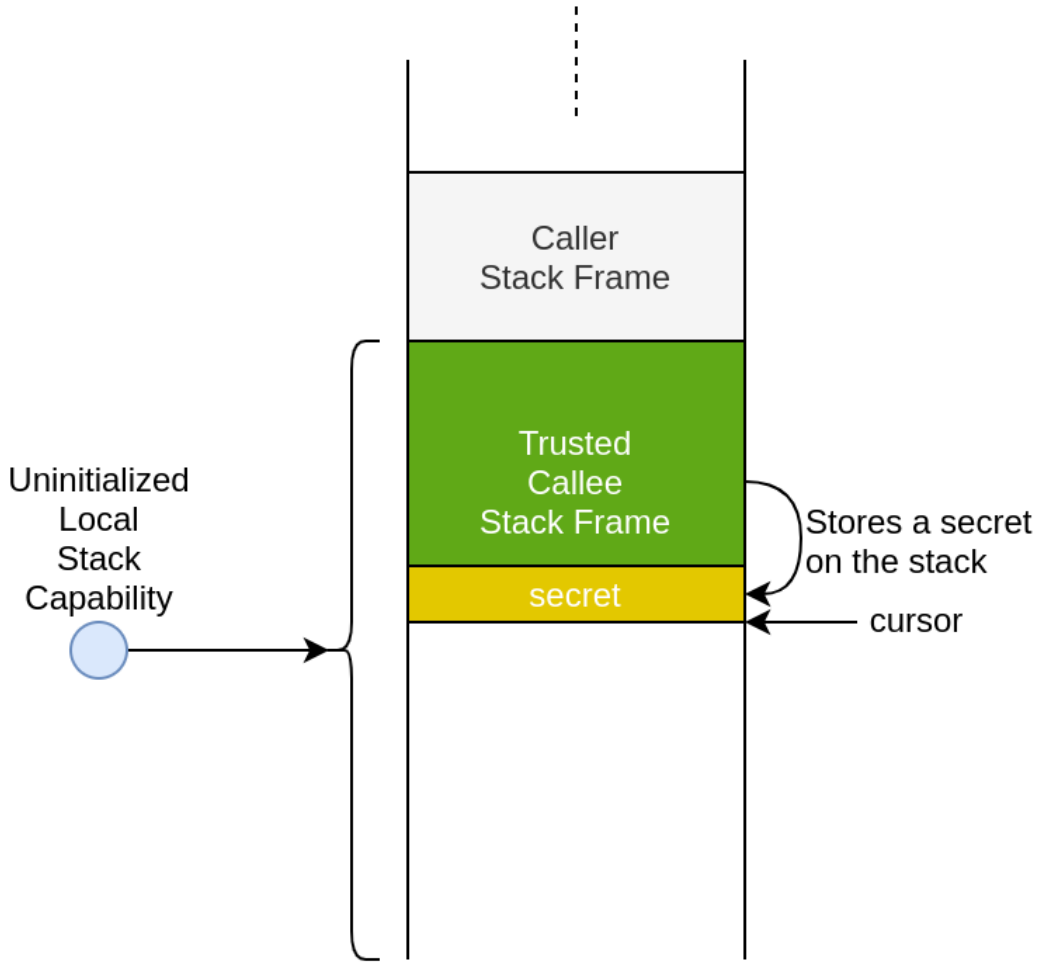
SECURE CALLING CONVENTION

EXAMPLE WITH ADVERSARY



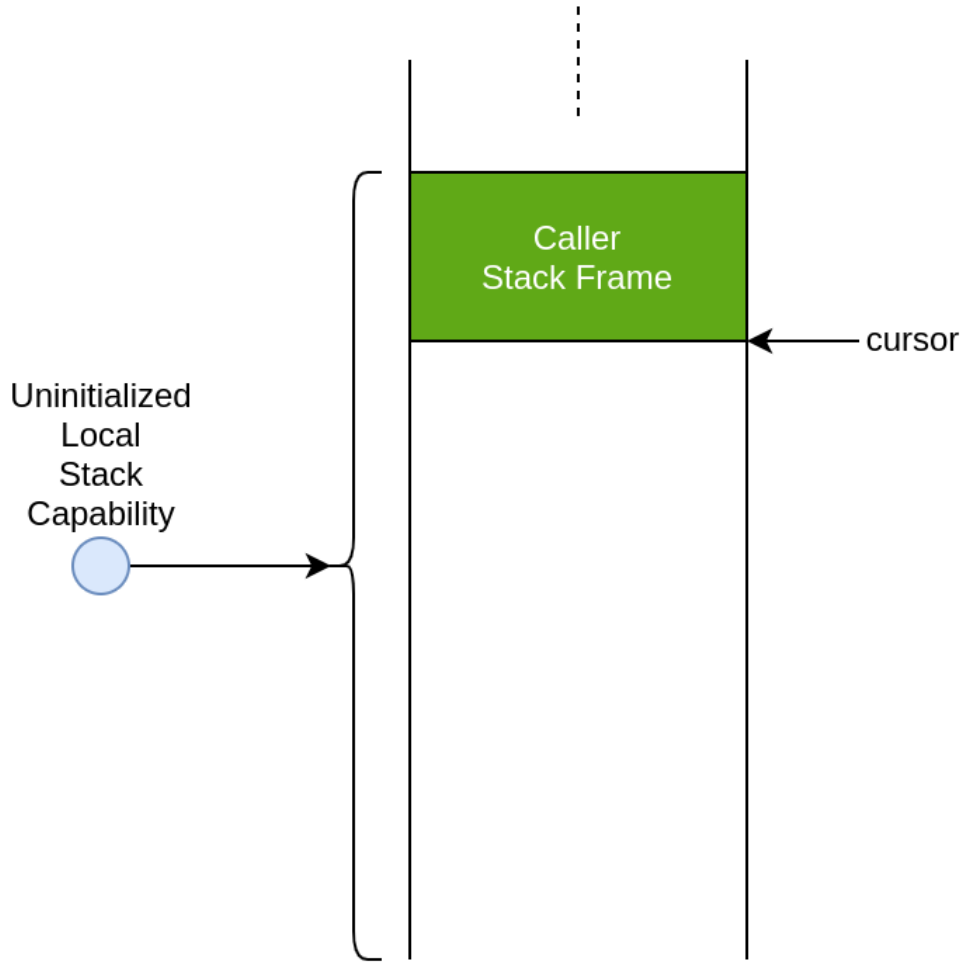
SECURE CALLING CONVENTION

EXAMPLE WITH ADVERSARY



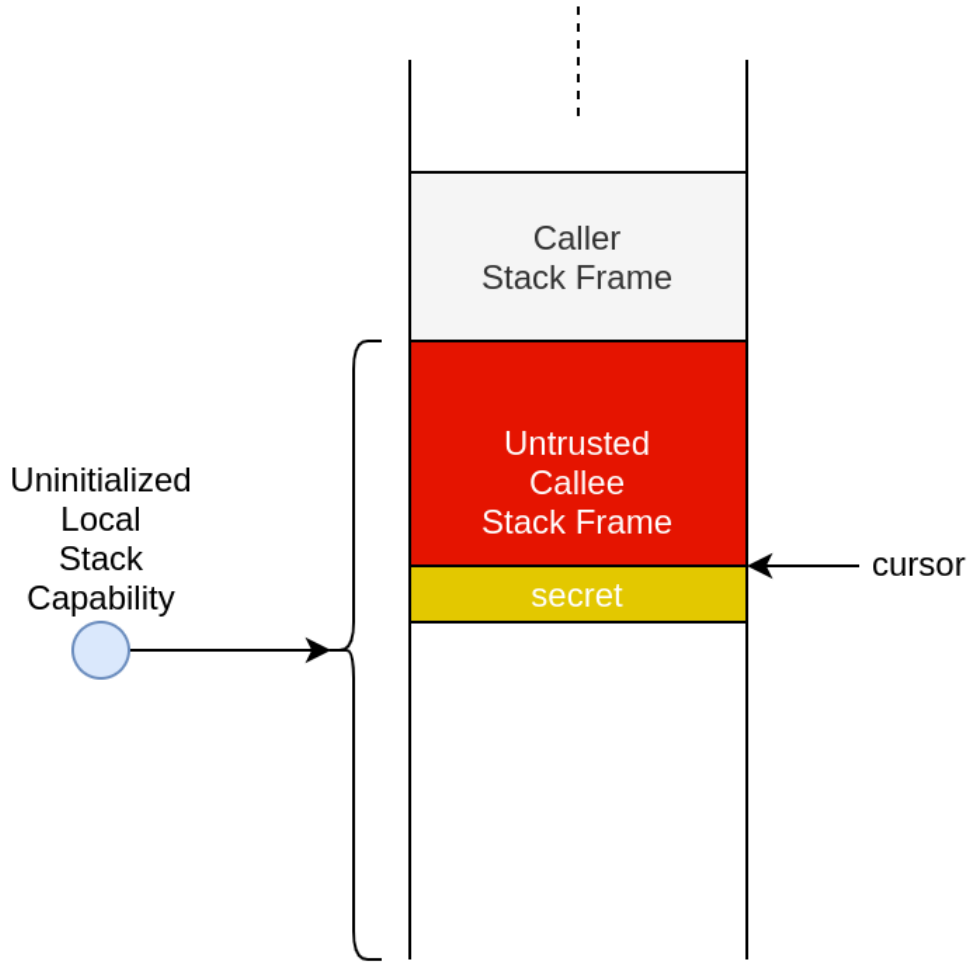
SECURE CALLING CONVENTION

EXAMPLE WITH ADVERSARY



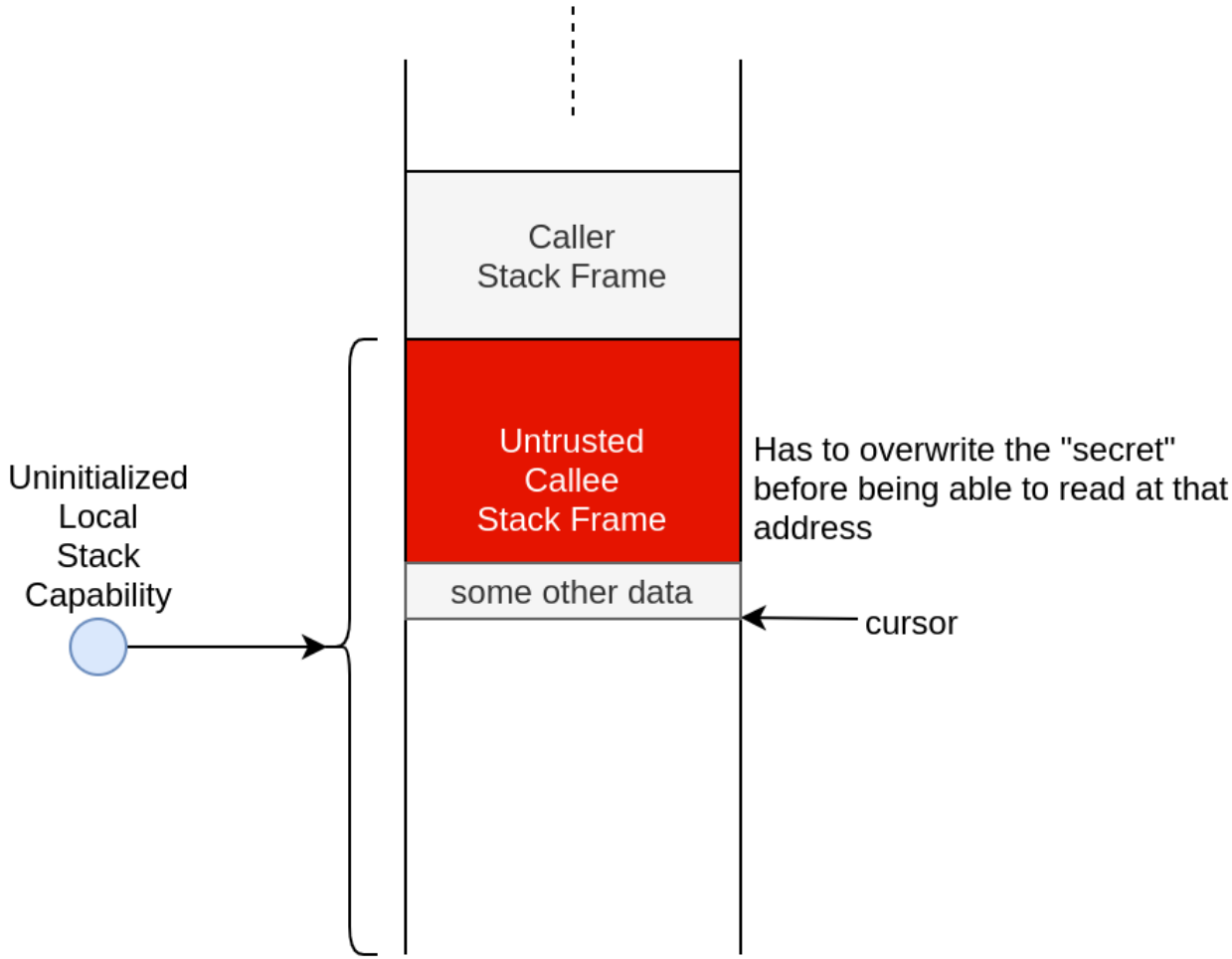
SECURE CALLING CONVENTION

EXAMPLE WITH ADVERSARY



SECURE CALLING CONVENTION

EXAMPLE WITH ADVERSARY



