Symbion

Interleaving Symbolic with Concrete Execution

***Fabio Gritti**, Lorenzo Fontana, Eric Gustafson, Fabio Pagani, Andrea Continella, Christopher Kruegel, and Giovanni Vigna*

University of California, Santa Barbara

Motivation

- **Symbolically execution of binaries is very useful.**
  - Identify bugs and security vulnerabilities
  - Reverse-engineer closed-source software
  - Formally verify properties

- **Scalability of symbolic execution is an issue**
  - State/path explosion
  - Program behaviors can't always be fully modeled by symbolic execution engines
    - Complex state initializations
    - Filesystem accesses
    - Network requests
    - Interrupts

Motivation

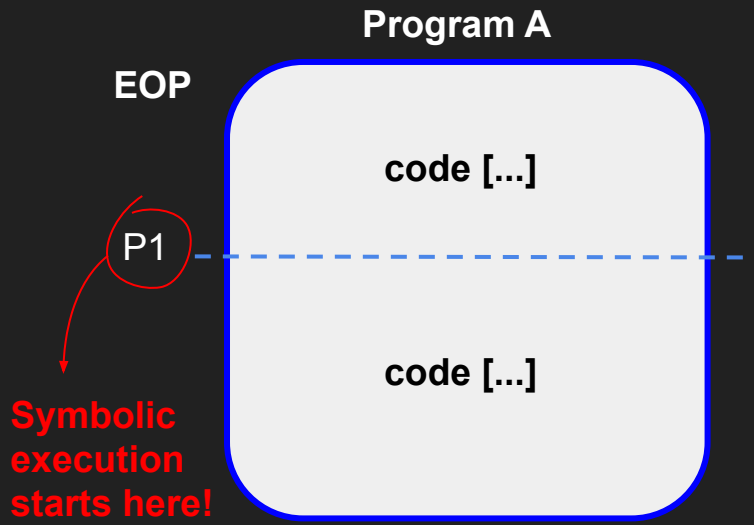
- **Symbolically execution of binaries is very useful.**
  - Identify bugs and security vulnerabilities
  - Reverse-engineer closed-source software
  - Formally verify properties

- **Scalability of symbolic execution is an issue**
  - State/path explosion
  - Program behaviors can't always be fully modeled by symbolic execution engines
    - Complex state initializations
    - Filesystem accesses
    - Network requests
    - Interrupts

CAN'T EXECUTE THE WHOLE PROGRAM SYMBOLICALLY!

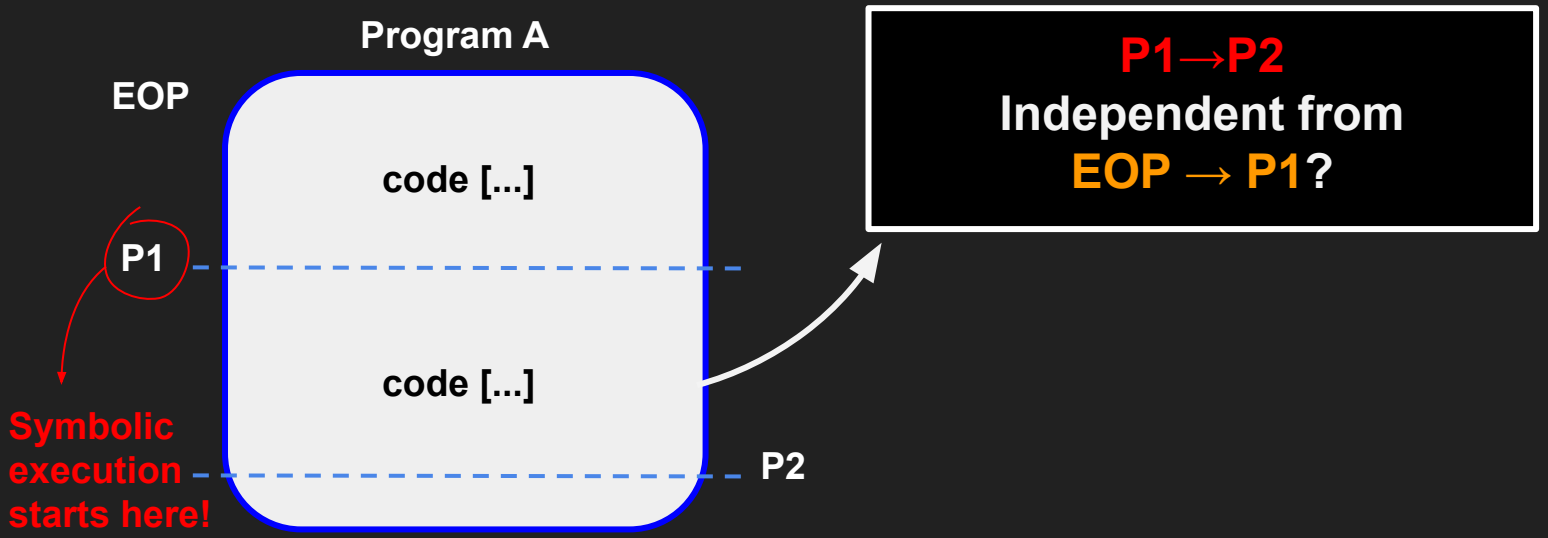
Motivation

- **Idea: why not just focus on a smaller portion of the code?**
  - Also known as **under-constrained symbolic execution.**



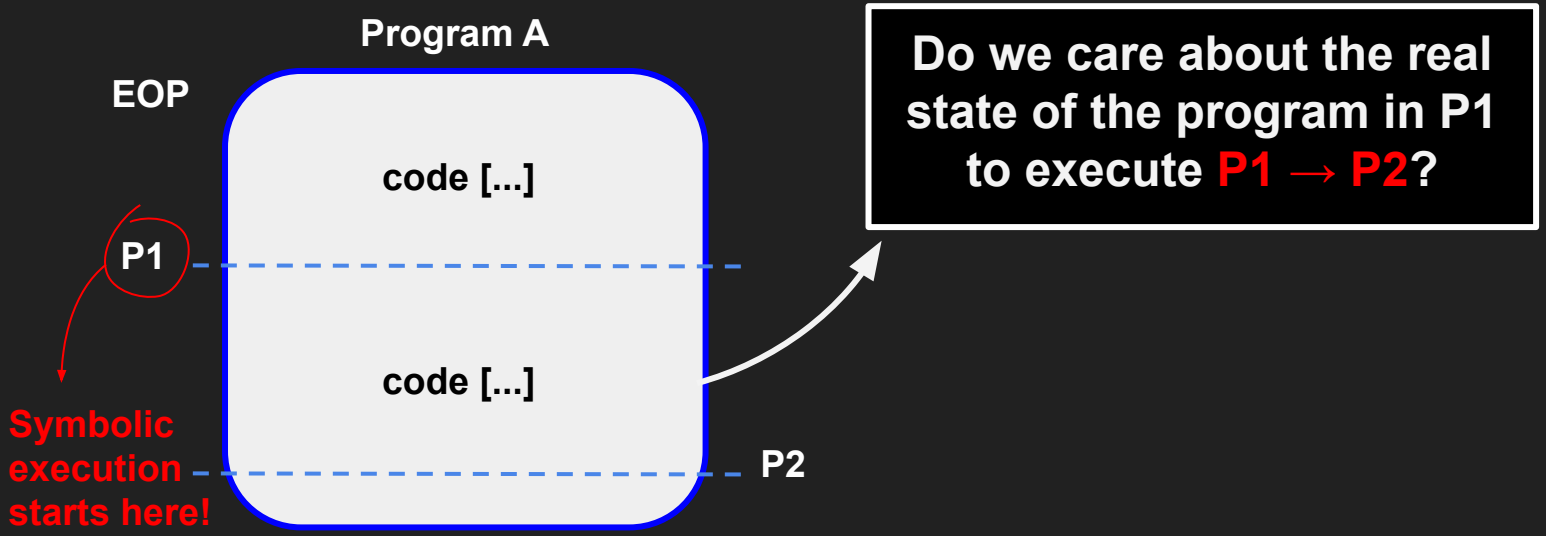
Motivation

- **Idea: why not just focus on a smaller portion of the code?**
  - Also known as **under-constrained symbolic execution.**



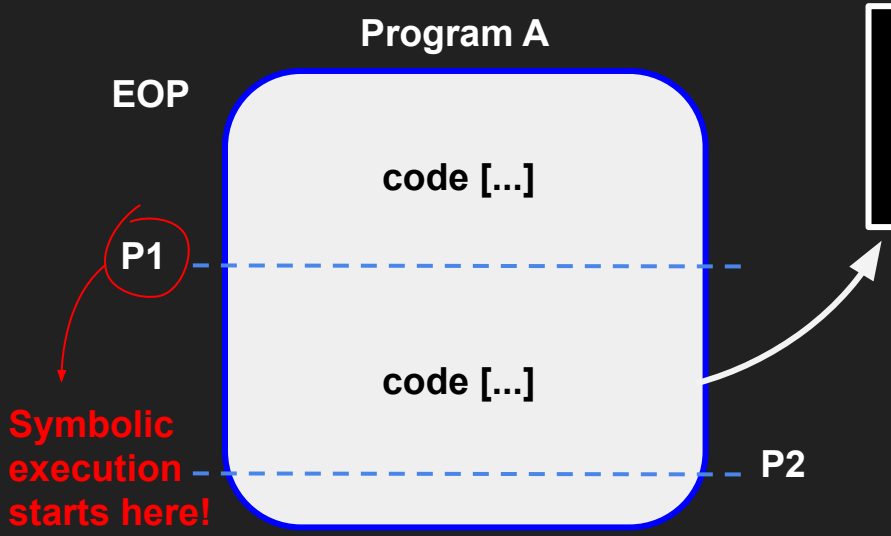
Motivation

- **Idea: why not just focus on a smaller portion of the code?**
  - Also known as **under-constrained symbolic execution.**



Motivation

- **Idea: why not just focus on a smaller portion of the code?**
  - Also known as **under-constrained symbolic execution.**

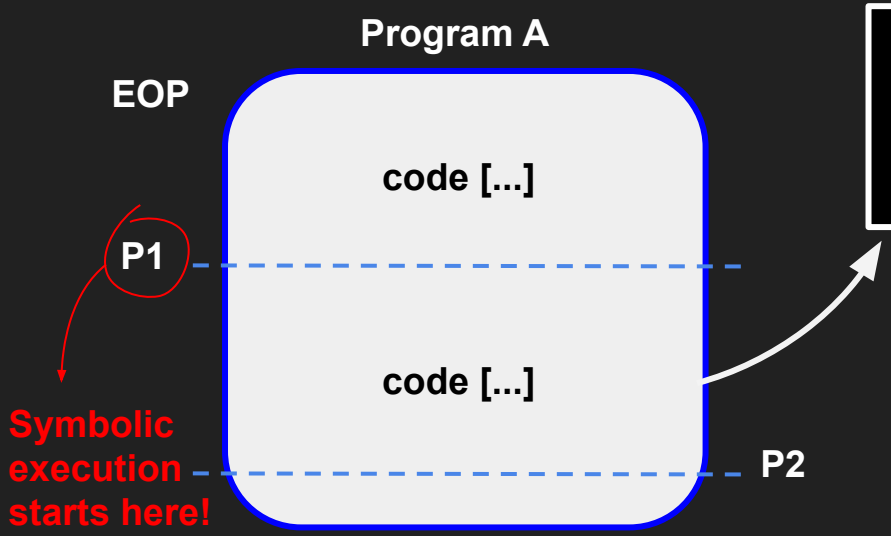


Do we care about the real state of the program in P1 to execute P1 → P2?

NO

Motivation

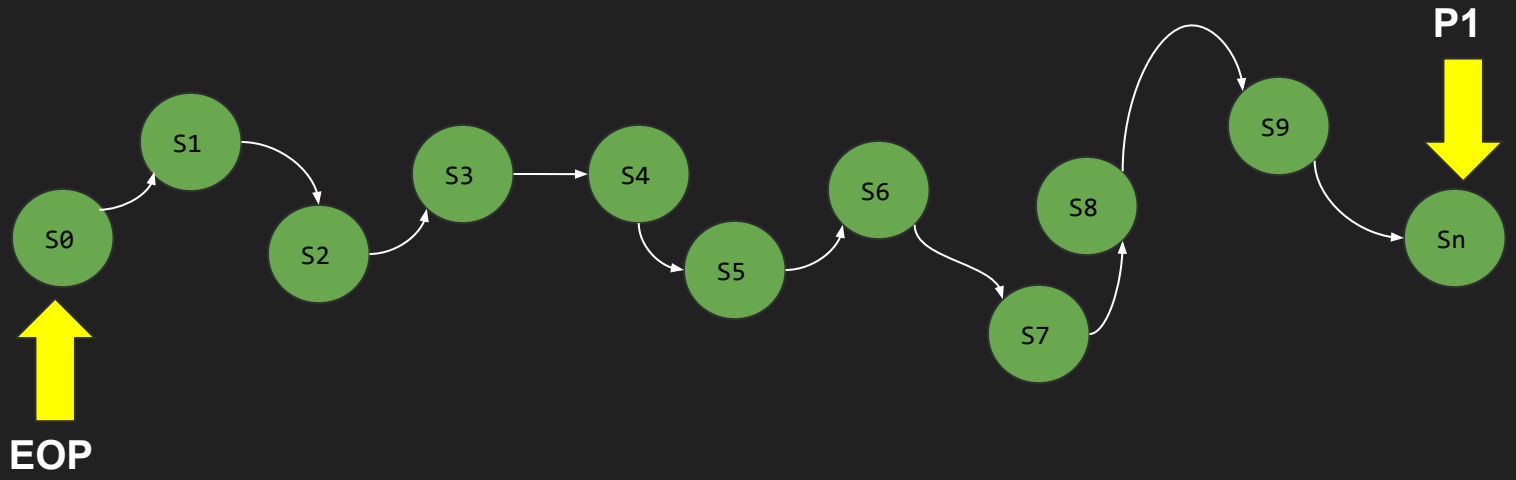
- **Idea: why not just focus on a smaller portion of the code?**
  - Also known as **under-constrained symbolic execution.**



Do we care about the real state of the program in P1 to execute P1 → P2?

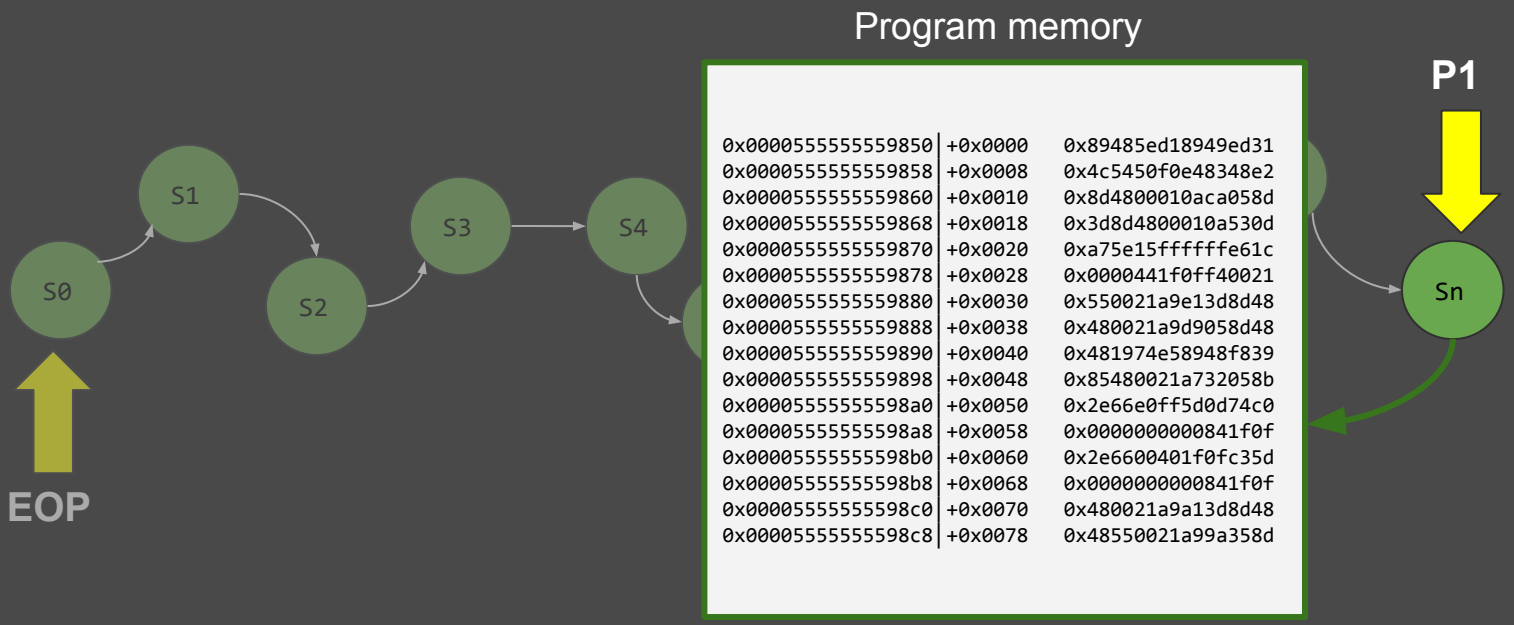
YES

Motivation



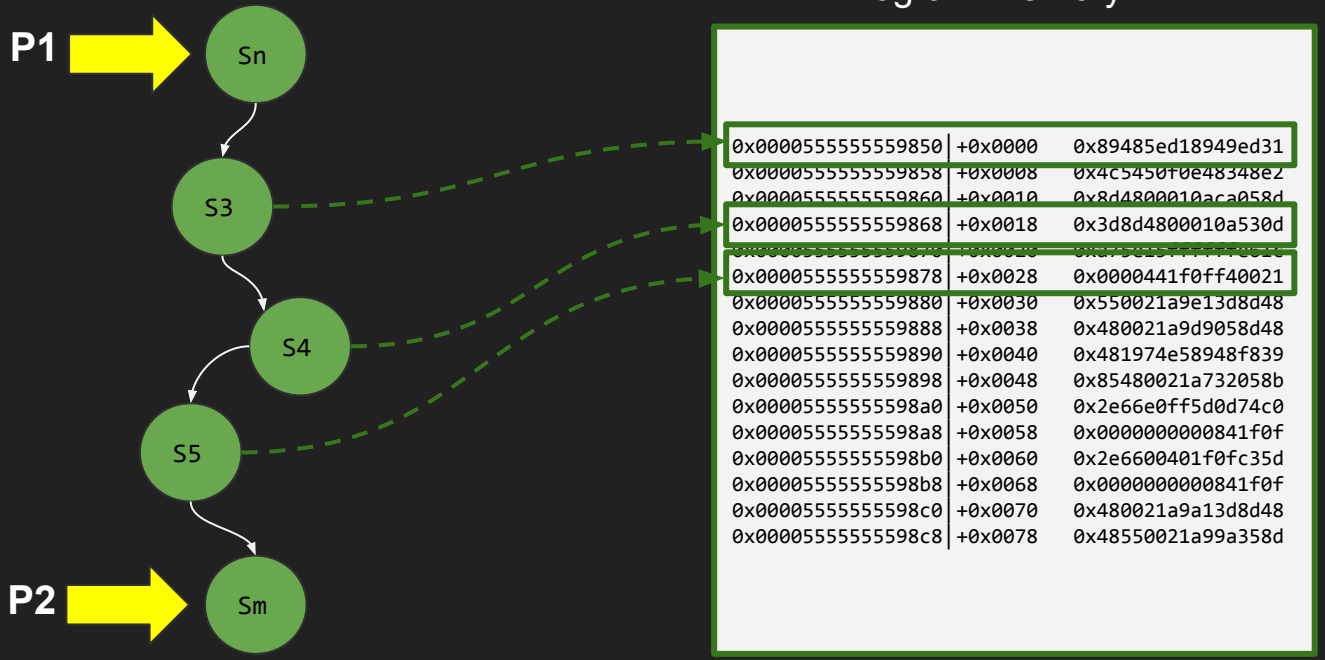
typical concrete execution

Motivation



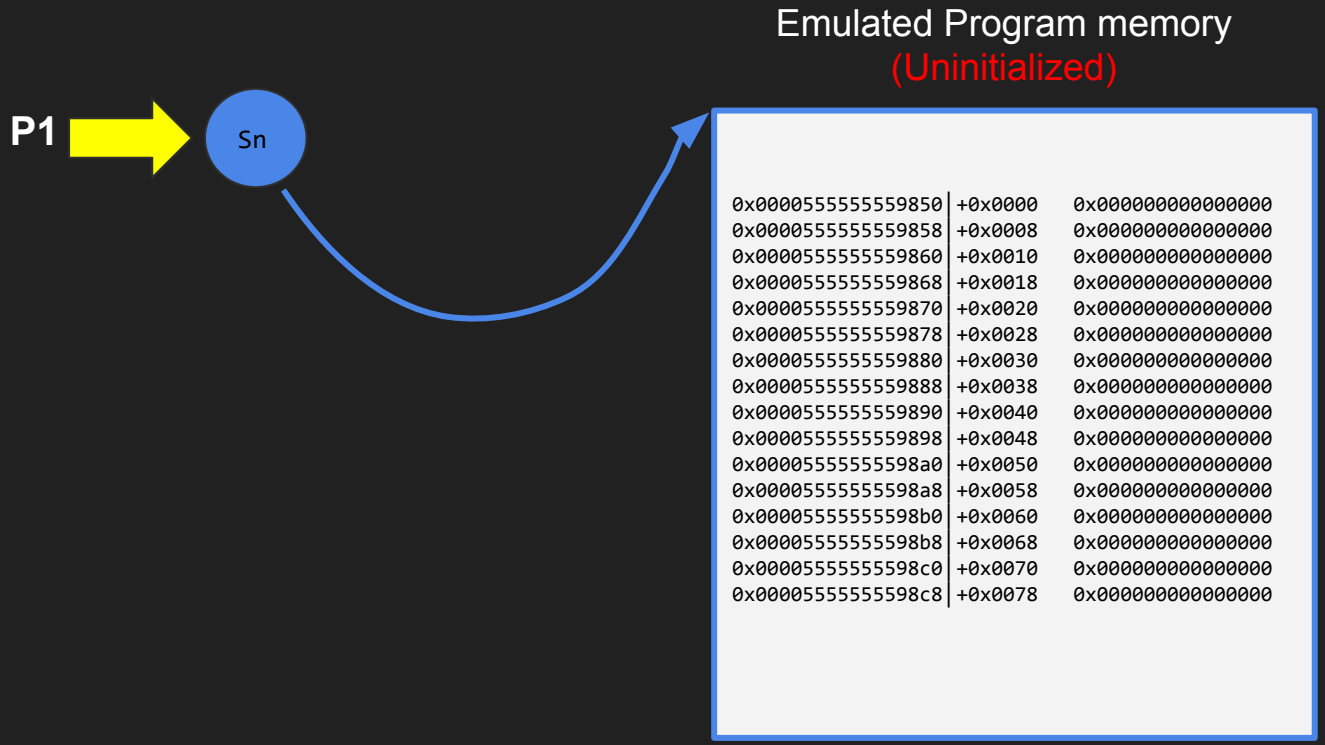
typical concrete execution

Motivation



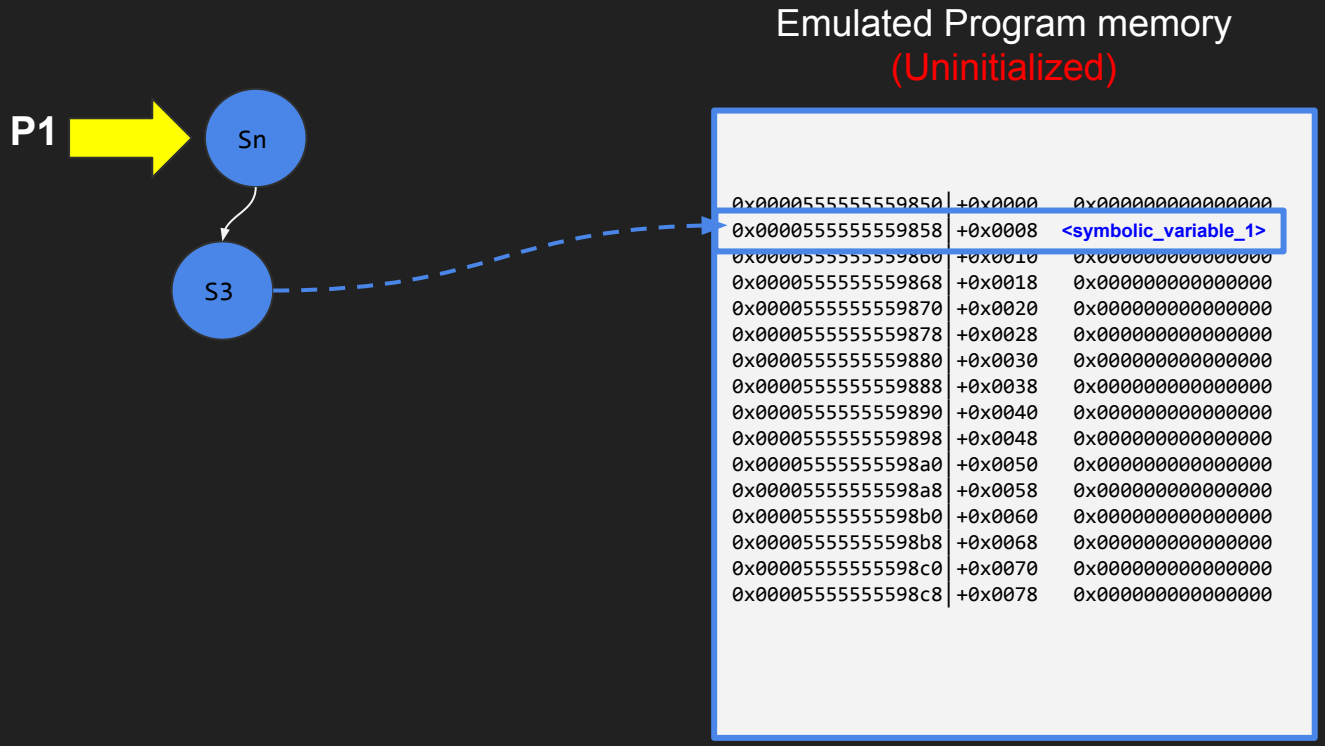
typical concrete execution

Motivation



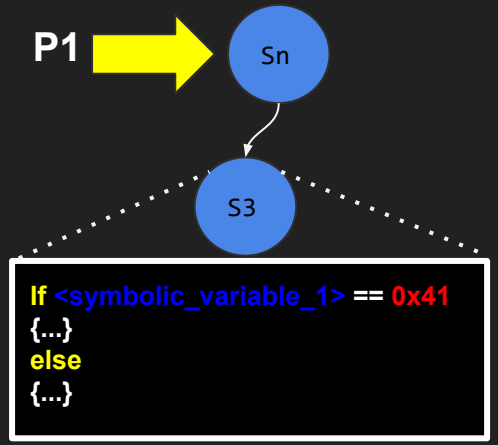
“under-constrained” symbolic execution

Motivation



“under-constrained” symbolic execution

Motivation

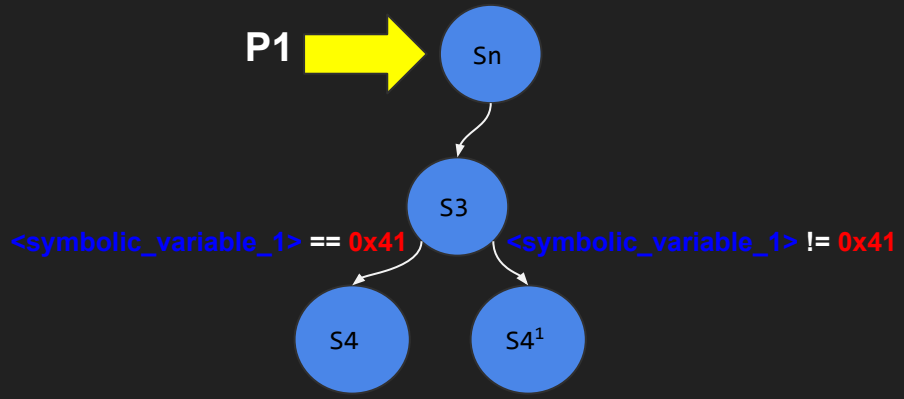


Emulated Program memory
(Uninitialized)

0x0000555555559858	+0x0008	<symbolic_variable_1>
0x0000555555559868	+0x0018	0x0000000000000000
0x0000555555559870	+0x0020	0x0000000000000000
0x0000555555559878	+0x0028	0x0000000000000000
0x0000555555559880	+0x0030	0x0000000000000000
0x0000555555559888	+0x0038	0x0000000000000000
0x0000555555559890	+0x0040	0x0000000000000000
0x0000555555559898	+0x0048	0x0000000000000000
0x00005555555598a0	+0x0050	0x0000000000000000
0x00005555555598a8	+0x0058	0x0000000000000000
0x00005555555598b0	+0x0060	0x0000000000000000
0x00005555555598b8	+0x0068	0x0000000000000000
0x00005555555598c0	+0x0070	0x0000000000000000
0x00005555555598c8	+0x0078	0x0000000000000000

“under-constrained” symbolic execution

Motivation

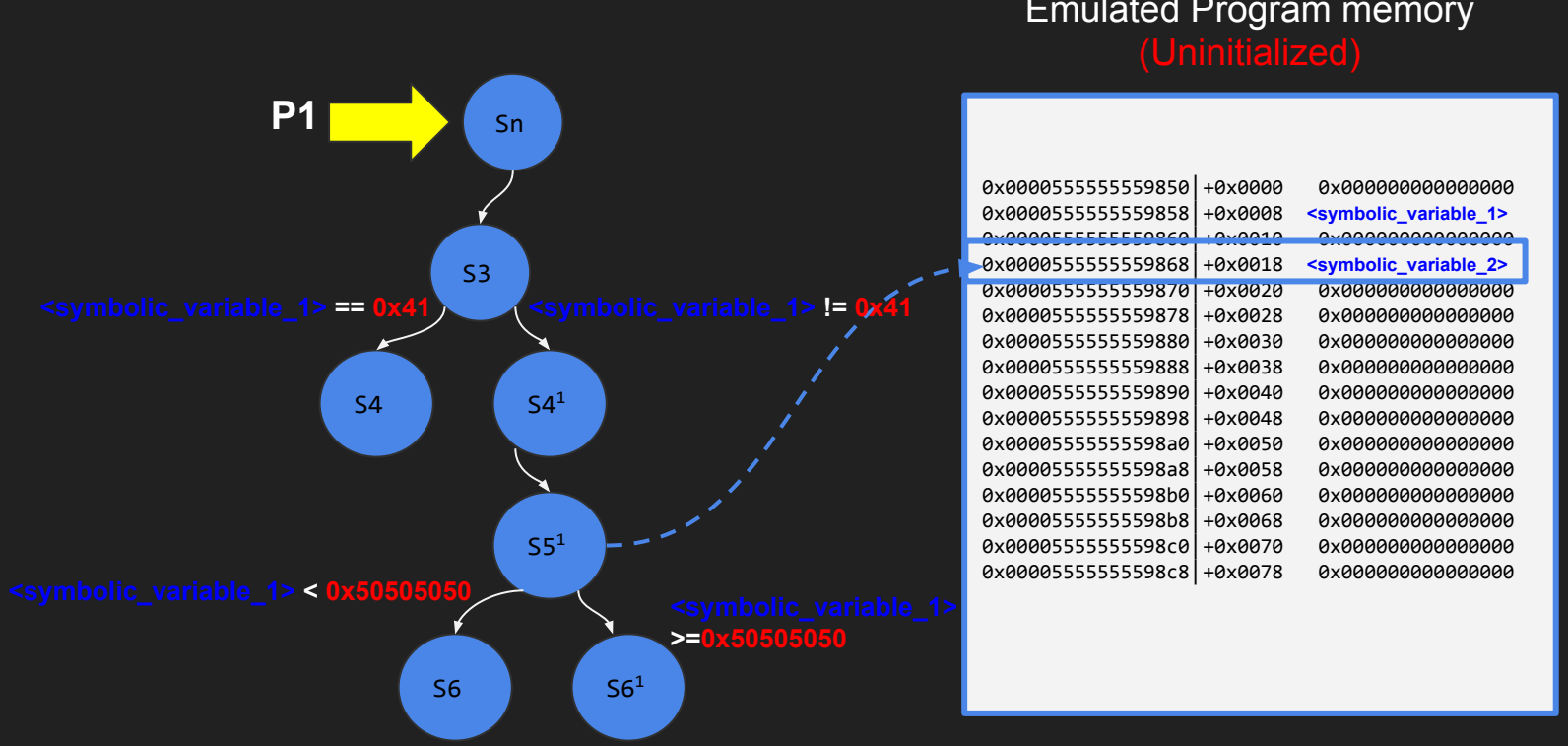


Emulated Program memory
(Uninitialized)

0x0000555555559858	+0x0008	<symbolic_variable_1>
0x0000555555559868	+0x0018	0x0000000000000000
0x0000555555559870	+0x0020	0x0000000000000000
0x0000555555559878	+0x0028	0x0000000000000000
0x0000555555559880	+0x0030	0x0000000000000000
0x0000555555559888	+0x0038	0x0000000000000000
0x0000555555559890	+0x0040	0x0000000000000000
0x0000555555559898	+0x0048	0x0000000000000000
0x00005555555598a0	+0x0050	0x0000000000000000
0x00005555555598a8	+0x0058	0x0000000000000000
0x00005555555598b0	+0x0060	0x0000000000000000
0x00005555555598b8	+0x0068	0x0000000000000000
0x00005555555598c0	+0x0070	0x0000000000000000
0x00005555555598c8	+0x0078	0x0000000000000000

