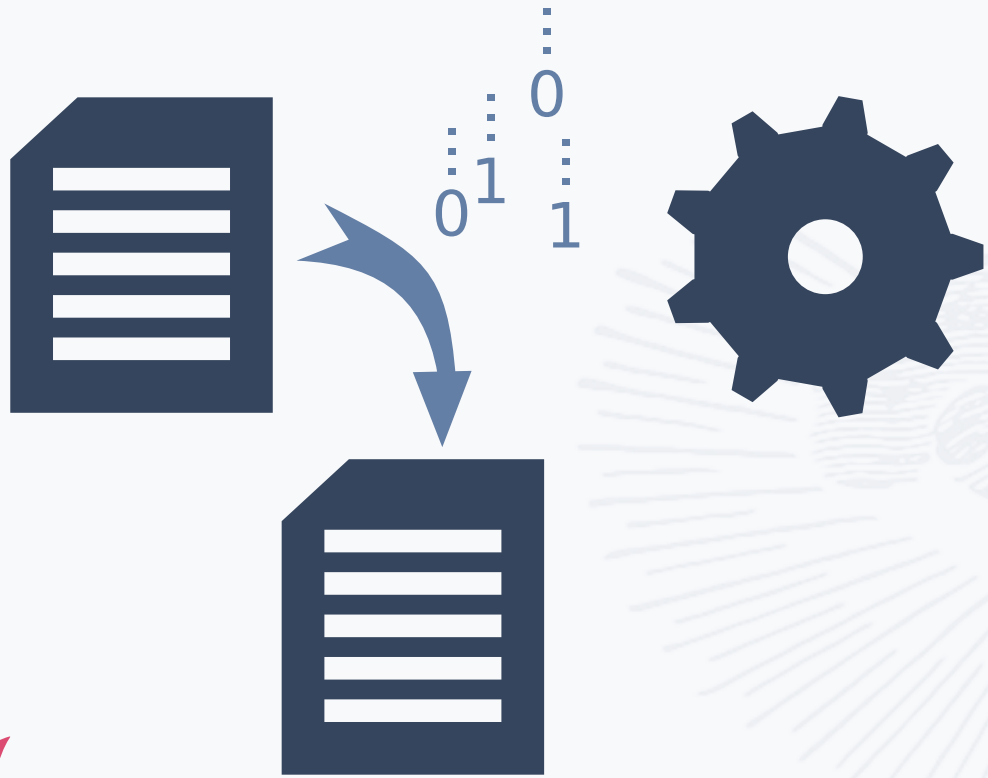
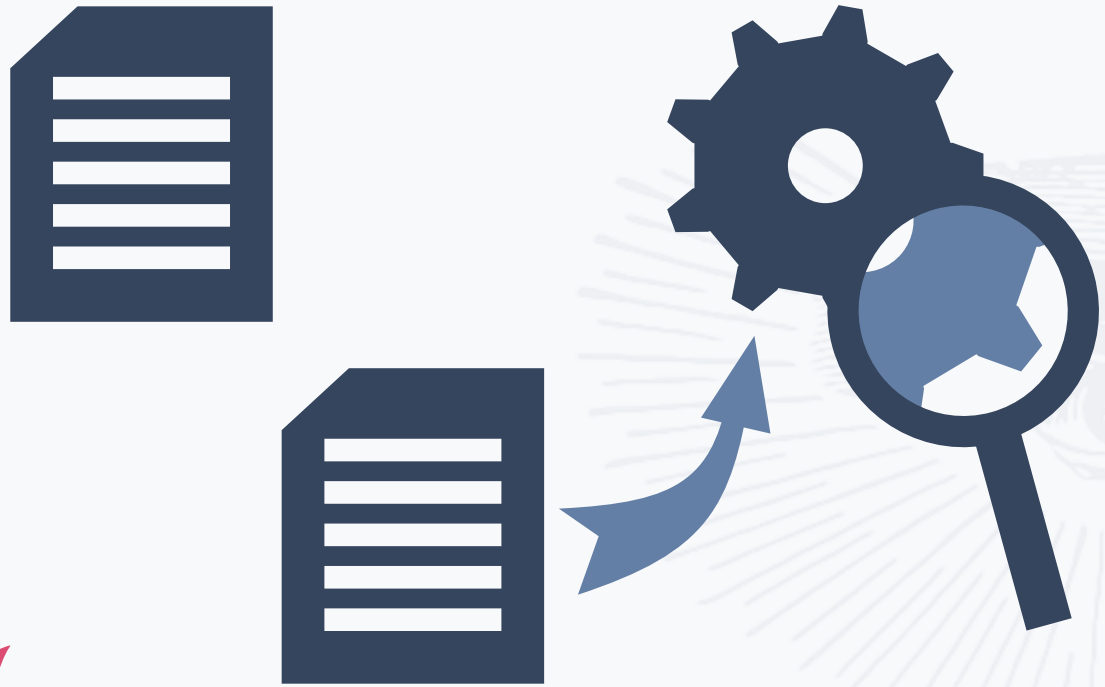
REDQUEEN

