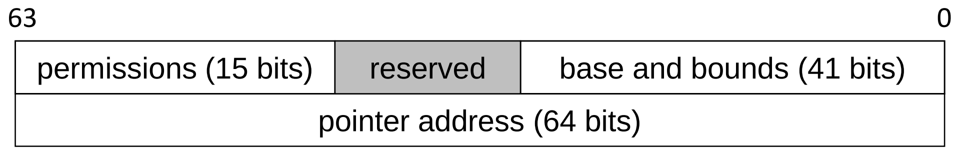
ESBMC-CHERI : Towards Verification of C Programs for CHERI Platforms with ESBMC

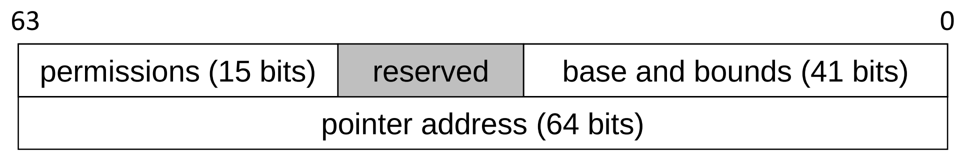
Capability Hardware Enhanced RISC Instructions (CHERI)

Capability Hardware Enhanced RISC Instructions (CHERI)

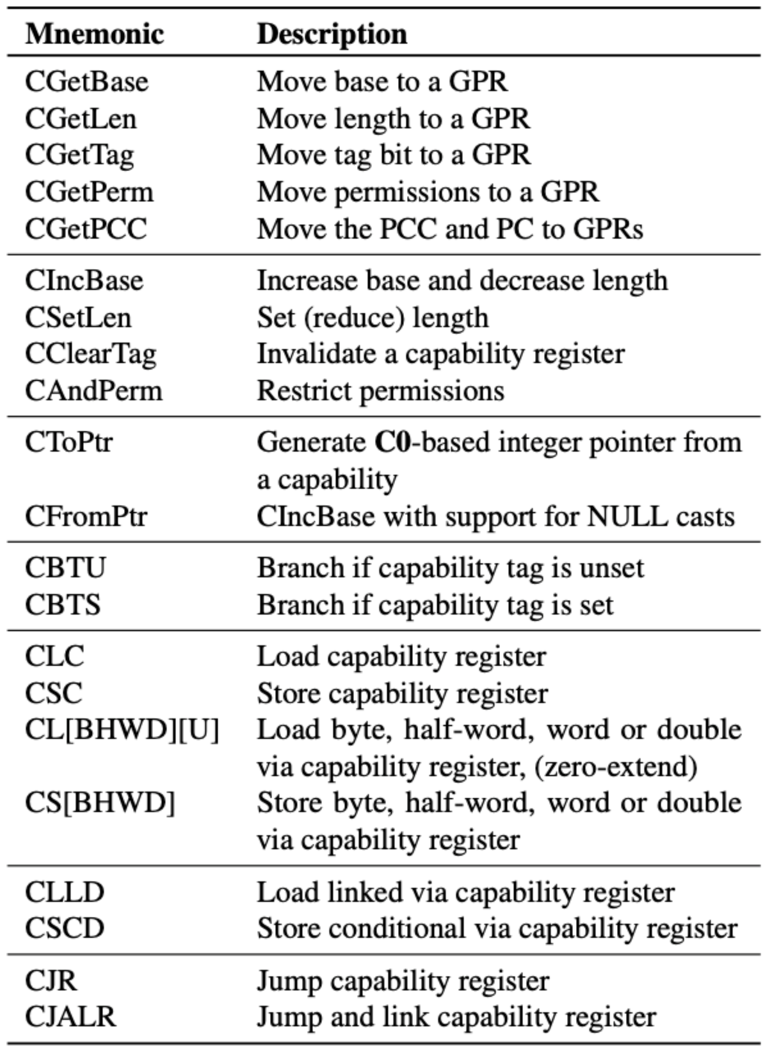


CHERI 128-bit capability

Capability Hardware Enhanced RISC Instructions (CHERI)



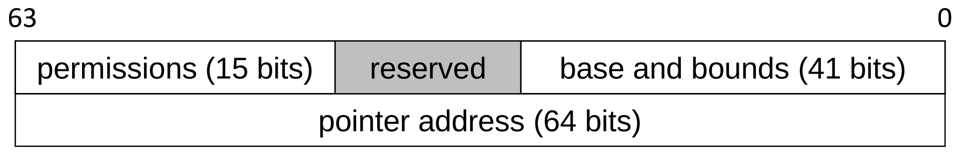
CHERI 128-bit capability



Mnemonic	**Description**
CGetBase	Move base to a GPR
CGetLen	Move length to a GPR
CGetTag	Move tag bit to a GPR
CGetPerm	Move permissions to a GPR
CGetPCC	Move the PCC and PC to GPRs
CIncBase	Increase base and decrease length
CSetLen	Set (reduce) length
CClearTag	Invalidate a capability register
CAndPerm	Restrict permissions
CToPtr	Generate **C0**-based integer pointer from a capability
CFromPtr	CIncBase with support for NULL casts
CBTU	Branch if capability tag is unset
CBTS	Branch if capability tag is set
CLC	Load capability register
CSC	Store capability register
CL[BHWD][U]	Load byte, half-word, word or double via capability register, (zero-extend)
CS[BHWD]	Store byte, half-word, word or double via capability register
CLLD	Load linked via capability register
CSCD	Store conditional via capability register
CJR	Jump capability register
CJALR	Jump and link capability register