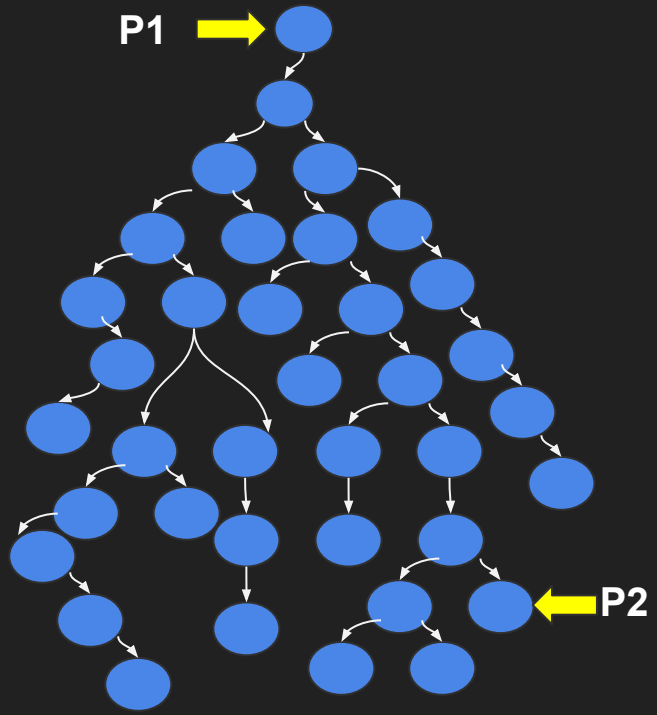
“under-constrained” symbolic execution

Motivation



"under-constrained" symbolic execution

Motivation

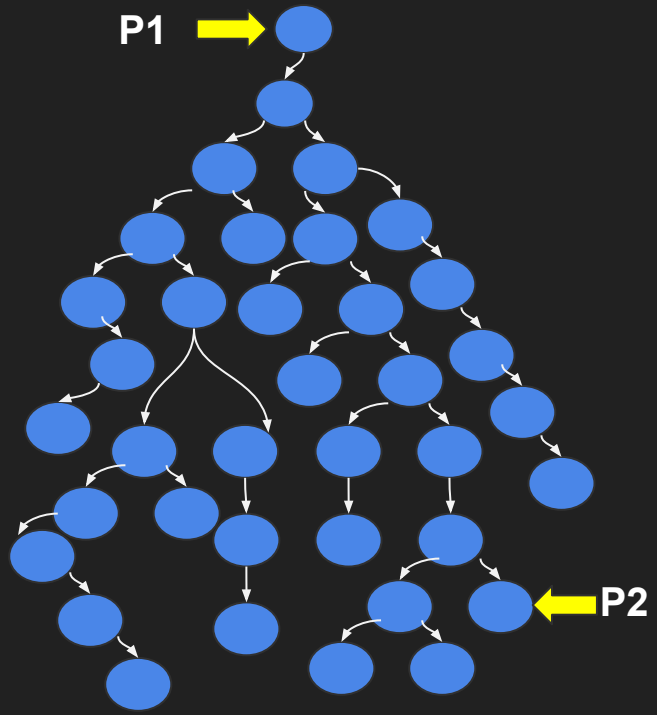


Emulated Program memory
(Uninitialized)

0x0000555555559850	+0x0000	<symbolic_variable_1>
0x0000555555559858	+0x0008	0x000000000ee0000
0x0000555555559860	+0x0010	0x0000000aaabbc34
0x0000555555559868	+0x0018	<symbolic_variable_2>
0x0000555555559870	+0x0020	0x000000000000000
0x0000555555559878	+0x0028	<symbolic_variable_3>
0x0000555555559880	+0x0030	0x000000000000000
0x0000555555559888	+0x0038	<symbolic_variable_4>
0x0000555555559890	+0x0040	<symbolic_variable_5>
0x0000555555559898	+0x0048	<symbolic_variable_6>
0x00005555555598a0	+0x0050	<symbolic_variable_7>
0x00005555555598a8	+0x0058	<symbolic_variable_8>
0x00005555555598b0	+0x0060	0x000000001231284
0x00005555555598b8	+0x0068	0x000000000001212
0x00005555555598c0	+0x0070	<symbolic_variable_9>
0x00005555555598c8	+0x0078	<symbolic_variable_a>
0x00005555555598c8	+0x0078	<symbolic_variable_b>
0x00005555555598c8	+0x0078	<symbolic_variable_c>
0x00005555555598c8	+0x0078	<symbolic_variable_d>
0x00005555555598c8	+0x0078	<symbolic_variable_e>
0x00005555555598c8	+0x0078	<symbolic_variable_f>
0x00005555555598c8	+0x0078	<symbolic_variable_10>
0x00005555555598c8	+0x0078	<symbolic_variable_11>

“under-constrained” symbolic execution

Motivation

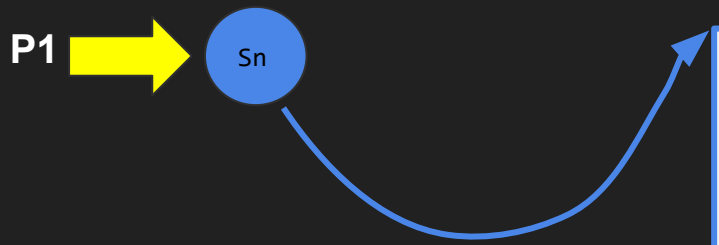


Emulated Program memory
(Uninitialized)

```
0x0000555555559850 | +0x0000 <symbolic_variable_1>
0x0000555555559858 | +0x0008   0x000000000ee0000
0x0000555555559860 | +0x0010   0x0000000aaabbc34
0x0000555555559868 | +0x0018 <symbolic_variable_2>
0x00005555555598c8 | +0x0078 <symbolic_variable_e>
0x00005555555598c8 | +0x0078 <symbolic_variable_f>
0x00005555555598c8 | +0x0078 <symbolic_variable_10>
0x00005555555598c8 | +0x0078 <symbolic_variable_11>
```

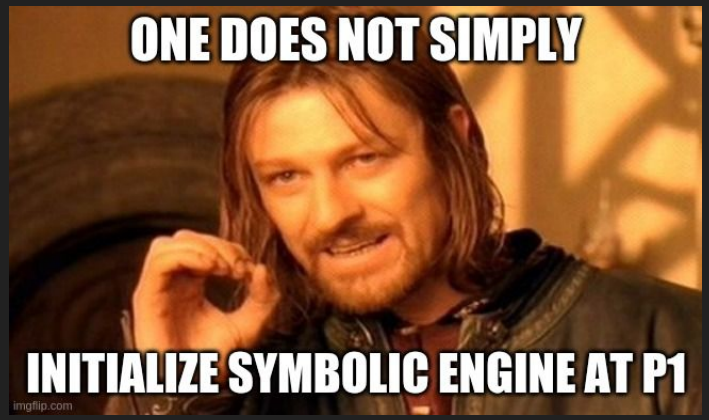
“under-constrained” symbolic execution

Motivation



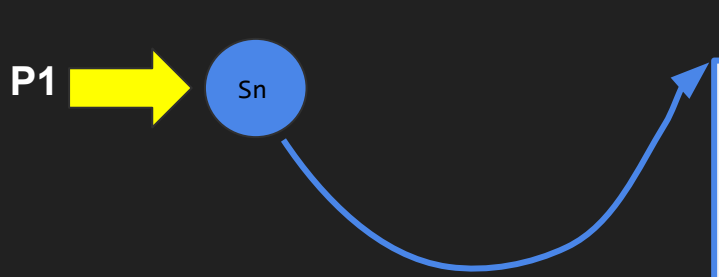
Emulated Program memory
(Uninitialized)

0x0000555555559850	+0x0000	0x0000000000000000
0x0000555555559858	+0x0008	0x0000000000000000
0x0000555555559860	+0x0010	0x0000000000000000
0x0000555555559868	+0x0018	0x0000000000000000
0x0000555555559870	+0x0020	0x0000000000000000
0x0000555555559878	+0x0028	0x0000000000000000
0x0000555555559880	+0x0030	0x0000000000000000
0x0000555555559888	+0x0038	0x0000000000000000
0x0000555555559890	+0x0040	0x0000000000000000
0x0000555555559898	+0x0048	0x0000000000000000
0x00005555555598a0	+0x0050	0x0000000000000000
0x00005555555598a8	+0x0058	0x0000000000000000
0x00005555555598b0	+0x0060	0x0000000000000000
0x00005555555598b8	+0x0068	0x0000000000000000
0x00005555555598c0	+0x0070	0x0000000000000000
0x00005555555598c8	+0x0078	0x0000000000000000



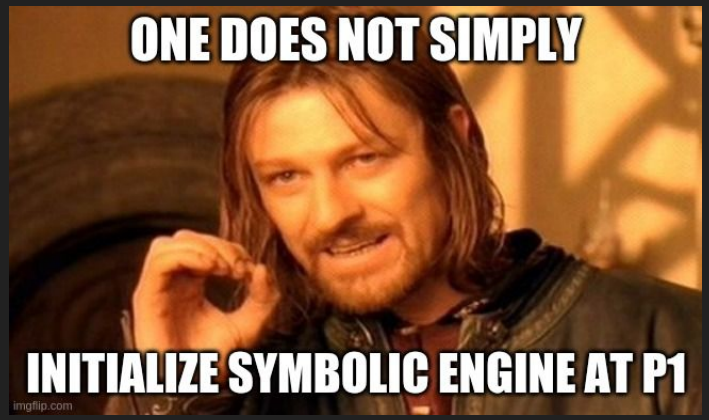
THIS WAS THE CAUSE!

Motivation



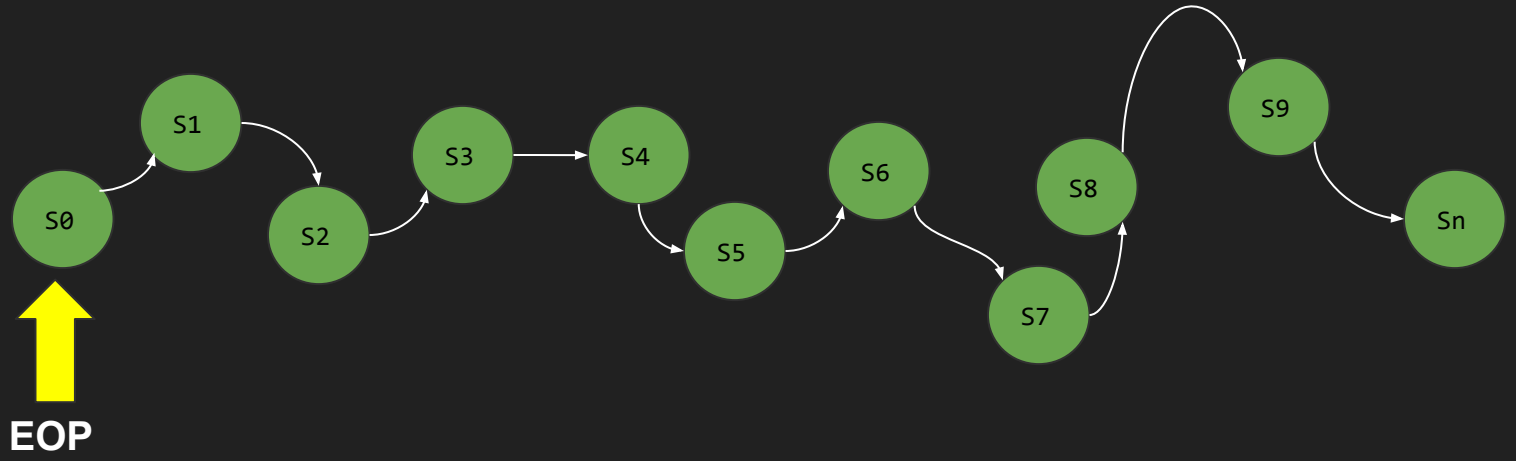
Emulated Program memory

0x0000555555559850	+0x0000	0x89485ed18949ed31
0x0000555555559858	+0x0008	0x4c5450f0e48348e2
0x0000555555559860	+0x0010	0x8d4800010aca058d
0x0000555555559868	+0x0018	0x3d8d4800010a530d
0x0000555555559870	+0x0020	0xa75e15ffffffe61c
0x0000555555559878	+0x0028	0x0000441f0ff40021
0x0000555555559880	+0x0030	0x550021a9e13d8d48
0x0000555555559888	+0x0038	0x480021a9d9058d48
0x0000555555559890	+0x0040	0x481974e58948f839
0x0000555555559898	+0x0048	0x85480021a732058b
0x00005555555598a0	+0x0050	0x2e66e0ff5d0d74c0
0x00005555555598a8	+0x0058	0x0000000000841f0f
0x00005555555598b0	+0x0060	0x2e6600401f0fc35d
0x00005555555598b8	+0x0068	0x0000000000841f0f
0x00005555555598c0	+0x0070	0x480021a9a13d8d48
0x00005555555598c8	+0x0078	0x48550021a99a358d



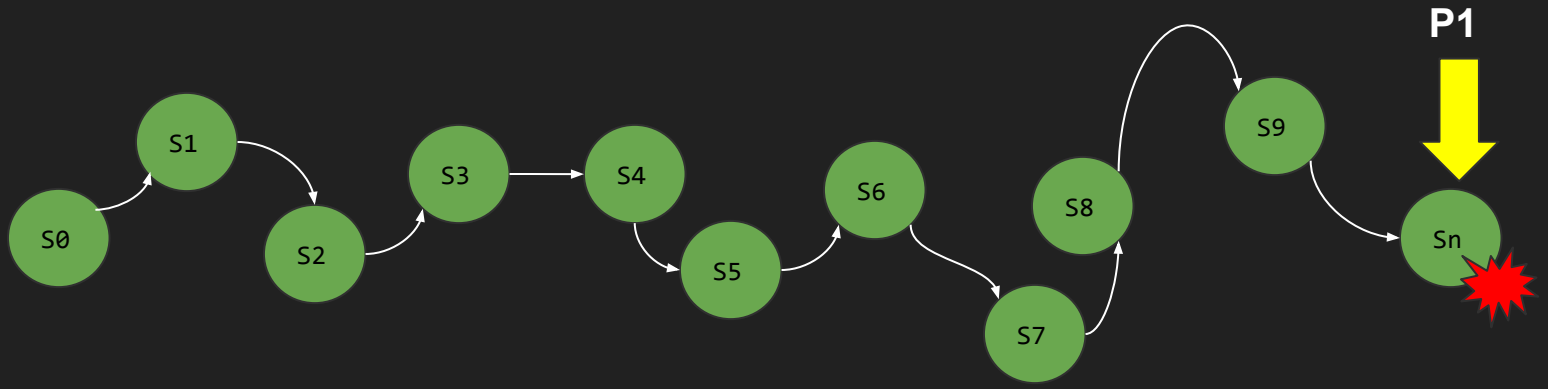
CAN WE HAVE THIS?

Approach



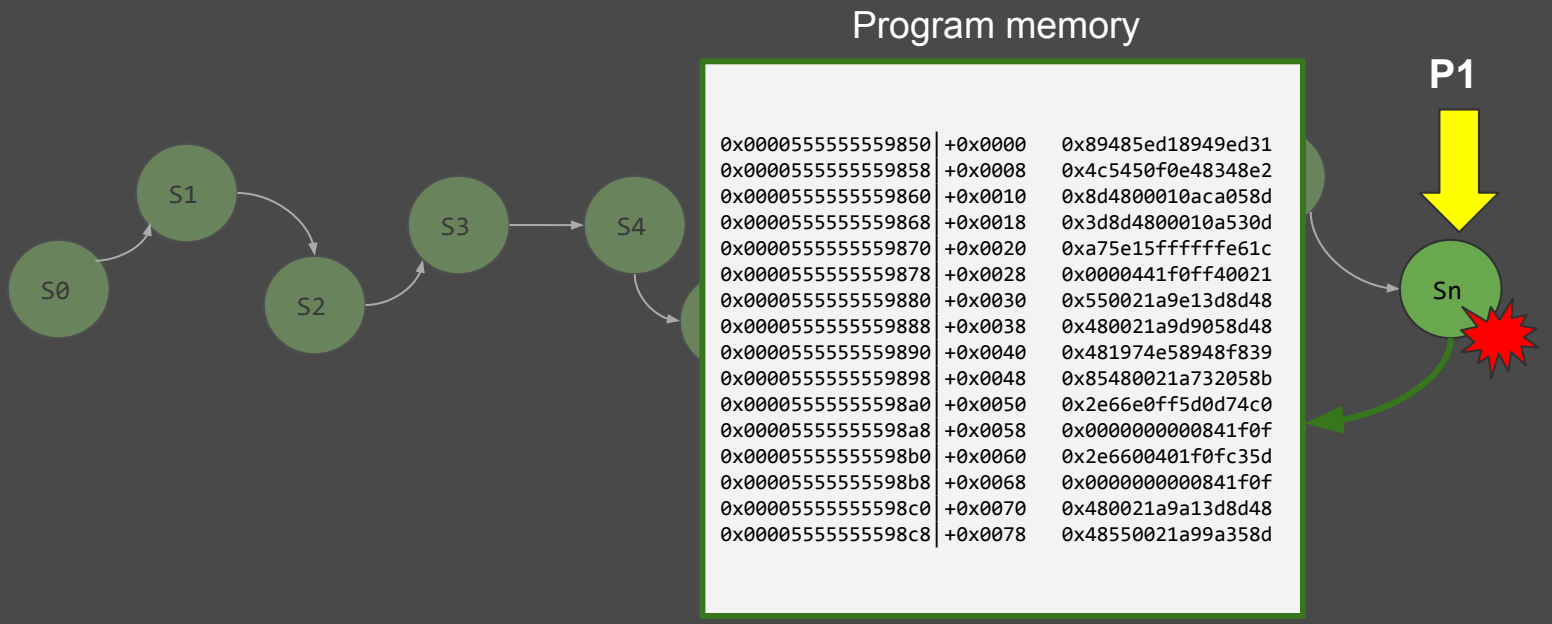
Interleaved symbolic execution

Approach



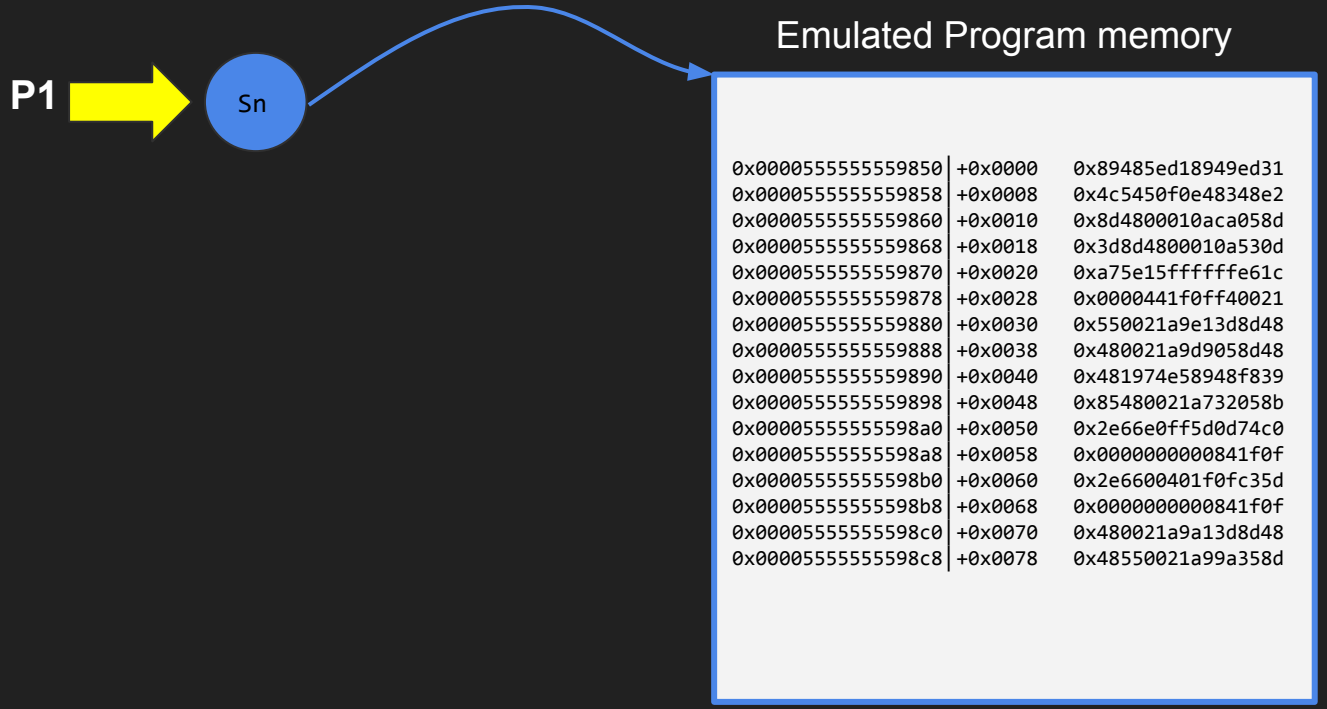
Interleaved symbolic execution

Approach



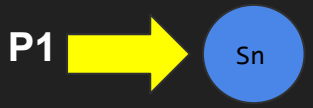
Interleaved symbolic execution

Approach



Interleaved symbolic execution

Approach



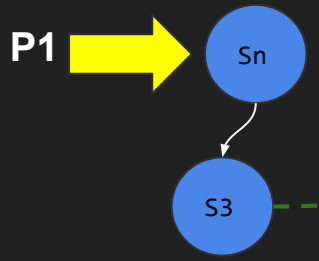
Emulated Program memory

0x0000555555559850	+0x0000	0x89485ed18949ed31
0x0000555555559858	+0x0008	0x4c5450f0e48348e2
0x0000555555559860	+0x0010	0x8d4800010aca058d
0x0000555555559868	+0x0018	<symbolic_variable_1>
0x0000555555559870	+0x0020	0xa75e15ffffffe61c
0x0000555555559878	+0x0028	0x0000441f0ff40021
0x0000555555559880	+0x0030	0x550021a9e13d8d48
0x0000555555559888	+0x0038	0x480021a9d9058d48
0x0000555555559890	+0x0040	0x481974e58948f839
0x0000555555559898	+0x0048	0x85480021a732058b
0x00005555555598a0	+0x0050	0x2e66e0ff5d0d74c0
0x00005555555598a8	+0x0058	0x0000000000841f0f
0x00005555555598b0	+0x0060	0x2e6600401f0fc35d
0x00005555555598b8	+0x0068	0x0000000000841f0f
0x00005555555598c0	+0x0070	0x480021a9a13d8d48
0x00005555555598c8	+0x0078	0x48550021a99a358d

User controlled

Interleaved symbolic execution

Approach

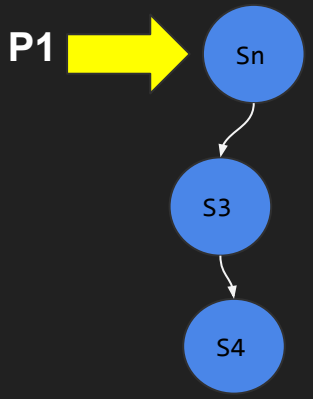


Emulated Program memory

0x0000555555559850	+0x0000	0x89485ed18949ed31
0x0000555555559858	+0x0008	0x4c5450f0e48348e2
0x0000555555559860	+0x0010	0x8d4800010aca058d
0x0000555555559868	+0x0018	<symbolic_variable_1>
0x0000555555559870	+0x0020	0xa75e15ffffffe61c
0x0000555555559878	+0x0028	0x0000441f0ff40021
0x0000555555559880	+0x0030	0x550021a9e13d8d48
0x0000555555559888	+0x0038	0x480021a9d9058d48
0x0000555555559890	+0x0040	0x481974e58948f839
0x0000555555559898	+0x0048	0x85480021a732058b
0x00005555555598a0	+0x0050	0x2e66e0ff5d0d74c0
0x00005555555598a8	+0x0058	0x0000000000841f0f
0x00005555555598b0	+0x0060	0x2e6600401f0fc35d
0x00005555555598b8	+0x0068	0x0000000000841f0f
0x00005555555598c0	+0x0070	0x480021a9a13d8d48
0x00005555555598c8	+0x0078	0x48550021a99a358d

Interleaved symbolic execution

Approach

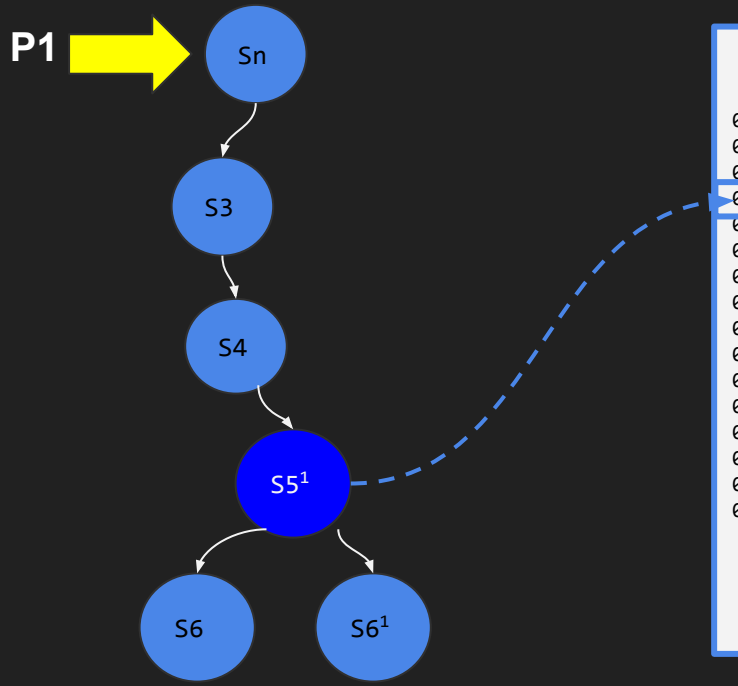


Emulated Program memory

0x0000555555559850	+0x0000	0x89485ed18949ed31
0x0000555555559858	+0x0008	0x4c5450f0e48348e2
0x0000555555559860	+0x0010	0x8d4800010aca058d
0x0000555555559868	+0x0018	<symbolic_variable_1>
0x0000555555559870	+0x0020	0xa75e15ffffffe61c
0x0000555555559878	+0x0028	0x0000441f0ff40021
0x0000555555559880	+0x0030	0x550021a9e13d8d48
0x0000555555559888	+0x0038	0x480021a9d9058d48
0x0000555555559890	+0x0040	0x481974e58948f839
0x0000555555559898	+0x0048	0x85480021a732058b
0x00005555555598a0	+0x0050	0x2e66e0ff5d0d74c0
0x00005555555598a8	+0x0058	0x0000000000841f0f
0x00005555555598b0	+0x0060	0x2e6600401f0fc35d
0x00005555555598b8	+0x0068	0x0000000000841f0f
0x00005555555598c0	+0x0070	0x480021a9a13d8d48
0x00005555555598c8	+0x0078	0x48550021a99a358d

Interleaved symbolic execution

Approach

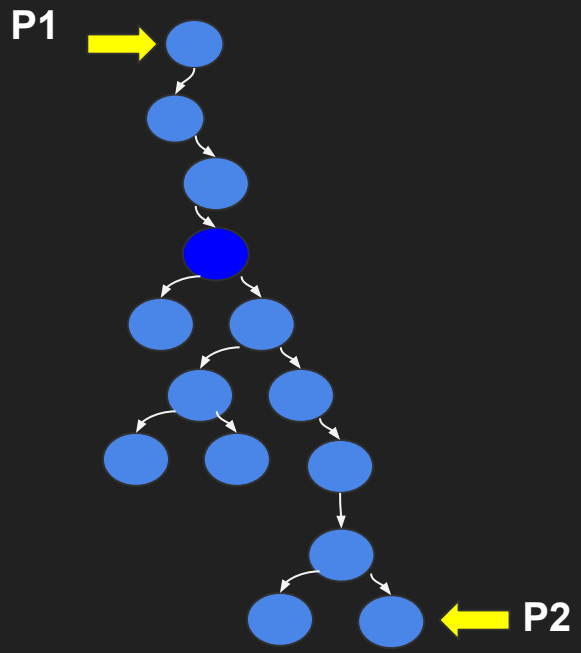


Emulated Program memory

0x0000555555559850	+0x0000	0x89485ed18949ed31
0x0000555555559858	+0x0008	0x4c5450f0e48348e2
0x0000555555559860	+0x0010	0x8d4800010aca058d
0x0000555555559868	+0x0018	<symbolic_variable_1>
0x0000555555559870	+0x0020	0xa75e15ffffffe61c
0x0000555555559878	+0x0028	0x0000441f0ff40021
0x0000555555559880	+0x0030	0x550021a9e13d8d48
0x0000555555559888	+0x0038	0x480021a9d9058d48
0x0000555555559890	+0x0040	0x481974e58948f839
0x0000555555559898	+0x0048	0x85480021a732058b
0x00005555555598a0	+0x0050	0x2e66e0ff5d0d74c0
0x00005555555598a8	+0x0058	0x0000000000841f0f
0x00005555555598b0	+0x0060	0x2e6600401f0fc35d
0x00005555555598b8	+0x0068	0x0000000000841f0f
0x00005555555598c0	+0x0070	0x480021a9a13d8d48
0x00005555555598c8	+0x0078	0x48550021a99a358d

Interleaved symbolic execution

Approach

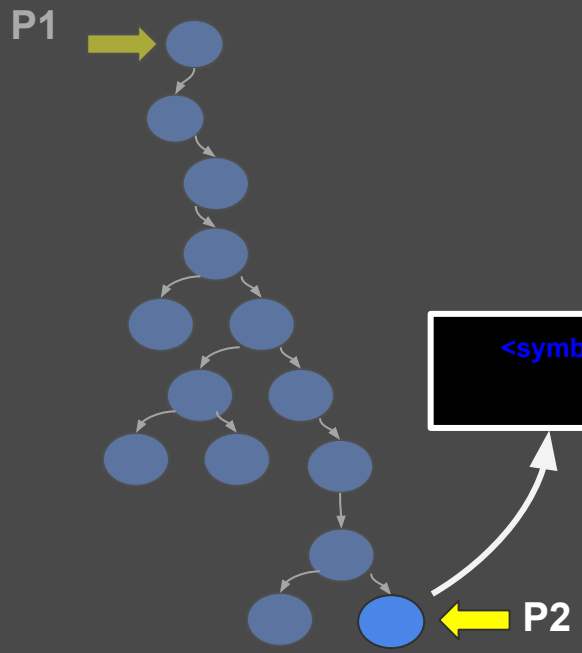


Emulated Program memory

0x0000555555559850	+0x0000	0x89485ed18949ed31
0x0000555555559858	+0x0008	0x4c5450f0e48348e2
0x0000555555559860	+0x0010	0x8d4800010aca058d
0x0000555555559868	+0x0018	<symbolic_variable_1>
0x0000555555559870	+0x0020	0xa75e15ffffffe61c
0x0000555555559878	+0x0028	0x0000441f0ff40021
0x0000555555559880	+0x0030	0x550021a9e13d8d48
0x0000555555559888	+0x0038	0x480021a9d9058d48
0x0000555555559890	+0x0040	0x481974e58948f839
0x0000555555559898	+0x0048	0x85480021a732058b
0x00005555555598a0	+0x0050	<symbolic_variable_2>
0x00005555555598a8	+0x0058	<symbolic_variable_3>
0x00005555555598b0	+0x0060	<symbolic_variable_4>
0x00005555555598b8	+0x0068	0x0000000000841f0f
0x00005555555598c0	+0x0070	0x480021a9a13d8d48
0x00005555555598c8	+0x0078	<symbolic_variable_5>

Interleaved symbolic execution

Approach



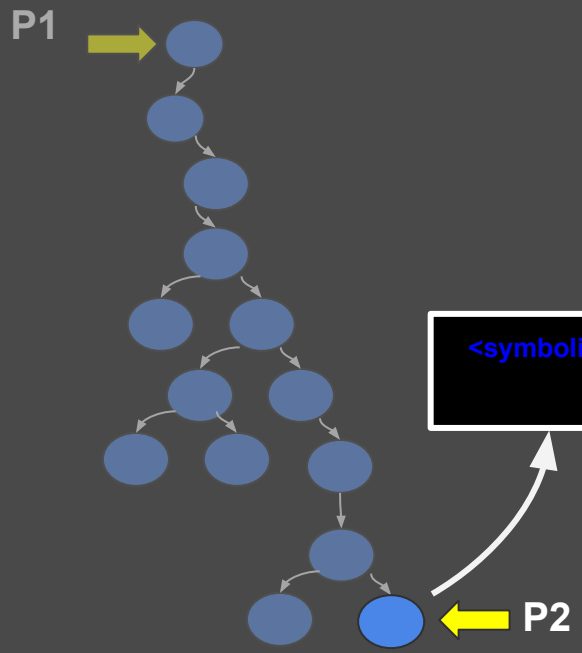
Emulated Program memory

0x0000555555559850	+0x0000	0x89485ed18949ed31
0x0000555555559858	+0x0008	0x4c5450f0e48348e2
0x0000555555559860	+0x0010	0x8d4800010aca058d
0x0000555555559868	+0x0018	<symbolic_variable_1>
0x0000555555559870	+0x0020	0xa75e15ffffffe61c
0x0000555555559878	+0x0028	0x0000441f0ff40021
0x0000555555559880	+0x0030	0x550021a9e13d8d48
	+0x0038	0x480021a9d9058d48
	0040	0x481974e58948f839
	0048	0x85480021a732058b
	0050	<symbolic_variable_2>
	0058	<symbolic_variable_3>
0x00005555555598b0	+0x0060	<symbolic_variable_4>
0x00005555555598b8	+0x0068	0x0000000000841f0f
0x00005555555598c0	+0x0070	0x480021a9a13d8d48
0x00005555555598c8	+0x0078	<symbolic_variable_5>