Fuzzing with Input-to-State Correspondence

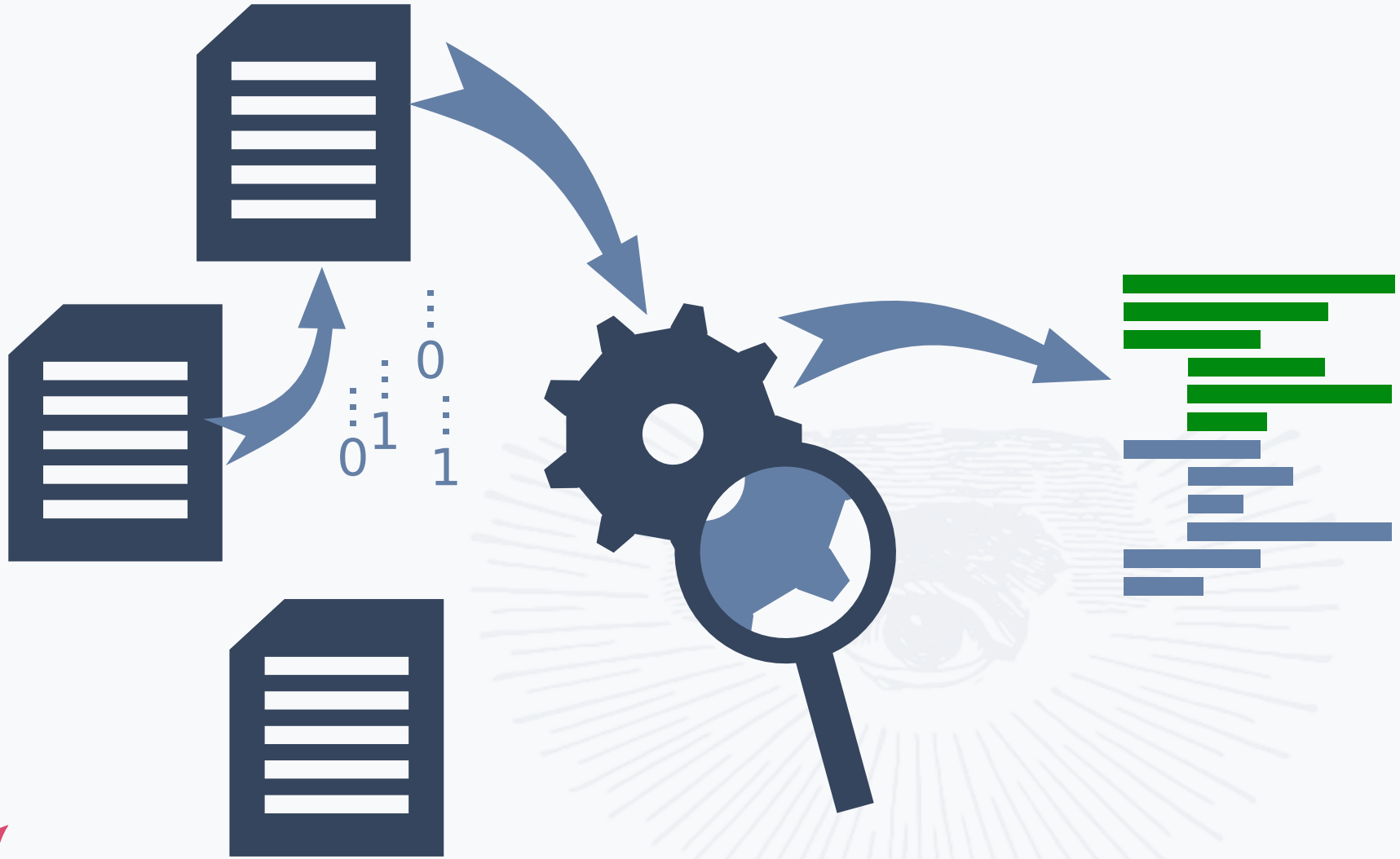
Cornelius Aschermann, Sergej Schumilo, Tim Blazytko, Robert Gawlik and Thorsten Holz
Ruhr-Universität Bochum