CHERI instruction-set extensions

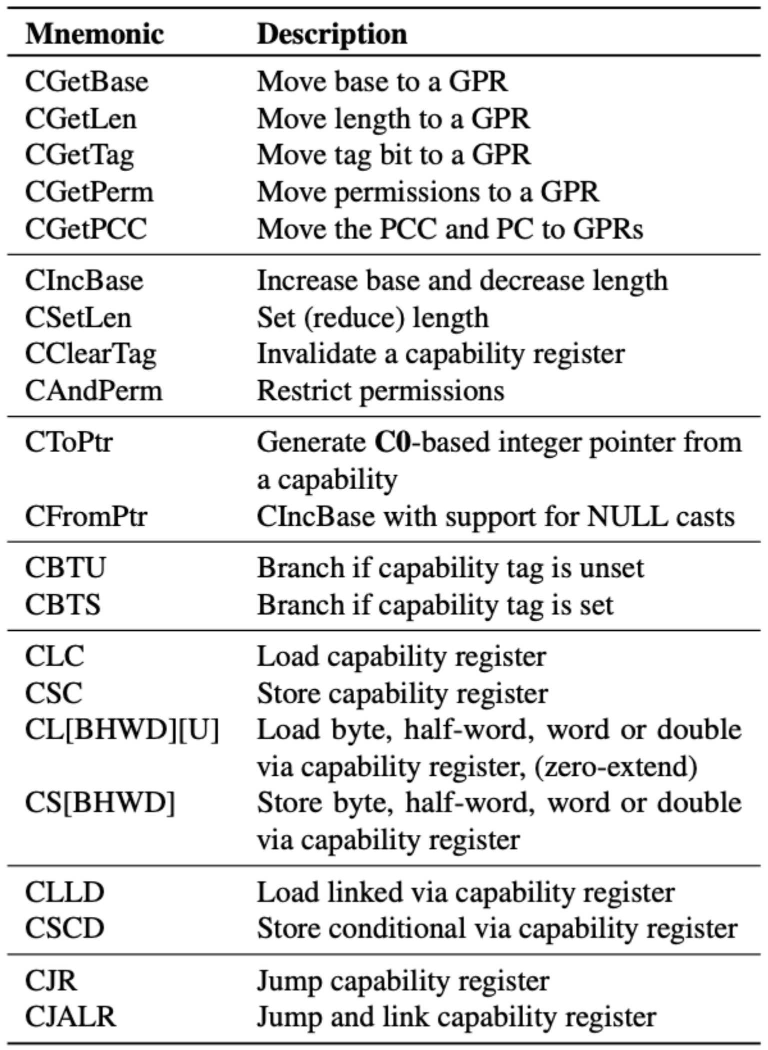
Capability Hardware Enhanced RISC Instructions (CHERI)



CHERI 128-bit capability

CHERI Clang/LLVM and **LLD**[1] - compiler and linker for CHERI ISAs

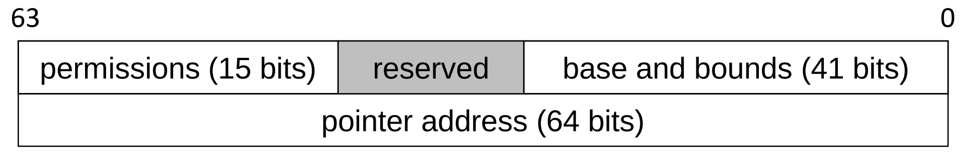
[1]https://www.cl.cam.ac.uk/research/security/ctsrd/cheri/cheri-llvm.html



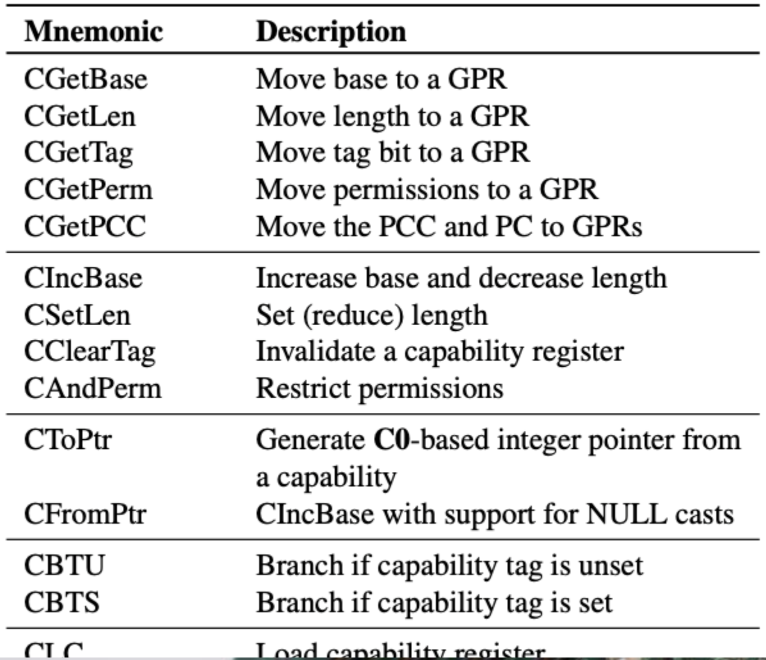
Mnemonic	**Description**
CGetBase	Move base to a GPR
CGetLen	Move length to a GPR
CGetTag	Move tag bit to a GPR
CGetPerm	Move permissions to a GPR
CGetPCC	Move the PCC and PC to GPRs
CIncBase	Increase base and decrease length
CSetLen	Set (reduce) length
CClearTag	Invalidate a capability register
CAndPerm	Restrict permissions
CToPtr	Generate **C0**-based integer pointer from a capability
CFromPtr	CIncBase with support for NULL casts
CBTU	Branch if capability tag is unset
CBTS	Branch if capability tag is set
CLC	Load capability register
CSC	Store capability register
CL[BHWD][U]	Load byte, half-word, word or double via capability register, (zero-extend)
CS[BHWD]	Store byte, half-word, word or double via capability register
CLLD	Load linked via capability register
CSCD	Store conditional via capability register
CJR	Jump capability register
CJALR	Jump and link capability register