<symbolic_variable_1> = ?????
To reach P2

Interleaved symbolic execution

Approach



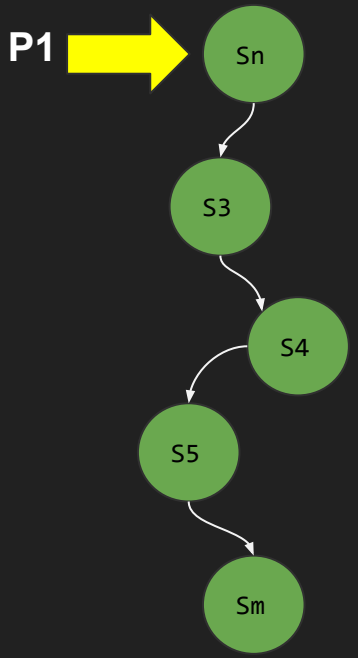
Emulated Program memory

0x0000555555559850	+0x0000	0x89485ed18949ed31
0x0000555555559858	+0x0008	0x4c5450f0e48348e2
0x0000555555559860	+0x0010	0x8d4800010aca058d
0x0000555555559868	+0x0018	<symbolic_variable_1>
0x0000555555559870	+0x0020	0xa75e15ffffffe61c
0x0000555555559878	+0x0028	0x0000441f0ff40021
0x0000555555559880	+0x0030	0x550021a9e13d8d48
	+0x0038	0x480021a9d9058d48
	0040	0x481974e58948f839
	0048	0x85480021a732058b
	0050	<symbolic_variable_2>
	0058	<symbolic_variable_3>
0x00005555555598b0	+0x0060	<symbolic_variable_4>
0x00005555555598b8	+0x0068	0x0000000000841f0f
0x00005555555598c0	+0x0070	0x480021a9a13d8d48
0x00005555555598c8	+0x0078	<symbolic_variable_5>

<symbolic_variable_1> = 0xdeadbeef
To reach P2

Interleaved symbolic execution

Approach

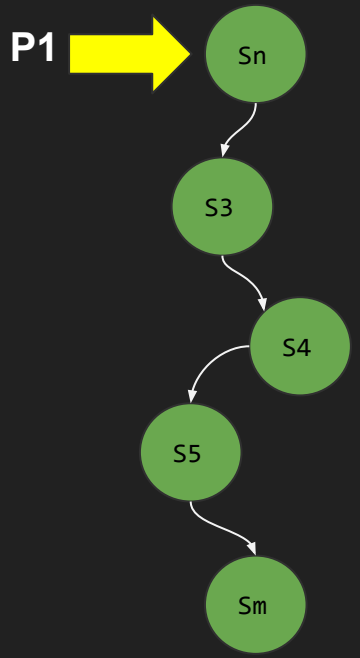


Program memory

```
0x0000555555559850 | +0x0000   0x89485ed18949ed31
0x0000555555559858 | +0x0008   0x4c5450f0e48348e2
0x0000555555559860 | +0x0010   0x8d4800010aca058d
0x0000555555559868 | +0x0018   0x8d4800010aca058d
0x0000555555559870 | +0x0020   0xa75e15ffffffe61c
0x0000555555559878 | +0x0028   0x0000441f0ff40021
0x0000555555559880 | +0x0030   0x550021a9e13d8d48
0x0000555555559888 | +0x0038   0x480021a9d9058d48
0x0000555555559890 | +0x0040   0x481974e58948f839
0x0000555555559898 | +0x0048   0x85480021a732058b
0x00005555555598a0 | +0x0050   0x2e66e0ff5d0d74c0
0x00005555555598a8 | +0x0058   0x0000000000841f0f
0x00005555555598b0 | +0x0060   0x2e6600401f0fc35d
0x00005555555598b8 | +0x0068   0x0000000000841f0f
0x00005555555598c0 | +0x0070   0x480021a9a13d8d48
0x00005555555598c8 | +0x0078   0x48550021a99a358d
```

Interleaved symbolic execution

Approach

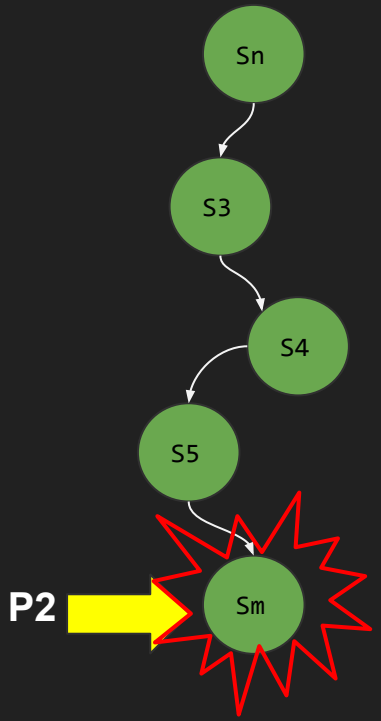


Program memory

```
0x0000555555559850 | +0x0000   0x89485ed18949ed31
0x0000555555559858 | +0x0008   0x4c5450f0e48348e2
0x0000555555559860 | +0x0010   0x8d4800010aca058d
0x0000555555559868 | +0x0018   0x00000000deadbeef
0x0000555555559870 | +0x0020   0xa75e15ffffffe61c
0x0000555555559878 | +0x0028   0x0000441f0ff40021
0x0000555555559880 | +0x0030   0x550021a9e13d8d48
0x0000555555559888 | +0x0038   0x480021a9d9058d48
0x0000555555559890 | +0x0040   0x481974e58948f839
0x0000555555559898 | +0x0048   0x85480021a732058b
0x00005555555598a0 | +0x0050   0x2e66e0ff5d0d74c0
0x00005555555598a8 | +0x0058   0x0000000000841f0f
0x00005555555598b0 | +0x0060   0x2e6600401f0fc35d
0x00005555555598b8 | +0x0068   0x0000000000841f0f
0x00005555555598c0 | +0x0070   0x480021a9a13d8d48
0x00005555555598c8 | +0x0078   0x48550021a99a358d
```

Interleaved symbolic execution

Approach



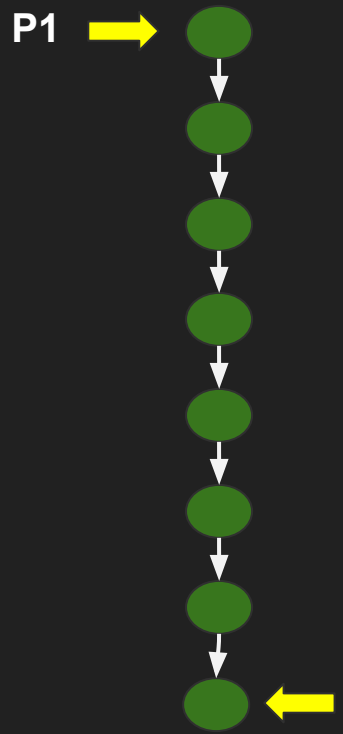
Program memory

```
0x0000555555559850 | +0x0000   0x0000000111111111
0x0000555555559858 | +0x0008   0x4c5450f0e48348e2
0x0000555555559860 | +0x0010   0x8d4800010aca058d
0x0000555555559868 | +0x0018   0x00000000deadbeef
0x0000555555559870 | +0x0020   0xa75e15ffffffe61c
0x0000555555559878 | +0x0028   0x0000441f0ff40021
0x0000555555559880 | +0x0030   0x1123012312310010
0x0000555555559888 | +0x0038   0x480021a9d9058d48
0x0000555555559890 | +0x0040   0x4141414141414141
0x0000555555559898 | +0x0048   0x85480021a732058b
0x00005555555598a0 | +0x0050   0x2e66e0ff5d0d74c0
0x00005555555598a8 | +0x0058   0x0000000000841f0f
0x00005555555598b0 | +0x0060   0x2e6600401f0fc35d
0x00005555555598b8 | +0x0068   0x0000000000841f0f
0x00005555555598c0 | +0x0070   0x0000100100000000
0x00005555555598c8 | +0x0078   0x48550021a99a358d
```

Interleaved symbolic execution

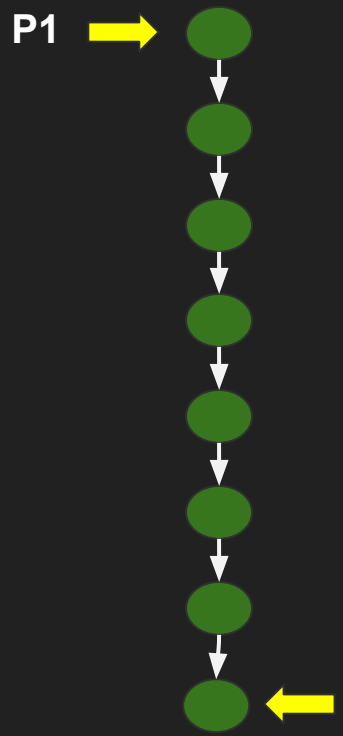
Approach

Approach

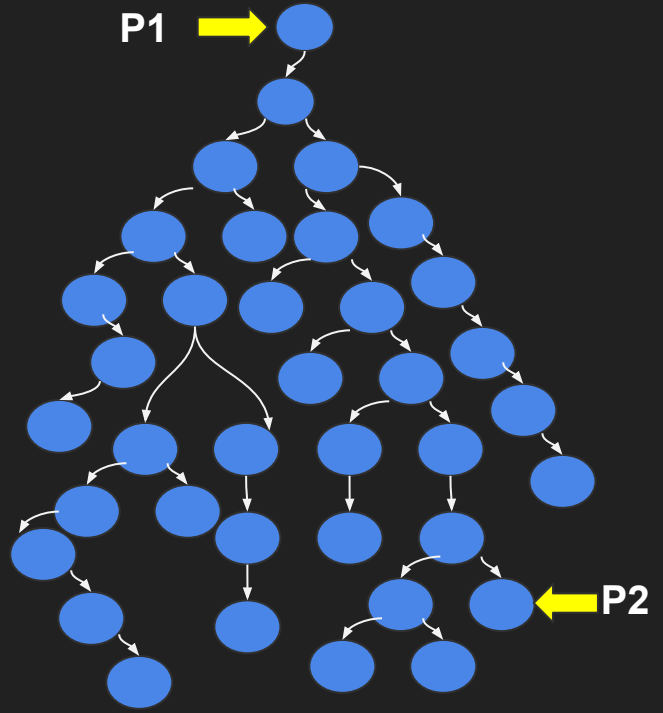


concrete execution

Approach

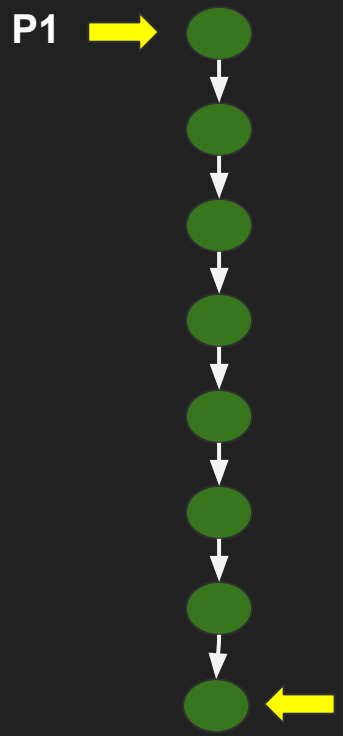


concrete execution

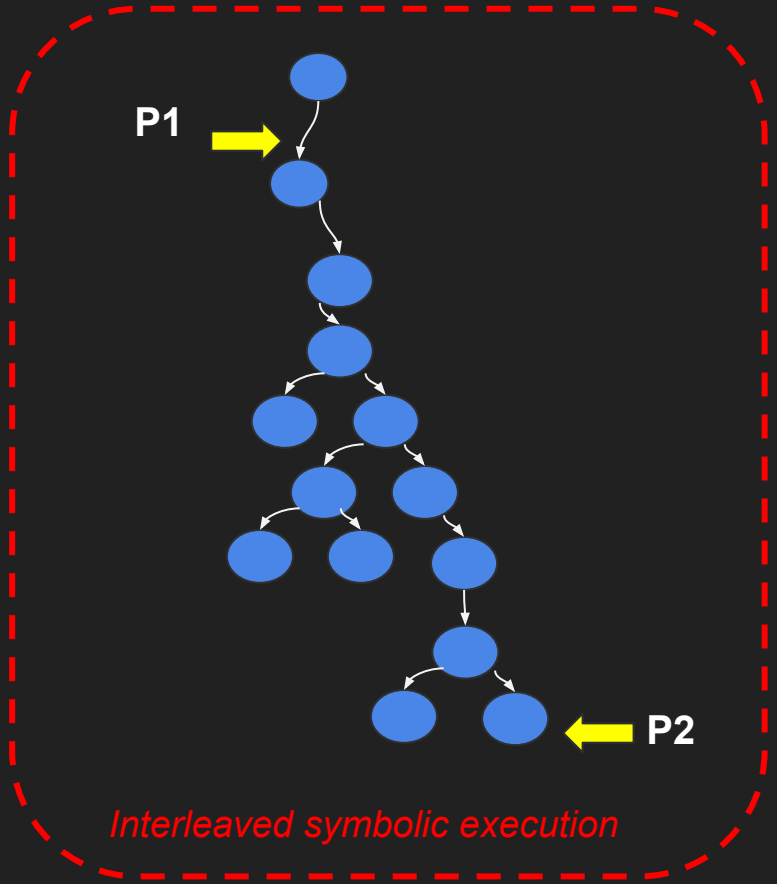


under-constrained symbolic exec.

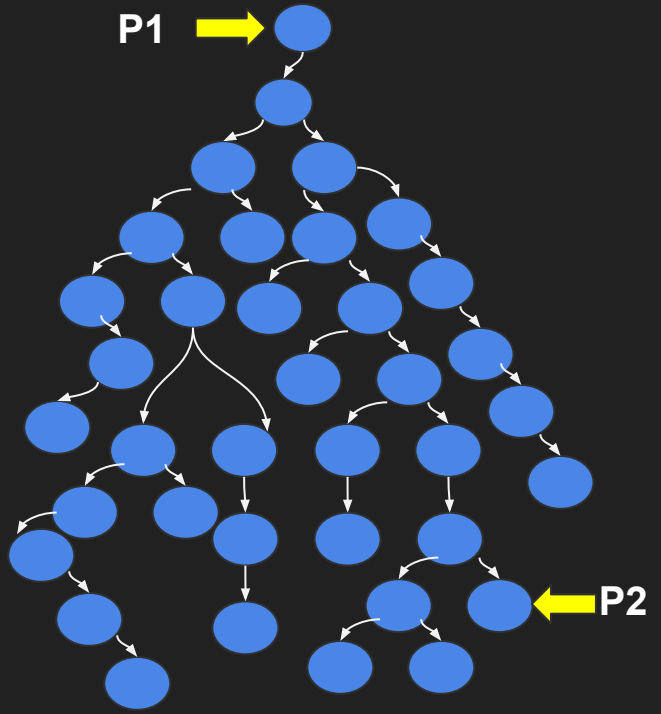
Approach



concrete execution

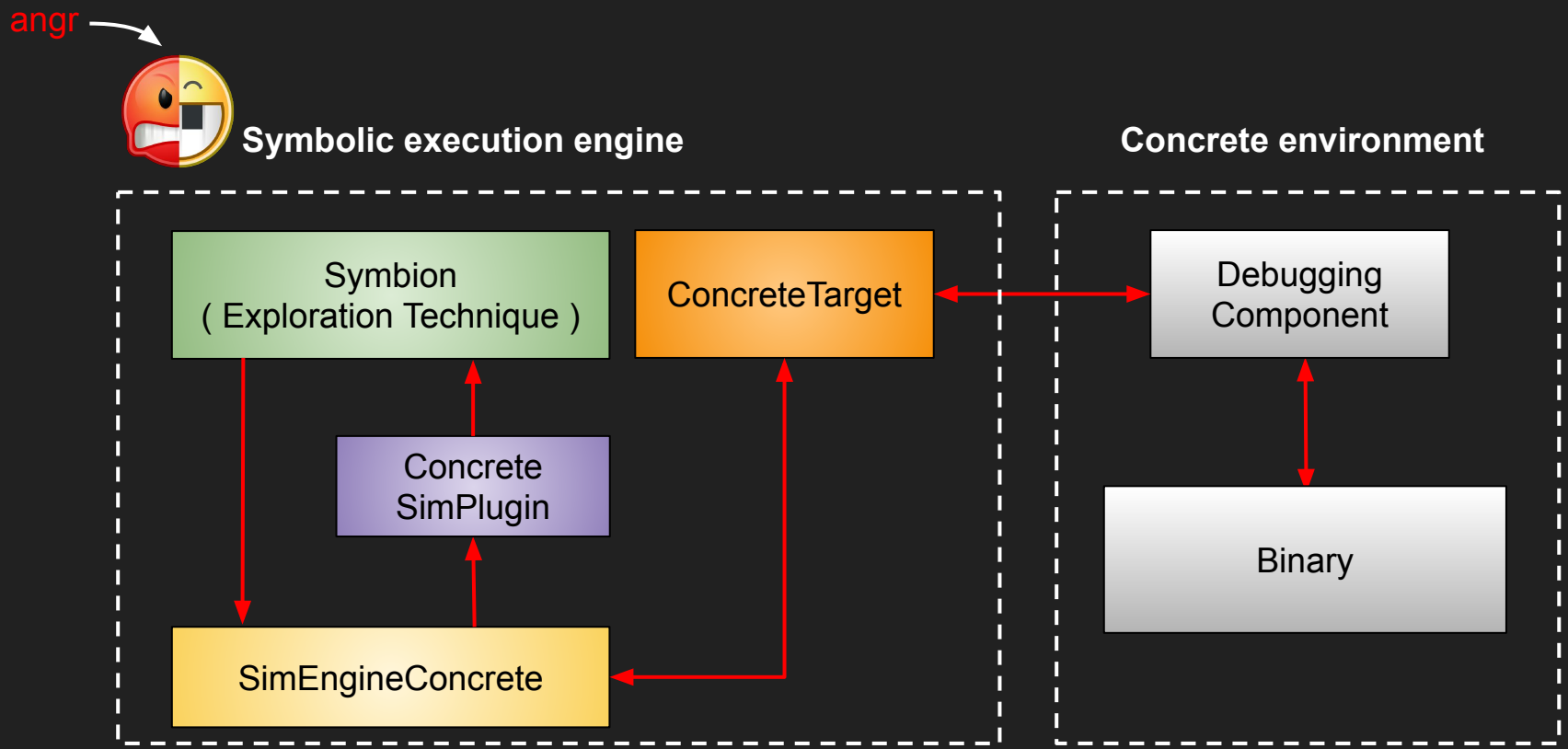


Interleaved symbolic execution



under-constrained symbolic exec.

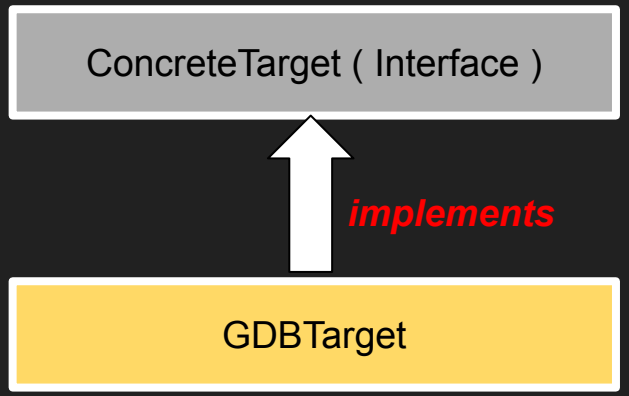
System Overview



ConcreteTarget

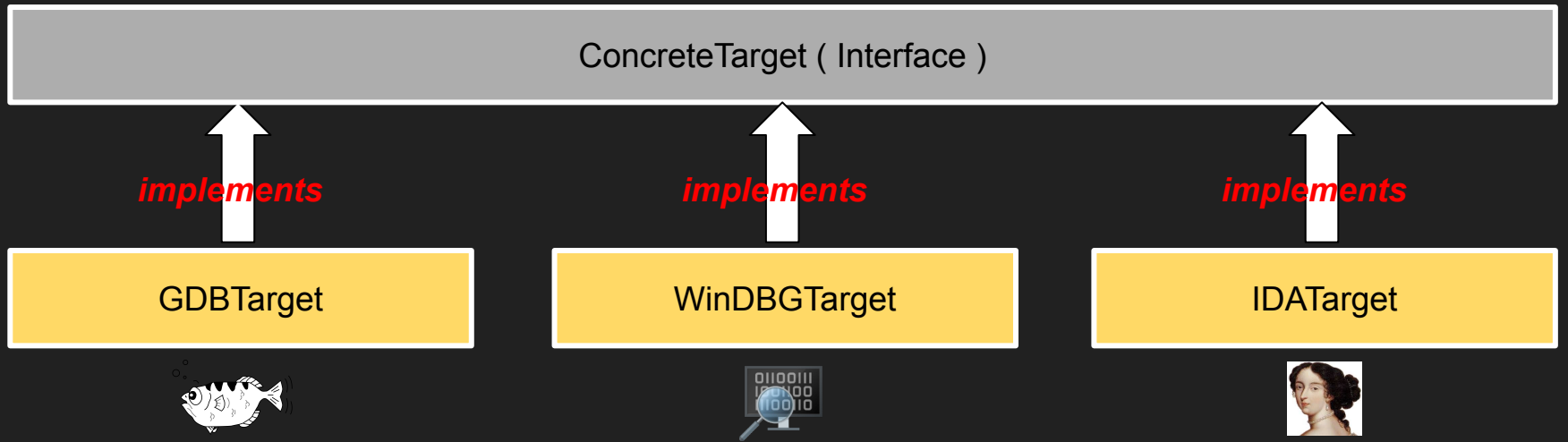
- Interface used to implement objects that will control the program executed inside the concrete analysis environment.

- Exposes the following methods:
  - `def read_memory(self, address, length)`
  - `def write_memory(self, address, data)`
  - `def read_register(self, register)`
  - `def write_register(self, register, value)`
  - `def set_breakpoint(self, address)`
  - `def remove_breakpoint(self, address)`
  - `def set_watchpoint(self, address)`
  - `def remove_watchpoint(self, address)`
  - `def get_mappings(self)`
  - `def run(self)`

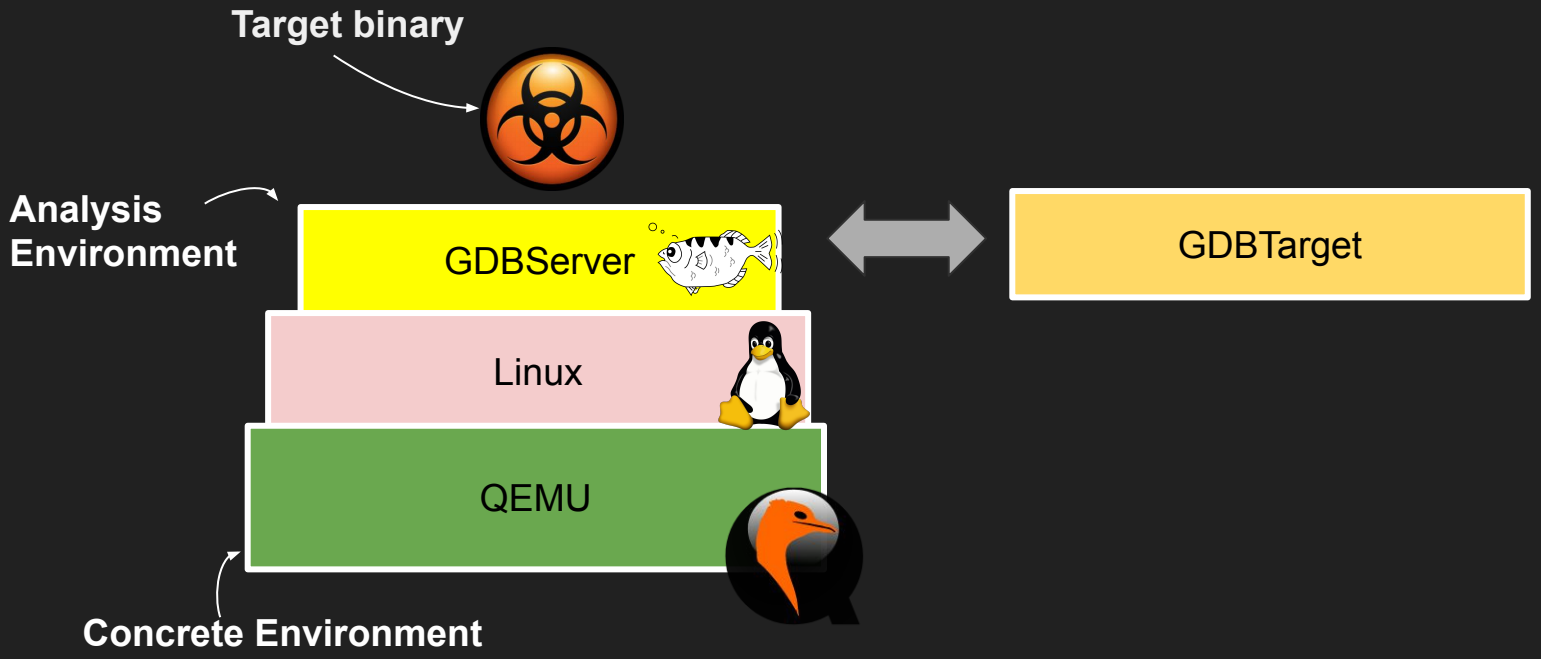


ConcreteTarget

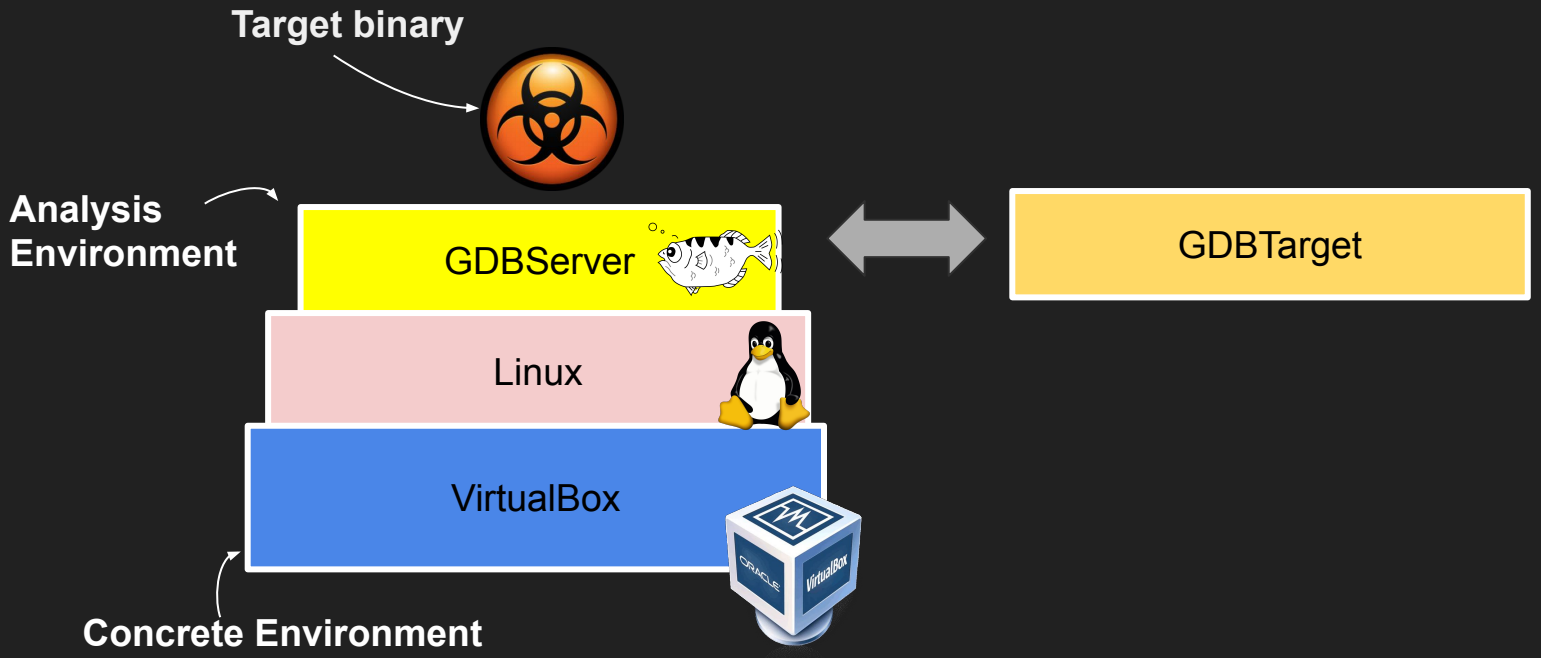
- It can have different interesting implementations!



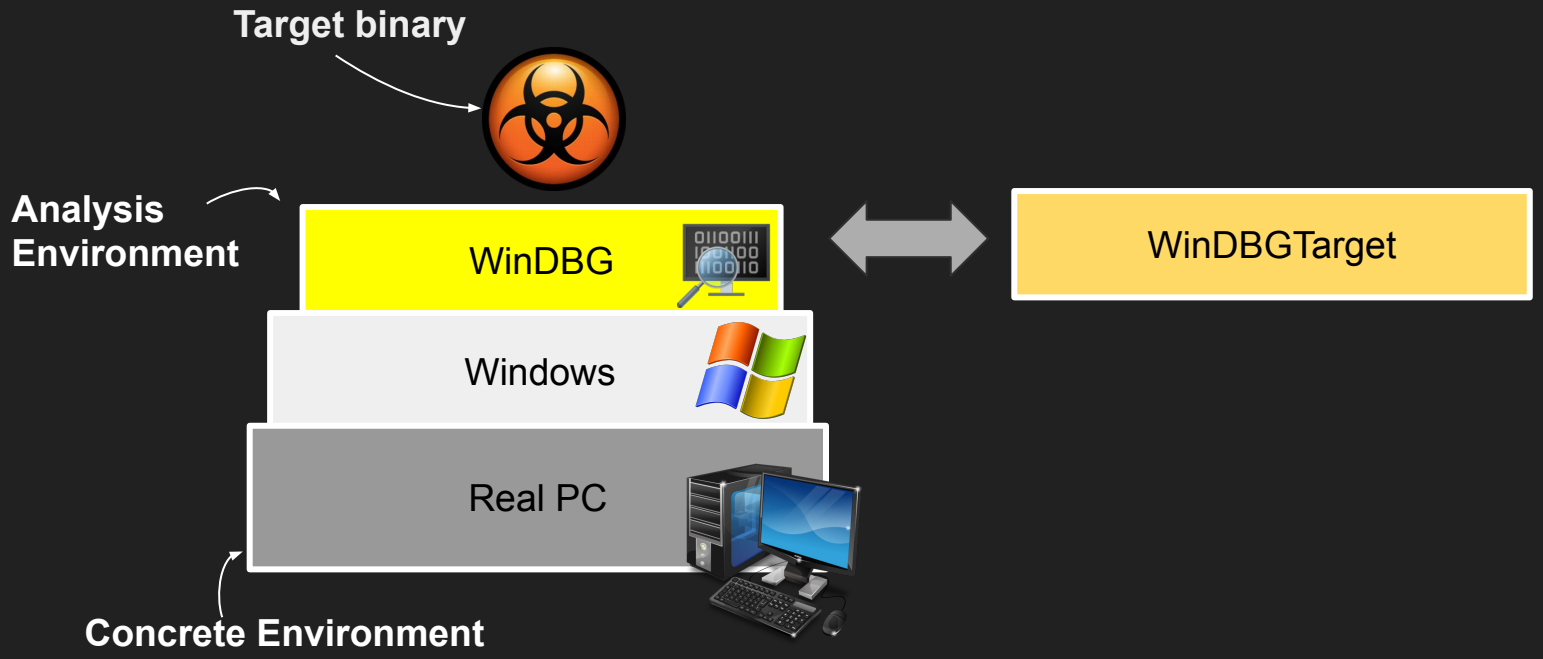
ConcreteTarget



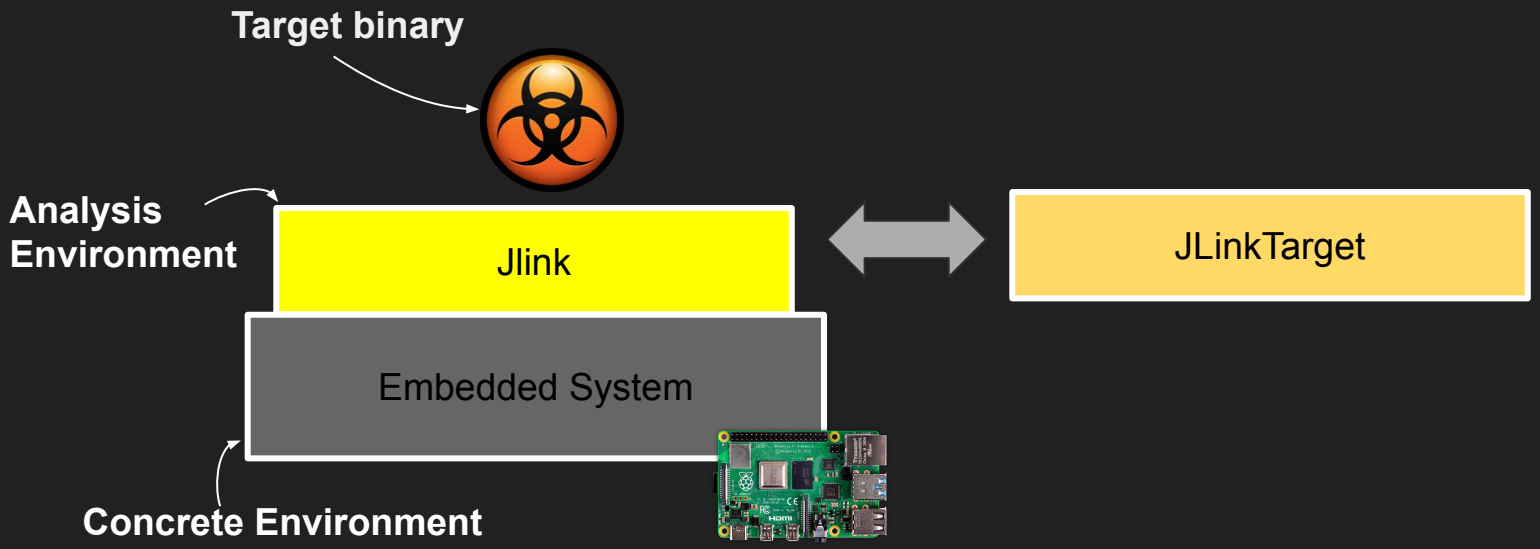
ConcreteTarget



ConcreteTarget



ConcreteTarget

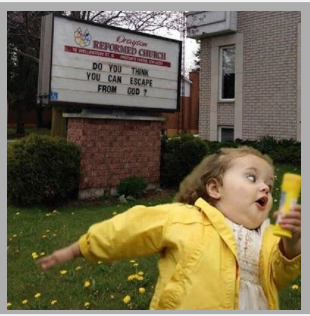


Let’s put all the pieces together

Use Cases
(malware reverse engineering)