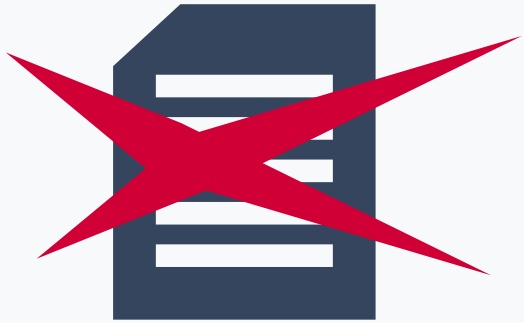
AFL

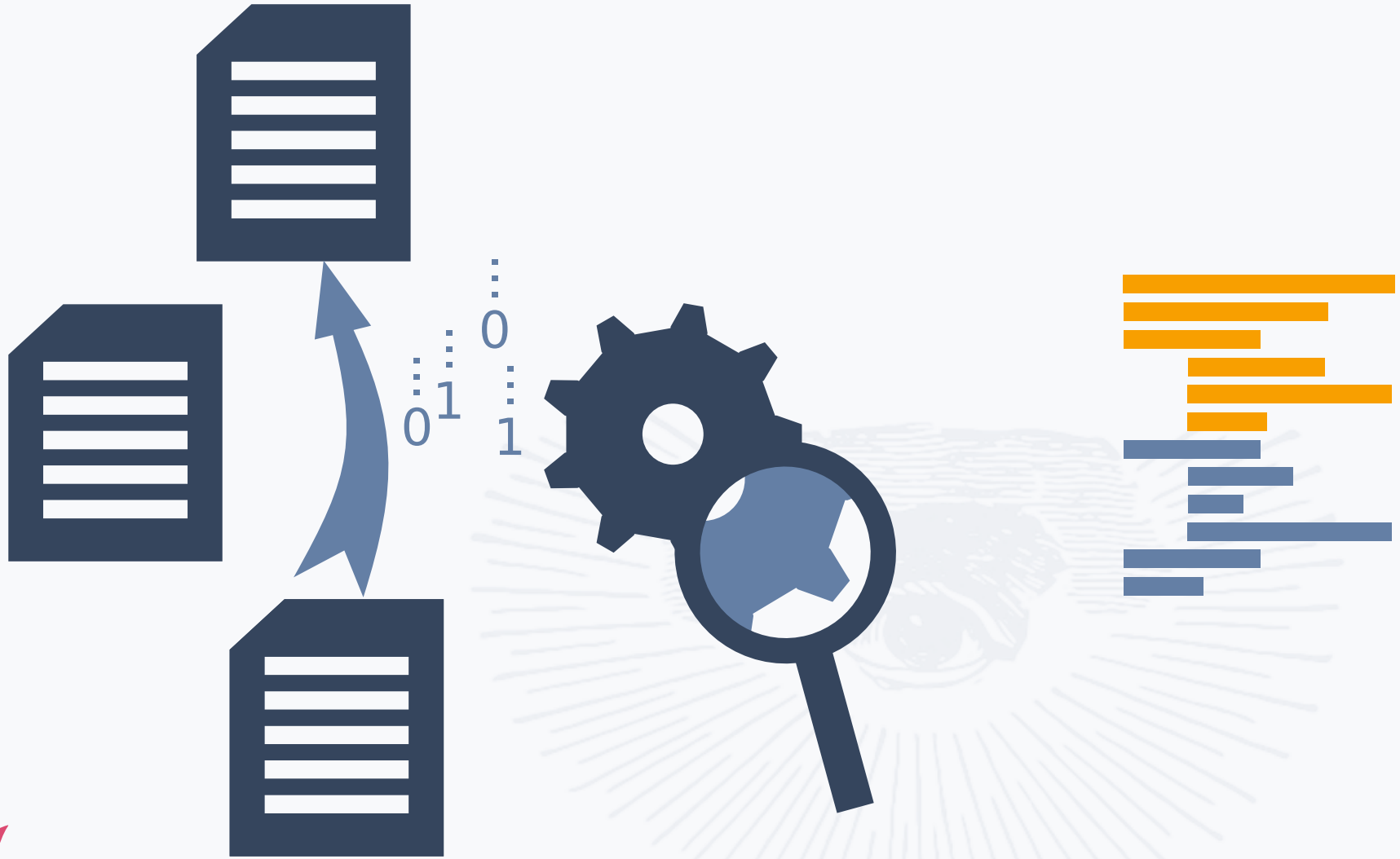

0
1
0
1

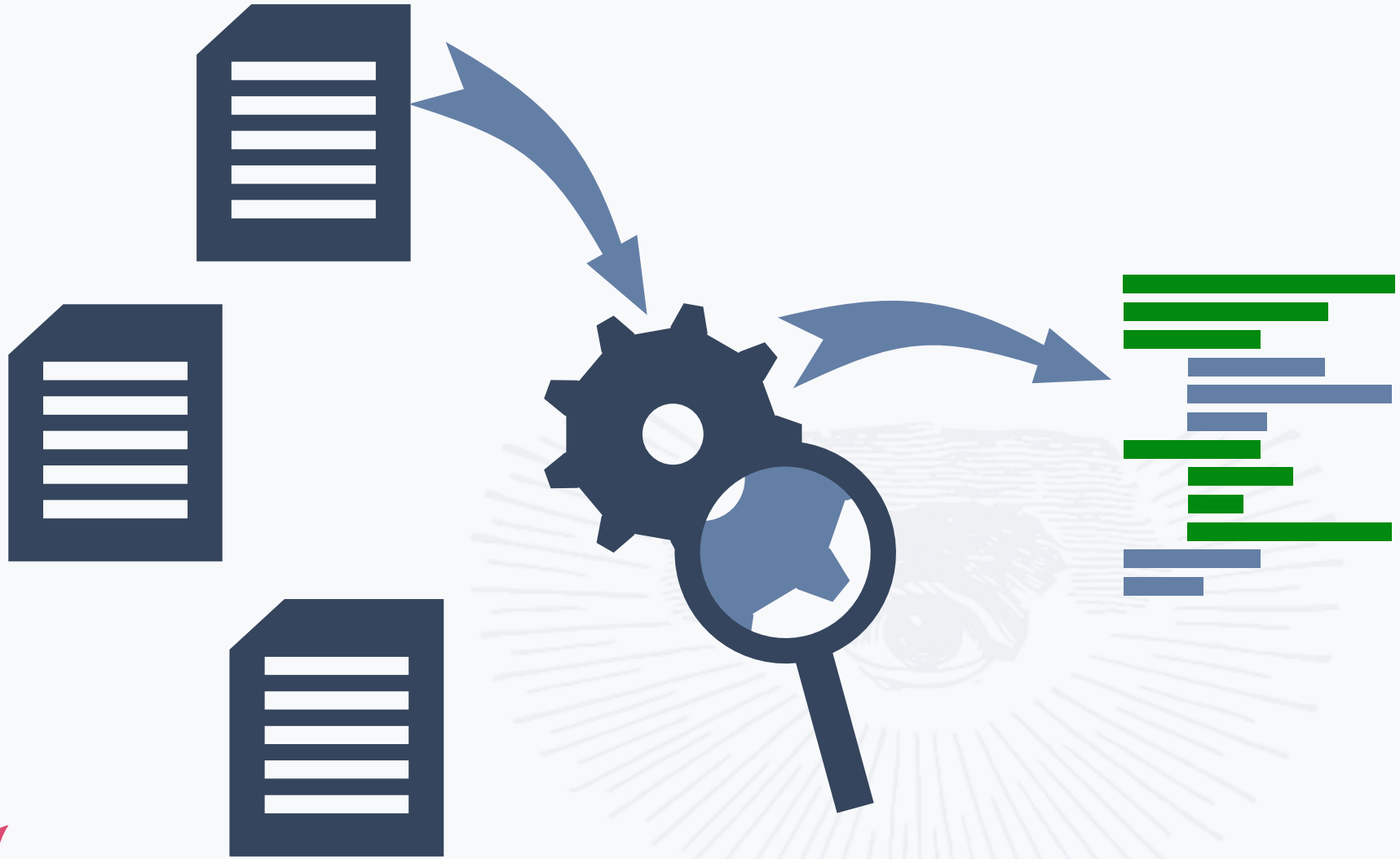



0
1
0
1


0
1
0
1




0
1
0
1



```
/* magic number */
if( u64(input) == u64("MAGICHDR"))
    bug(1);
```


```
/* magic number */
if( u64(input) == u64("MAGICHDR"))
    bug(1);
```

Unlikely

```
/* nested checksum */
if( u64(input) == hash(input[8..len]) )
    if( u64(input+8) == hash(input[16..len]) )
        if( input[16] == 'R' && input[17] == 'Q')
            bug(2);
```

```
/* nested checksum */
if( u64(input) == hash(input[8..len]) )
    if( u64(input+8) == hash(input[16..len]) )
        if( input[16] == 'R' && input[17] == 'Q')
            bug(2);
```

\x46\x01\0\0\0\0\0\0xa3\0\0\0\0\0\0\0RQ

```
/* nested checksum */
if( u64(input) == hash(input[8..len]) )
    if( u64(input+8) == hash(input[16..len]) )
        if( input[16] == 'R' && input[17] == 'Q')
            bug(2);
```

\x46\x01\0\0\0\0\0\0\xa3\0\0\0\0\0\0\0RQ

```
/* nested checksum */
if( u64(input) == hash(input[8..len]) )
    if( u64(input+8) == hash(input[16..len]) )
        if( input[16] == 'R' && input[17] == 'Q')
            bug(2);
```

\x46\x01\0\0\0\0\0\0\xa3\0\0\0\0\0\0\0RQ

```
/* nested checksum */
if( u64(input) == hash(input[8..len]) )
    if( u64(input+8) == hash(input[16..len]) )
        if( input[16] == 'R' && input[17] == 'Q')
            bug(2);
```

\x46\x01\0\0\0\0\0\0xa3\0\0\0\0\0\0\0RQ

easy

```
/* nested checksum */
if( u64(input) == hash(input[8..len]) )
    if( u64(input+8) == hash(input[16..len]) )
        if( input[16] == 'R' && input[17] == 'Q')
            bug(2);
```

\x46\x01\0\0\0\0\0\xa3\0\0\0\0\0\0RQ

easy

Approaches	Goals
AFL Laf-Intel	
Angora	
Driller	
Flayer	
KLEE	
Taintscope	
T-Fuzz	
Vuzzer	

Approaches	Goals
AFL Laf-Intel Angora Driller Flayer KLEE Taintscope T-Fuzz Vuzzer	Magic Numbers

Approaches	Goals
~~AFL~~ ~~LAF~~ ~~INTEL~~	Magic Numbers
~~ANGORA~~	Nested Hashes
~~DRILLER~~	
FLAYER	
KLEE	
TAINTSCOPE	
T-FUZZ	
~~VUZZER~~	