CHERI instruction-set extensions

Capability Hardware Enhanced RISC Instructions (CHERI)



CHERI 128-bit capability



Mnemonic	Description
CGetBase	Move base to a GPR
CGetLen	Move length to a GPR
CGetTag	Move tag bit to a GPR
CGetPerm	Move permissions to a GPR
CGetPCC	Move the PCC and PC to GPRs
CIncBase	Increase base and decrease length
CSetLen	Set (reduce) length
CClearTag	Invalidate a capability register
CAndPerm	Restrict permissions
CToPtr	Generate **C0**-based integer pointer from a capability
CFromPtr	CIncBase with support for NULL casts
CBTU	Branch if capability tag is unset
CBTS	Branch if capability tag is set
CLC	Load capability register

CHERI Clang/LLVM and **LLD**[1] - compiler and linker for CHERI ISAs

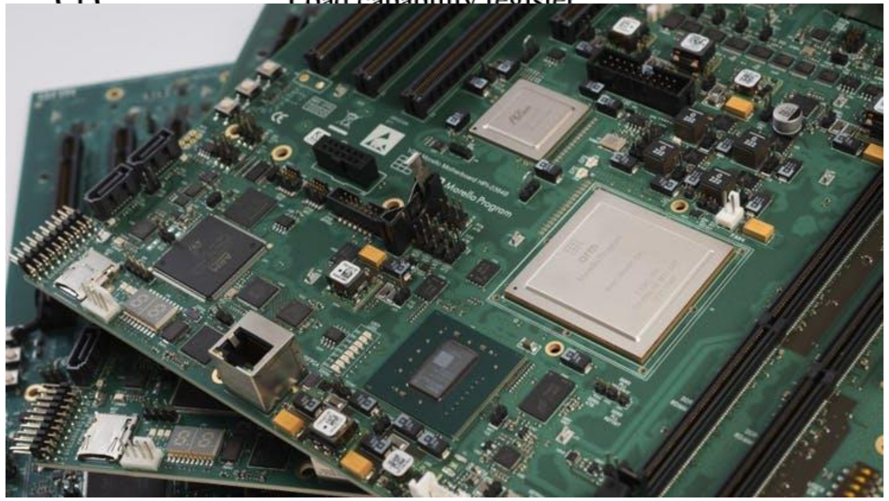
[1]https://www.cl.cam.ac.uk/research/security/ctsrd/cheri/cheri-llvm.html

CheriBSD[2] - adaptation of FreeBSD to support CHERI ISAs

[2]https://www.cl.cam.ac.uk/research/security/ctsrd/cheri/cheribsd.html

ARM Morello[3] - SoC development board with a CHERI-extended ARMv8-A processor

[3]https://www.arm.com/architecture/cpu/morello



CHERI-C program

```
#include <stdlib.h>
#include <string.h>
#include <cheri/cheric.h>

void main() {
   int n = nondet_uint() % 1024;                              /* models arbitrary user input */
   char a[n+1], *__capability b = cheri_ptr(a, n+1);
   b[n] = 17;                                                 /* succeeds */
   char *__capability c = cheri_setbounds(b-1, n);            /* fails: not the same object */
   /* ... */                                                  /* more CHERI-C API checks */
   memset_c(c, 42, n);                                        /* setting memory through a capability */
}
```

CHERI-C program

```
#include <stdlib.h>
#include <string.h>
#include <cheri/cheric.h>

void main() {
   int n = nondet_uint() % 1024;                               /* models arbitrary user input */
   char a[n+1], *__capability b = cheri_ptr(a, n+1);
   b[n] = 17;                                                  /* succeeds */
   char *__capability c = cheri_setbounds(b-1, n);             /* fails: not the same object */
   /* ... */                                                   /* more CHERI-C API checks */
   memset_c(c, 42, n);                                         /* setting memory through a capability */
}
```

New capability types

CHERI-C program

CHERI-C API

```
#include <stdlib.h>
#include <string.h>
#include <cheri/cheric.h>

void main() {
    int n = nondet_uint() % 1024;                            /* models arbitrary user input */
    char a[n+1], *__capability b = cheri_ptr(a, n+1);
    b[n] = 17;                                               /* succeeds */
    char *__capability c = cheri_setbounds(b-1, n);          /* fails: not the same object */
    /* ... */                                                /* more CHERI-C API checks */
    memset_c(c, 42, n);                                      /* setting memory through a capability */
}
```