Detect DGA

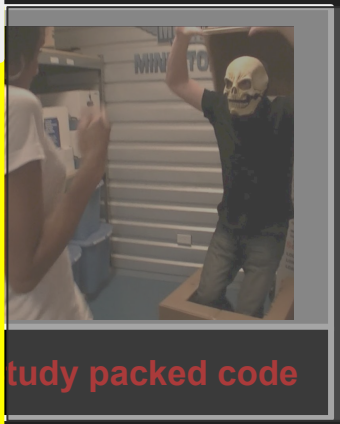
Study packed code

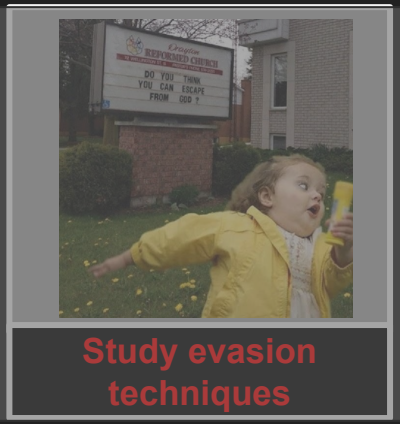


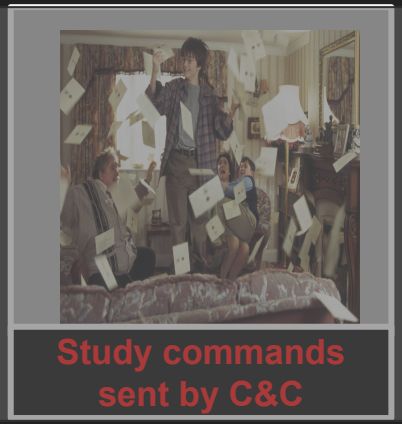
Study evasion techniques

Study commands sent by C&C

Use Cases
(malware reverse engineering)

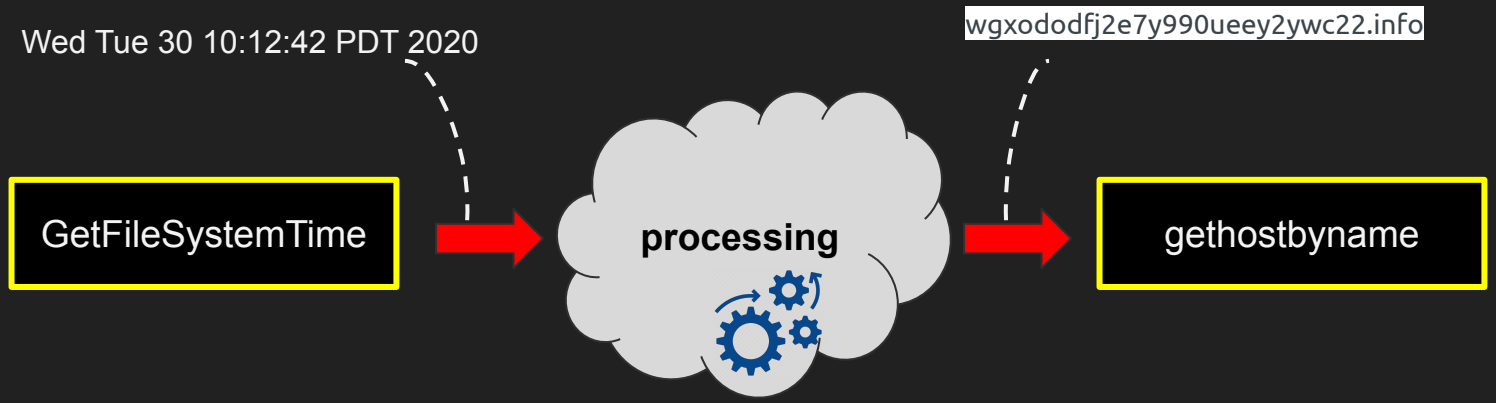






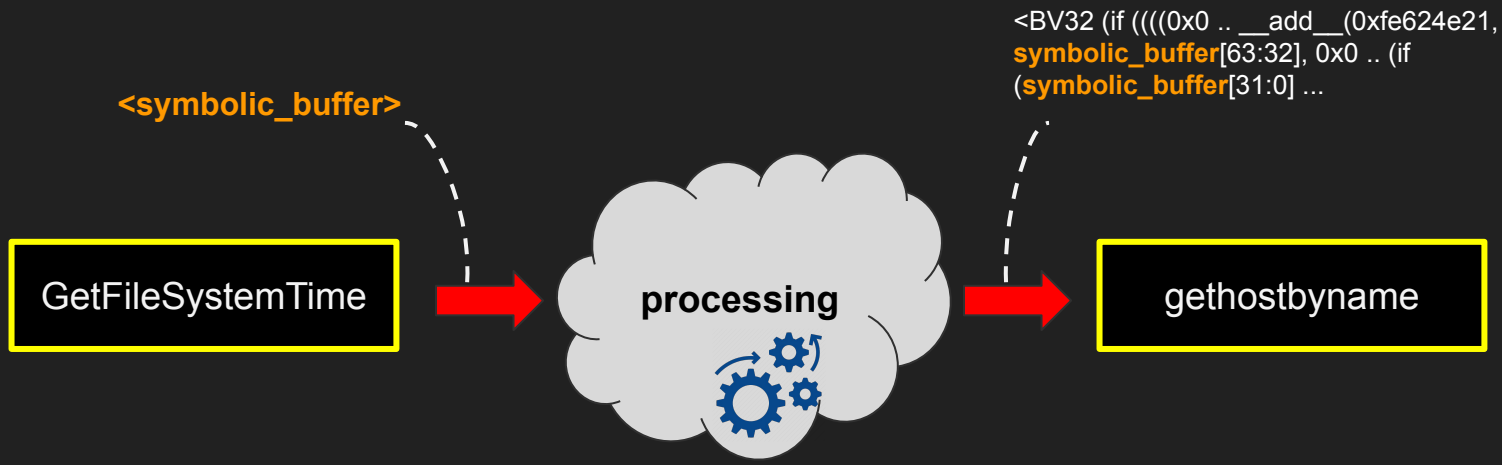
Use Case

- Symmi Trojan
  - Detecting a domain generation algorithm (DGA) inside the binary.



Use Case

- Symmi Trojan
  - Detecting a domain generation algorithm (DGA) inside the binary.



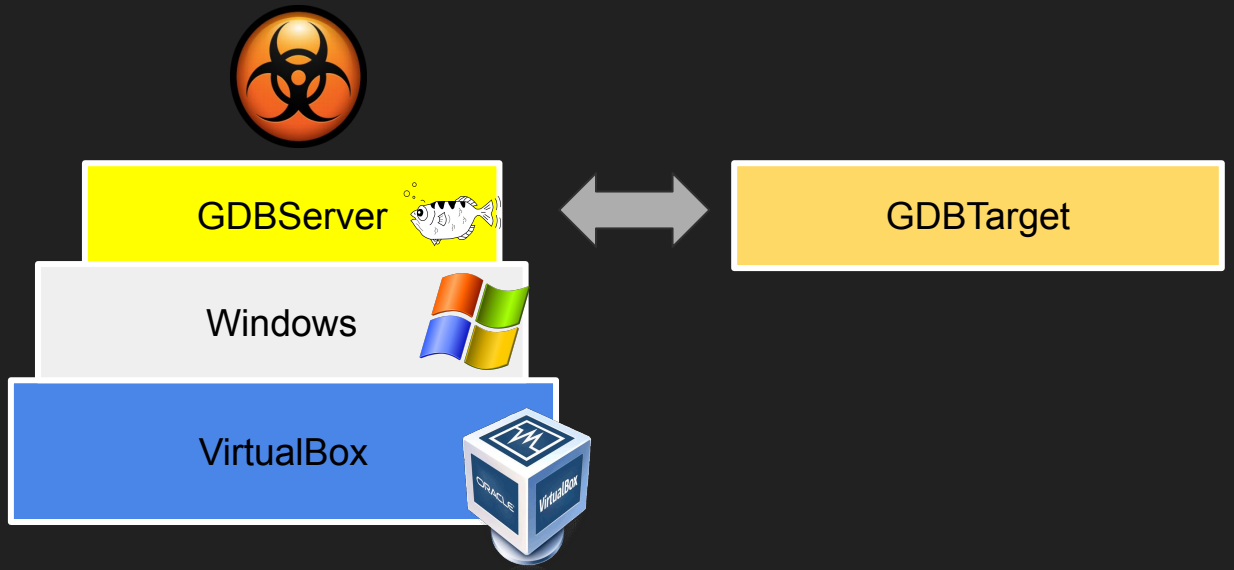
Use Case

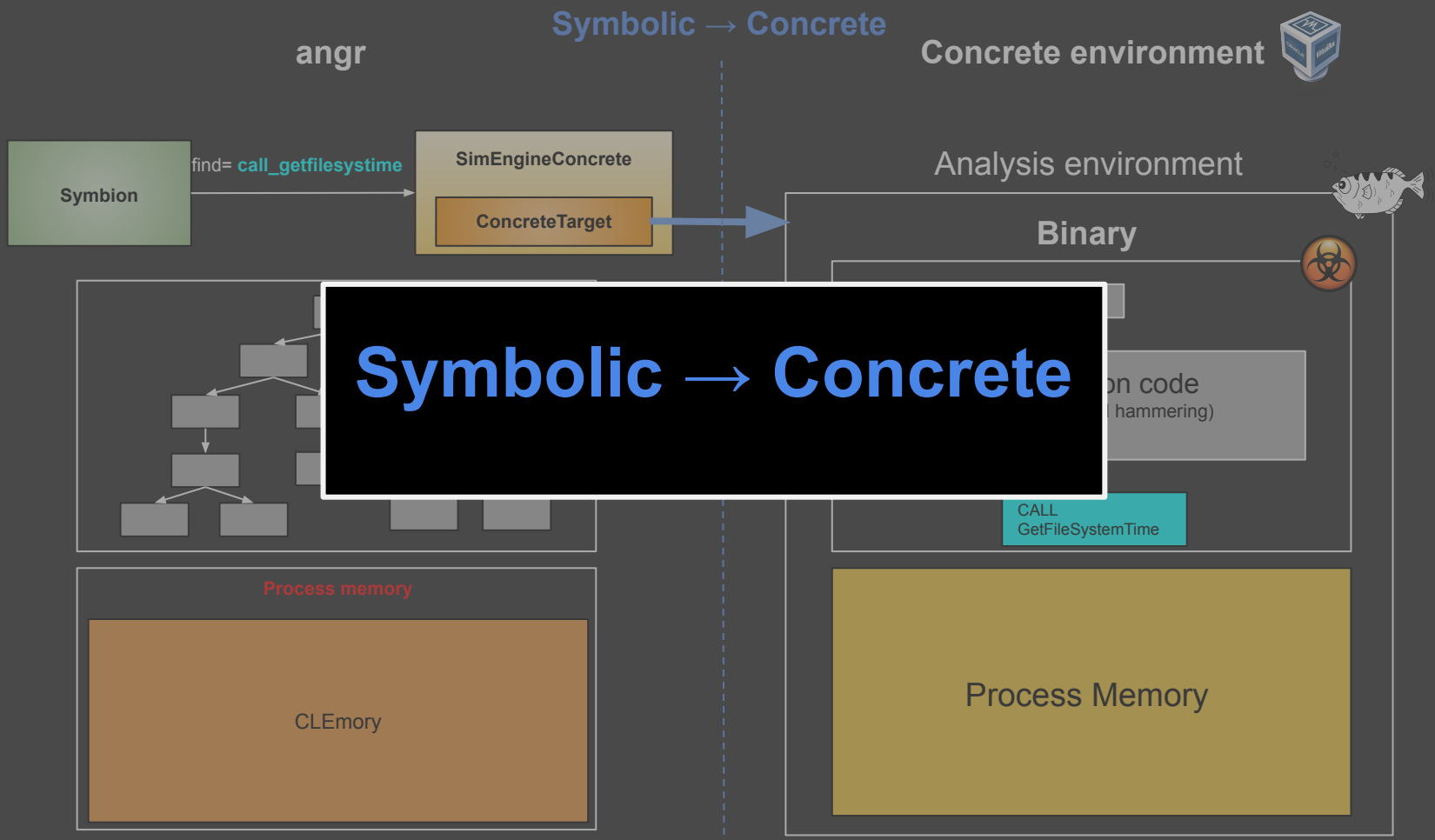
- Symmi Trojan
  - Detecting a domain generation algorithm (DGA) inside the binary.

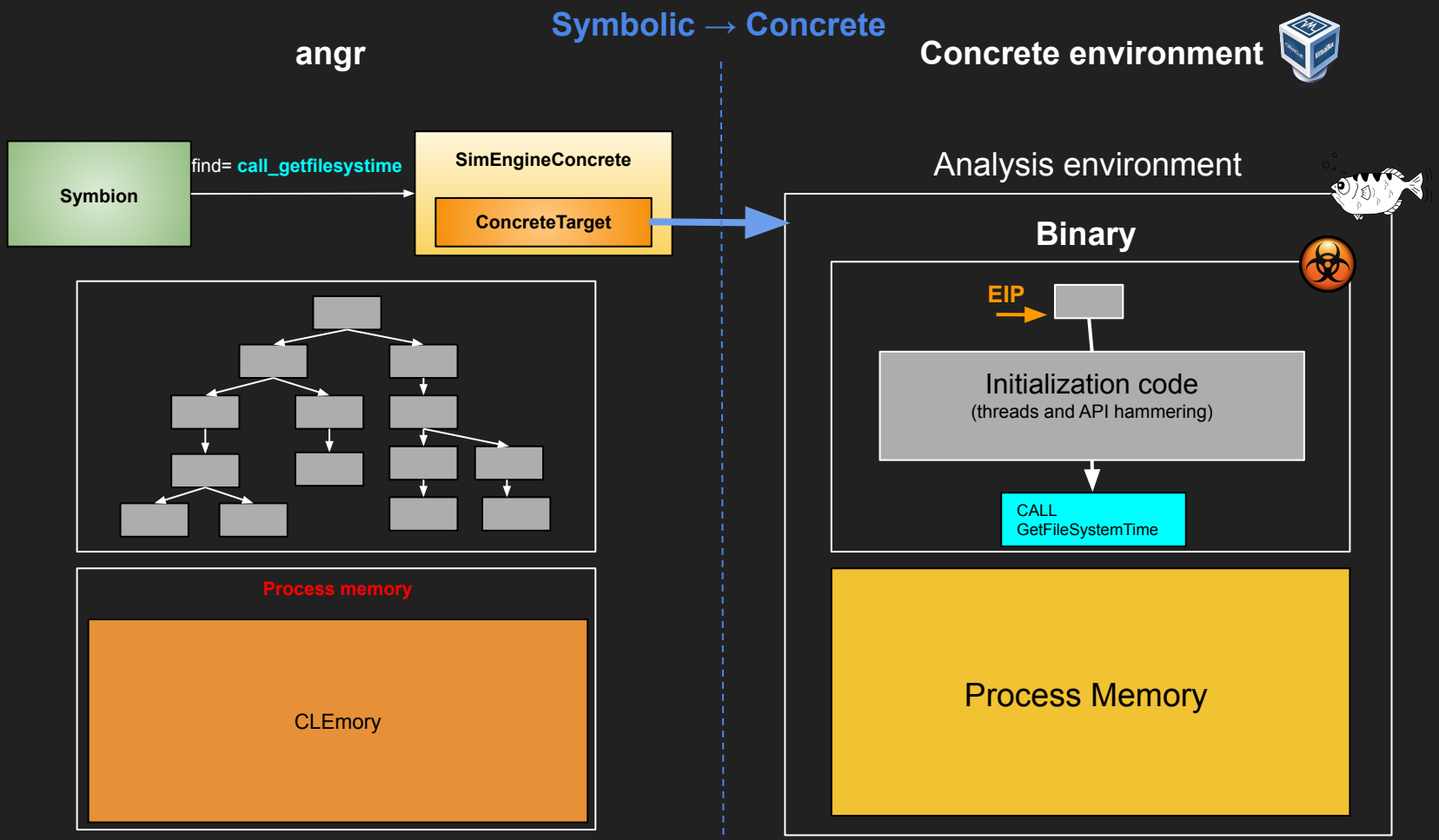
  - **Challenges**:
    - Malware has noisy initialization code and evasion:
      - “API Hammering”
      - Junk code
      - Self-checks
    - Vanilla symbolic execution or under-constrained symbolic execution won’t work.

Use Case

- Symmi Trojan
  - Detecting a domain generation algorithm (DGA) inside the binary.




Symbolic → Concrete
angr
Concrete environment
Symbion
find= call_getfilesystime
SimEngineConcrete
ConcreteTarget
Analysis environment
Binary
Symbolic → Concrete
on code
hammering)
CALL
GetFileSystemTime
Process memory
CLEmory
Process Memory


Symbolic → Concrete
angr
Concrete environment
Symbion
find= call_getfilesystime
SimEngineConcrete
ConcreteTarget
Process memory
CLEmory
Analysis environment
Binary
EIP
Initialization code
(threads and API hammering)
CALL
GetFileSystemTime
Process Memory

Symbolic → Concrete

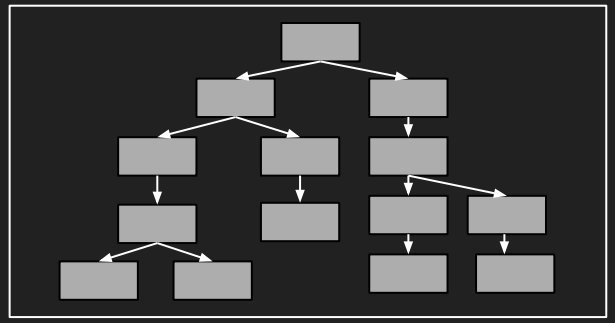
angr

Symbion

find= call_getfilesystime

SimEngineConcrete

ConcreteTarget



Process memory

CLEmory

Concrete environment

Analysis environment

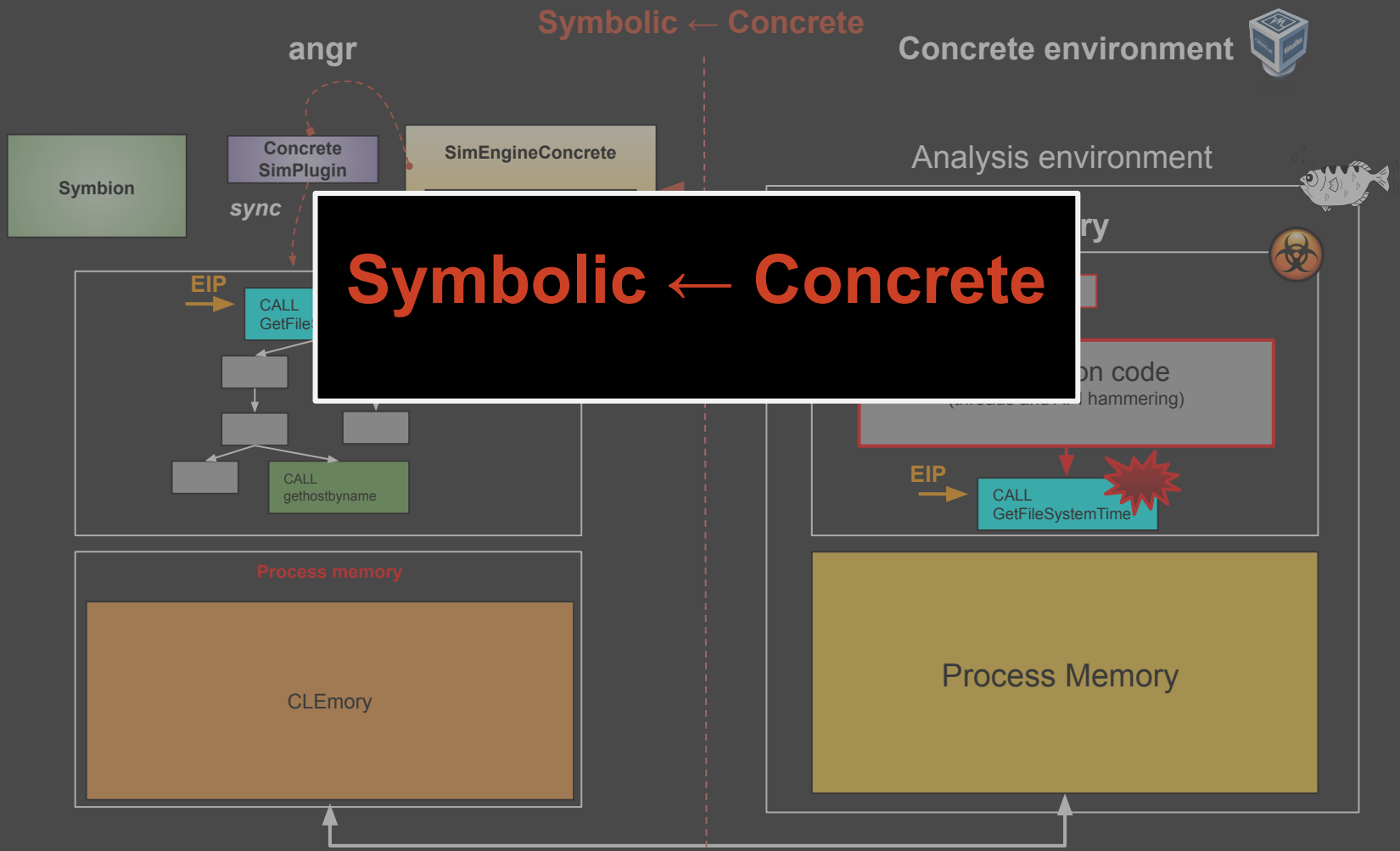
Binary

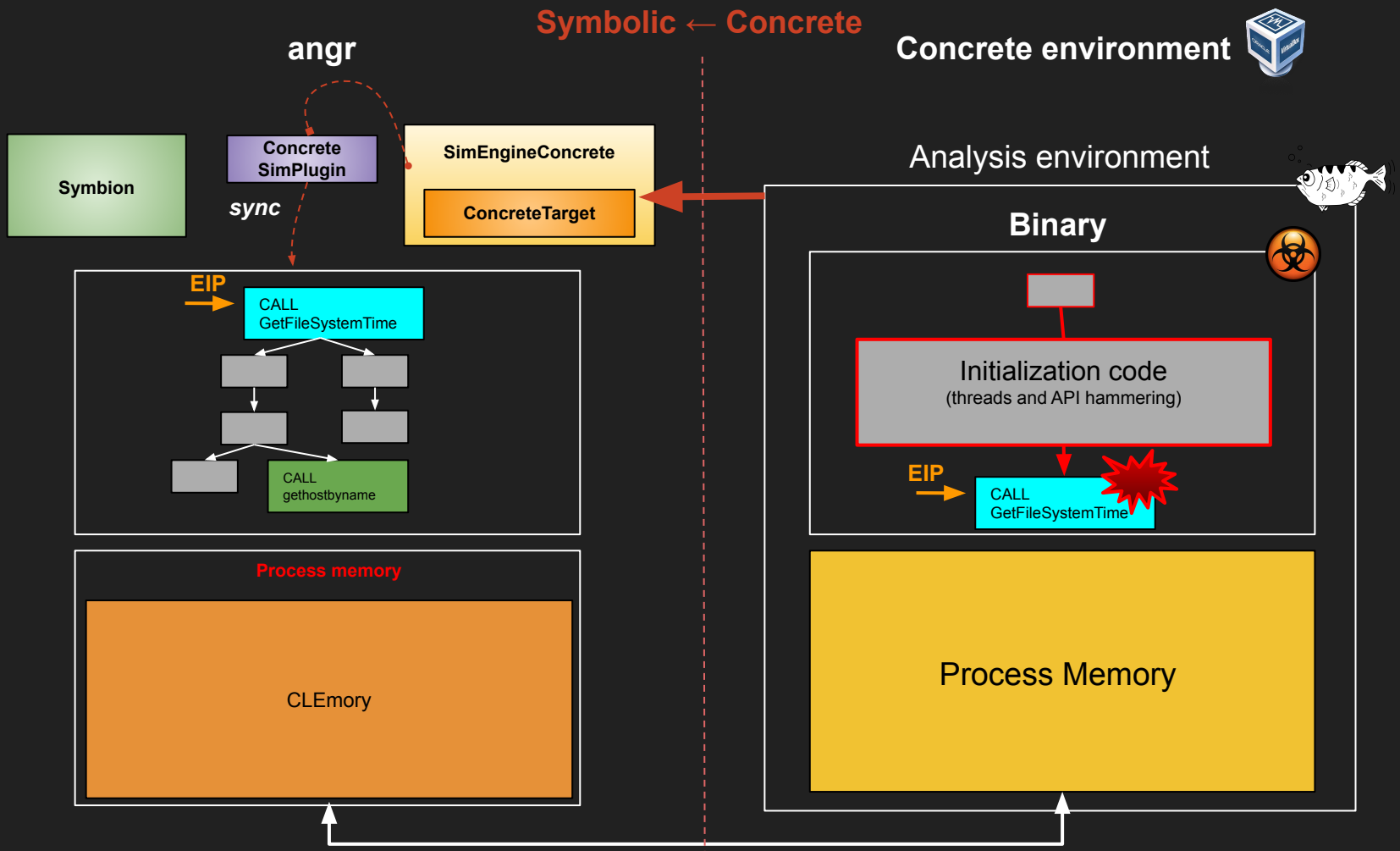
Initialization code
(threads and API hammering)