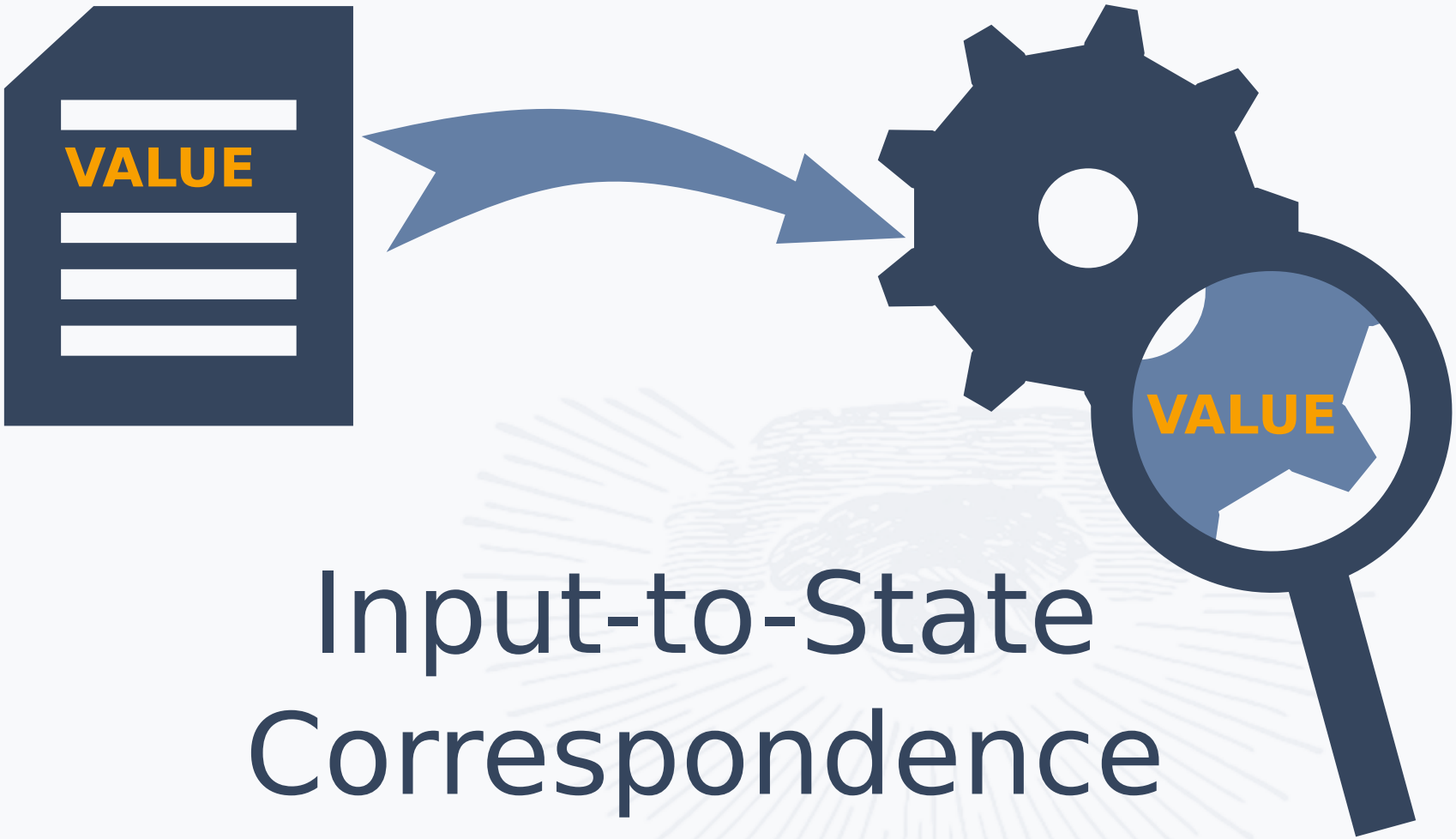
Approaches	Goals
~~AFL~~ ~~LAF~~ ~~INTEL~~	Magic Numbers
~~ANGORA~~	Nested Hashes
~~DRILLER~~	Automatic
~~FLAYER~~	
KLEE	
TAINTSCOPE	
T-FUZZ	
~~VUZZER~~	

Approaches	Goals
~~AFL~~ ~~LAF~~ ~~INTEL~~	Magic Numbers
~~ANGORA~~	Nested Hashes
~~DRILLER~~	Automatic
~~FLAYER~~	Binary Only
~~KLEE~~	
TAINTSCOPE	
T-FUZZ	
~~VUZZER~~	

Approaches	Goals
~~AFL~~ ~~LAF~~ ~~INTEL~~	Magic Numbers
~~ANGORA~~	Nested Hashes
~~DRILLER~~	Automatic
~~FLAYER~~	Binary Only
~~KLEE~~	No Env. Model
~~TAINTSCOPE~~	
~~T-FUZZ~~	
~~VUZZER~~	