New capability types

CHERI-C program

```
#include <stdlib.h>
#include <string.h>
#include <cheri/cheric.h>

void main() {
   int n = nondet_uint() % 1024;                       /* models arbitrary user input */
   char a[n+1], *__capability b = cheri_ptr(a, n+1);
   b[n] = 17;                                          /* succeeds */
   char *__capability c = cheri_setbounds(b-1, n);     /* fails: not the same object */
   /* ... */                                           /* more CHERI-C API checks */
   memset_c(c, 42, n);                                 /* setting memory through a capability */
}
```

CHERI-C API

New capability types

Pure-capability CHERI-C model

```
#include <stdlib.h>
#include <string.h>
#include <cheri/cheric.h>

void main() {
   int n = nondet_uint() % 1024;
   char a[n+1], *__capability b = cheri_ptr(a, n+1);
   b[n] = 17;
   char *__capability c = cheri_setbounds(b-1, n);
   /* ... */
   memset_c(c, 42, n);
}
```

Pure-capability CHERI-C model

```
#include <stdlib.h>
#include <string.h>
#include <cheri/cheric.h>

void main() {
   int n = nondet_uint() % 1024;
   char a[n+1], *__capability b = cheri_ptr(a, n+1);
   b[n] = 17;
   char *__capability c = cheri_setbounds(b-1, n);
   /* ... */
   memset_c(c, 42, n);
}
```

```
#include <string.h>
#include <stdio.h>

void main(void) {
   int n = nondet_uint() % 1024;
   char a[n+1], *b = a;
   b[n] = 17;
   char *c = b-1;
   memset(c, 42, n);
}
```

Pure-capability CHERI-C model

```
#include <stdlib.h>
#include <string.h>
#include <cheri/cheric.h>

void main() {
   int n = nondet_uint() % 1024;
   char a[n+1], *__capability b = cheri_ptr(a, n+1);
   b[n] = 17;
   char *__capability c = cheri_setbounds(b-1, n);
   /* ... */
   memset_c(c, 42, n);
}
```

```
#include <string.h>
#include <stdio.h>

void main(void) {
   int n = nondet_uint() % 1024;
   char a[n+1], *b = a;
   b[n] = 17;
   char *c = b-1;
   memset(c, 42, n);
}
```

All pointers are automatically replaced with capabilities by the **CHERI Clang/LLVM** compiler

Pure-capability CHERI-C model

```
#include <stdlib.h>
#include <string.h>
#include <cheri/cheric.h>

void main() {
   int n = nondet_uint() % 1024;
   char a[n+1], *__capability b = cheri_ptr(a, n+1);
   b[n] = 17;
   char *__capability c = cheri_setbounds(b-1, n);
   /* ... */
   memset_c(c, 42, n);
}
```

```
#include <string.h>
#include <stdio.h>

void main(void) {
   int n = nondet_uint() % 1024;
   char a[n+1], *b = a;
   b[n] = 17;
   char *c = b-1;
   memset(c, 42, n);
}
```

All pointers are automatically replaced with capabilities by the **CHERI Clang/LLVM** compiler

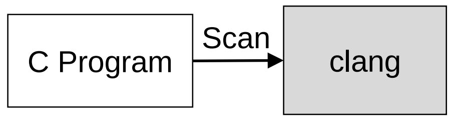
ESBMC-CHERI is the *first* tool capable of formally verifying C programs for CHERI platforms