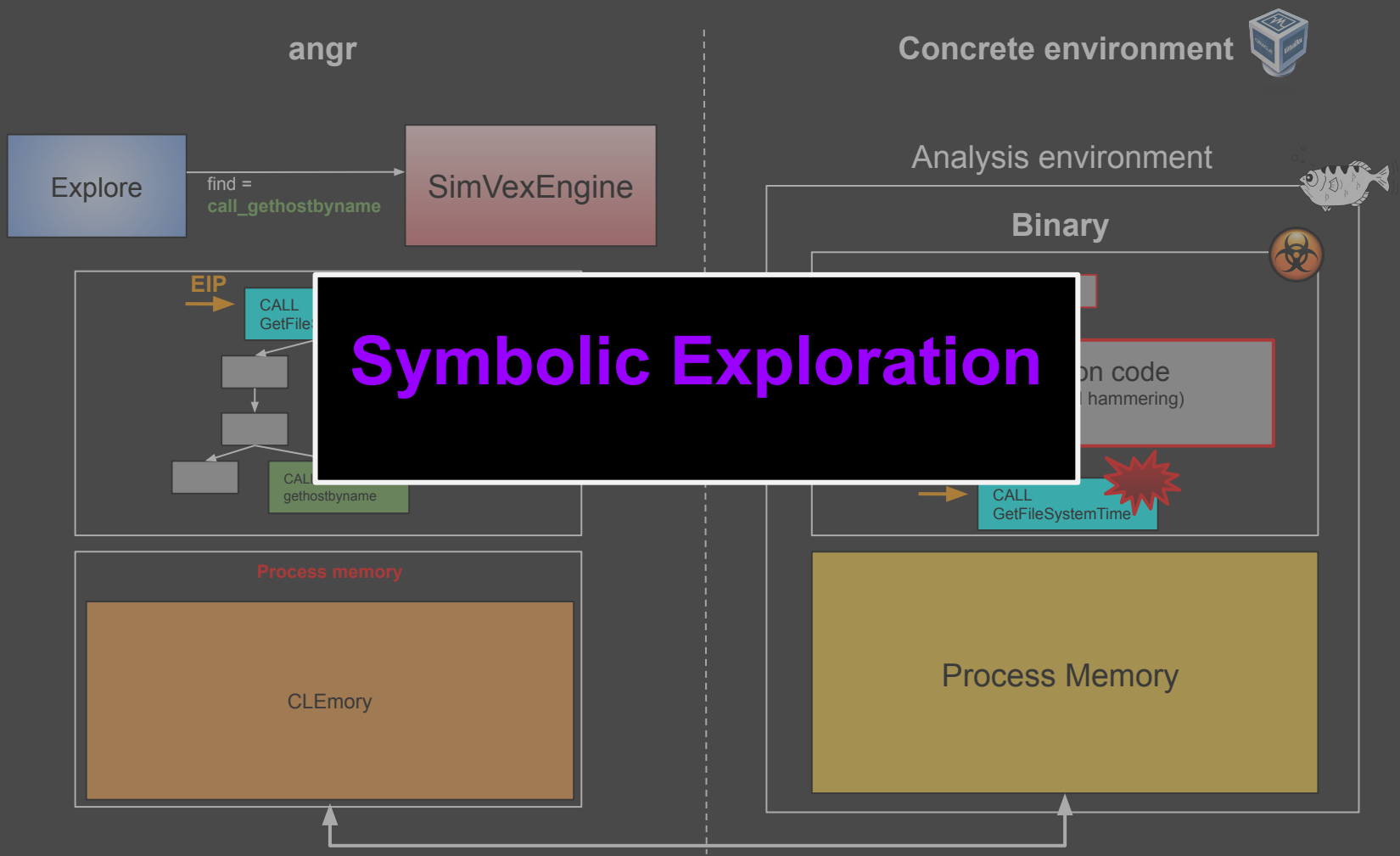
EIP

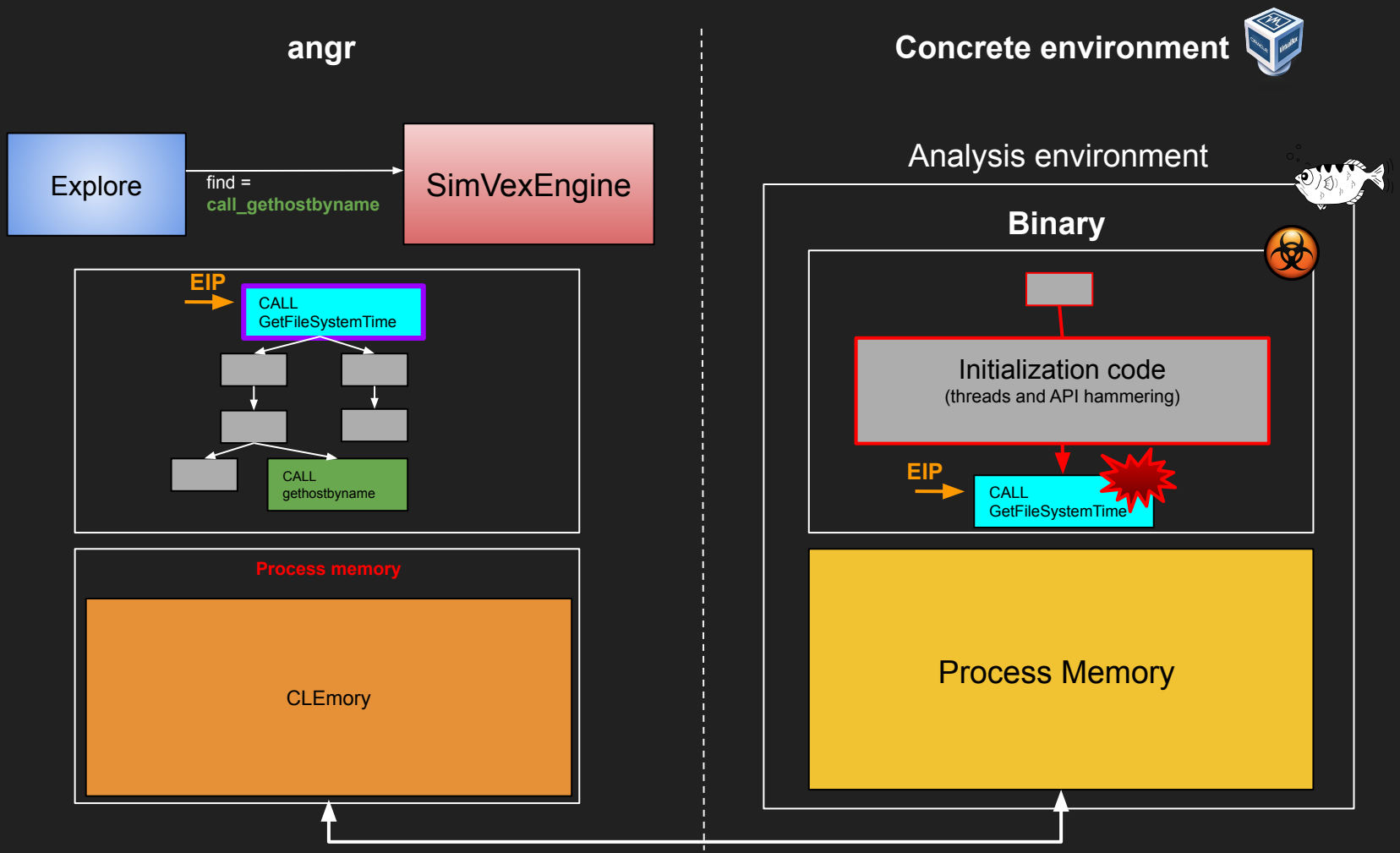
CALL
GetFileSystemTime

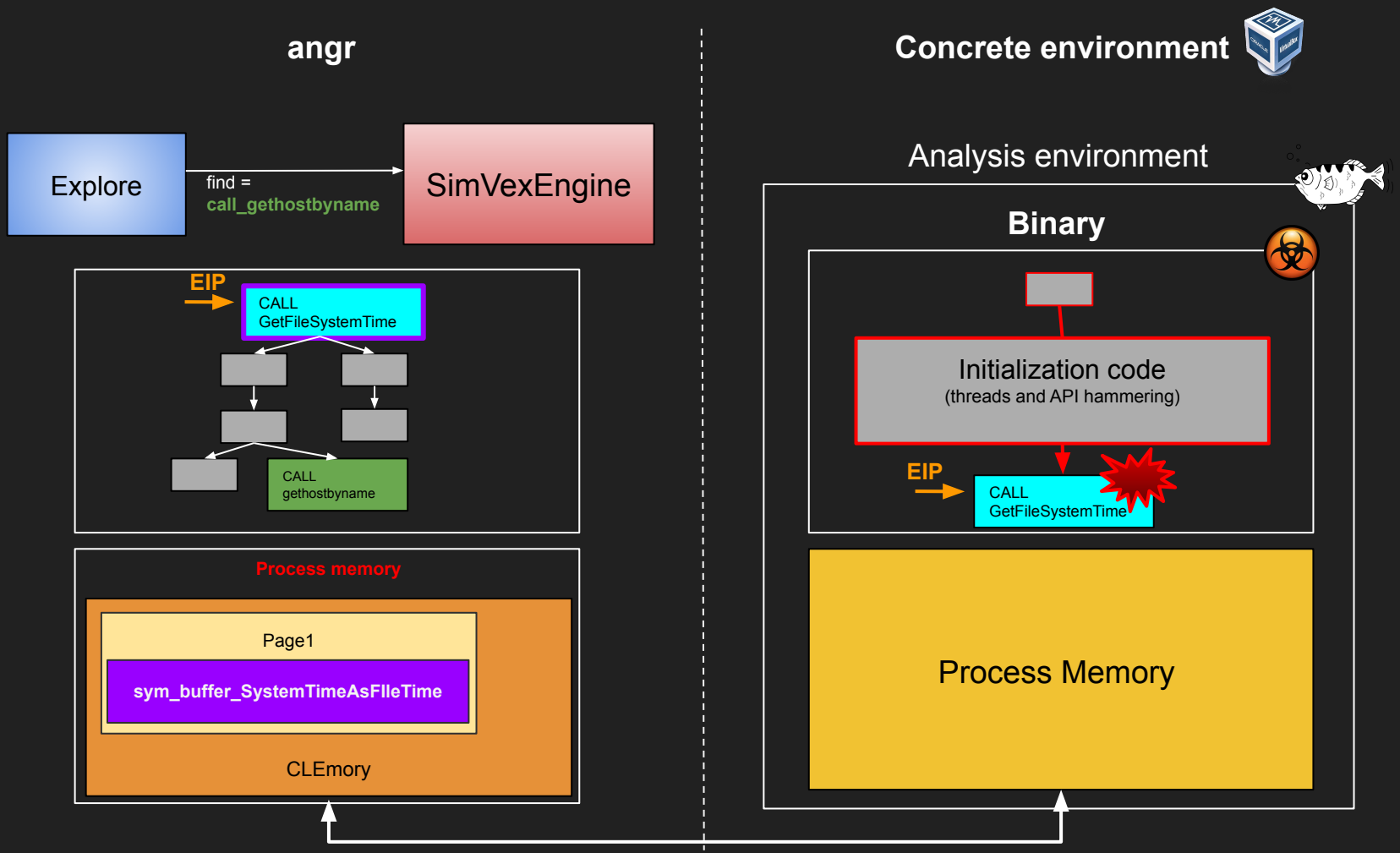
Process Memory

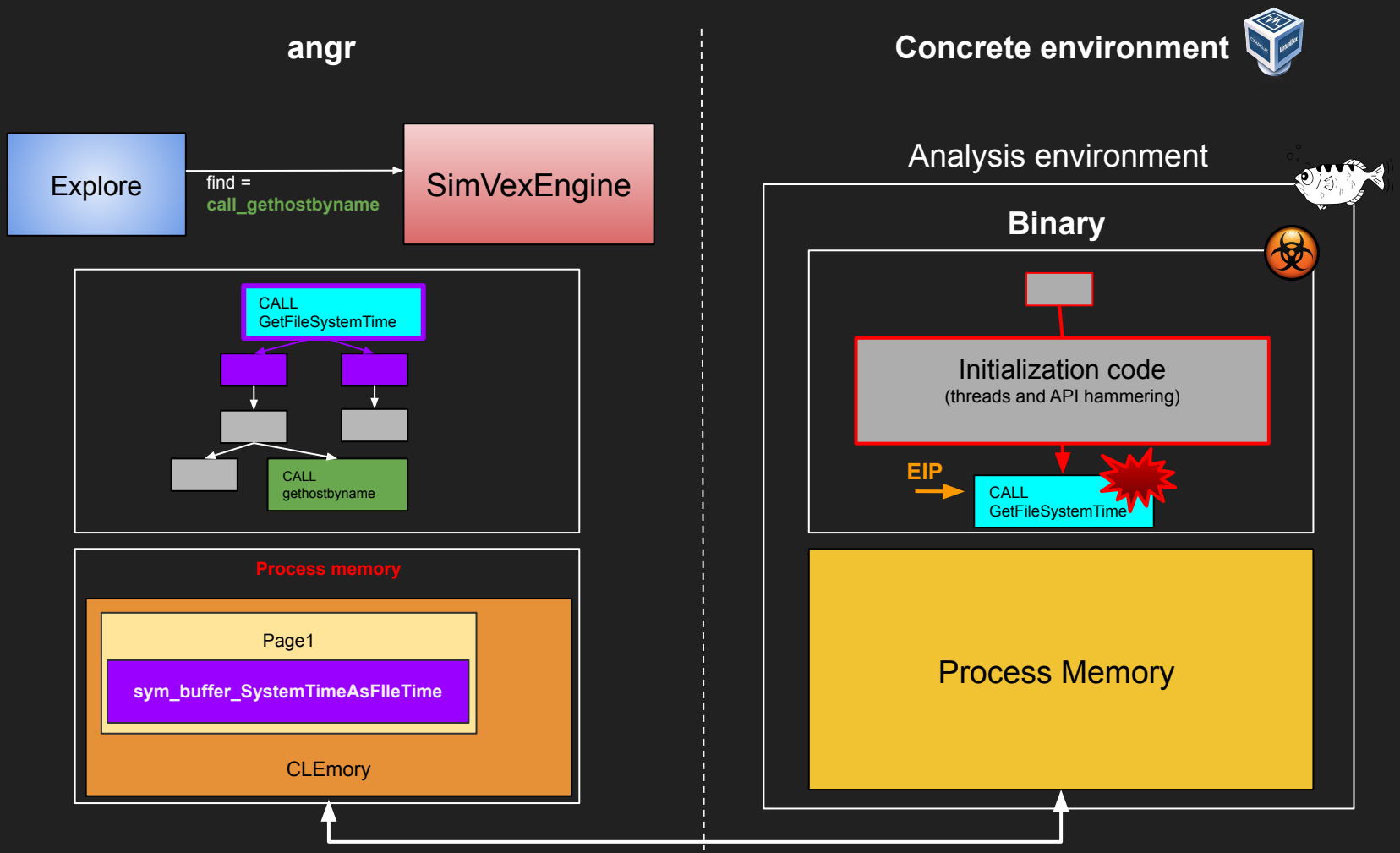

Symbolic ← Concrete
angr
Concrete environment
Symbion
Concrete SimPlugin
SimEngineConcrete
sync
Analysis environment
Symbolic ← Concrete
EIP
CALL GetFile
CALL gethostbyname
hammering)
EIP
CALL GetFileSystemTime
Process memory
CLEmory
Process Memory

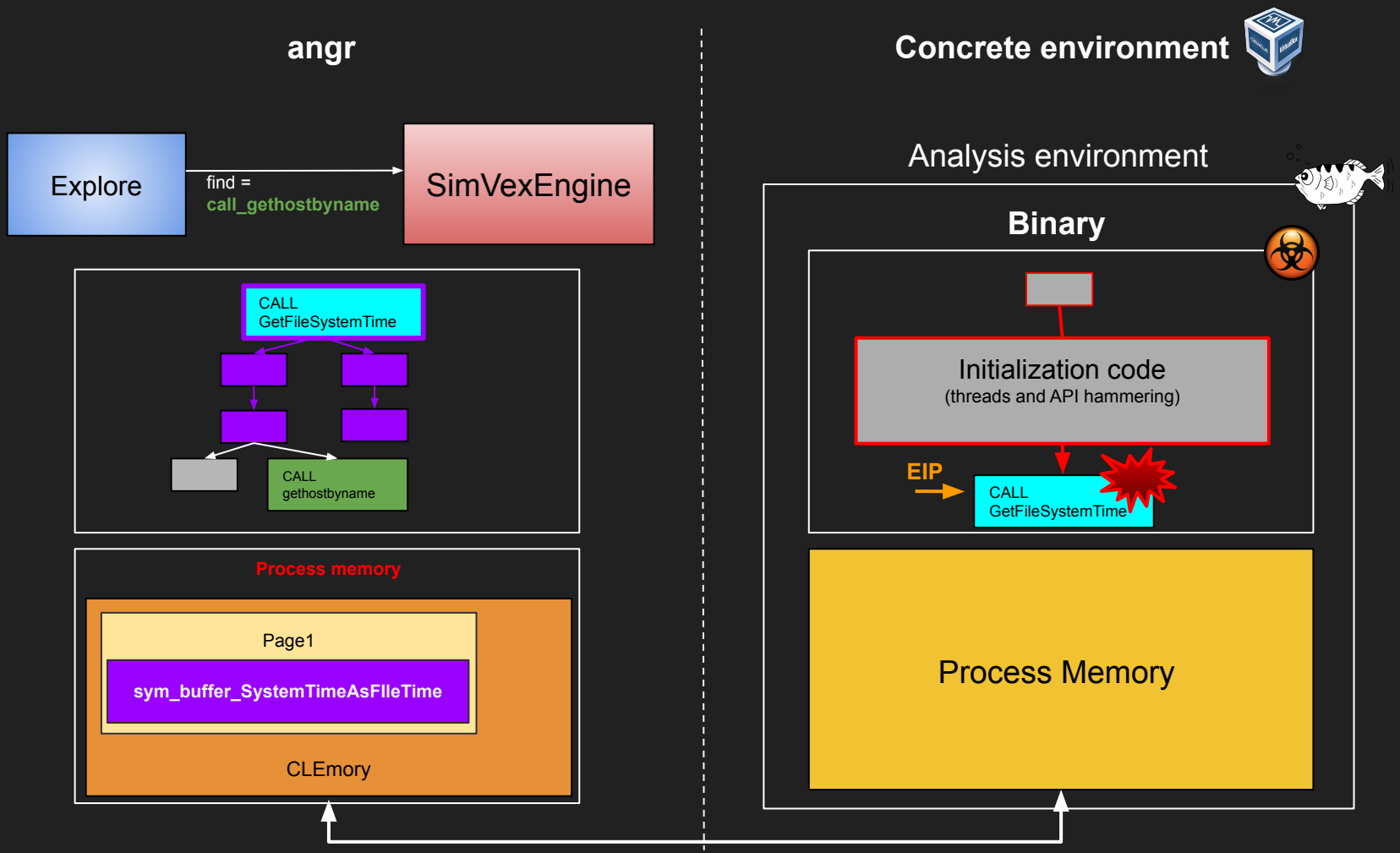

Symbolic ← Concrete
angr
Concrete environment
Symbion
Concrete SimPlugin
sync
SimEngineConcrete
ConcreteTarget
Analysis environment
Binary
EIP
CALL GetFileSystemTime
CALL gethostbyname
Initialization code
(threads and API hammering)
EIP
CALL GetFileSystemTime
Process memory
CLEmory
Process Memory

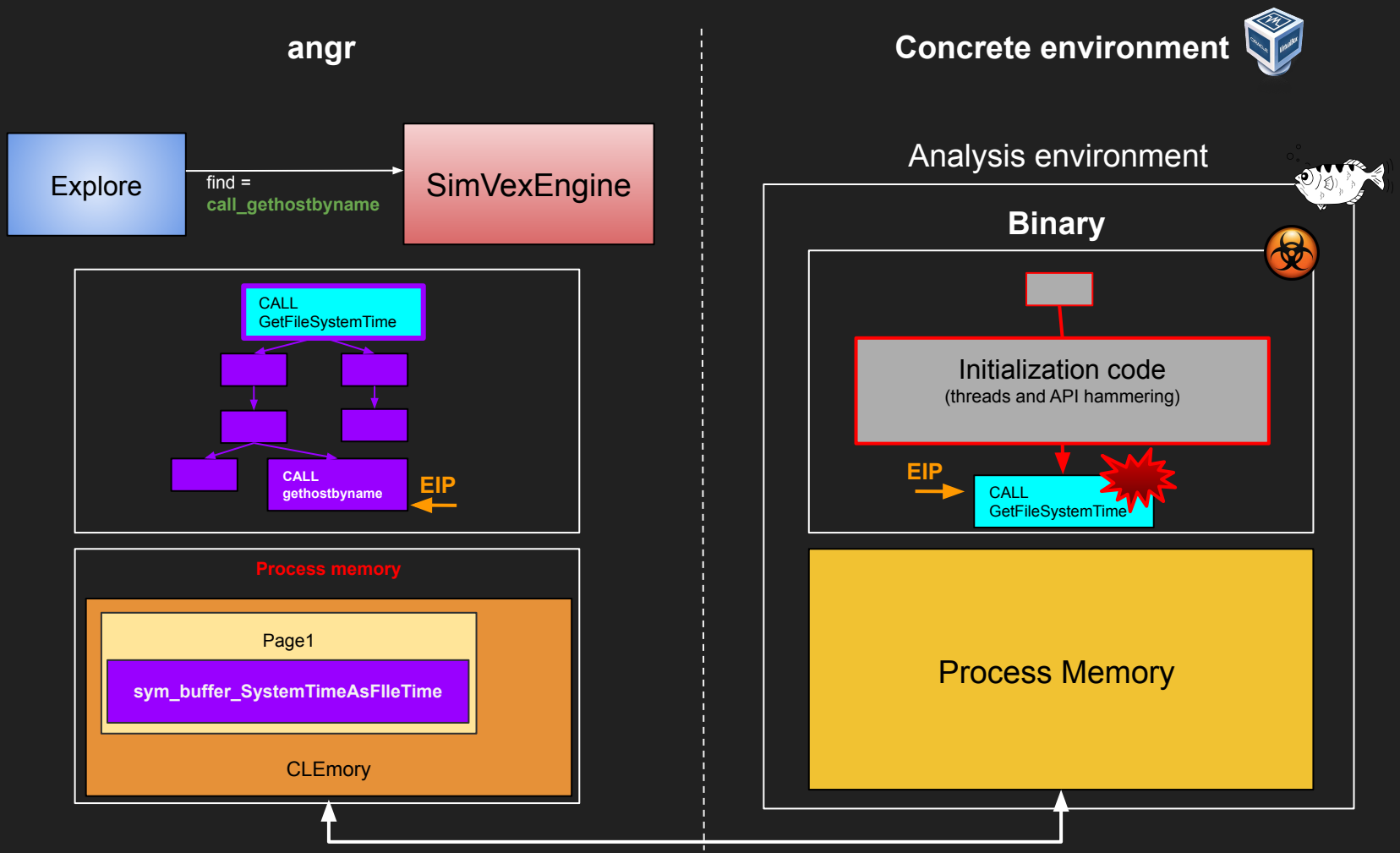

angr
Concrete environment
Explore
find =
call_gethostbyname
SimVexEngine
Analysis environment
Binary
EIP
CALL
GetFile
gethostbyname
Symbolic Exploration
on code
hammering)
CALL
GetFileSystemTime
Process memory
CLEmory
Process Memory

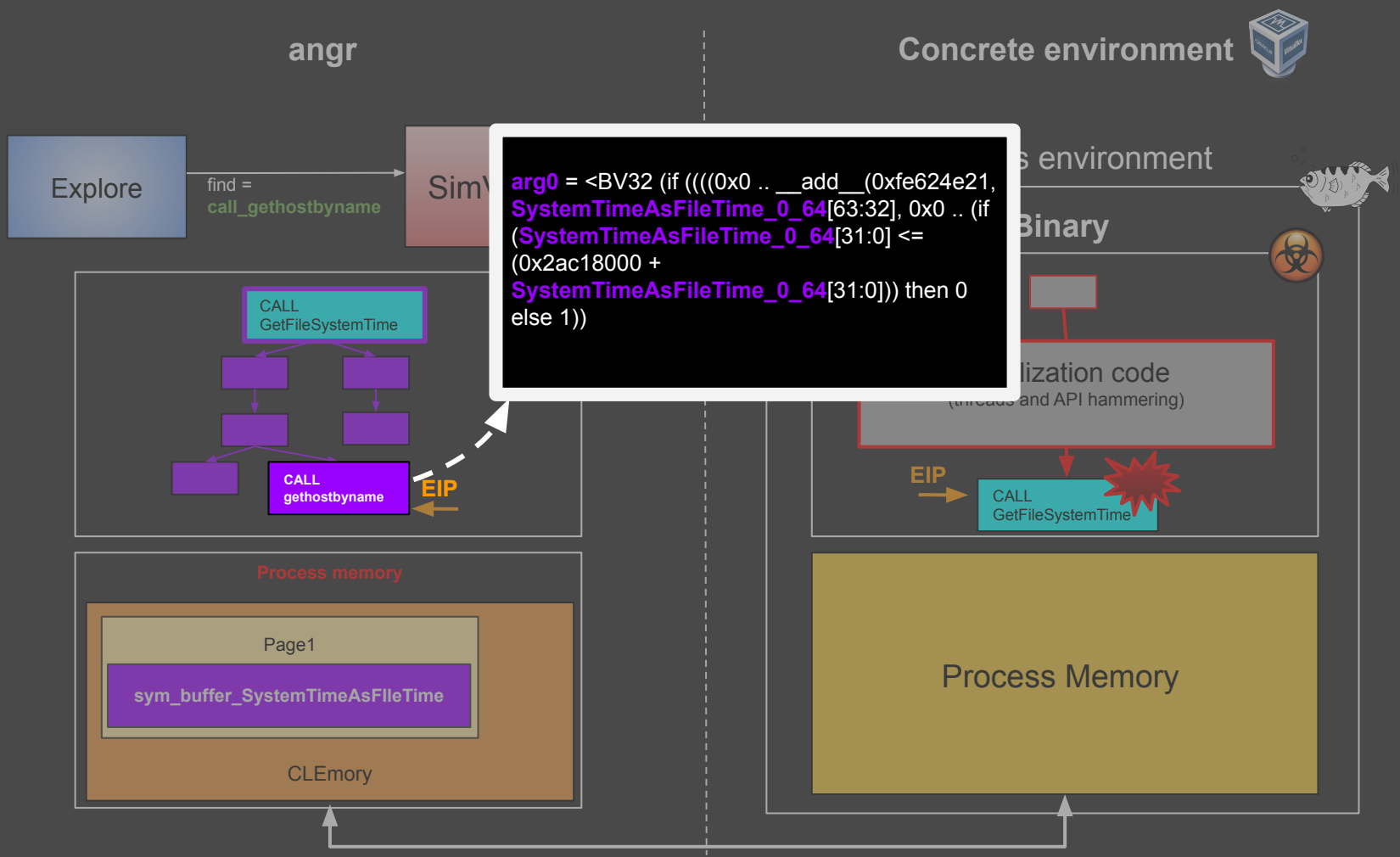

angr
Explore
find =
call_gethostbyname
SimVexEngine
EIP
CALL
GetFileSystemTime
CALL
gethostbyname
Process memory
CLEmory
Concrete environment
Analysis environment
Binary
Initialization code
(threads and API hammering)
EIP
CALL
GetFileSystemTime
Process Memory


angr
Concrete environment
Explore
find =
call_gethostbyname
SimVexEngine
EIP
CALL
GetFileSystemTime
CALL
gethostbyname
Process memory
Page1
sym_buffer_SystemTimeAsFIleTime
CLEmory
Analysis environment
Binary
Initialization code
(threads and API hammering)
EIP
CALL
GetFileSystemTime
Process Memory


angr
Explore
find =
call_gethostbyname
SimVexEngine
CALL
GetFileSystemTime
CALL
gethostbyname
Process memory
Page1
sym_buffer_SystemTimeAsFIleTime
CLEmory
Concrete environment
Analysis environment
Binary
Initialization code
(threads and API hammering)
EIP
CALL
GetFileSystemTime
Process Memory

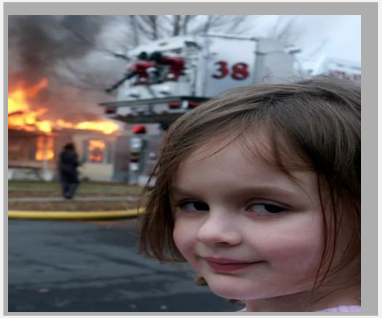

angr
Explore
find =
call_gethostbyname
SimVexEngine
CALL
GetFileSystemTime
CALL
gethostbyname
Process memory
Page1
sym_buffer_SystemTimeAsFIleTime
CLEmory
Concrete environment
Analysis environment
Binary
Initialization code
(threads and API hammering)
EIP
CALL
GetFileSystemTime
Process Memory


angr
Concrete environment
Explore
find =
call_gethostbyname
SimVexEngine
CALL
GetFileSystemTime
CALL
gethostbyname
EIP
Process memory
Page1
sym_buffer_SystemTimeAsFIleTime
CLEmory
Analysis environment
Binary
Initialization code
(threads and API hammering)
EIP
CALL
GetFileSystemTime
Process Memory


angr
Concrete environment
Explore
find =
call_gethostbyname
arg0 = <BV32 (if ((((0x0 .. __add__(0xfe624e21, SystemTimeAsFileTime_0_64[63:32], 0x0 .. (if (SystemTimeAsFileTime_0_64[31:0] <= (0x2ac18000 + SystemTimeAsFileTime_0_64[31:0])) then 0 else 1))
s environment
Binary
CALL
GetFileSystemTime
lization code
(threads and API hammering)
CALL
gethostbyname
EIP
EIP
CALL
GetFileSystemTime
Process memory
Page1
sym_buffer_SystemTimeAsFIleTime
CLEmory
Process Memory

(More) Use Cases

Vulnerabilities Hunting



Exploit Writing/Generation

More!

Comparison

- Question prediction: Why isn’t this just “Concolic Execution?”

Comparison

- Question prediction: Why isn't this just "Concolic Execution?"
- **Concolic execution** has the goal of improving code coverage of vanilla symbolic execution.
- The techniques are orthogonal and can be chained together

Comparison

- Other similar tools have been developed in the past:
  - Avatar2
  - Triton
  - S2E
  - Mayhem (not freely available to the community)

- None was really making available this kind of technique in a customizable, general purpose and easy to use/programmatic way

Limitations

- Program execution correctness not guaranteed by default
  - Users could force executions that are not feasible
  - Solutions to mitigate this can be implemented on top of the technique
- Desynchronized environment interactions
  - Only registers and memory are synchronized
  - States of other objects (socket,file,stdin/stdout) are not sync with the symbolic engine
- Targets support
  - Limited amount of Concrete Targets
  - “Lazy developing” (as needed)

Takeaways

1. Symbion is a building block that can empower different new analyses applied to many scenarios

2. Supporting symbolic execution at real-world-program scale is essential

3. Symbion provides a compromise between the power of symbolic execution and the ability to operate on real-world programs

Support

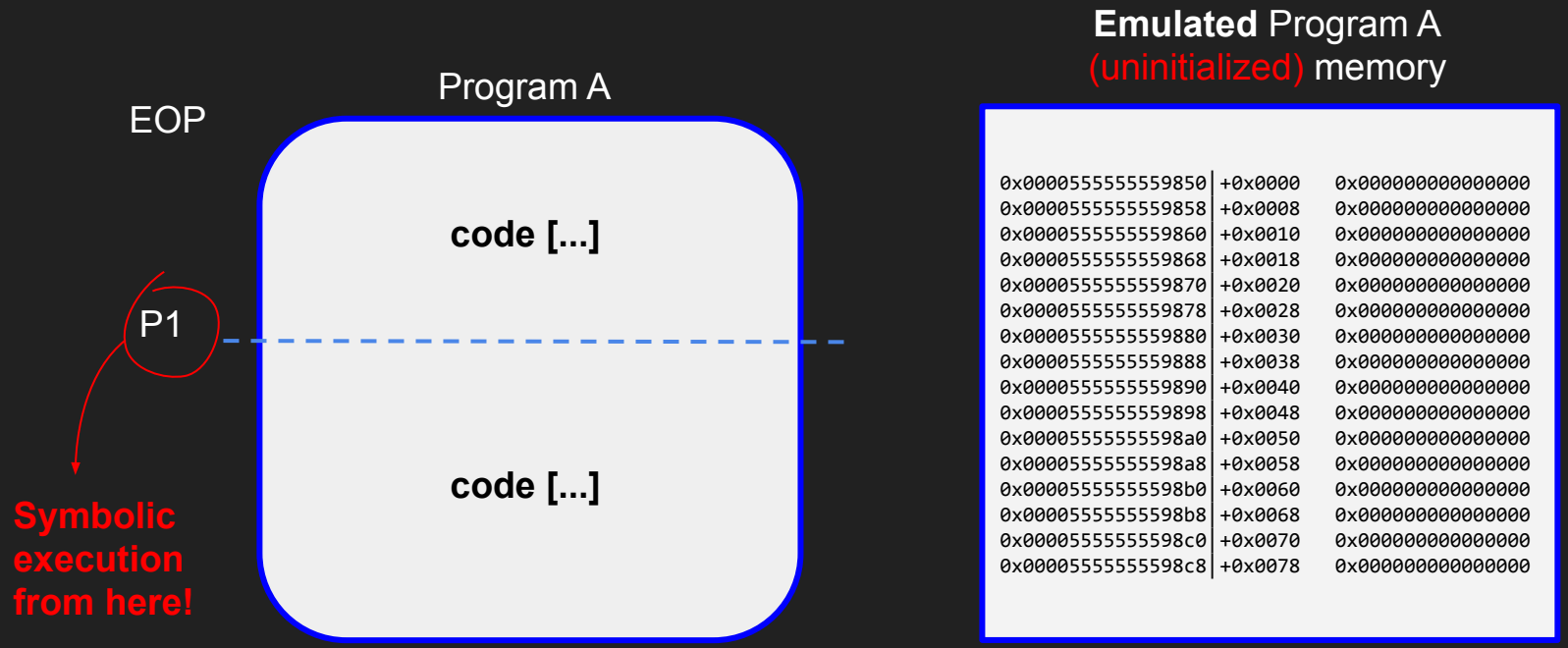
- Open source
  - https://github.com/angr/angr
  - https://github.com/degrigis/symbion-use-cases
  - https://github.com/angr/angr-targets
- Docs & Tutorials
  - https://angr.io/blog/angr_symbion/
  - https://docs.angr.io/advanced-topics/symbion
- Support
  - https://angr.io/invite/
  - Just yell in #help or directly ping me @degrigis

Thanks!