IDEA

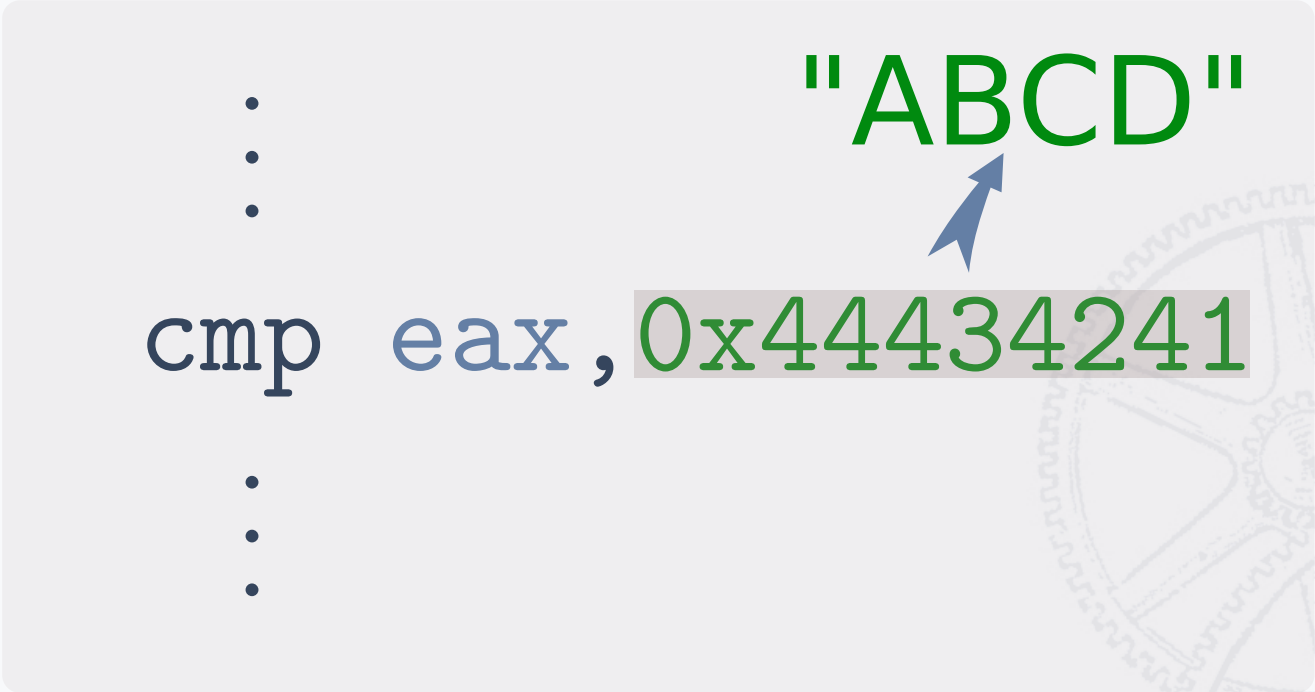
Input-to-State Correspondence



Input-to-State Correspondence

DESIGN

```
    .
    .
    .
cmp eax,0x44434241
    .
    .
    .
```

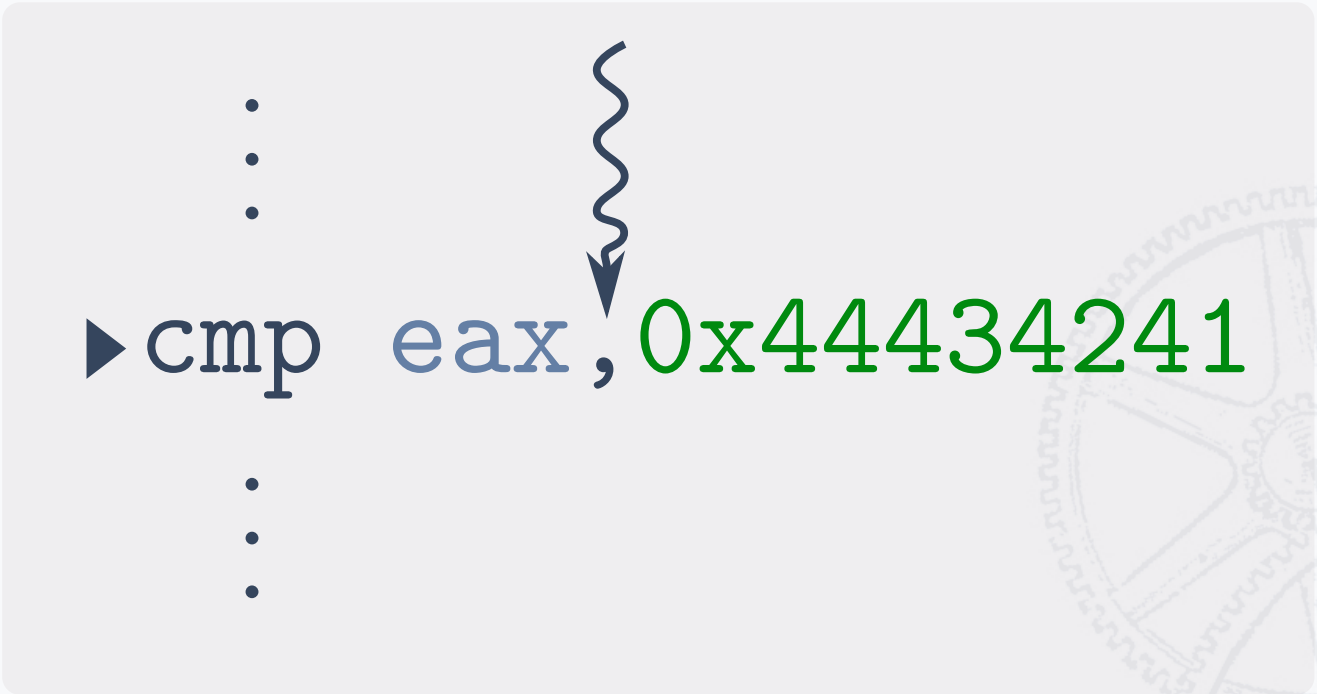

"ABCD"
cmp eax,0x44434241

```
    .
    .
    .
cmp eax,0x44434241
    .
    .
    .
```

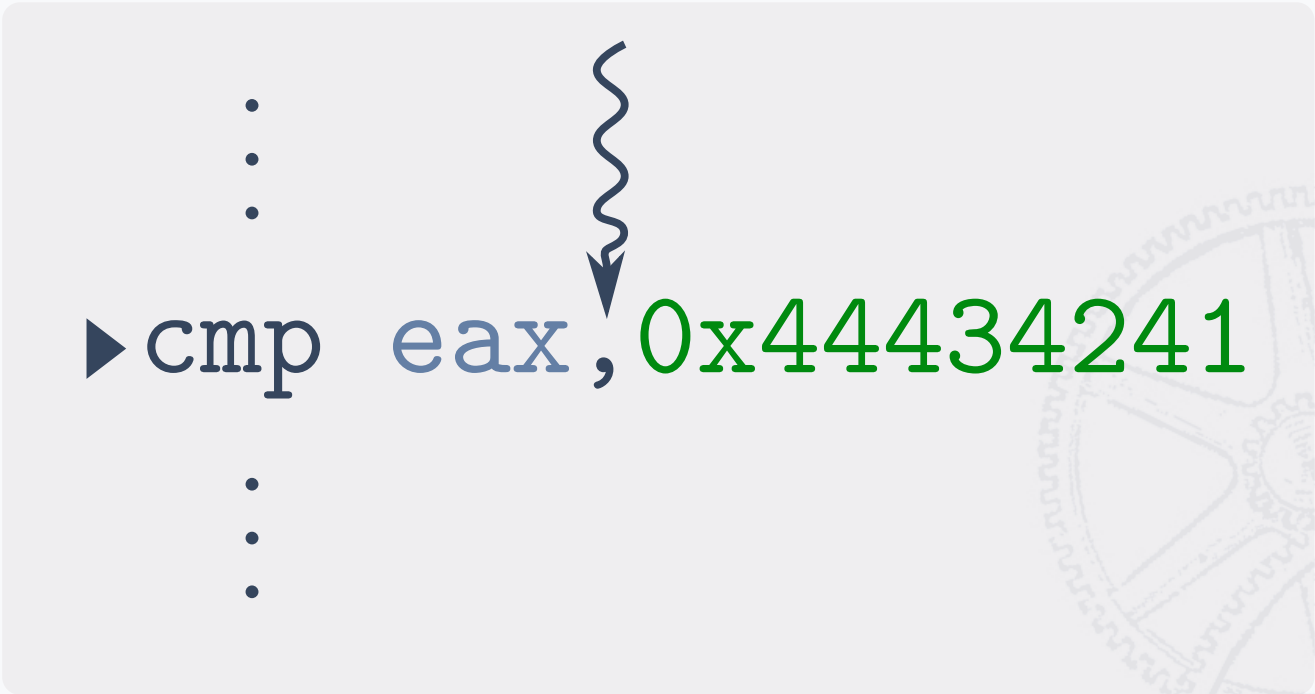
Input: "SEEDVALUE"

```
    .
    .
    .
cmp eax,0x44434241
    .
    .
    .
```