ESBMC

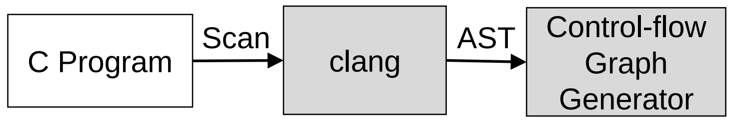
ESBMC

C Program

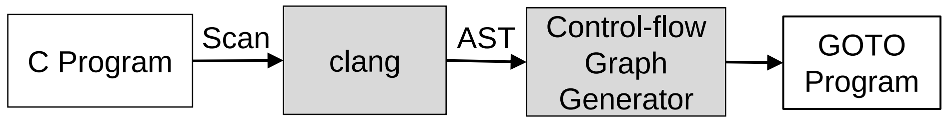
ESBMC



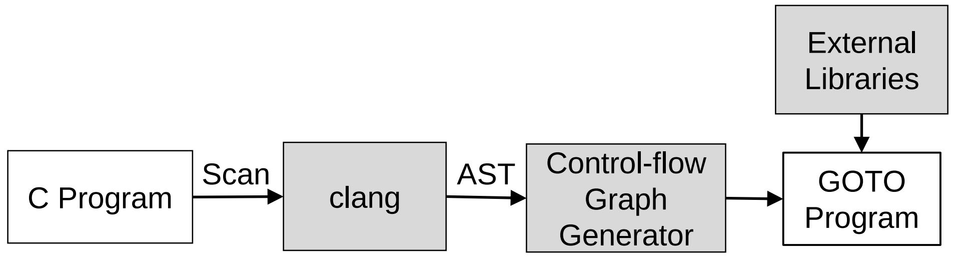
ESBMC



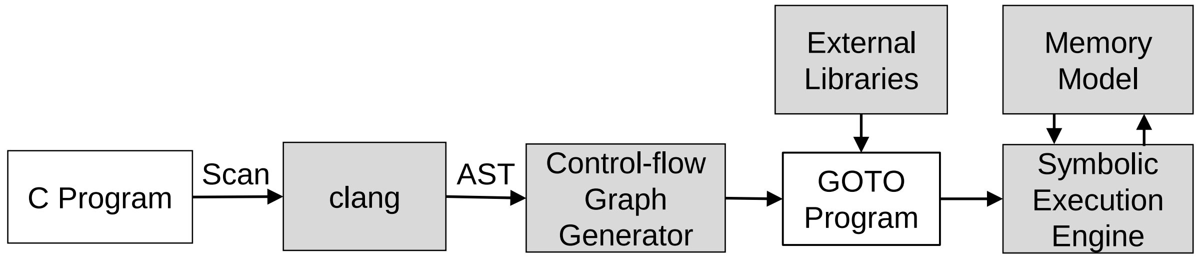
ESBMC


C Program
Scan
clang
AST
Control-flow Graph Generator
GOTO Program

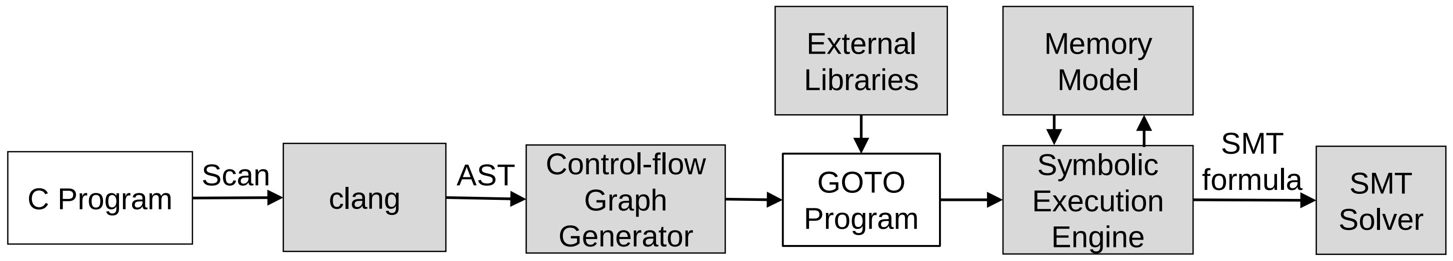
ESBMC


External Libraries
C Program
Scan
clang
AST
Control-flow Graph Generator
GOTO Program

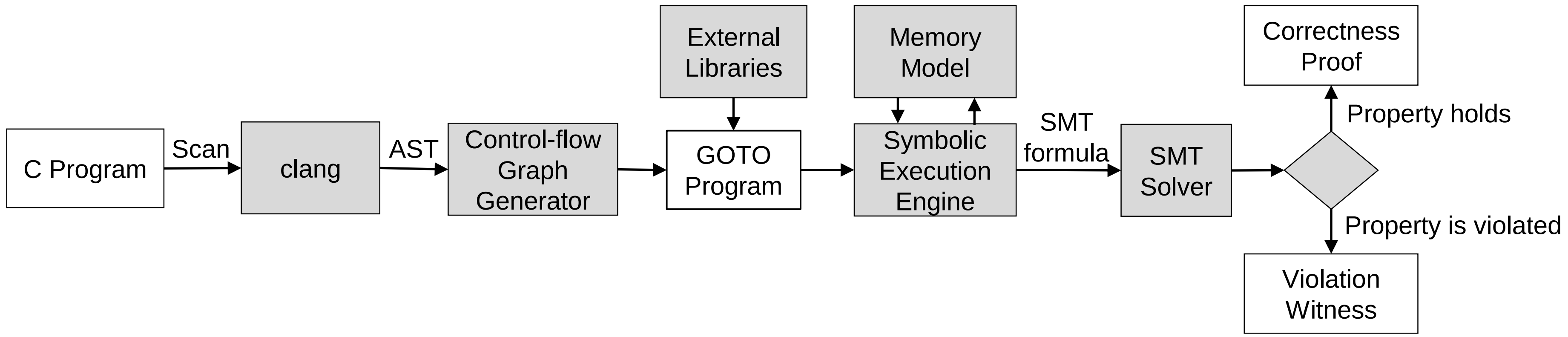
ESBMC


External Libraries
Memory Model
C Program
Scan
clang
AST
Control-flow Graph Generator
GOTO Program
Symbolic Execution Engine