degrigis@cs.ucsb.edu

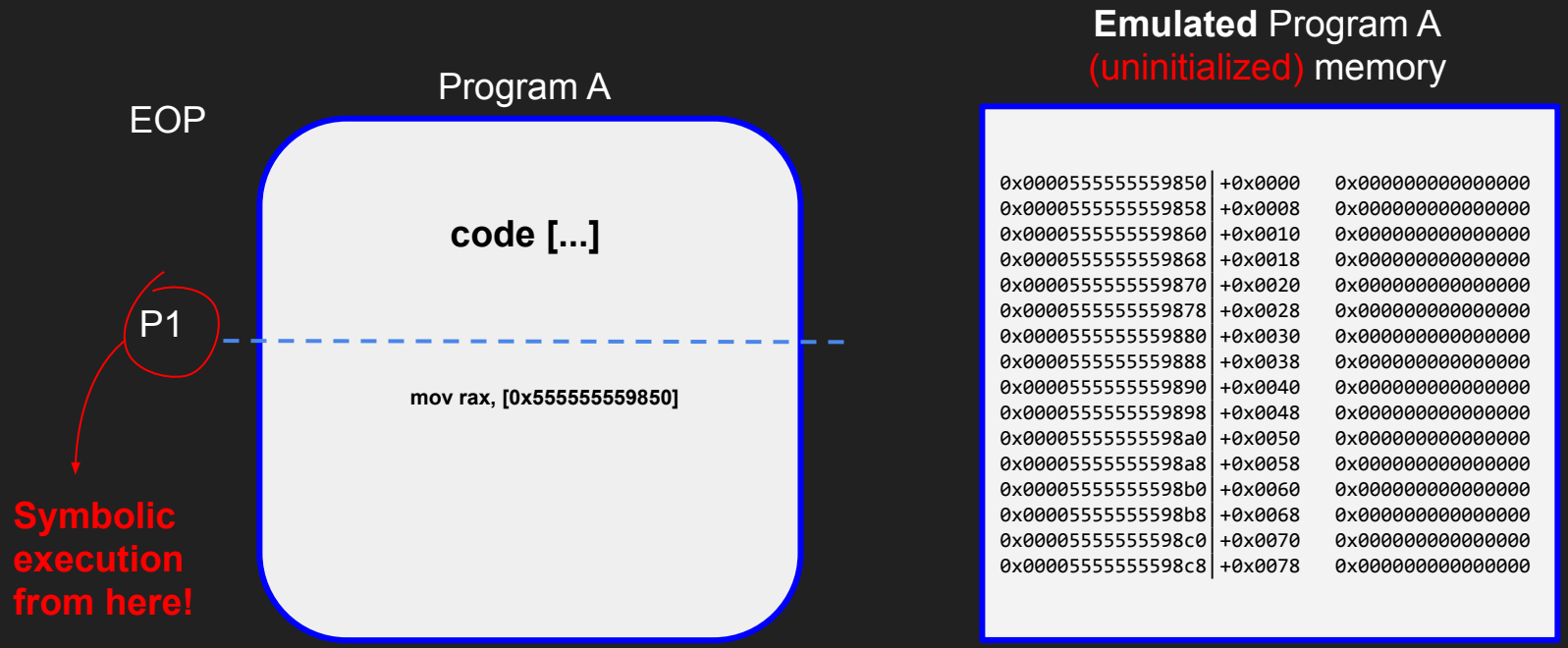
@degrigis

Motivation



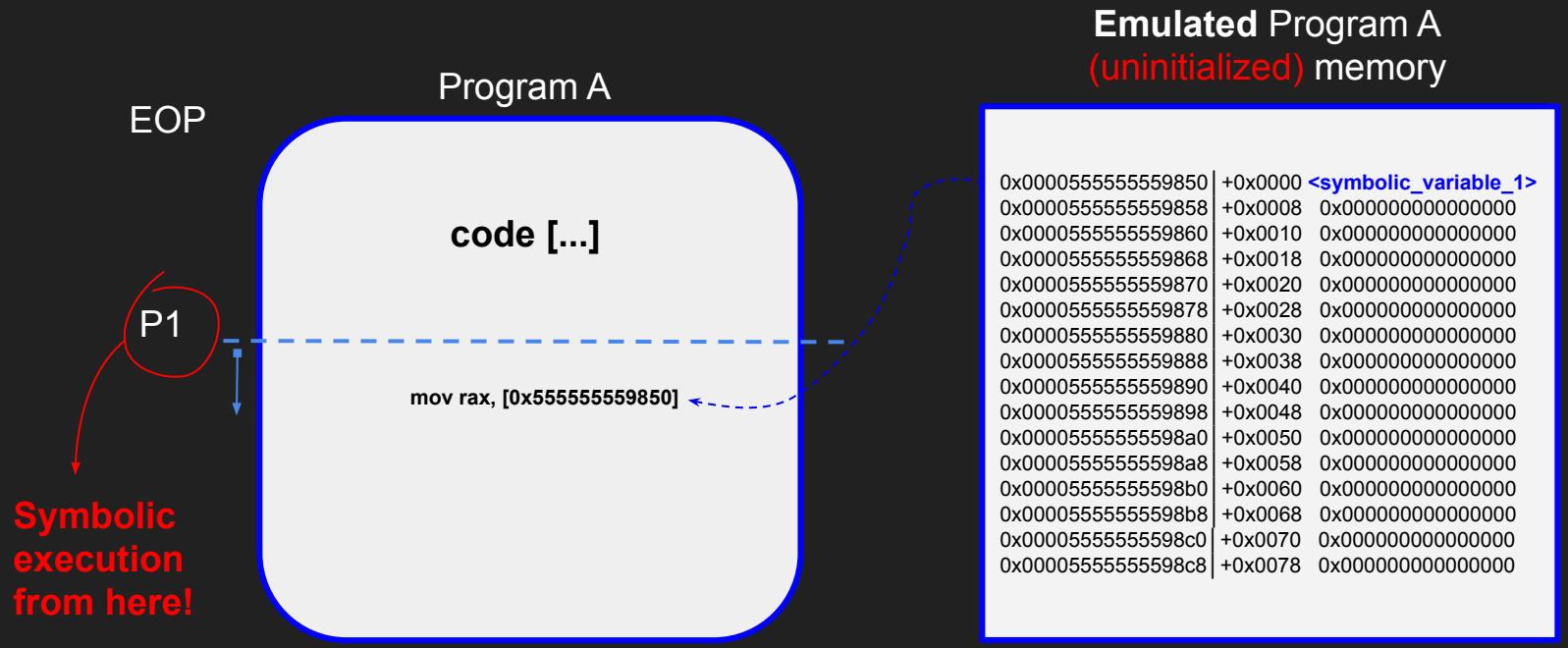
"under-constrained" symbolic execution

Motivation



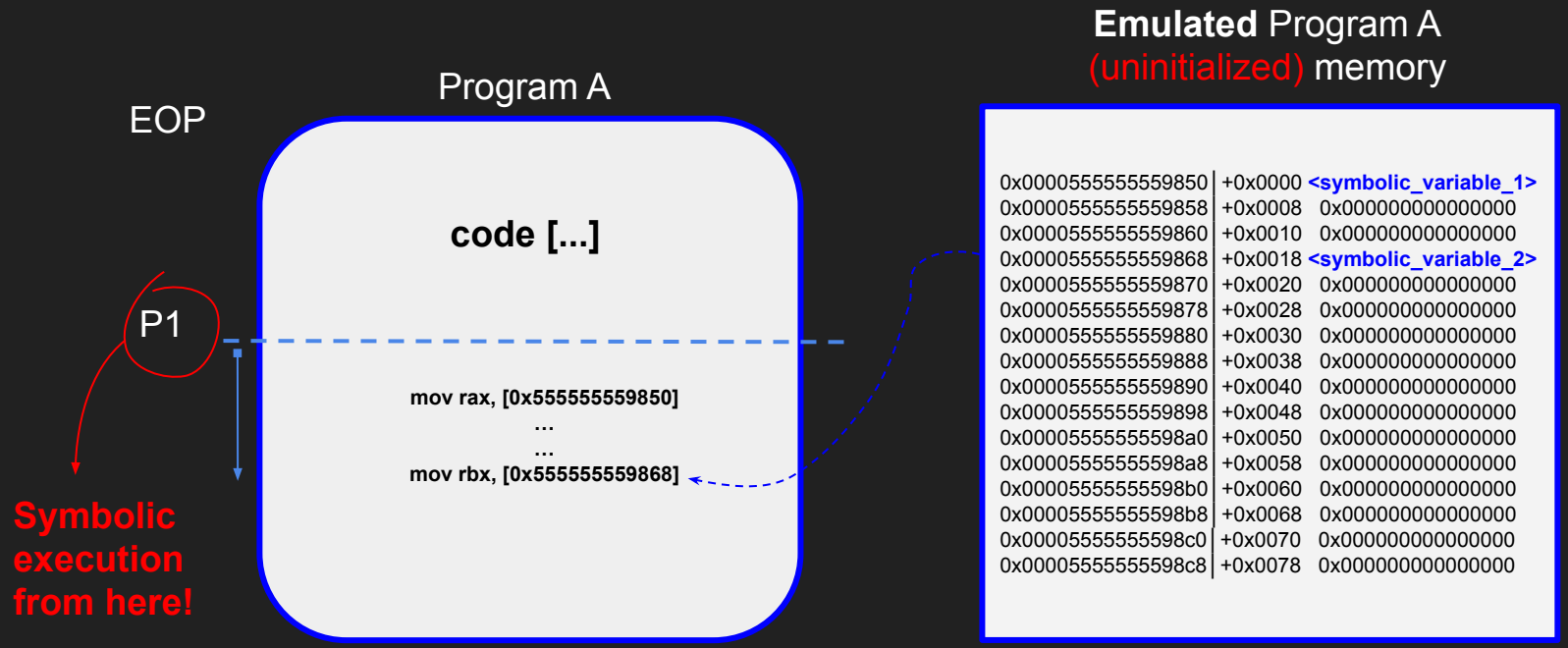
“under-constrained” symbolic execution

Motivation



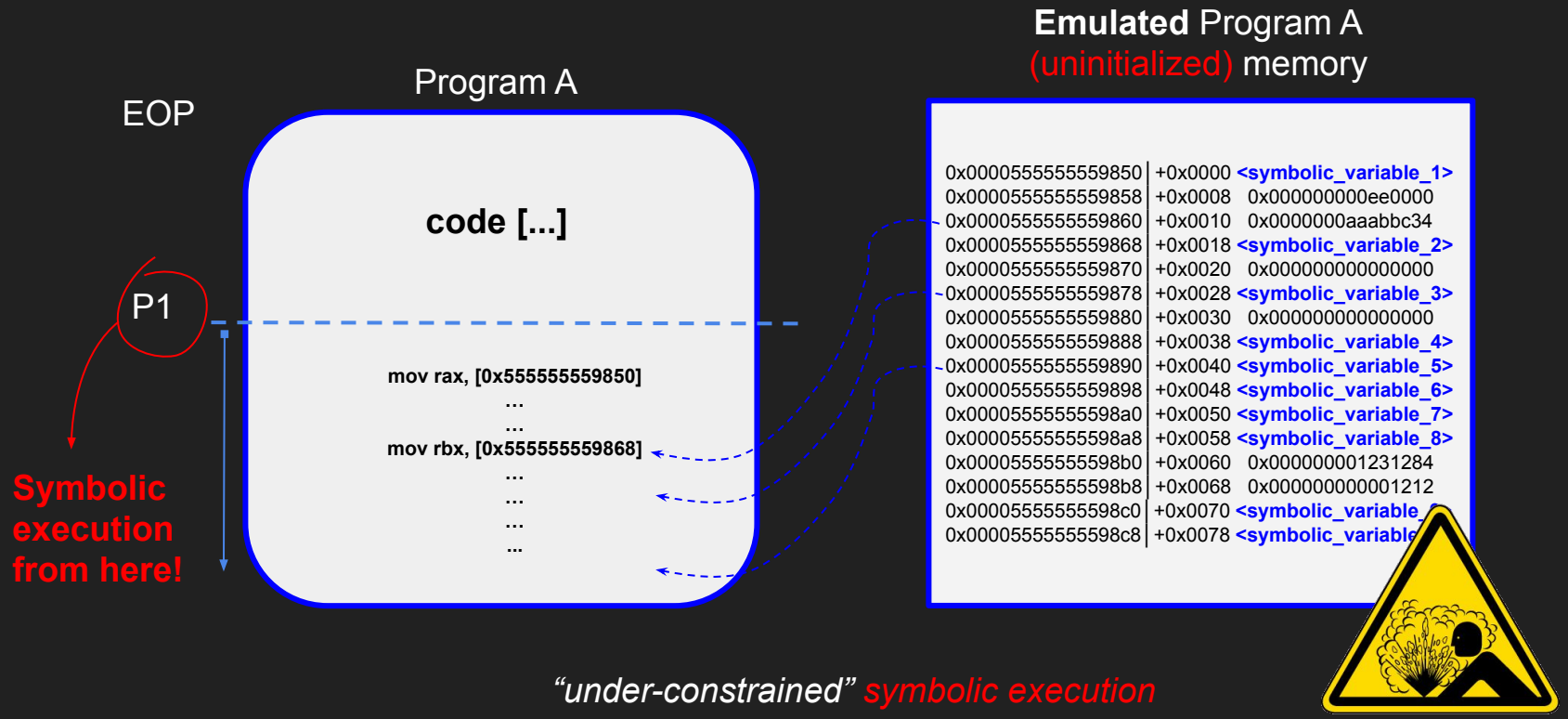
"under-constrained" symbolic execution

Motivation



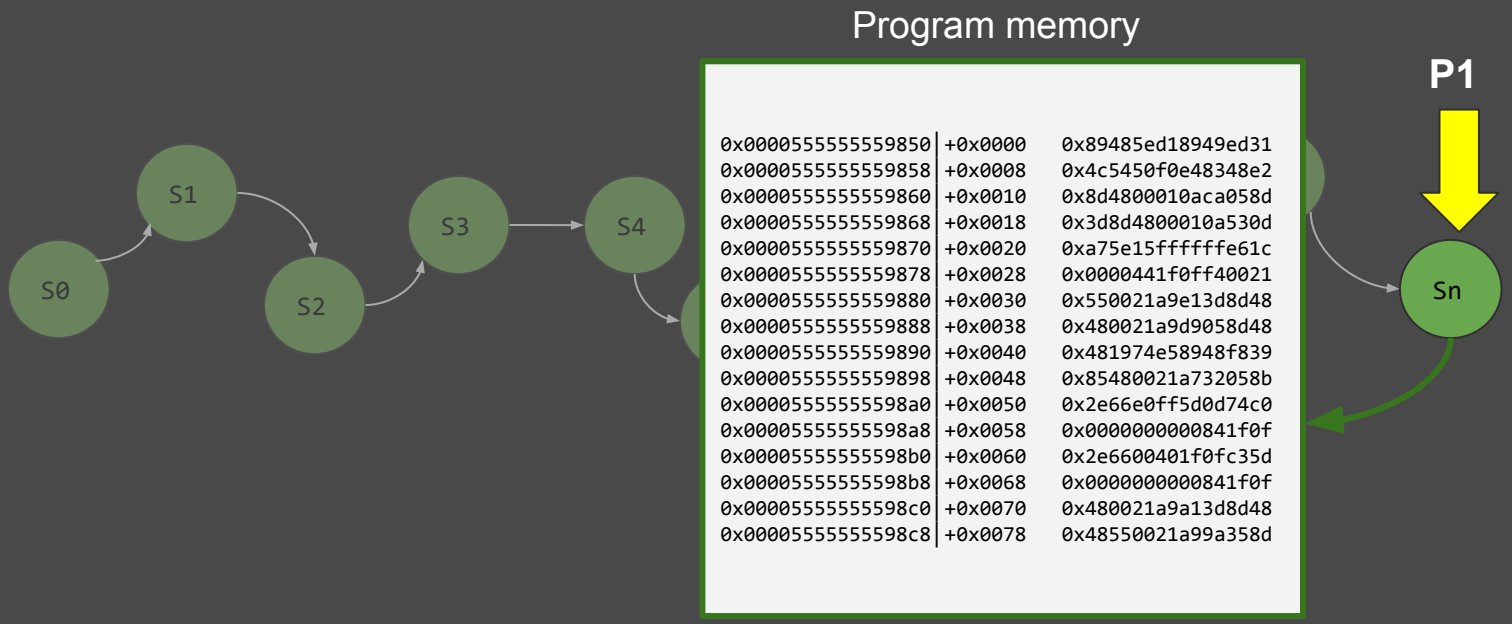
"under-constrained" symbolic execution

Motivation



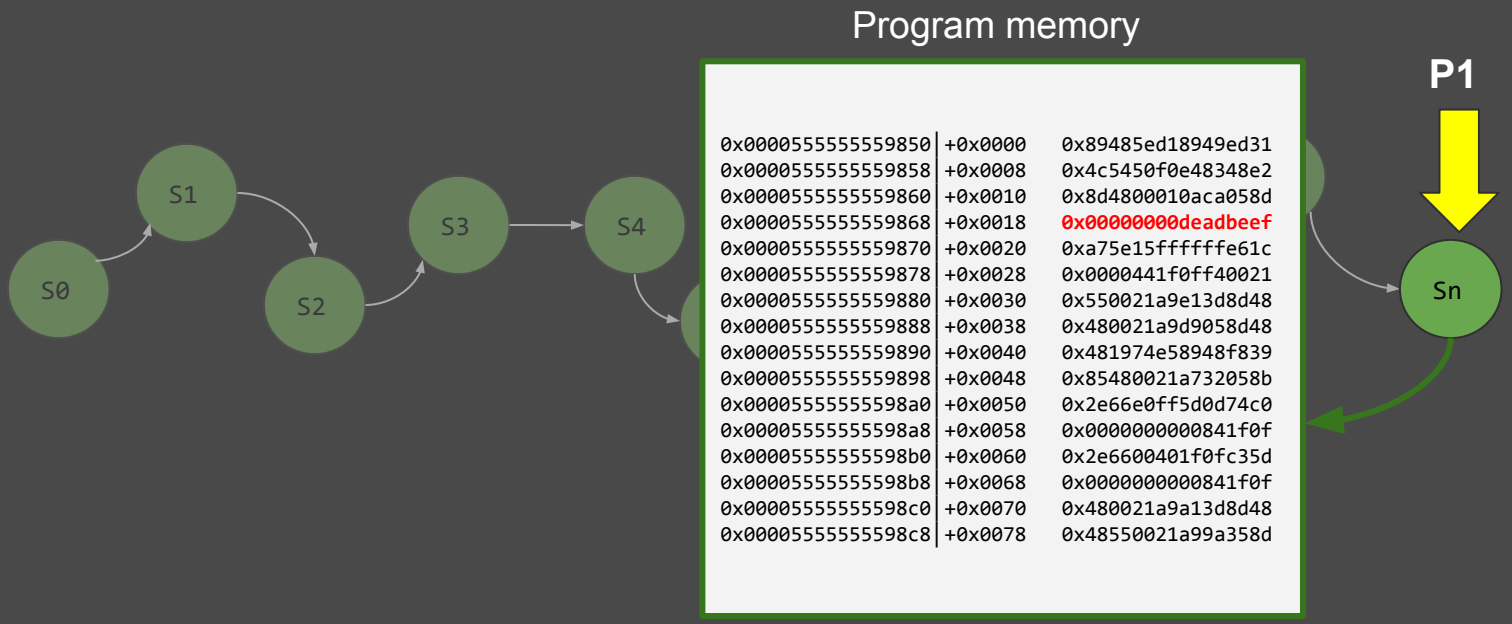
"under-constrained" symbolic execution

Approach



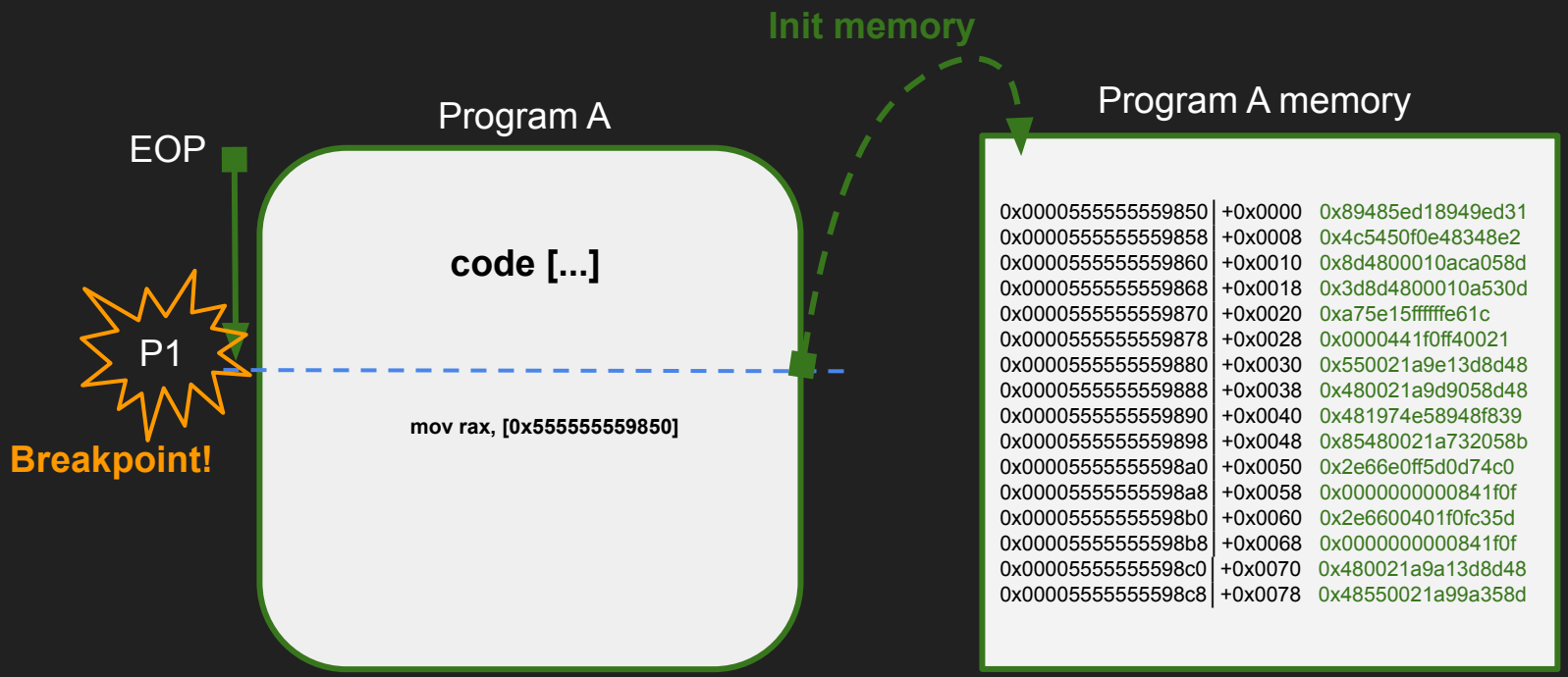
Interleaved symbolic execution

Approach



Interleaved symbolic execution

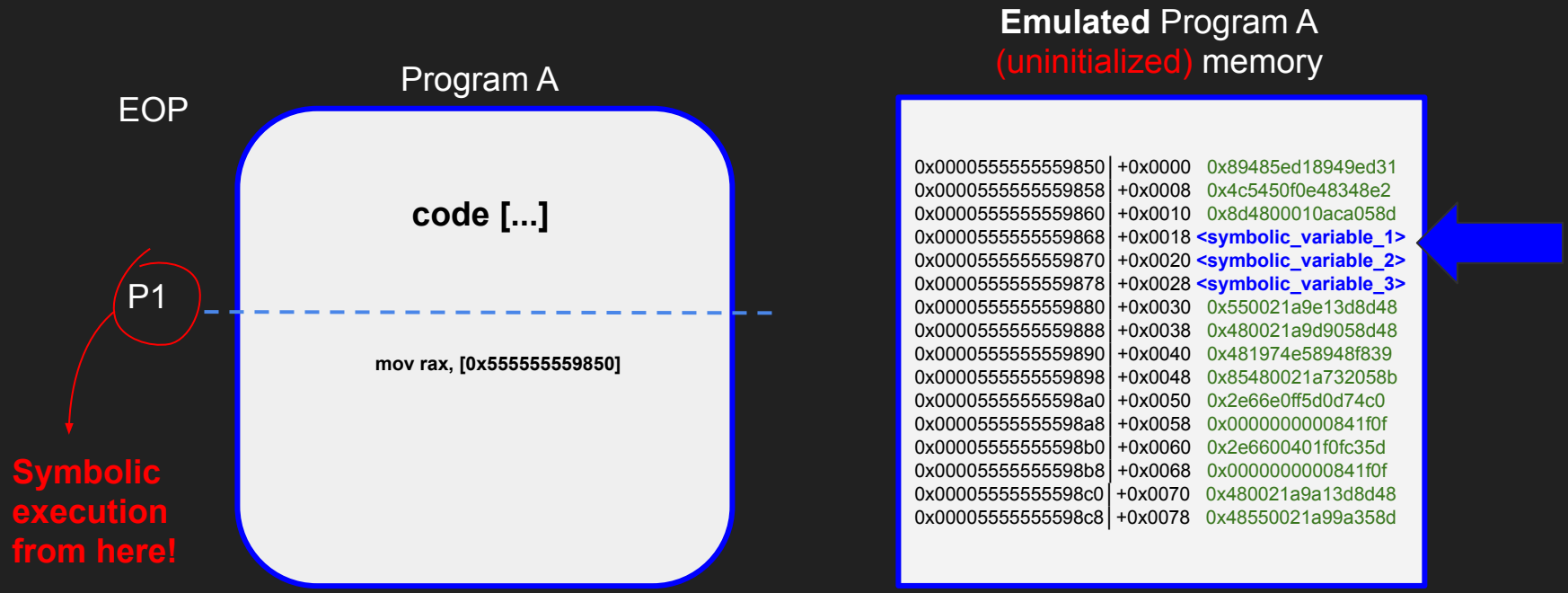
Approach



Interleaved symbolic execution

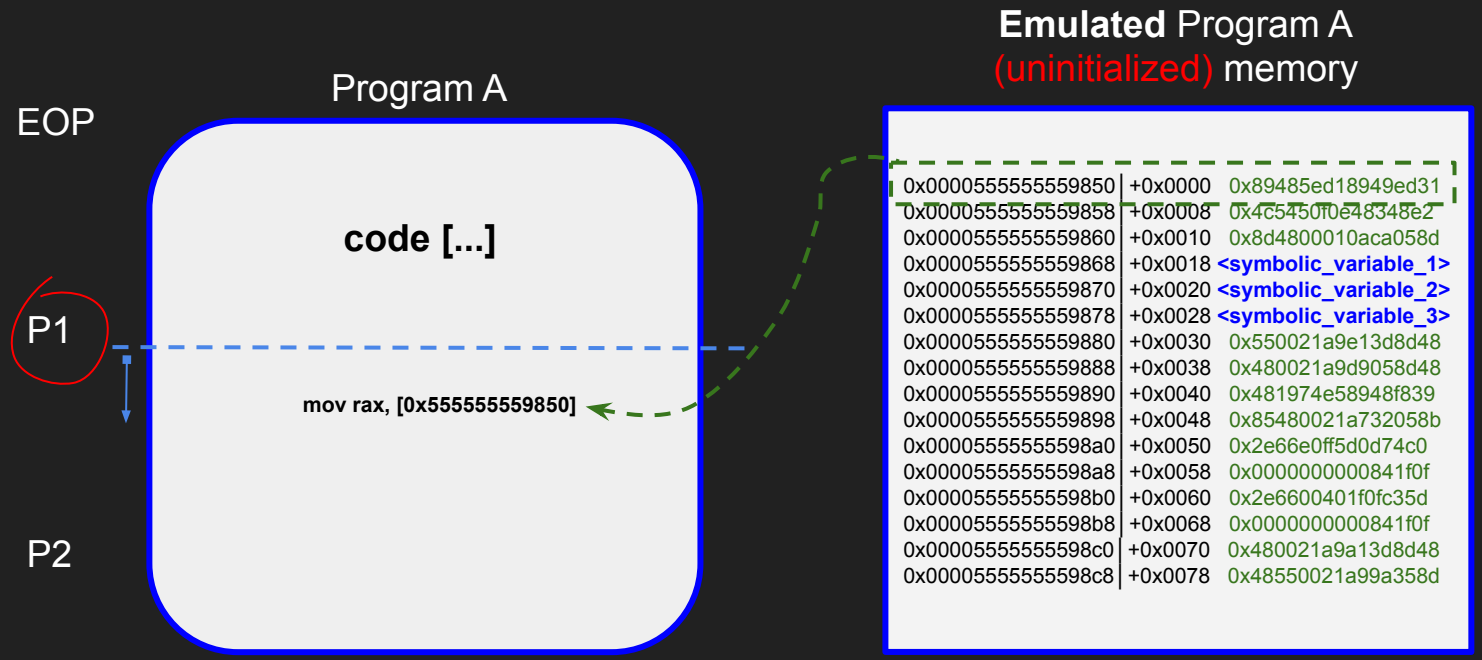
(Phase 1: concrete execution to P1)

Approach



Interleaved symbolic execution
(Phase 2: setup symbolic data)

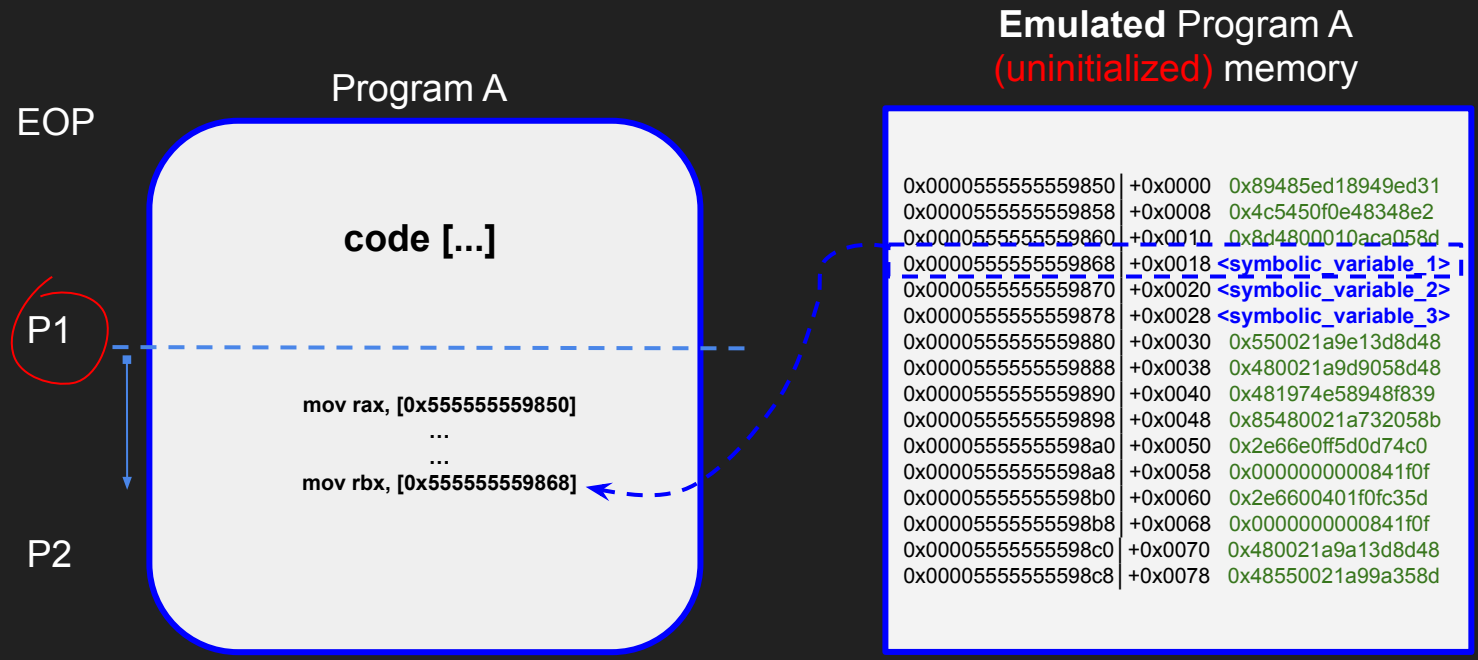
Approach



Interleaved symbolic execution

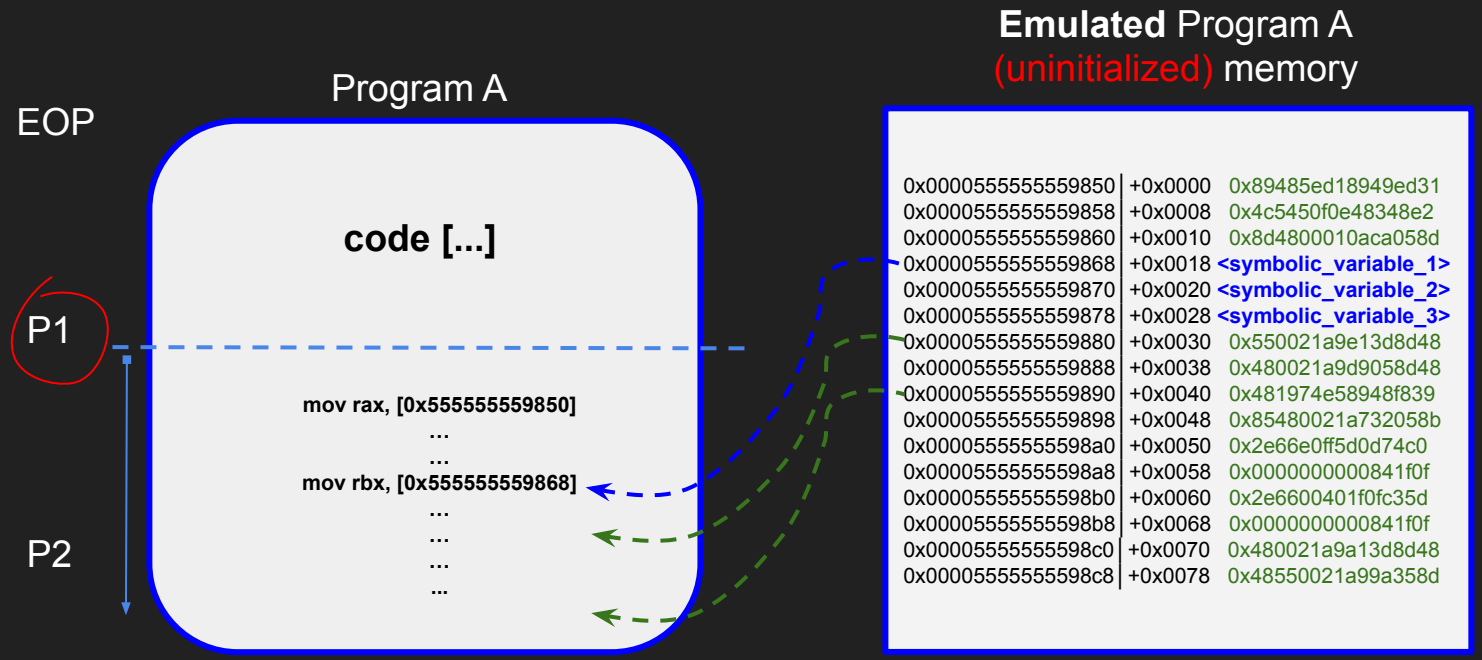
(Phase 3: symbolic execution)

Approach



Interleaved symbolic execution
(Phase 3: symbolic execution)

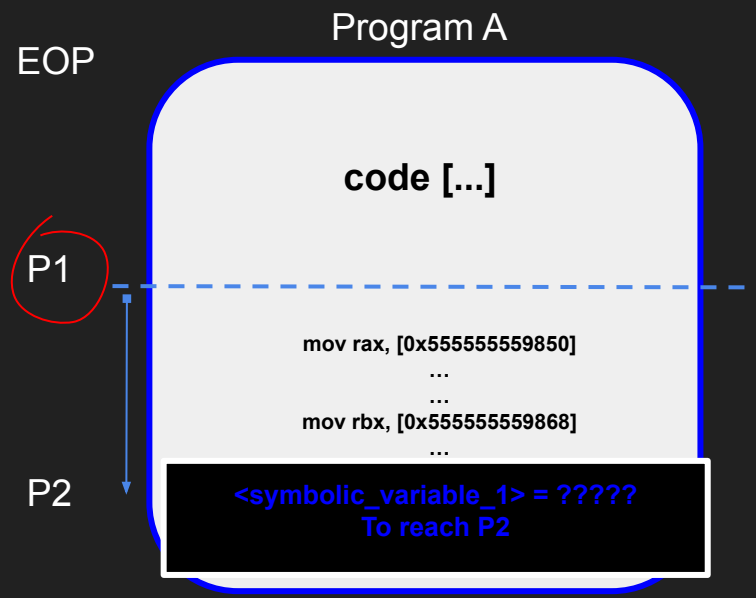
Approach



Interleaved symbolic execution

(Phase 3: symbolic execution)

Approach

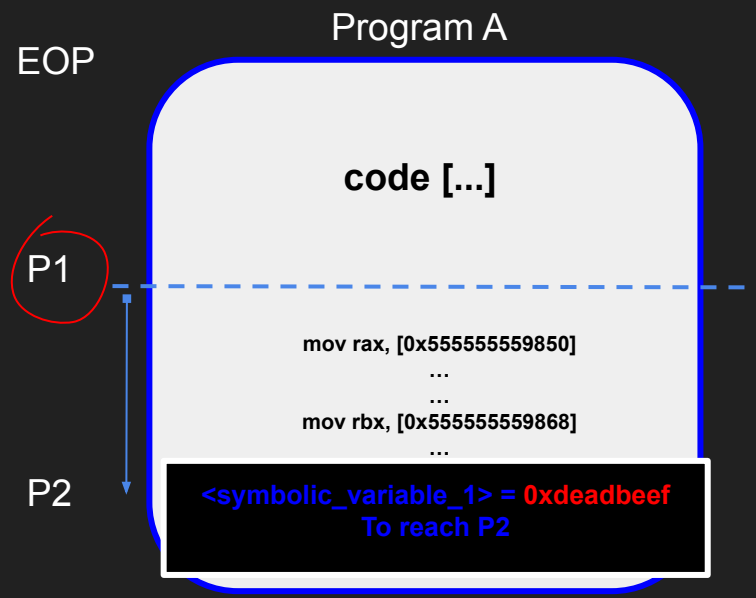


Emulated Program A
(uninitialized) memory

0x0000555555559850	+0x0000	0x89485ed18949ed31
0x0000555555559858	+0x0008	0x4c5450f0e48348e2
0x0000555555559860	+0x0010	0x8d4800010aca058d
0x0000555555559868	+0x0018	<symbolic_variable_1>
0x0000555555559870	+0x0020	<symbolic_variable_2>
0x0000555555559878	+0x0028	<symbolic_variable_3>
0x0000555555559880	+0x0030	0x550021a9e13d8d48
0x0000555555559888	+0x0038	0x480021a9d9058d48
0x0000555555559890	+0x0040	0x481974e58948f839
0x0000555555559898	+0x0048	0x85480021a732058b
0x00005555555598a0	+0x0050	0x2e66e0ff5d0d74c0
0x00005555555598a8	+0x0058	0x0000000000841f0f
0x00005555555598b0	+0x0060	0x2e6600401f0fc35d
0x00005555555598b8	+0x0068	0x0000000000841f0f
0x00005555555598c0	+0x0070	0x480021a9a13d8d48
0x00005555555598c8	+0x0078	0x48550021a99a358d

Interleaved symbolic execution
(Phase 3: symbolic execution)

Approach

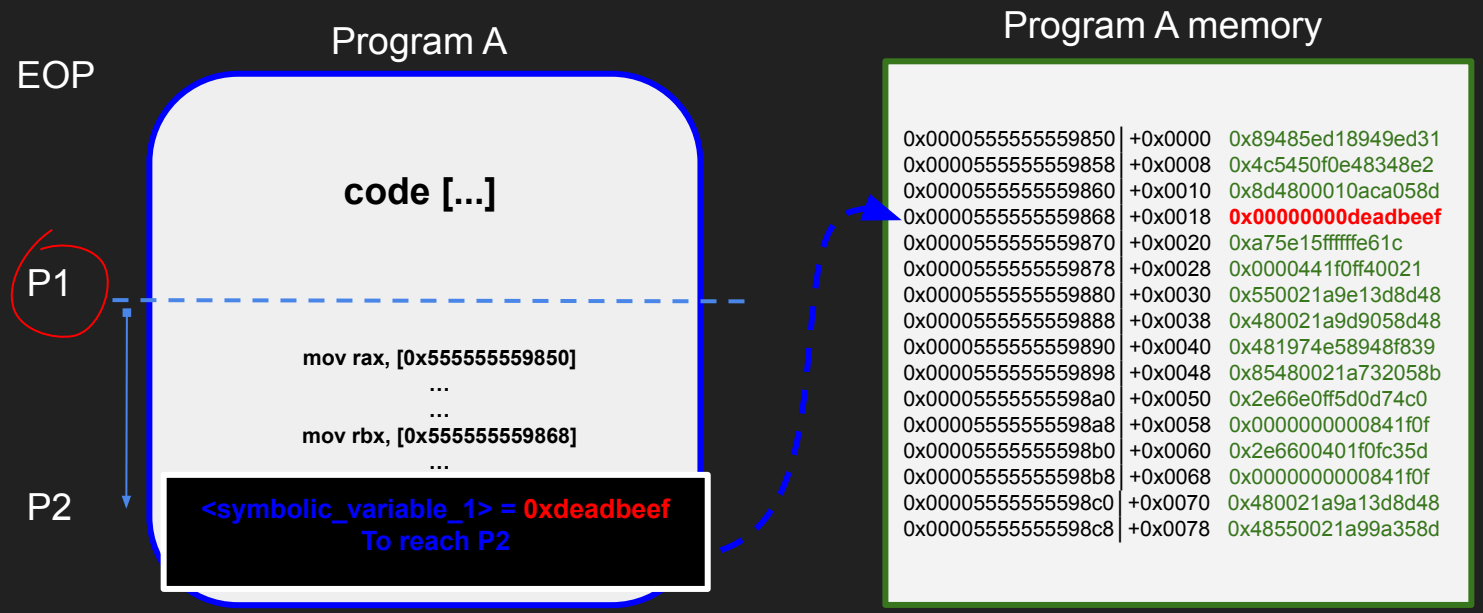


Emulated Program A
(uninitialized) memory

0x0000555555559850	+0x0000	0x89485ed18949ed31
0x0000555555559858	+0x0008	0x4c5450f0e48348e2
0x0000555555559860	+0x0010	0x8d4800010aca058d
0x0000555555559868	+0x0018	<symbolic_variable_1>
0x0000555555559870	+0x0020	<symbolic_variable_2>
0x0000555555559878	+0x0028	<symbolic_variable_3>
0x0000555555559880	+0x0030	0x550021a9e13d8d48
0x0000555555559888	+0x0038	0x480021a9d9058d48
0x0000555555559890	+0x0040	0x481974e58948f839
0x0000555555559898	+0x0048	0x85480021a732058b
0x00005555555598a0	+0x0050	0x2e66e0ff5d0d74c0
0x00005555555598a8	+0x0058	0x0000000000841f0f
0x00005555555598b0	+0x0060	0x2e6600401f0fc35d
0x00005555555598b8	+0x0068	0x0000000000841f0f
0x00005555555598c0	+0x0070	0x480021a9a13d8d48
0x00005555555598c8	+0x0078	0x48550021a99a358d