Input: "SEEDVALUE"

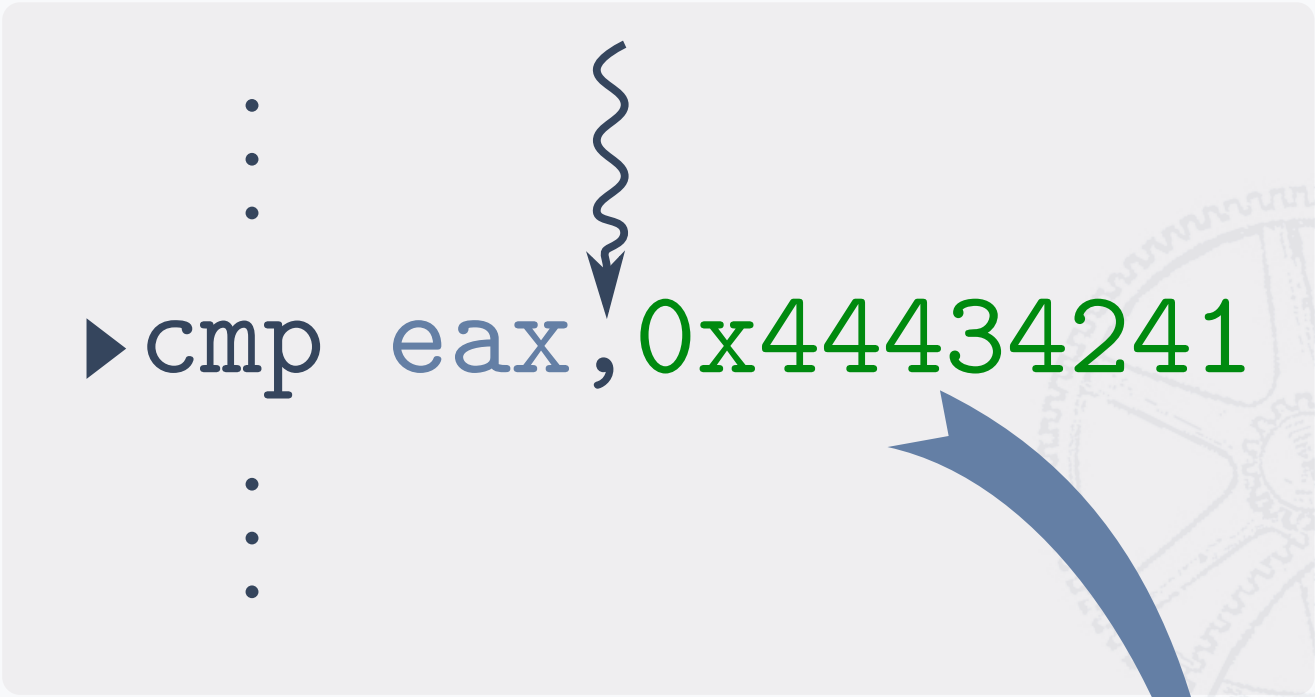


Input: "SEEDVALUE"



Observe: `eax = 0x554c4156`

Input: "SEEDVALUE"



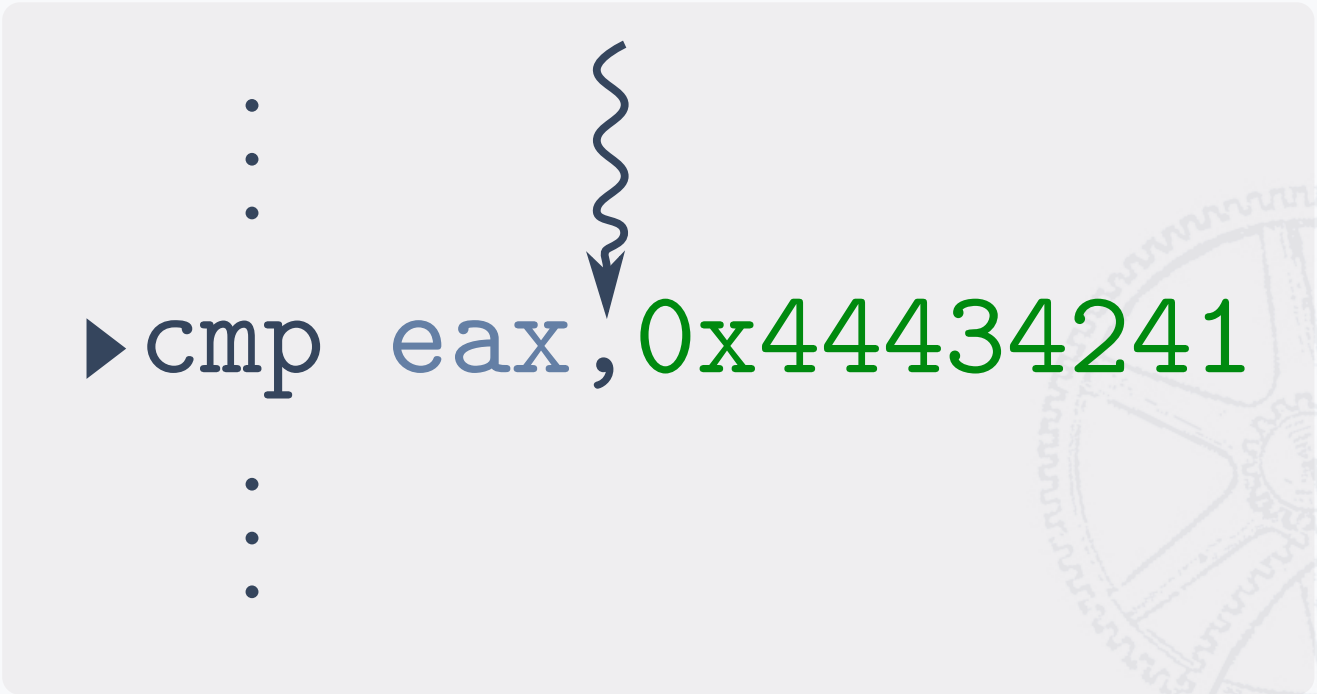
Observe: eax = 0x554c4156

Replace(0x554c4156, 0x44434241)