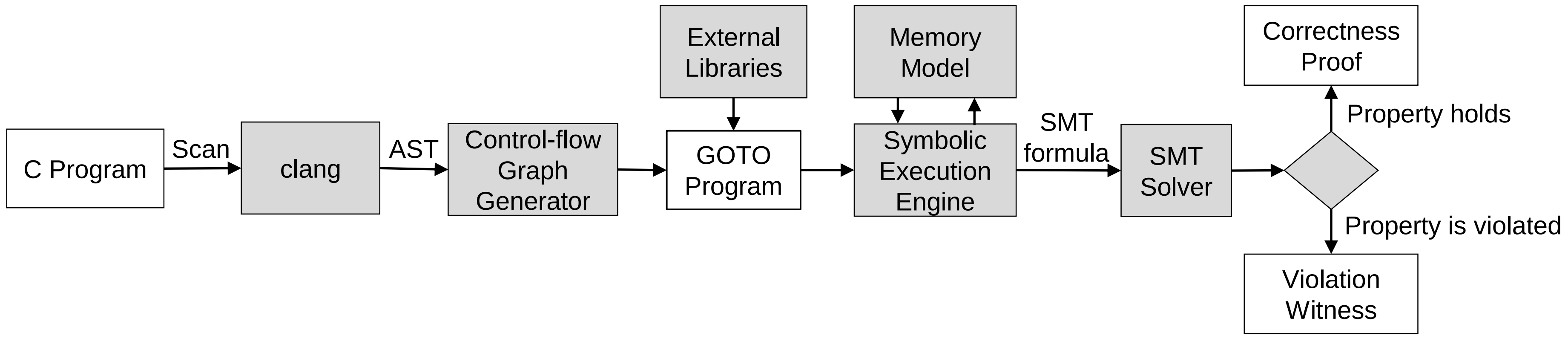
ESBMC


External Libraries
Memory Model
C Program
Scan
clang
AST
Control-flow Graph Generator
GOTO Program
Symbolic Execution Engine
SMT formula
SMT Solver

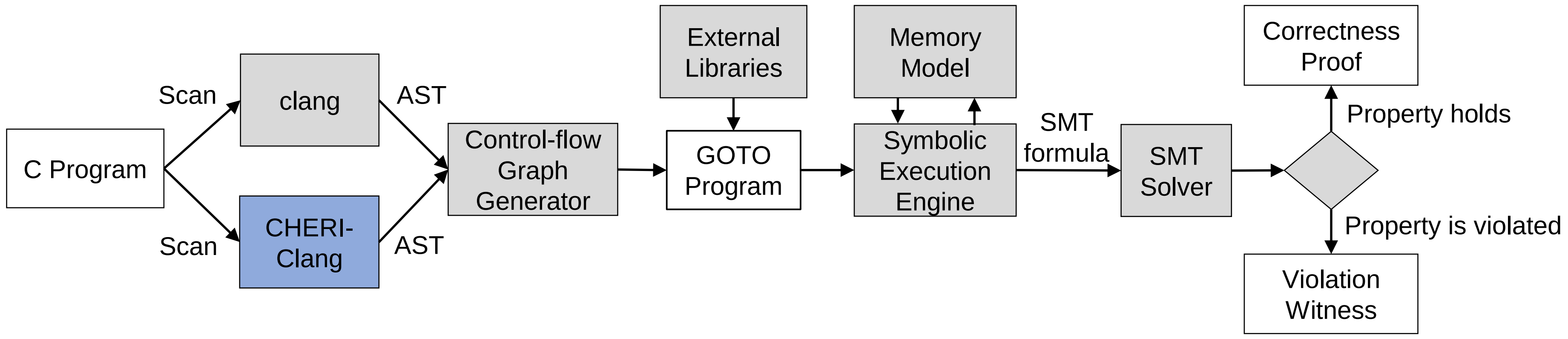
ESBMC


External Libraries
Memory Model
Correctness Proof
Property holds
C Program
Scan
clang
AST
Control-flow Graph Generator
GOTO Program
Symbolic Execution Engine
SMT formula
SMT Solver
Property is violated
Violation Witness

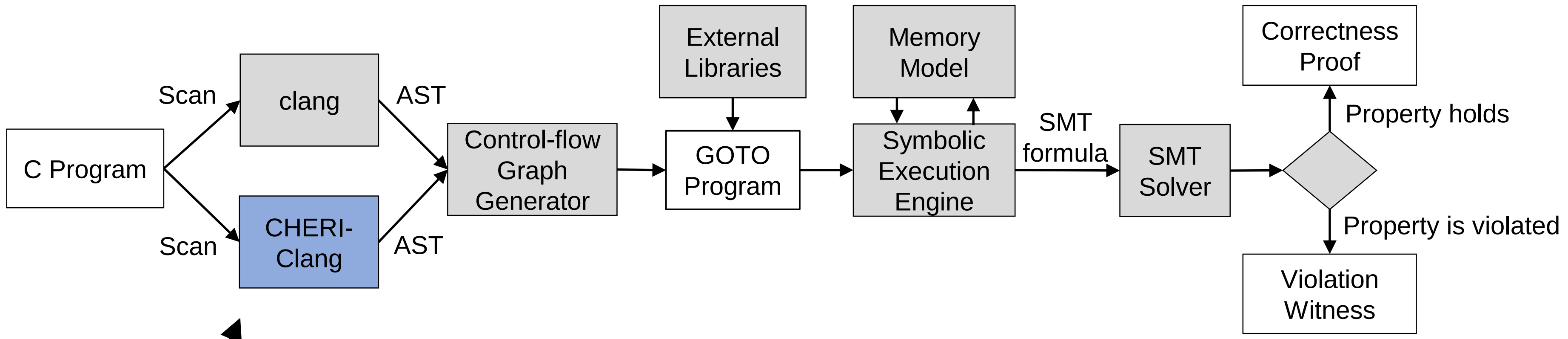
ESBMC-CHERI


External Libraries
Memory Model
Correctness Proof
C Program
Scan
clang
AST
Control-flow Graph Generator
GOTO Program
Symbolic Execution Engine
SMT formula
SMT Solver
Property holds
Property is violated
Violation Witness