Interleaved symbolic execution
(Phase 3: symbolic execution)

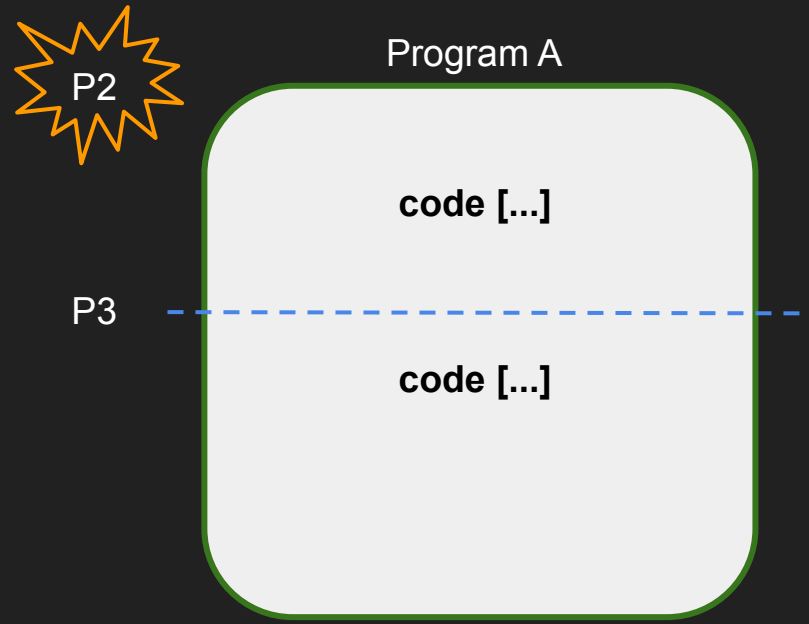
Approach



Interleaved symbolic execution

(Phase 4: Edit program A concrete memory)

Approach



Program A memory

0x0000555555559850	+0x0000	0x89485ed18949ed31
0x0000555555559858	+0x0008	0x4c5450f0e48348e2
0x0000555555559860	+0x0010	0x8d4800010aca058d
0x0000555555559868	+0x0018	**0x00000000deadbeef**
0x0000555555559870	+0x0020	0xa75e15ffffffe61c
0x0000555555559878	+0x0028	0x0000441f0ff40021
0x0000555555559880	+0x0030	0x550021a9e13d8d48
0x0000555555559888	+0x0038	0x480021a9d9058d48
0x0000555555559890	+0x0040	0x481974e58948f839
0x0000555555559898	+0x0048	0x85480021a732058b
0x00005555555598a0	+0x0050	0x2e66e0ff5d0d74c0
0x00005555555598a8	+0x0058	0x0000000000841f0f
0x00005555555598b0	+0x0060	0x2e6600401f0fc35d
0x00005555555598b8	+0x0068	0x0000000000841f0f
0x00005555555598c0	+0x0070	0x480021a9a13d8d48
0x00005555555598c8	+0x0078	0x48550021a99a358d

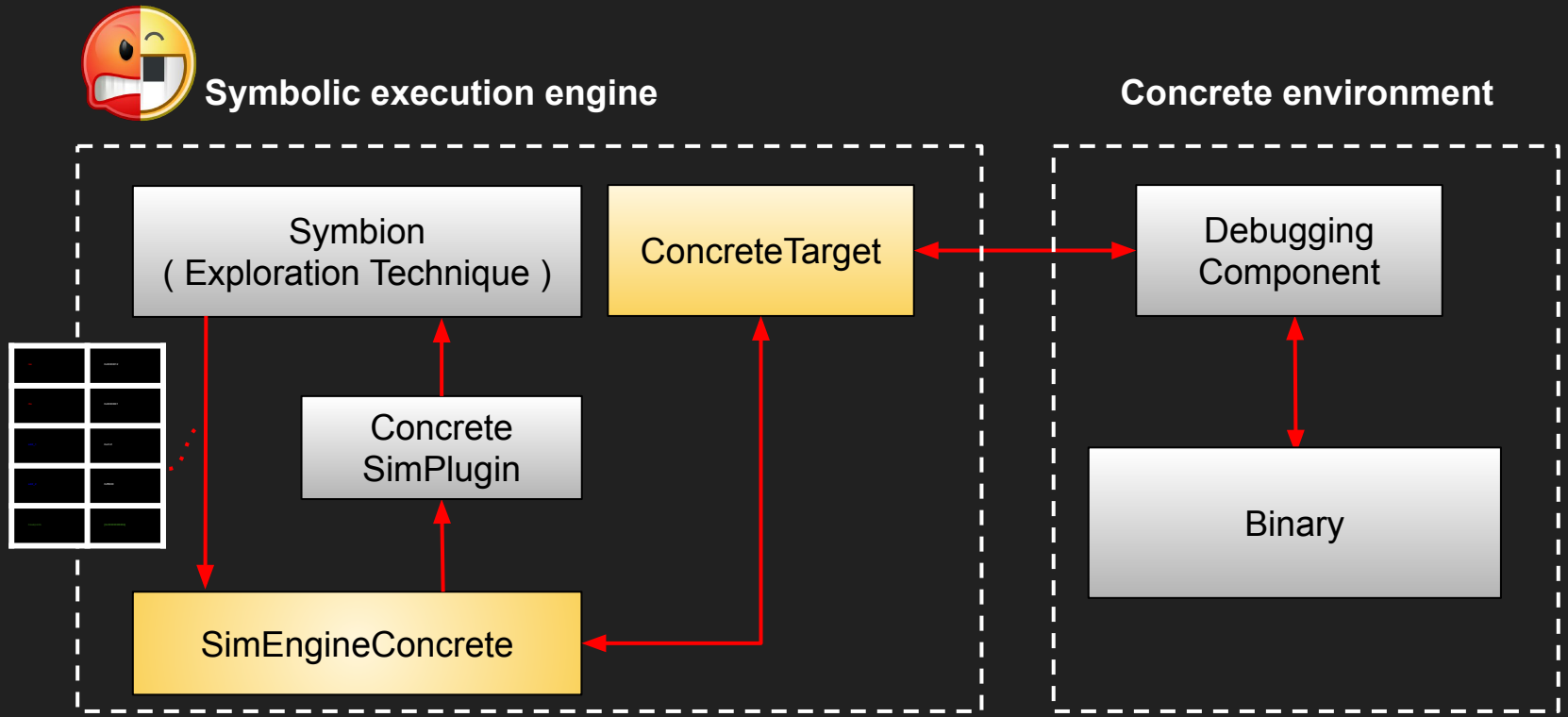
Interleaved symbolic execution

(Phase 5: Resume concrete execution)

Symbion - Exploration Technique

- API provided to the users in order to control the *concrete execution* of the binary inside the *concrete environment*

System Overview



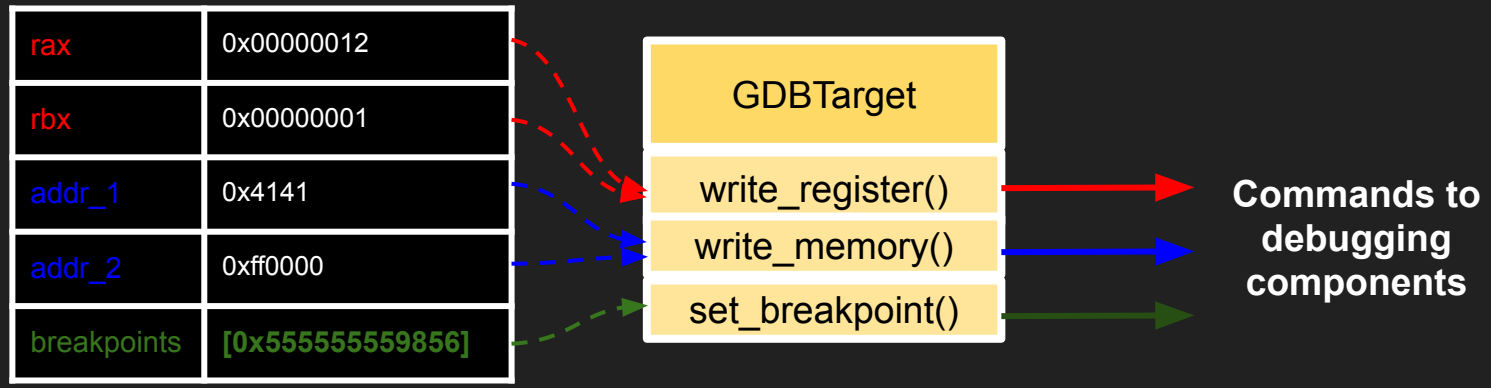
SimEngineConcrete

- Engine used by the **Symbion Exploration Technique** in order to step the *concrete execution* of the binary in the analysis environment.

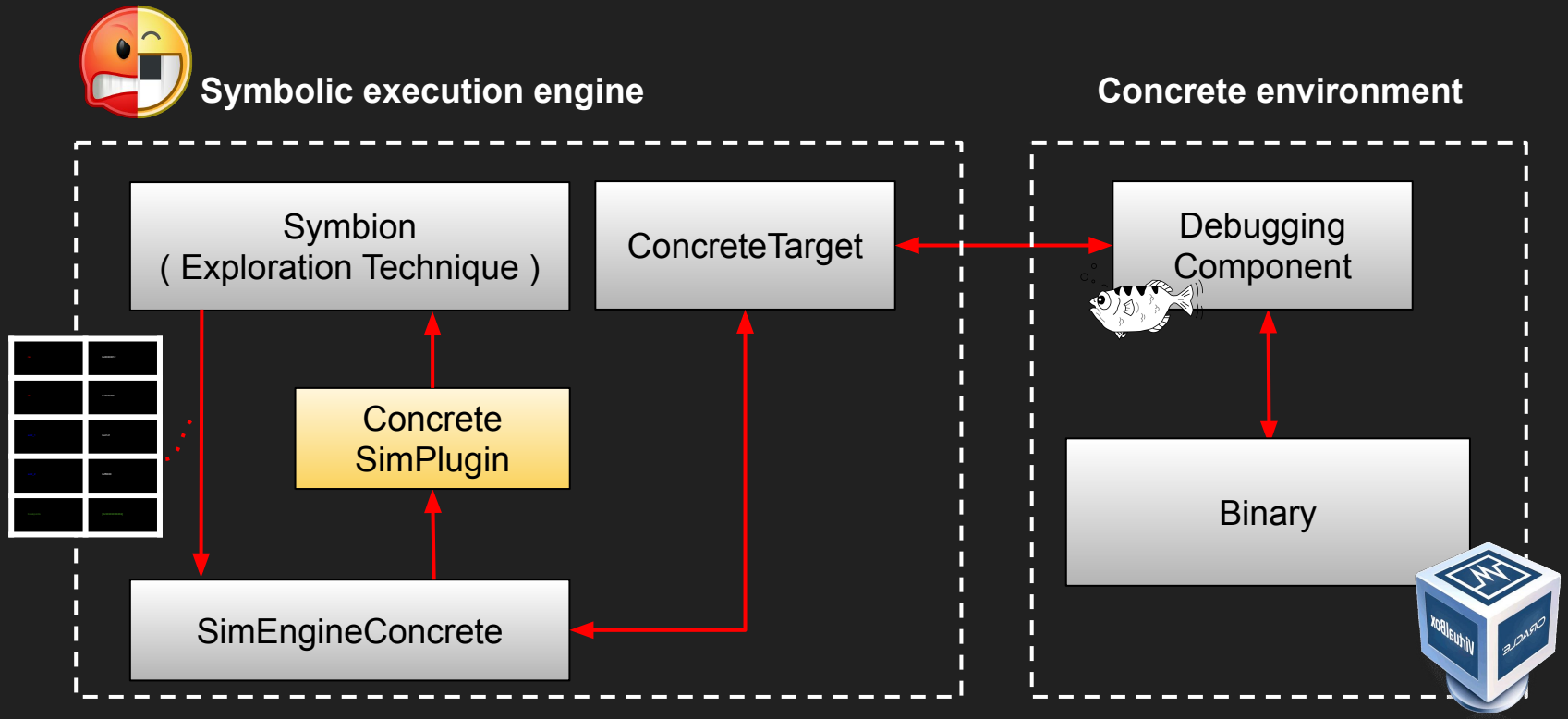
- Consists of two main parts:
  - **to_engine**( )
    - Handle the “jump” **inside** the concrete world!
  - **from_engine**( )
    - Handle the “jump” **outside** the concrete world leveraging the **Concrete SimPlugin**.

SimEngineConcrete

- **to_engine( ):**
  - Leverages the **ConcreteTarget** object to:
    - Set breakpoints on the concrete execution instance of the program.
    - Modify the concrete memory.
    - Resume the concrete execution by exploiting.

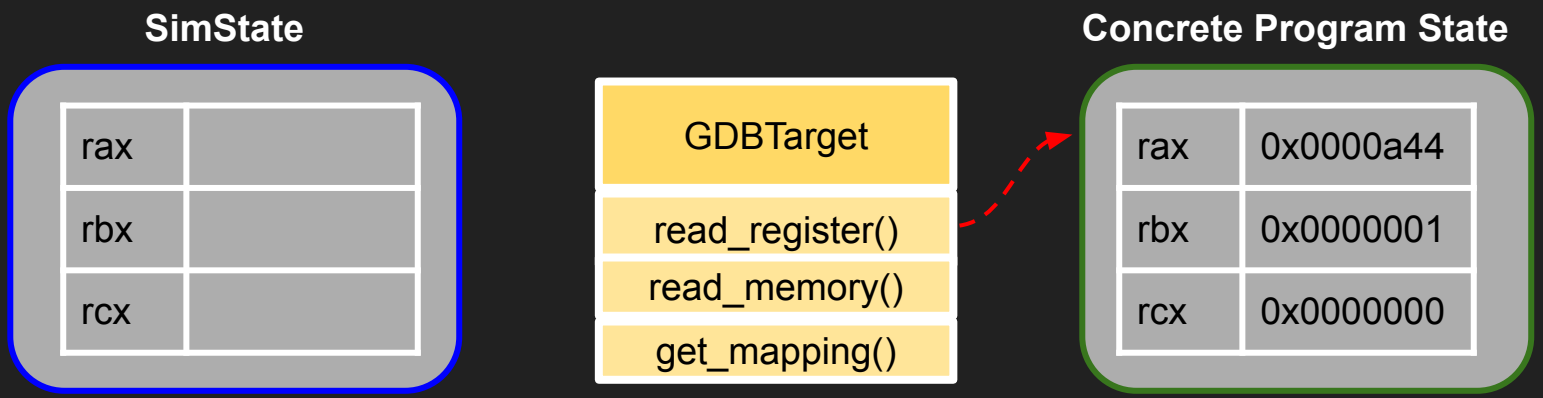


System Overview



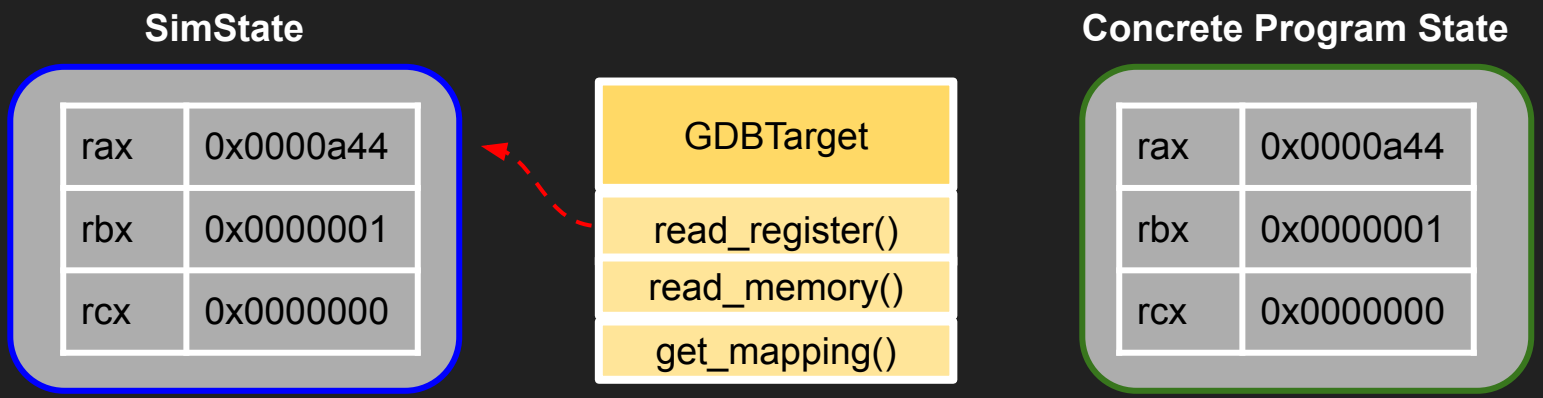
SimConcrete Plugin

- Synchronize the concrete process with **angr** and returns a a new SimState.
  - Copy values of ALL registers.



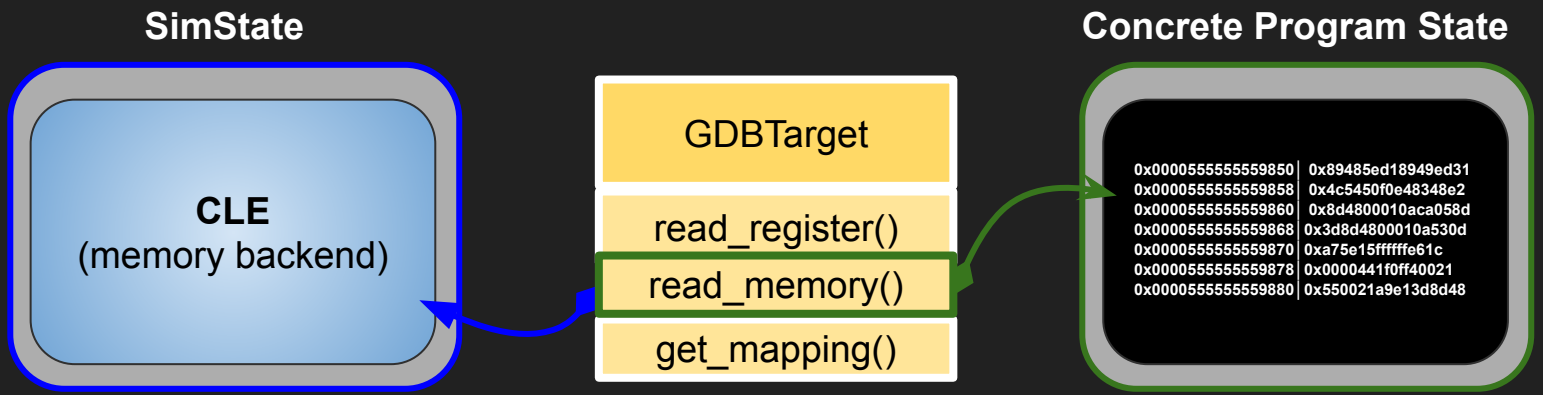
SimConcrete Plugin

- Synchronize the concrete process with **angr** and returns a a new SimState.
  - Copy values of ALL registers.



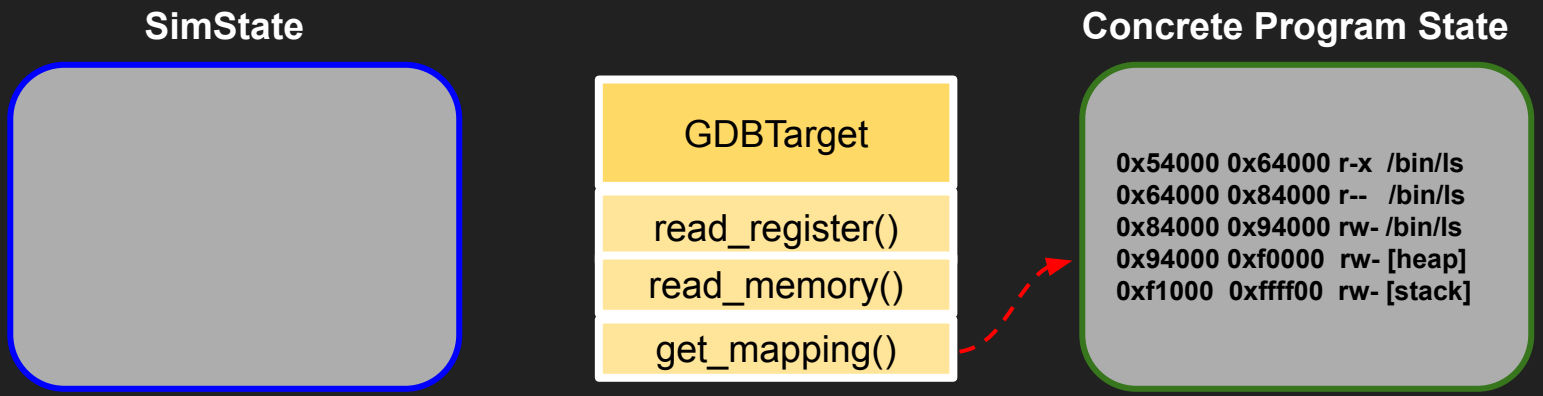
SimConcrete Plugin

- Synchronize the concrete process with **angr** and returns a a new SimState.
  - Copy values of ALL registers.
  - Hook new SimState memory backend to redirect reads to concrete process.



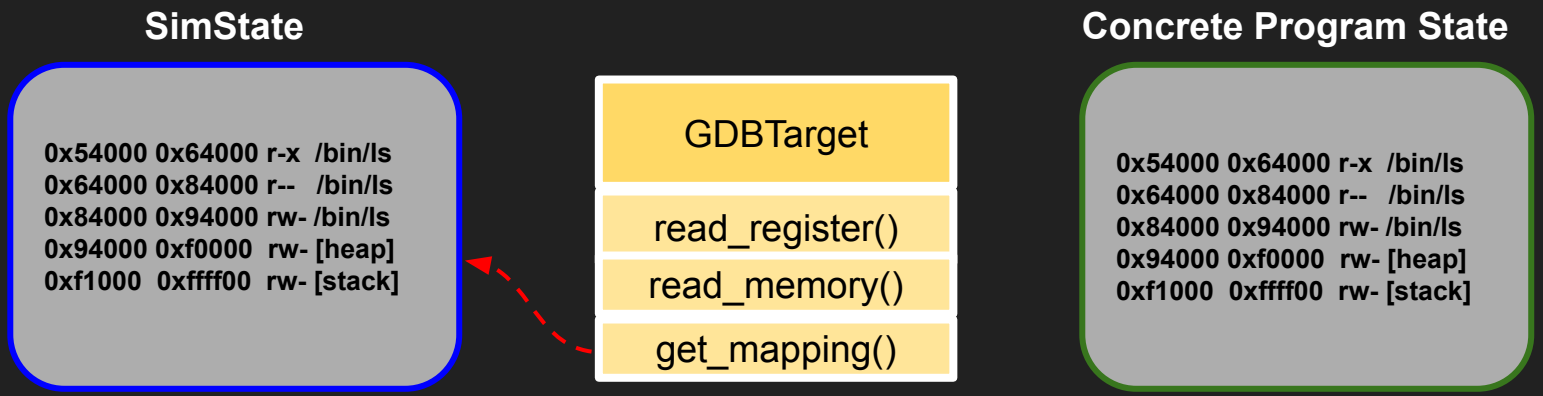
SimConcrete Plugin

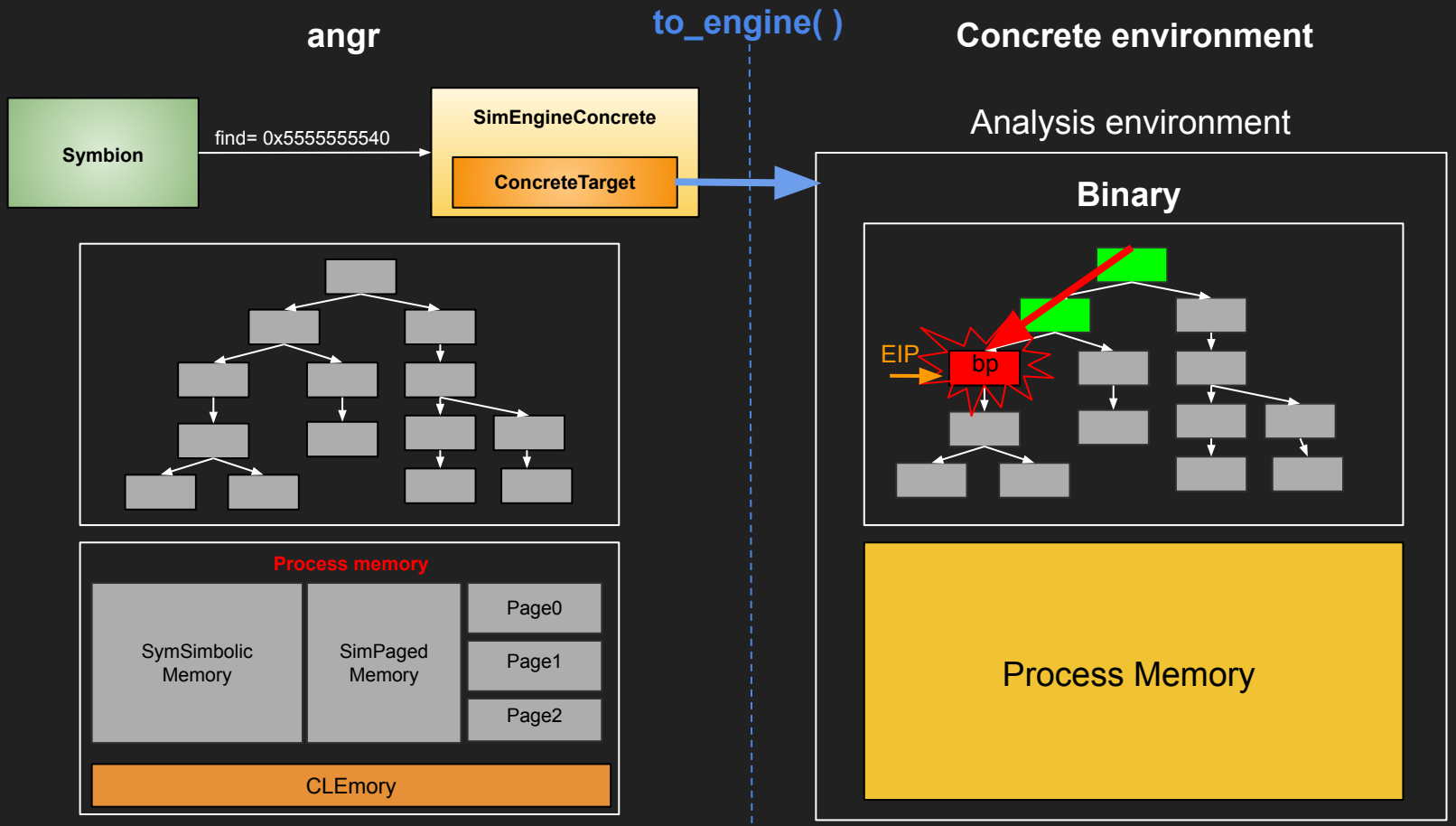
- Synchronize the concrete process with **angr** and returns a a new SimState.
  - Copy values of ALL registers.
  - Hook new SimState memory backend to redirect reads to concrete process.
  - Updates memory mapping information.