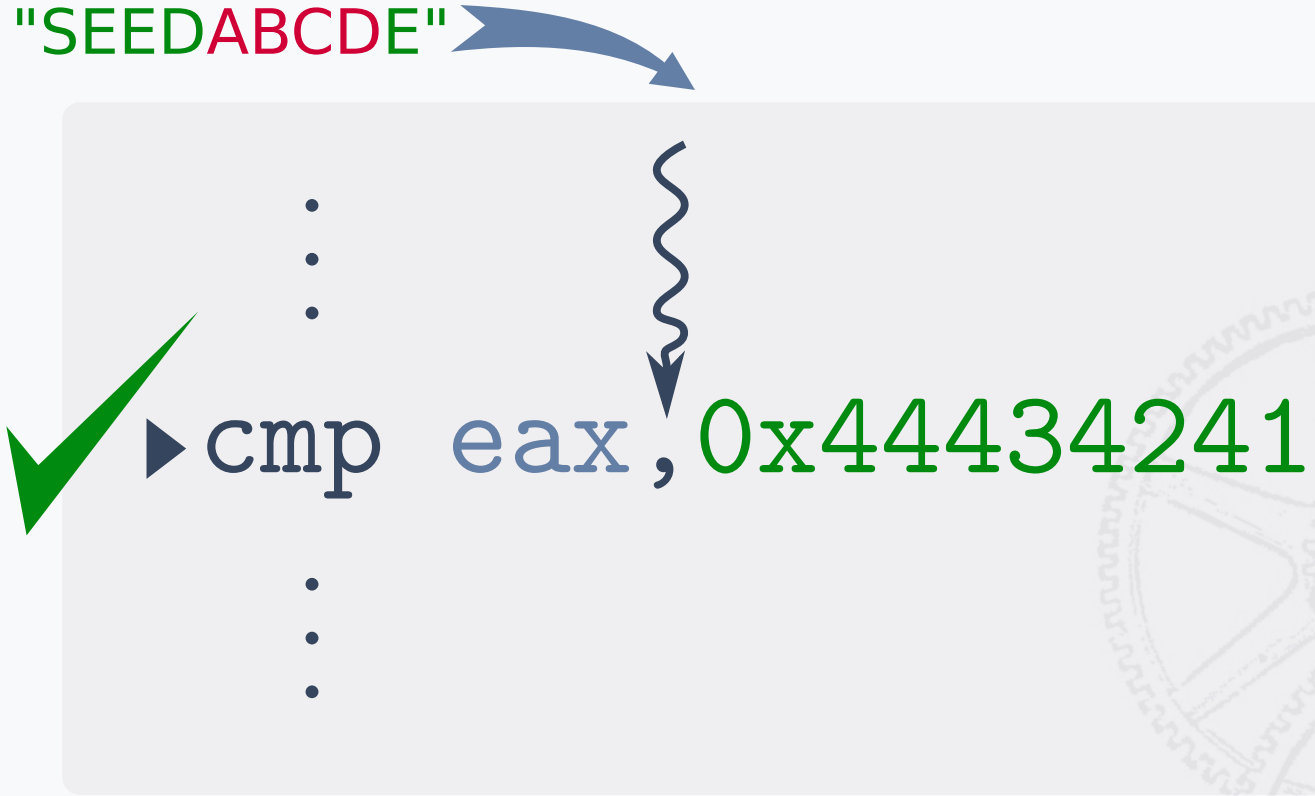
Input: "SEEDABCDE"

```
    .
    .
    .
    cmp eax,0x44434241
    .
    .
    .
```

Input: "SEEDABCDE"



Input: "SEEDABCDE"



Encodings?

Encodings?

atoi

atoll

atol

hex-decimal

base-64

sign-extend

little-endian

hex

big-endian

Encodings?

Replace(0x554c4156, 0x44434241)

Replace(enc(0x554c4156), enc(0x44434241))

$$x < y ?$$

x < y ?

Replace(0x554c4156, 0x44434241)

Replace(0x554c4156, 0x44434241+1)
Replace(0x554c4156, 0x44434241-1)

Slow?

Slow?

Executions: $10^8 \ldots 10^{14}$

Queue Entries: $10^3 \ldots 10^5$

cmp ?

cmp ?

xor lea sub

add

cmp ?

xor

lea

sub

call

add

Large Inputs?

Large Inputs?

```
af 00 00 00 ff ff ff ff  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
                      :
                      :     (64 KB later)
                      :
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Large Inputs?

Replace(0x0, 0x44)

```
af 00 00 00 ff ff ff ff  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
                      :     (64 KB later)
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

Replace(0x0, 0x44)

```
af 00 00 00
ff ff ff ff
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
    .
    .
    .
```

Replace(0x0, 0x44)

```
af 00 00 00
ff ff ff ff
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
   .
   .
   .
```

Replace(0xb1, 0x44)

```
af 00 00 00
b1 06 77 7a
45 ea 6c 3b
dd a6 3e b1
cc 2d 9d f0
ef 64 4d 45
32 04 54 08
c6 5e f3 e7
   .
   .
   .
```

Replace(0x0, 0x44)

```
af 00 00 00
ff ff ff ff
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
   .
   .
   .
```

Replace(0xb1, 0x44)

```
af 00 00 00
b1 06 77 7a
45 ea 6c 3b
dd a6 3e b1
cc 2d 9d f0
ef 64 4d 45
32 04 54 08
c6 5e f3 e7
   .
   .
   .
```

Replace(0x0, 0x44)

```
af 00 00 00
ff ff ff ff
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
   .
   .
   .
```

Replace(0xb1, 0x44)

```
af 00 00 00
b1 06 77 7a
45 ea 6c 3b
dd a6 3e b1
cc 2d 9d f0
ef 64 4d 45
32 04 54 08
c6 5e f3 e7
   .
   .
   .
```

```
/* magic number */
if( u64(input) == u64("MAGICHDR"))
    bug(1);
```

```
/* magic number */
if( u64(input) == u64("MAGICHDR"))
    bug(1);
```

```
/* nested checksum */
if( u64(input) == hash(input[8..len]) )
    if( u64(input+8) == hash(input[16..len]) )
        if( input[16] == 'R' && input[17] == 'Q')
            bug(2);
```

Patch with `cmp al,al`

False Positives?

False Positives?

$\Rightarrow$ Fix new queue entries

False Positives?

⇒ Fix new queue entries

⇒ OR ELSE remove patch

Nesting?

Nesting?