ESBMC-CHERI

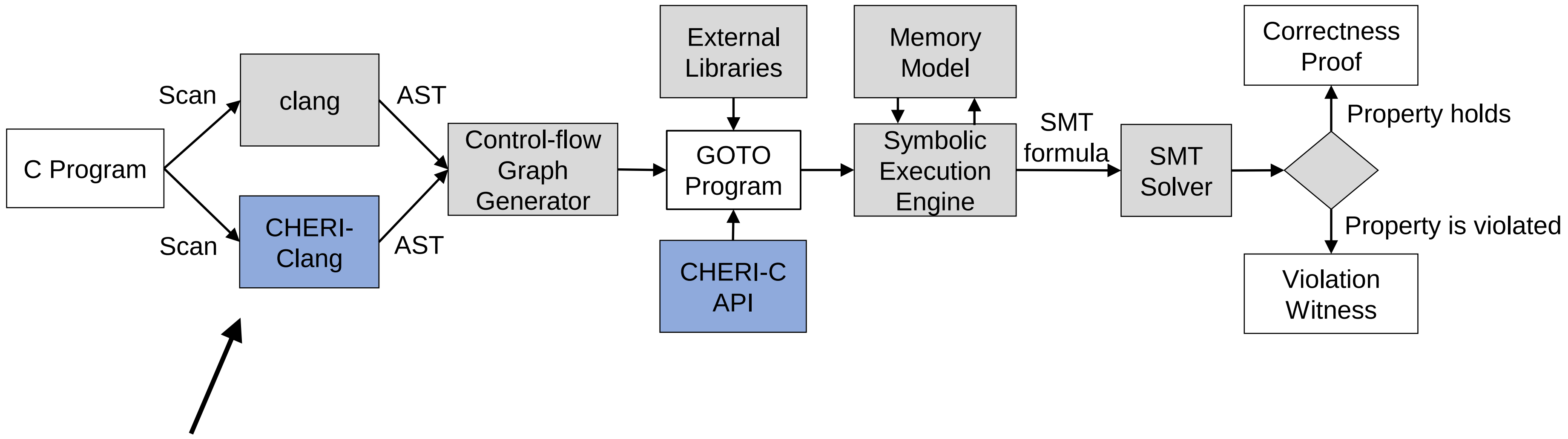

C Program
Scan
Scan
clang
CHERI-Clang
AST
AST
Control-flow Graph Generator
External Libraries
GOTO Program
Memory Model
Symbolic Execution Engine
SMT formula
SMT Solver
Property holds
Correctness Proof
Property is violated
Violation Witness