OUTLINE

1. Introduction
2. Uninitialized Capabilities
3. Secure Calling Convention
4. Evaluation
5. Future Work
6. Conclusions

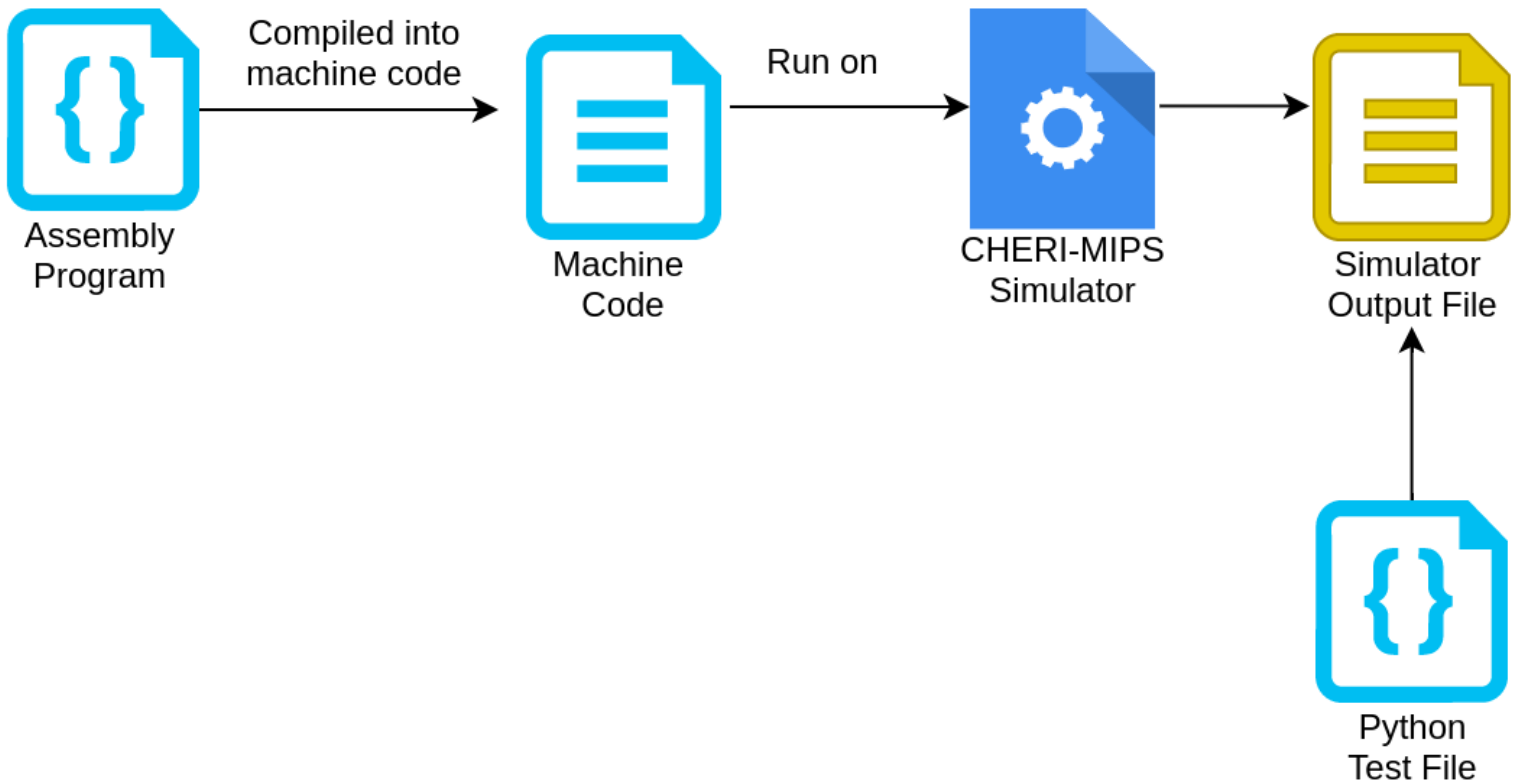
EVALUATION

ASSEMBLER

- LLVM Assembler
  - CHERI-MIPS Backend
- New instructions added

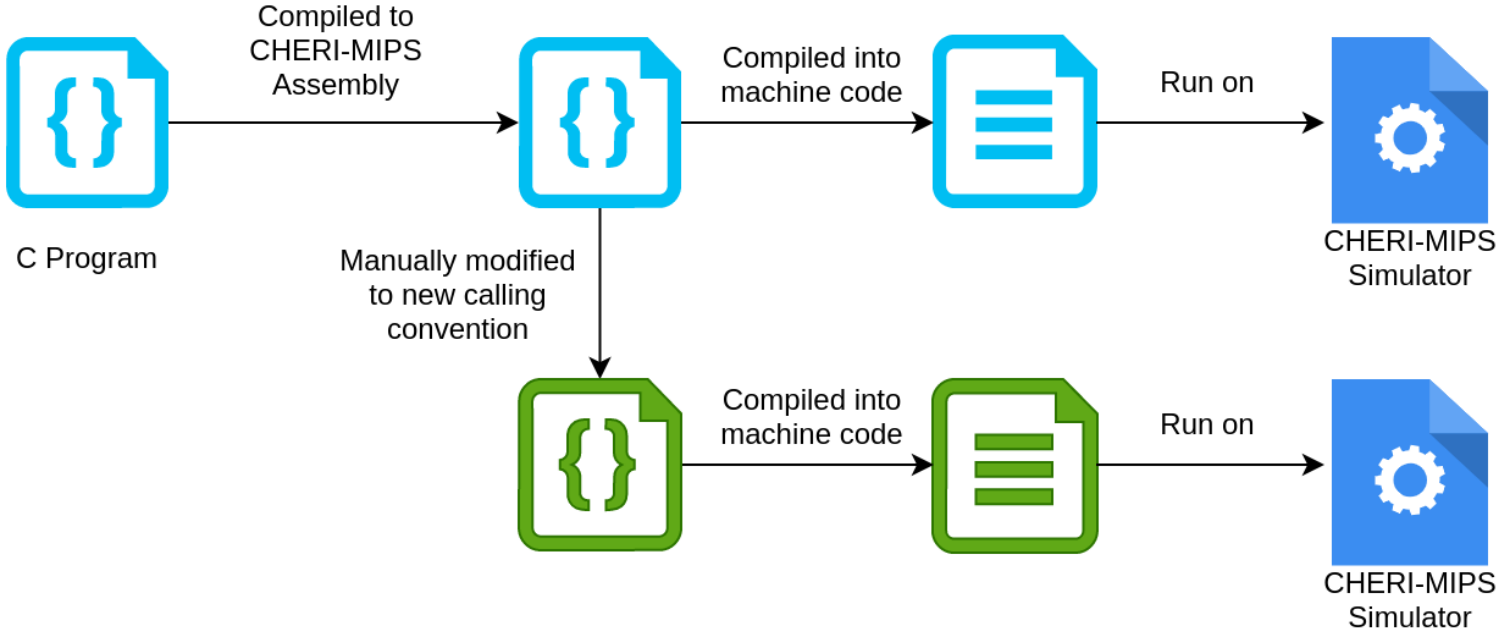
EVALUATION

UNIT TESTING INSTRUCTIONS



EVALUATION

CALLING CONVENTION



EVALUATION

EXPERIMENTS

- C Programs