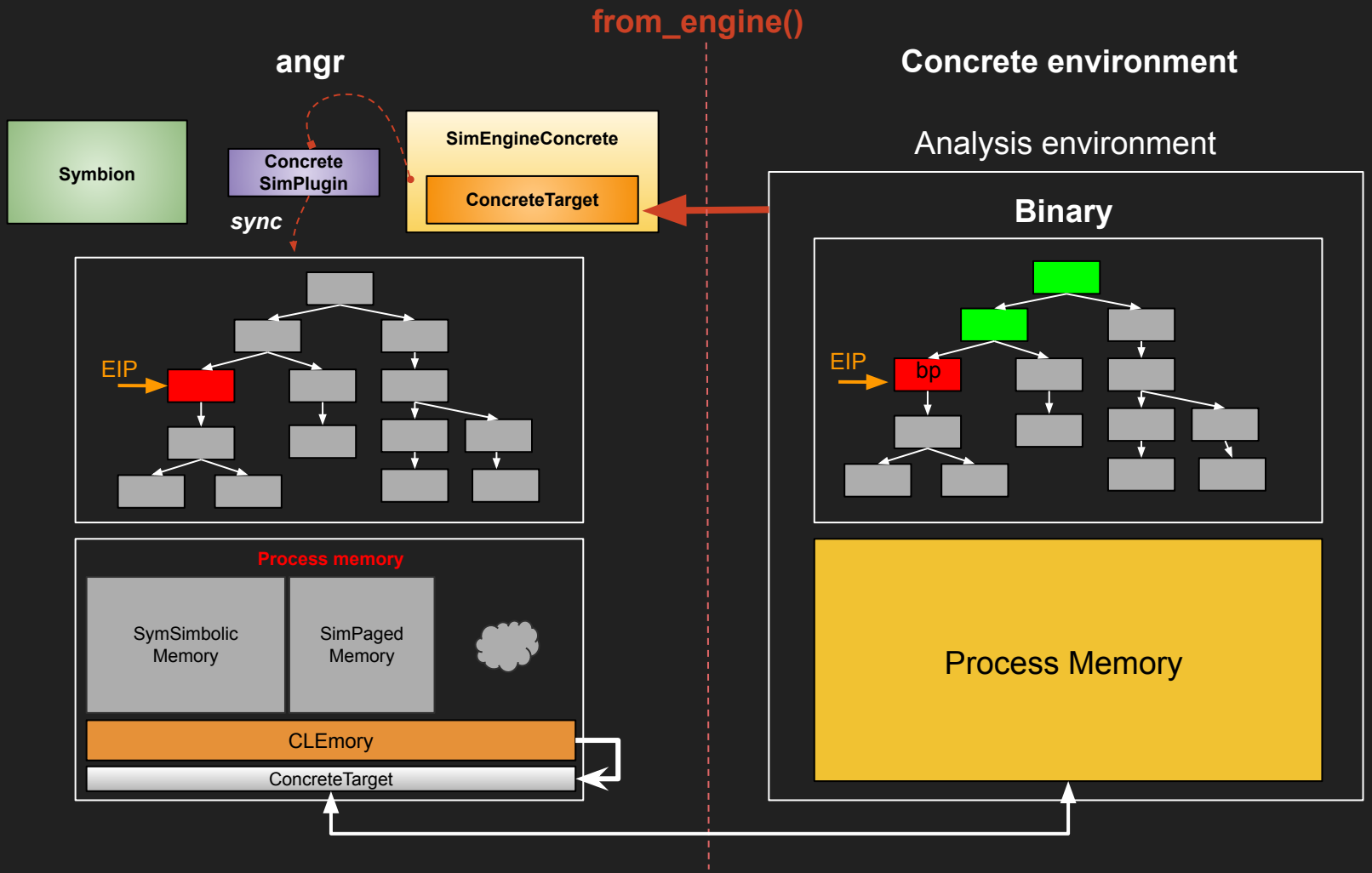


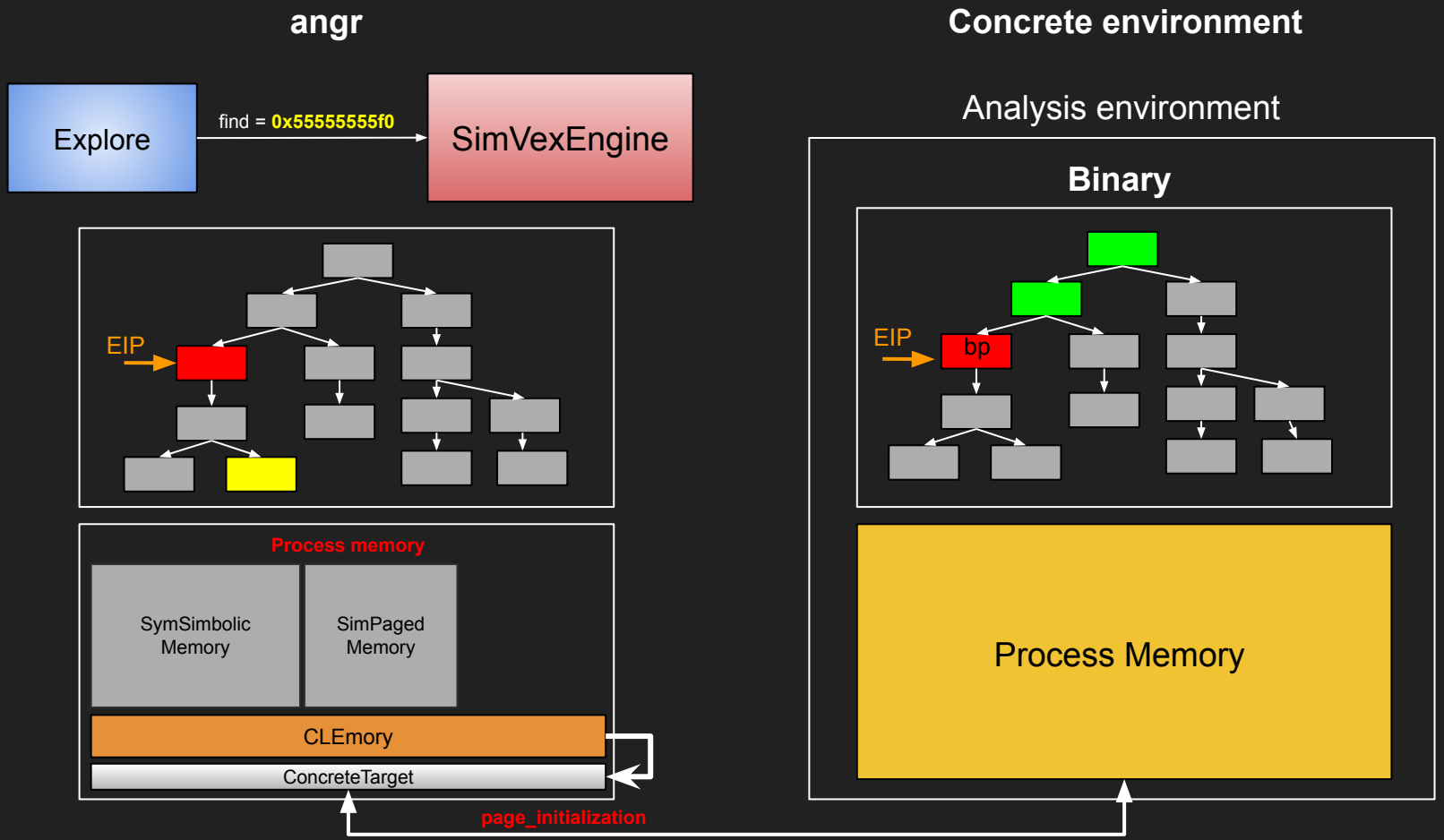
SimConcrete Plugin

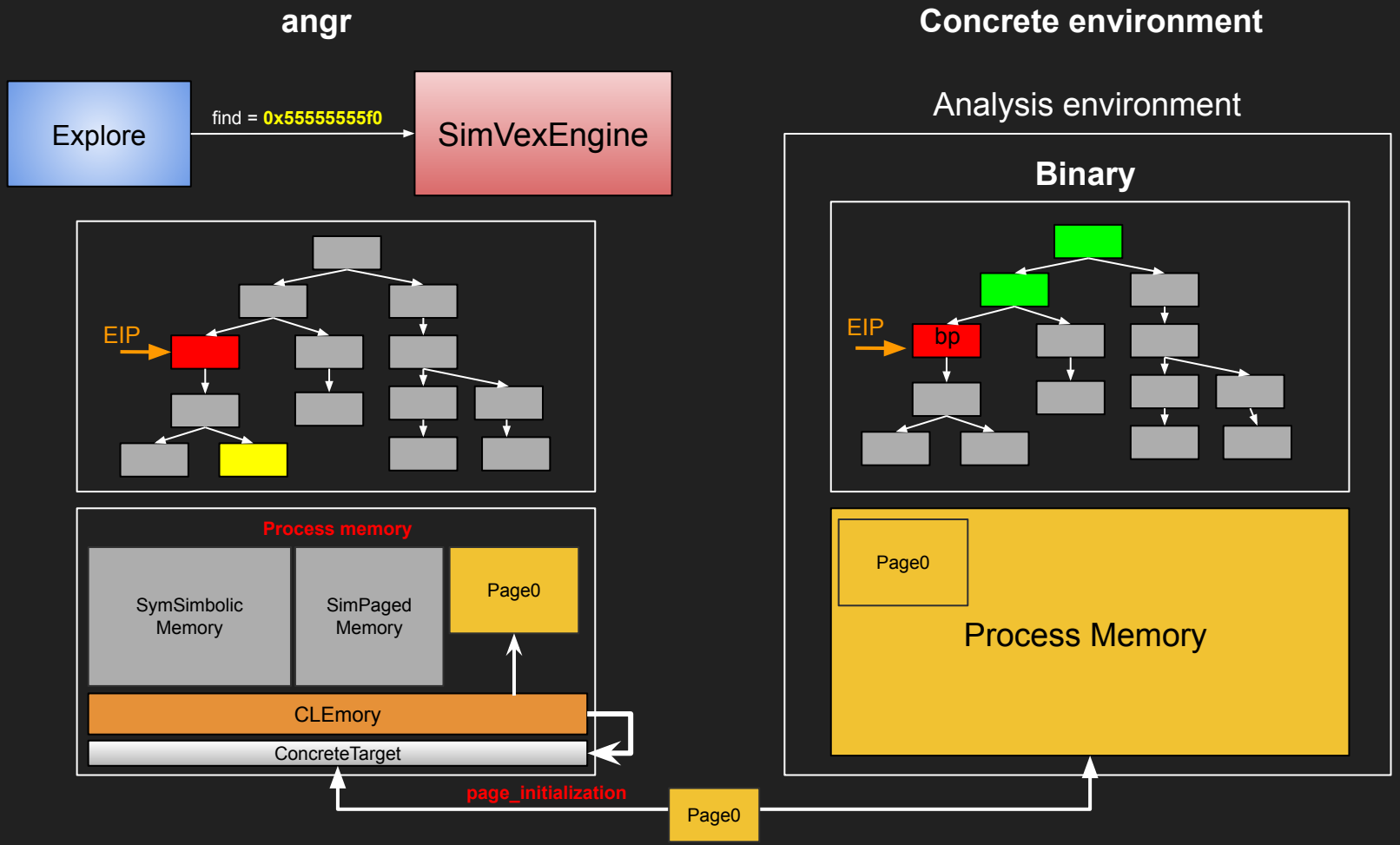
- Synchronize the concrete process with **angr** and returns a a new SimState.
  - Copy values of ALL registers.
  - Hook new SimState memory backend to redirect reads to concrete process.
  - Updates memory mapping information.

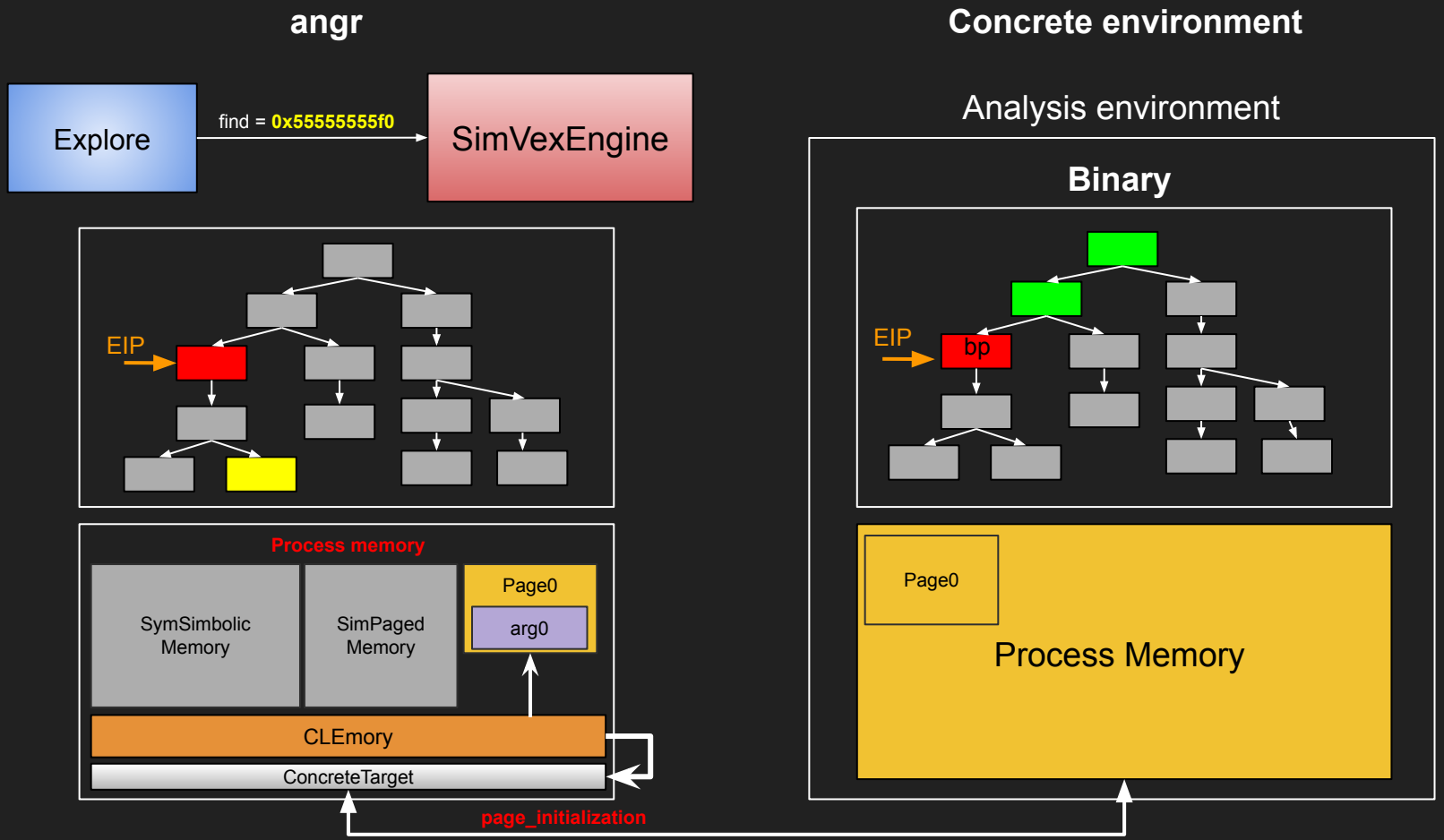


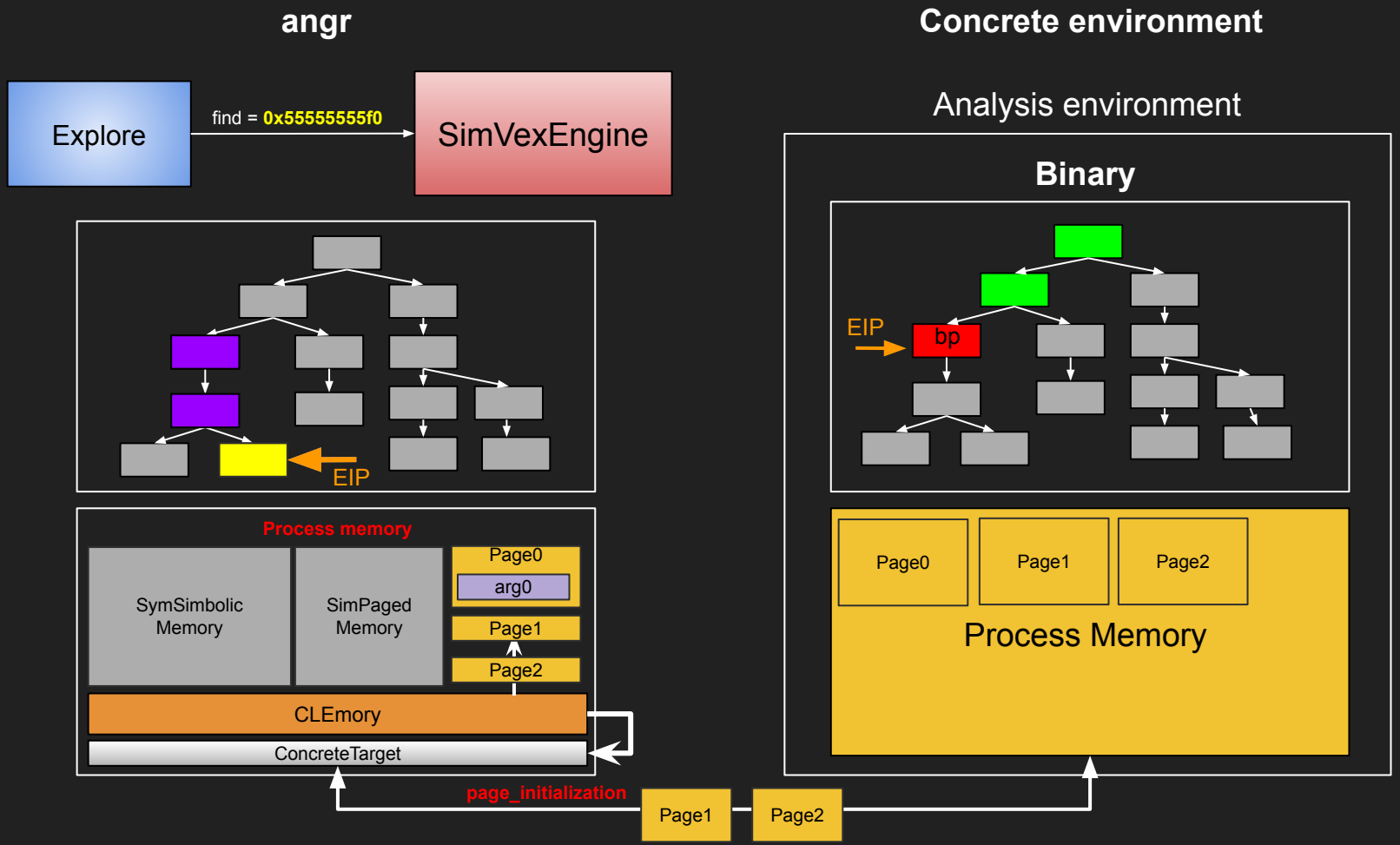

angr
to_engine( )
Concrete environment
Symbion
find= 0x5555555540
SimEngineConcrete
ConcreteTarget
Analysis environment
Binary
EIP
bp
Process memory
SymSimbolic Memory
SimPaged Memory
Page0
Page1
Page2
CLEmory
Process Memory

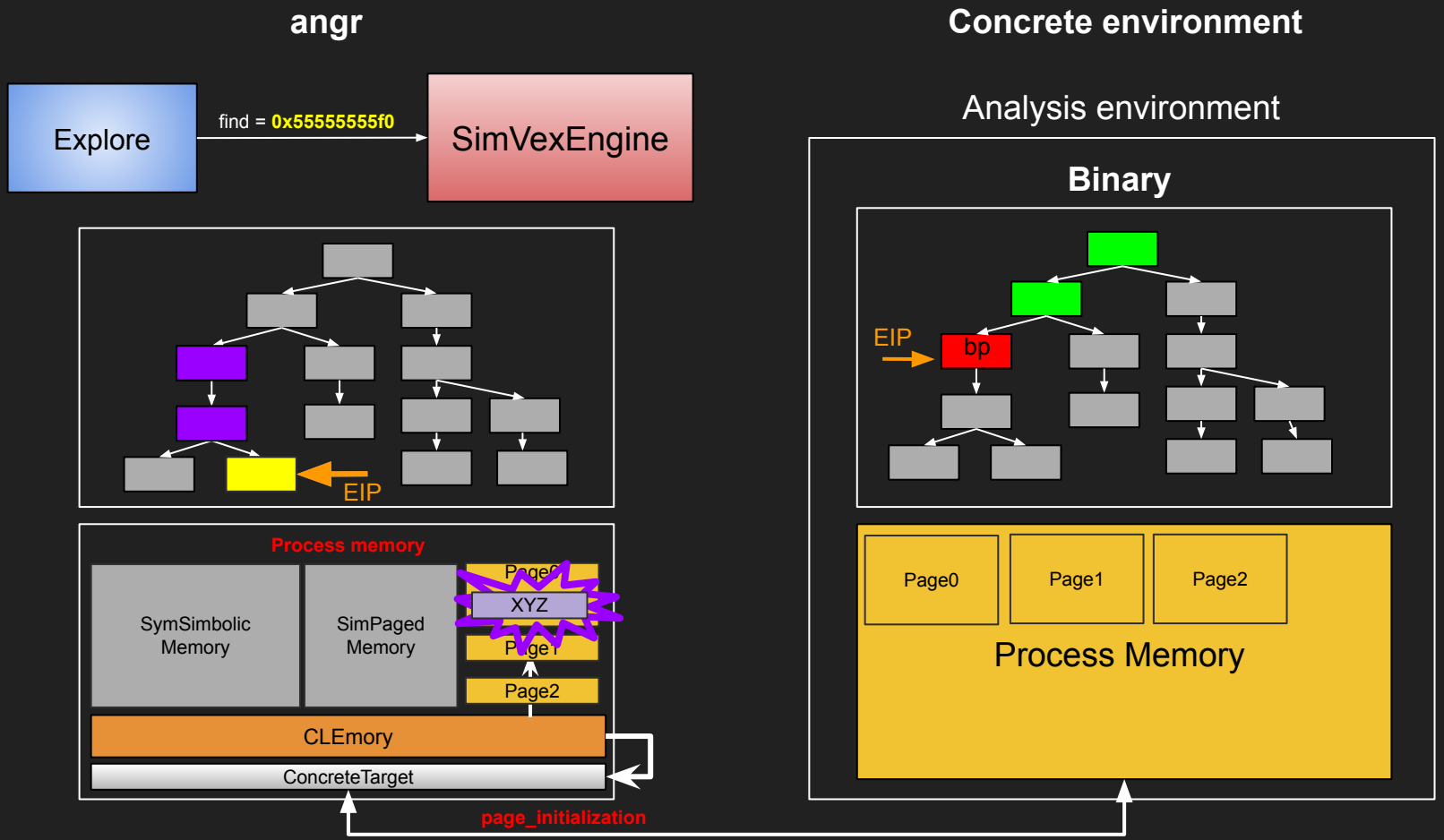

from_engine()
angr
Concrete environment
Symbion
Concrete SimPlugin
sync
SimEngineConcrete
ConcreteTarget
Analysis environment
Binary
EIP
EIP
bp
Process memory
SymSimbolic Memory
SimPaged Memory
CLEmory
ConcreteTarget
Process Memory

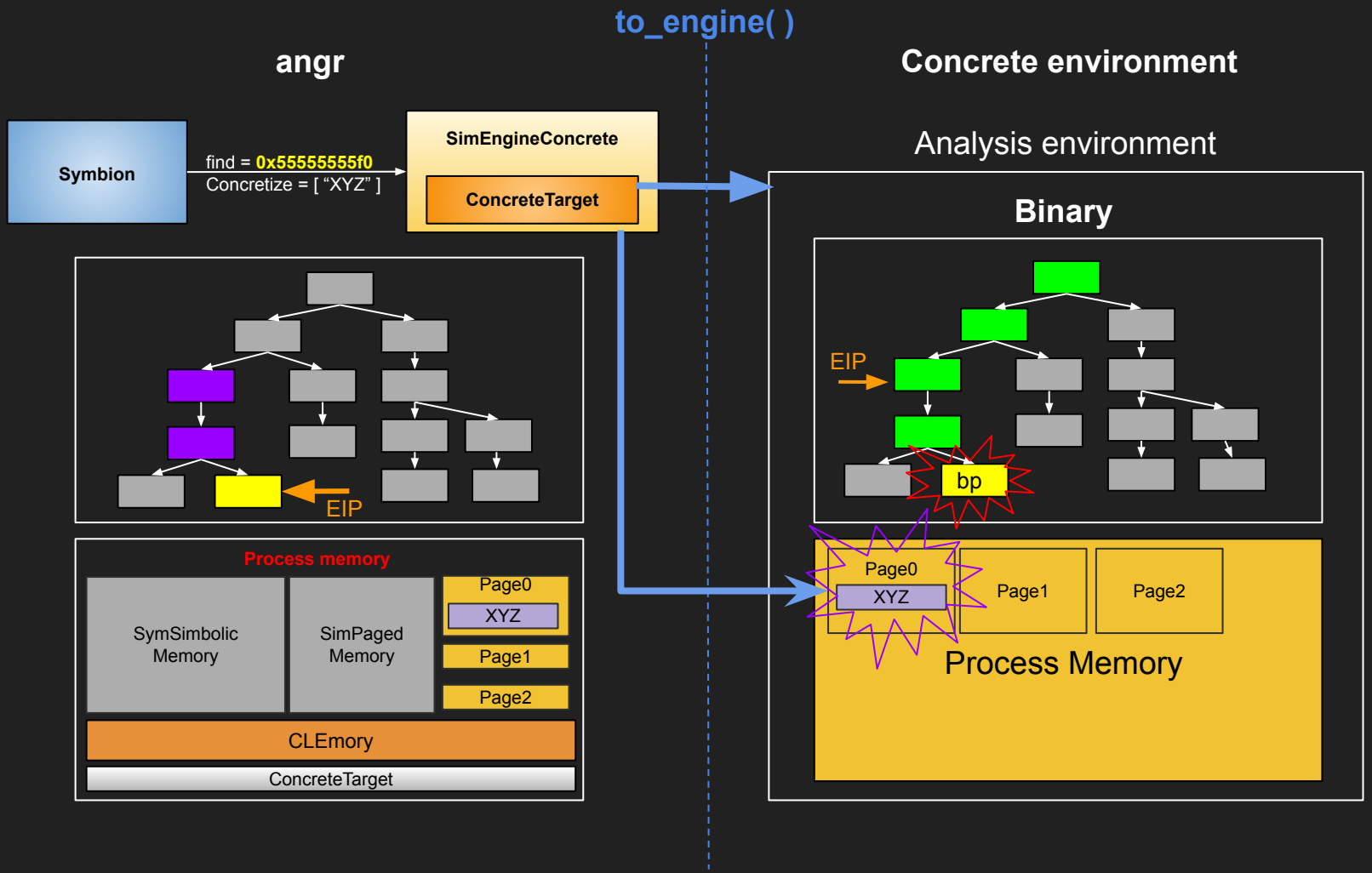

angr
Concrete environment
Explore
find = 0x55555555f0
SimVexEngine
Analysis environment
Binary
EIP
EIP
bp
Process memory
SymSimbolic Memory
SimPaged Memory
CLEmory
ConcreteTarget
Process Memory
page_initialization

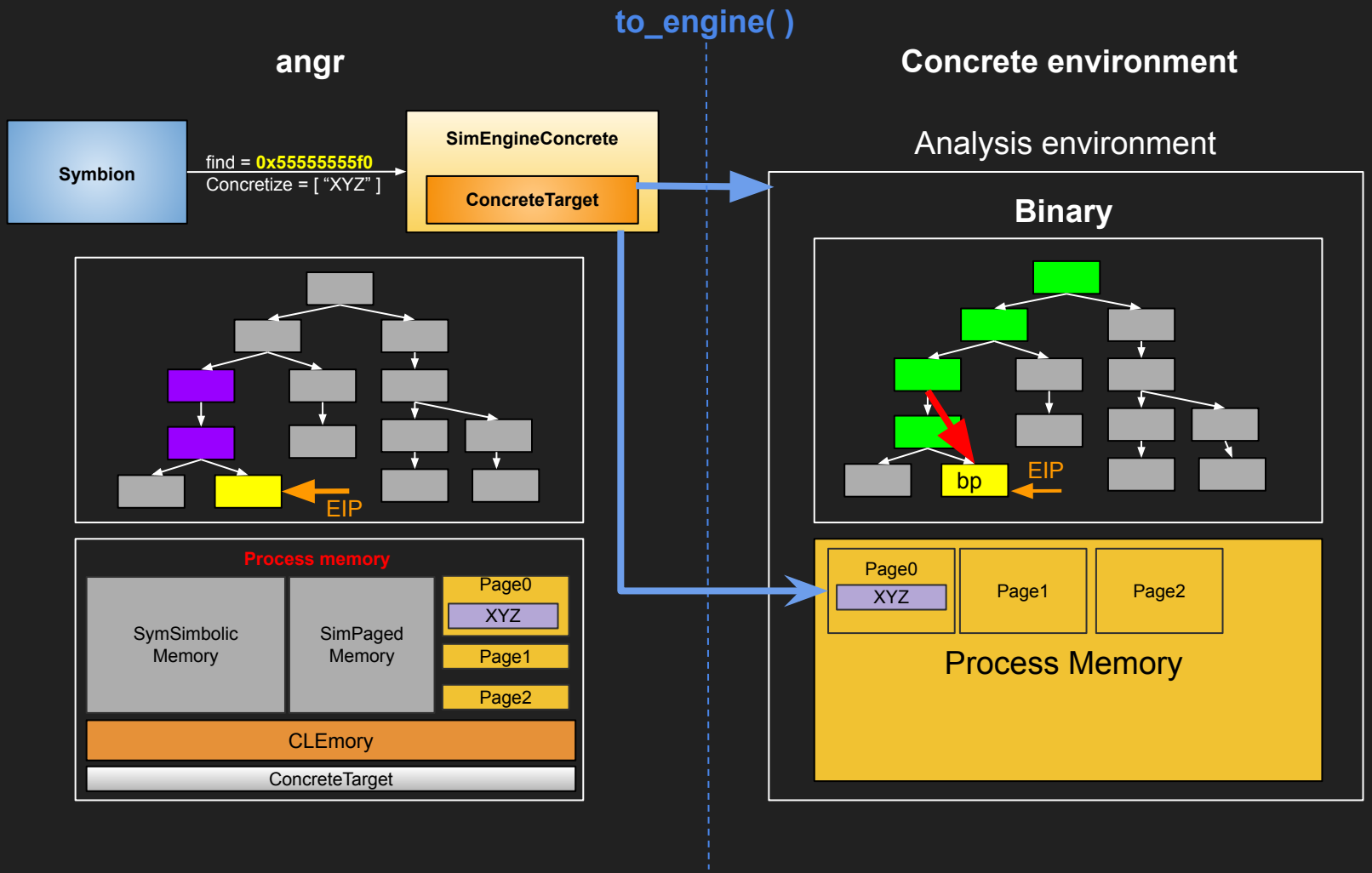

angr
Concrete environment
Explore
find = 0x55555555f0
SimVexEngine
Analysis environment
Binary
EIP
EIP
bp
Process memory
SymSimbolic Memory
SimPaged Memory
Page0
CLEmory
ConcreteTarget
Page0
Process Memory
page_initialization
Page0


angr
Concrete environment
Explore
find = 0x55555555f0
SimVexEngine
Analysis environment
Binary
EIP
EIP
bp
Process memory
SymSimbolic Memory
SimPaged Memory
Page0
arg0
CLEmory
ConcreteTarget
Page0
Process Memory
page_initialization

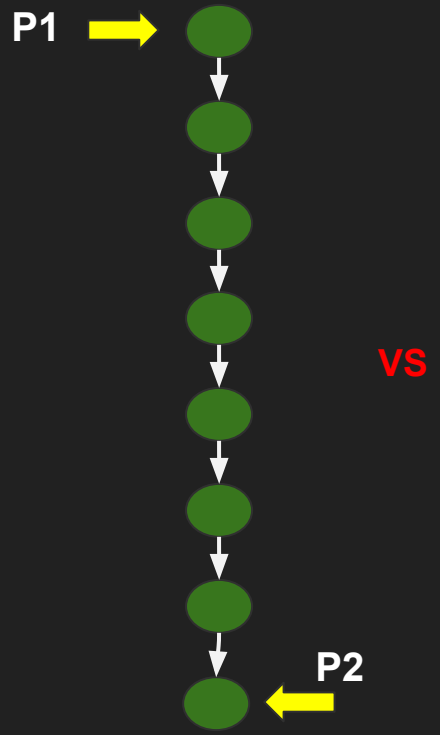

angr
Concrete environment
Explore
find = 0x55555555f0
SimVexEngine
Analysis environment
Binary
EIP
bp
EIP
Process memory
SymSimbolic Memory
SimPaged Memory
Page0
arg0
Page1
Page2
CLEmory
ConcreteTarget
Page0
Page1
Page2
Process Memory
page_initialization
Page1
Page2


angr
Concrete environment
Explore
find = 0x55555555f0
SimVexEngine
Analysis environment
Binary
EIP
bp
EIP
Process memory
SymSimbolic Memory
SimPaged Memory
XYZ
Page2
CLEmory
ConcreteTarget
Page0
Page1
Page2
Process Memory
page_initialization

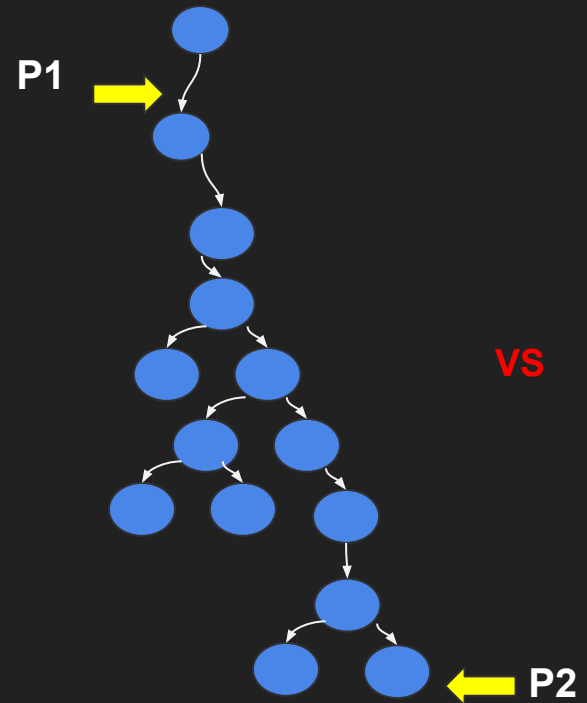

to_engine( )
angr
Concrete environment
Symbion
find = 0x55555555f0
Concretize = ["XYZ"]
SimEngineConcrete
ConcreteTarget
Analysis environment
Binary
EIP
bp
EIP
Process memory
SymSimbolic Memory
SimPaged Memory
Page0
XYZ
Page1
Page2
CLEmory
ConcreteTarget
Page0
XYZ
Page1
Page2
Process Memory


to_engine( )
angr
Concrete environment
Symbion
find = 0x55555555f0
Concretize = ["XYZ"]
SimEngineConcrete
ConcreteTarget
Analysis environment
Binary
EIP
bp
EIP
Process memory
SymSimbolic Memory
SimPaged Memory
Page0
XYZ
Page1
Page2
CLEmory
ConcreteTarget
Page0
XYZ
Page1
Page2
Process Memory

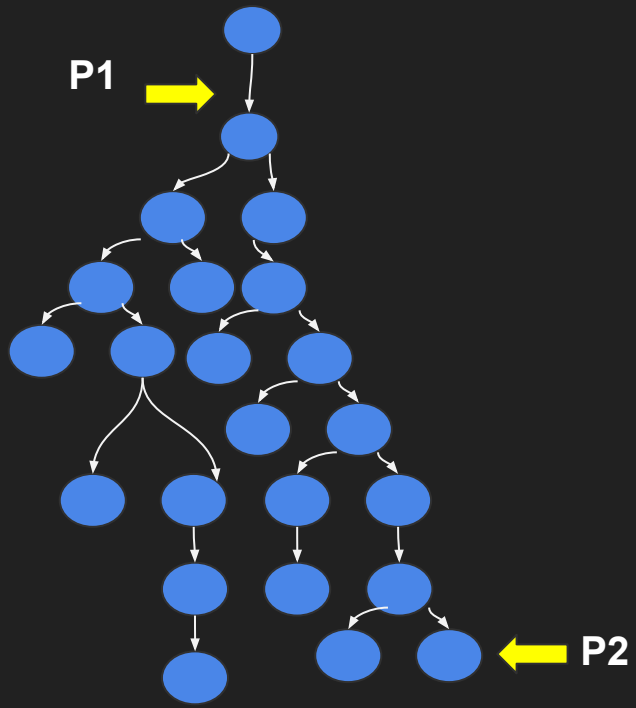
Approach



VS



VS



concrete execution

Interleaved symbolic execution

under-constrained symbolic exec.

Approach

- Idea: **Interleaving symbolic and concrete execution**
  - Concrete execute EOP → P1

Approach

- Idea: **Interleaving symbolic and concrete execution**
  - Concrete execute EOP → P1
  - Synchronize state at P1 inside symbolic engine

Approach

- Idea: **Interleaving symbolic and concrete execution**
  - Concrete execute EOP → P1
  - Synchronize state at P1 inside symbolic engine
  - User defines symbolic variables for analysis

Approach

- Idea: **Interleaving symbolic and concrete execution**
  - Concrete execute EOP → P1
  - Synchronize state at P1 inside symbolic engine
  - User defines symbolic variables for analysis
  - Symbolically execute P1 → P2

Approach

- Idea: **Interleaving symbolic and concrete execution**
  - Concrete execute EOP → P1
  - Synchronize state at P1 inside symbolic engine
  - User defines symbolic variables for analysis
  - Symbolically execute P1 → P2
  - Ask constraints solver for solutions

Approach

- Idea: **Interleaving symbolic and concrete execution**
  - Concrete execute EOP → P1
  - Synchronize state at P1 inside symbolic engine
  - User defines symbolic variables for analysis
  - Symbolically execute P1 → P2
  - Ask constraints solver for solutions
  - Overwrite solutions inside program's real memory

Approach

- Idea: **Interleaving symbolic and concrete execution**
  - Concrete execute EOP → P1
  - Synchronize state at P1 inside symbolic engine
  - User defines symbolic variables for analysis
  - Symbolically execute P1 → P2
  - Ask constraints solver for solutions
  - Overwrite solutions inside program's real memory
  - Concrete execute P1 → P2

Approach

- Idea: **Interleaving symbolic and concrete execution**
  - Concrete execute EOP → P1
  - Synchronize state at P1 inside symbolic engine
  - User defines symbolic variables for analysis
  - Symbolically execute P1 → P2
  - Ask constraints solver for solutions
  - Overwrite solutions inside program's real memory
  - Concrete execute P1 → P2
  - Repeat!