ESBMC-CHERI



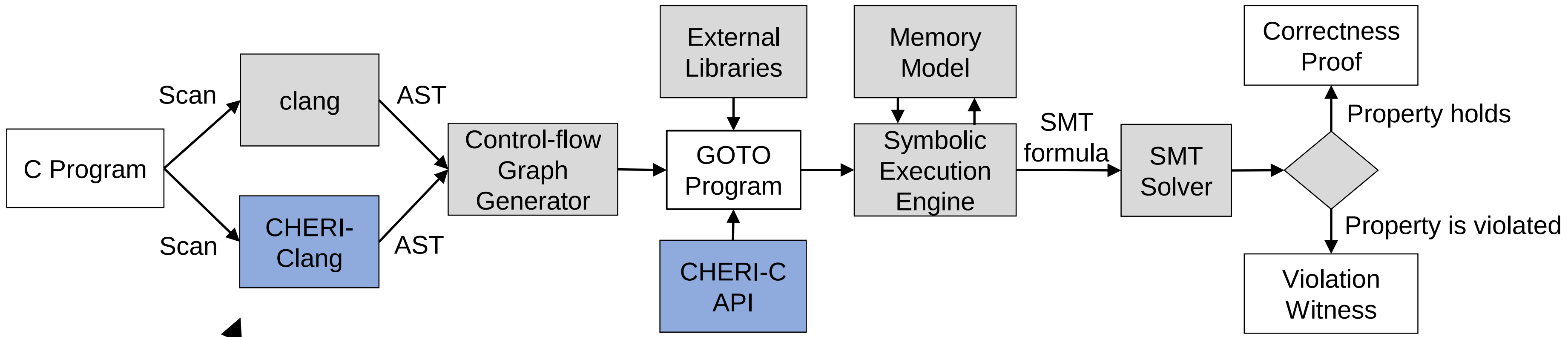
CHERI Clang/LLVM compiler

ESBMC-CHERI



CHERI Clang/LLVM
compiler

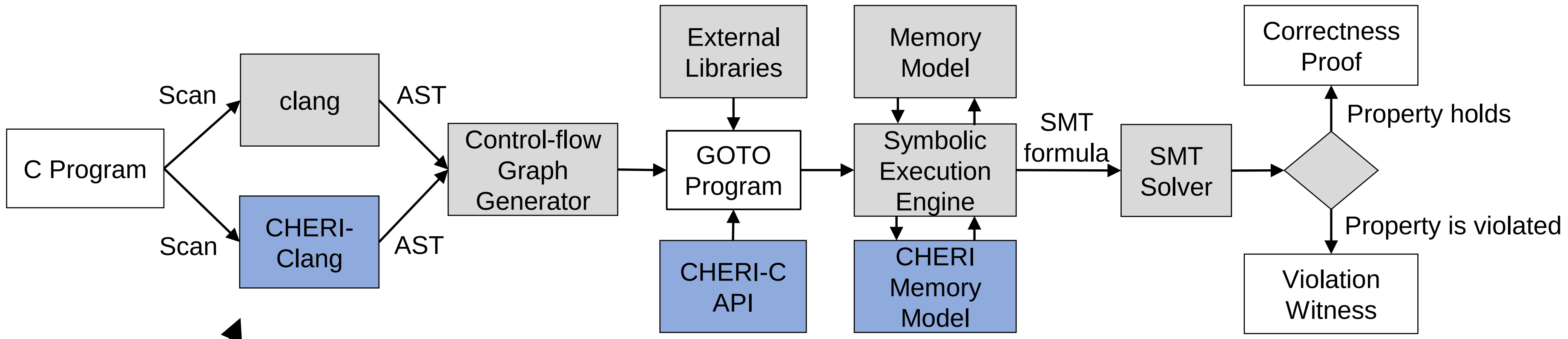
ESBMC-CHERI



CHERI Clang/LLVM
compiler

Implement computational model for CHERI-C API functions inside ESBMC (e.g., ***cheri_setbounds***)

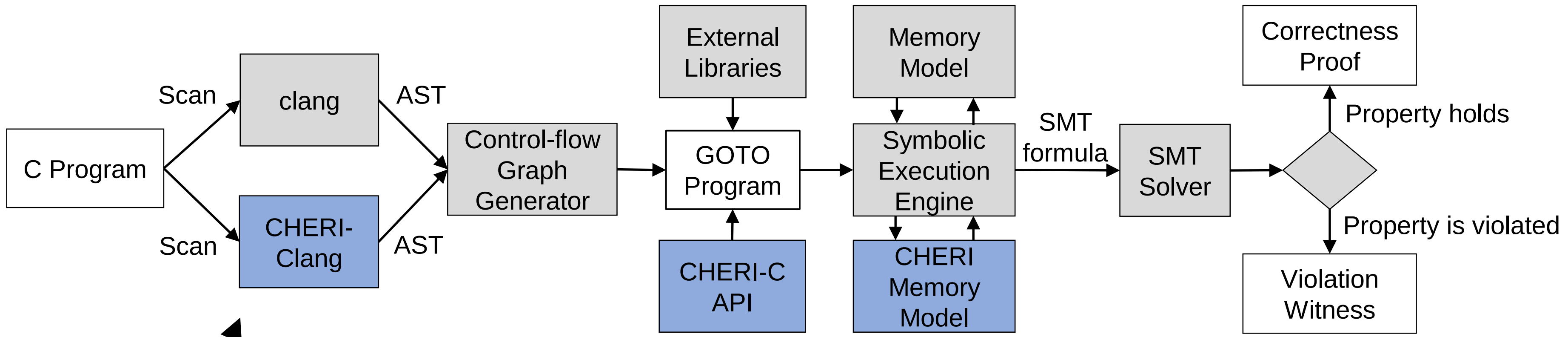
ESBMC-CHERI



CHERI Clang/LLVM
compiler

Implement computational model for CHERI-C API functions inside ESBMC (e.g., ***cheri_setbounds***)

ESBMC-CHERI



CHERI Clang/LLVM compiler

Implement computational model for CHERI-C API functions inside ESBMC (e.g., ***cheri_setbounds***)

- New capability types
- Tagged memory
- Capability dereferencing

Computational model of *cheri_setbounds*

```
/* modelled after UCAM-CL-TR-951 semantics of CHERI-MIPS instruction CSetBounds */
void *__capability cheri_setbounds(void *__capability cap, __SIZE_TYPE__ sz)
{
#if 1
  union __esbmc_cheri_cap128 u = { cap };
  cc128_cap_t comp;
  cc128_decompress_mem(u.pesbt, u.cursor, true /* tag */, &comp);
  __PTRADDR_TYPE__ cursor = comp._cr_cursor;
  __PTRADDR_TYPE__ base = comp.cr_base;
  __PTRADDR_TYPE__ top = comp._cr_top;
#else
  __PTRADDR_TYPE__ cursor = (__PTRADDR_TYPE__)cap;
  __PTRADDR_TYPE__ base = cheri_getbase(cap);
  __PTRADDR_TYPE__ top = cheri_gettop(cap);
#endif
  __ESBMC_assert(cheri_gettag(cap), "tag-violation c2exception");
  __ESBMC_assert(base <= cursor, "length-violation c2exception");
  __uint128_t newTop = cursor;
  newTop += sz;
  bool exact = cc128_setbounds(&comp, cursor, newTop);
  (void)exact; /* ignore */
  u.pesbt = cc128_compress_mem(&comp);
  __ESBMC_assert(__ESBMC_POINTER_OBJECT((__cheri_fromcap void *)u.cap) == __ESBMC_POINTER_OBJECT((__cheri_fromcap void *)cap), "error: not same pointer_object");
  __ESBMC_assert(__ESBMC_POINTER_OFFSET((__cheri_fromcap void *)u.cap) == __ESBMC_POINTER_OFFSET((__cheri_fromcap void *)cap), "error: not same pointer_offset");
  __ESBMC_assume(__ESBMC_same_object((__cheri_fromcap void *)u.cap, (__cheri_fromcap void *)cap));
  return u.cap;
}
```

Thank you for watching!!!