MANCHESTER 1824
The University of Manchester

ESBMC-CHERI: Towards Verification of C Programs for CHERI Platforms with ESBMC

Franz Brauße

CHERI and CHERI-C

Motivation

Memory errors in software are one of the main problems in computer security.

1	CWE-787	Out-of-bounds Write
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
4	CWE-20	Improper Input Validation
5	CWE-125	Out-of-bounds Read
6	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
7	CWE-416	Use After Free
8	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
9	CWE-352	Cross-Site Request Forgery (CSRF)
10	CWE-434	Unrestricted Upload of File with Dangerous Type

CHERI[1]

- combined hardware architecture/software design to avoid exploitation of memory errors
- mainly intended for low-level systems software written in C/C++
- software: adds features (and a few restrictions) to these programming languages
- hardware: ensures that memory errors are caught by the CPU

[1] Capability Hardware Enhanced RISC Instructions

ESBMC

Bounded model checker for C programs:

- static program verification, ahead of compilation
- supports single- and multi-threaded code
- validity, safety properties, arbitrary C expressions as assumptions/assertions
- proofs by symbolic execution, k-induction, loop-unrolling, interval analysis, SMT solvers
- provides counter-examples: program traces of assignments

Also targets memory safety bugs, e.g.:

- spatial: out-of-bounds read/write

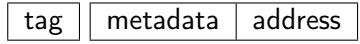
```
int a[5];
[...]
int x = a[17]; // flagged by ESBMC
```

- temporal: dynamic memory

```
int *p = malloc(n);
[...]
free(p);
[...]
*p = x; // flagged by ESBMC
```

Objective

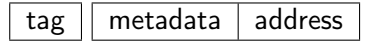
Static verification of CHERI-C programs in ESBMC

$1 + 2 \cdot 64$ bits: 

Objective

Static verification of CHERI-C programs in ESBMC

$1 + 2 \cdot 64$ bits:



CHERI-enabled ESBMC:

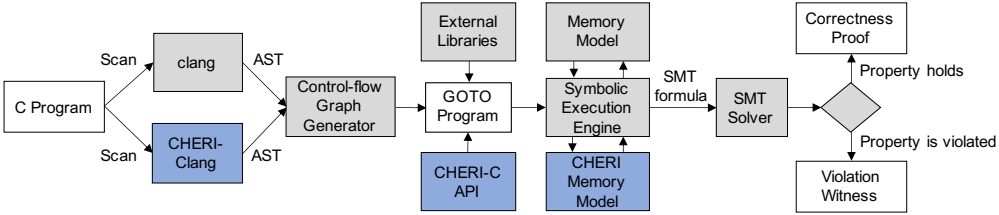
- both uncompressed and concentrate capabilities modelled bit-precise:
  - allows for constraints on in-memory bit pattern of CHERI metadata
- reasoning in CHERI-BSD execution environments of capability hardware platforms RISC-V, ARM Morello, MIPS
- both, hybrid and purecap mode
- optionally detect conditions for CHERI hardware exceptions
- model tagged memory

Effects on proving memory safety properties:

- spatial reasoning is augmented by CHERI-C semantics
- temporal reasoning ability of ESBMC is inherited

High-level Goals for CHERI-enabled ESBMC

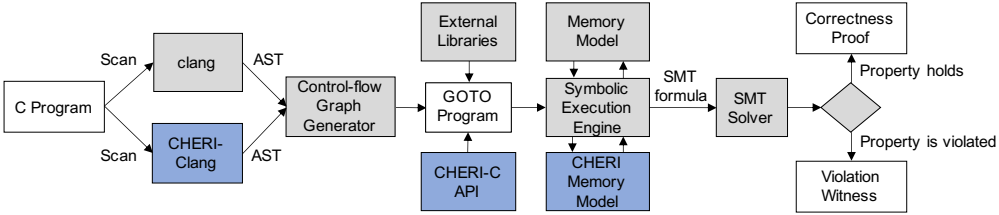
Challenges and benefits



Enable two directions for verification of CHERI-C programs

High-level Goals for CHERI-enabled ESBMC

Challenges and benefits



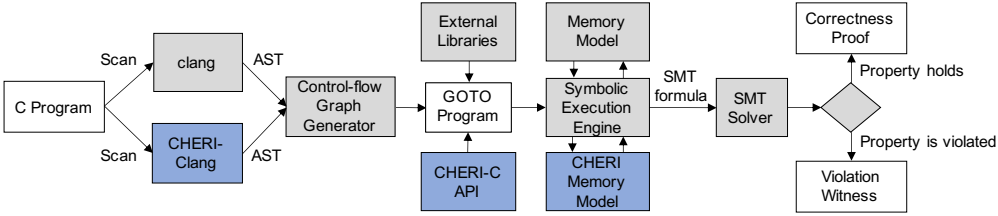
Enable two directions for verification of CHERI-C programs:

1. Proving absence of CHERI-exceptions (cf. other architecture-specific semantics)

2. Assuming absence of CHERI-exceptions

High-level Goals for CHERI-enabled ESBMC

Challenges and benefits



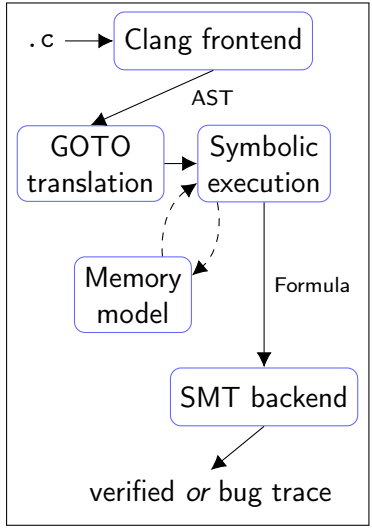
Enable two directions for verification of CHERI-C programs:

1. Proving absence of CHERI-exceptions (cf. other architecture-specific semantics):
   - Bit-precise model of in-memory representation of capabilities
   - Base/bounds might differ from those ESBMC already maintains
   - Additional checks for permissions
2. Assuming absence of CHERI-exceptions:
   - Reasoning about unmodelled external C functions (libraries) via pointer provenance
   - Simplification of ESBMC's spatial memory safety checks
   - Optimisation of operational models of `libc` functions

Approach

Static verification of CHERI-C programs in ESBMC

- replace Clang in frontend by CHERI-Clang
- support new types:
  - `[u]intcap_t`
  - `*__capability`
- support pointer/integer casts in memory model
- model tagged memory for valid pointer provenance and unforgeability $\rightsquigarrow$ track capability-sized r/w
- support CHERI-C API `cheri_*()`
  - utilise `cheri-compressed-cap` library
- add cross-platform verification support
- rework internal representation of `union`



Status

Static verification of CHERI-C programs in ESBMC

CHERI-enabled pre-release `v6.9-cheri` on Github. Details:

Rafael Menezes, et al. Towards Verification of C Programs for CHERI Platforms. ISSTA'22.

Status

Static verification of CHERI-C programs in ESBMC

CHERI-enabled pre-release `v6.9-cheri` on Github. Details:

Rafael Menezes, et al. Towards Verification of C Programs for CHERI Platforms. ISSTA'22.

Finds the OOB accesses on lines 9–11 in this CHERI-C program:

```
1  #include <cheri/cheric.h>
2
3  void main(void)
4  {
5      int n = nondet_uint() % 1024; // models user input
6      char a[n+1], *__capability b = cheri_ptr(a, n+1);
7      b[n] = 17;                    // succeeds
8
9      char *__capability c = cheri_setbounds(b-1, n);
10     /* ... */
11     memset_c(c, 42, n);
12 }
```

Status

Static verification of CHERI-C programs in ESBMC

- Integration of CHERI-Clang front-end
- Support for cross-platform verification
- Internal representation of `union`
- *deprecated:* in-mem representation of all capabilities via `cheri-compressed-cap` library

(De-)compression has significant effect on verification performance.

Status

Static verification of CHERI-C programs in ESBMC

- Integration of CHERI-Clang front-end
- Support for cross-platform verification
- Internal representation of `union`
- *deprecated:* in-mem representation of all capabilities via `cheri-compressed-cap` library

(De-)compression has significant effect on verification performance.

Focus of currently ongoing work:

- Tagged memory
  - Semantics of new types (`[u]intcap_t` and `*__capability`)
- Metadata: Bit-precise reasoning (where necessary), symbolic (if possible)
- Operational models of CHERI-C API mostly via intrinsics
- Pointer/integer casts in memory model

Conclusion & Future work

- Formalise semantics of CHERI-C constructs in ESBMC.
- Fully use the benefits CHERI-C has for automated program verification.
- Guarantee spatial and temporal memory safety ahead of runtime.
- Automated manipulation/removal of capabilities and runtime bounds checks. .75
  - Improve performance of verified user programs.

CHERI-enabled ESBMC https://github.com/fbrausse/esbmc/tree/fb/cheri

SCorCH https://scorch-project.github.io

ESBMC http://esbmc.org