  - Function calls
  - Arrays
  - Pointer arithmetic
- Semantics Preserved
- Measure execution time
- Number of instructions

EVALUATION

RESULTS

- Unit Tests Pass
- Semantics are preserved
- Overhead for secure calling convention
  - Stack frame clearing, depends on stack frame sizes
- Number of instructions doubles for secure calling convention

OUTLINE

1. Introduction
2. Uninitialized Capabilities
3. Secure Calling Convention
4. Evaluation
5. Future Work
6. Conclusions

FUTURE WORK

HARDWARE IMPLEMENTATION

- Should be possible
  - Uninitialized Capabilities only require one extra bit
  - New instructions similar to existing instructions
- Out of scope of thesis

FUTURE WORK

CLANG/LLVM

- Calling Convention currently not implemented in Clang/LLVM...
- ... but exploration of Clang/LLVM compiler for calling convention provided in thesis

OUTLINE

1. Introduction
2. Uninitialized Capabilities
3. Secure Calling Convention
4. Evaluation
5. Future Work
6. Conclusions

CONCLUSIONS

- Uninitialized Capabilities
  - Semantics
  - ISA Extension
    - Instantiated for CHERI-MIPS
- Calling Convention
  - Enforces WBCF and LSE
  - Security comes at a cost (overhead)
- Exploration of Clang/LLVM compiler
  - To implement new calling convention