$\Rightarrow$ Topological Sort

REDQUEEN

Target

kAFL

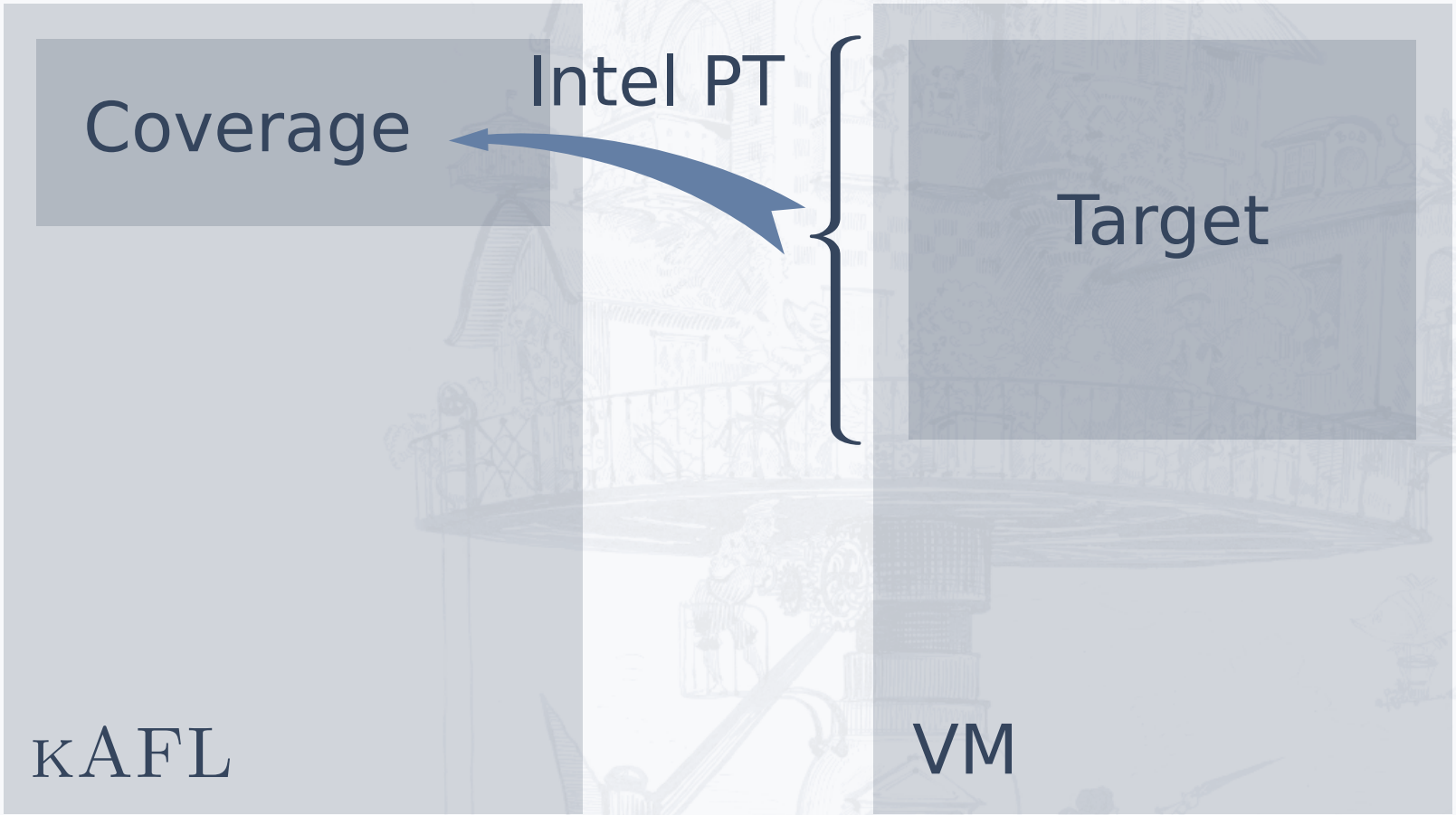
VM

Coverage

kAFL

Target

VM


Coverage
Intel PT
Target
kAFL
VM

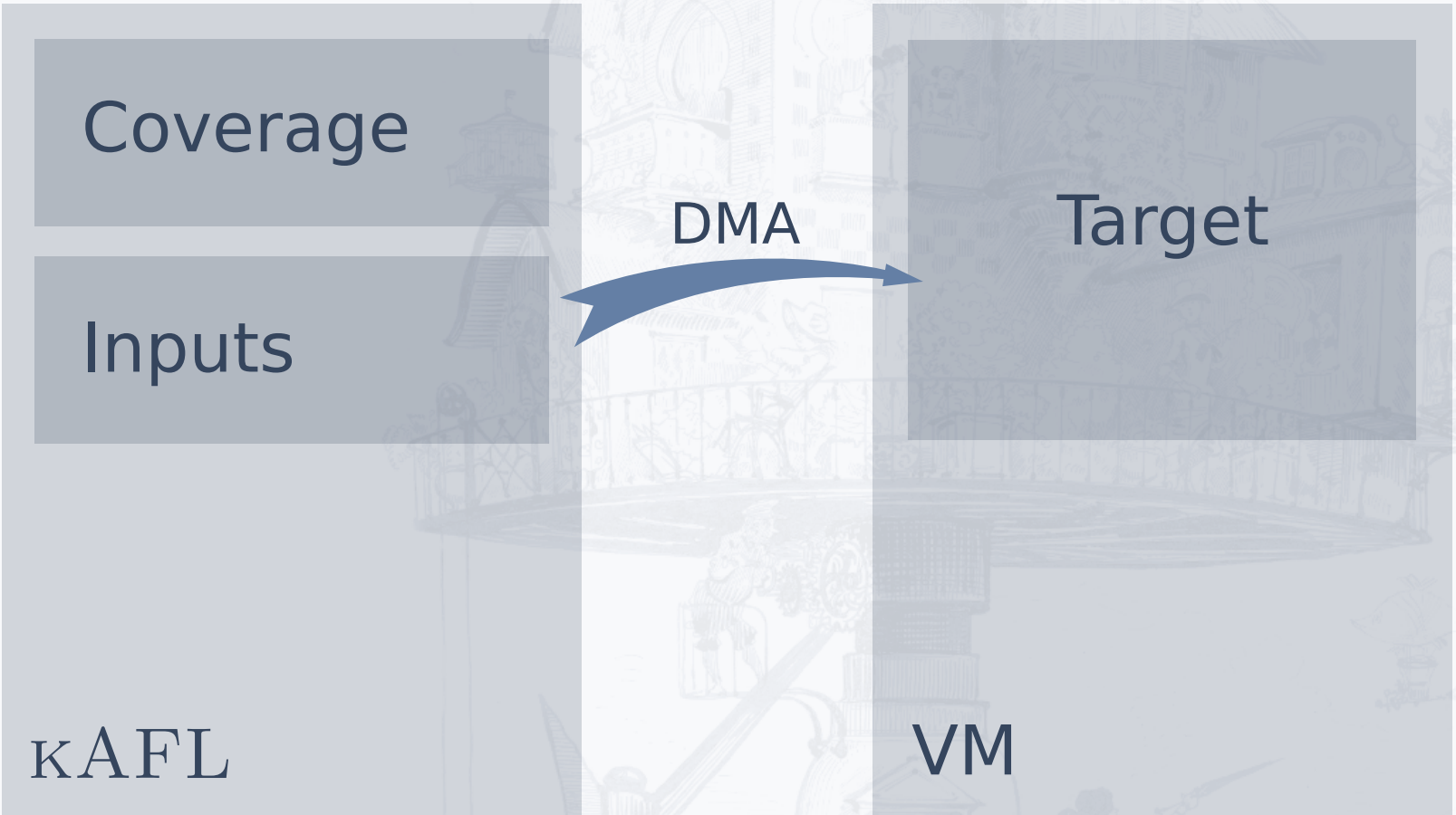
Coverage

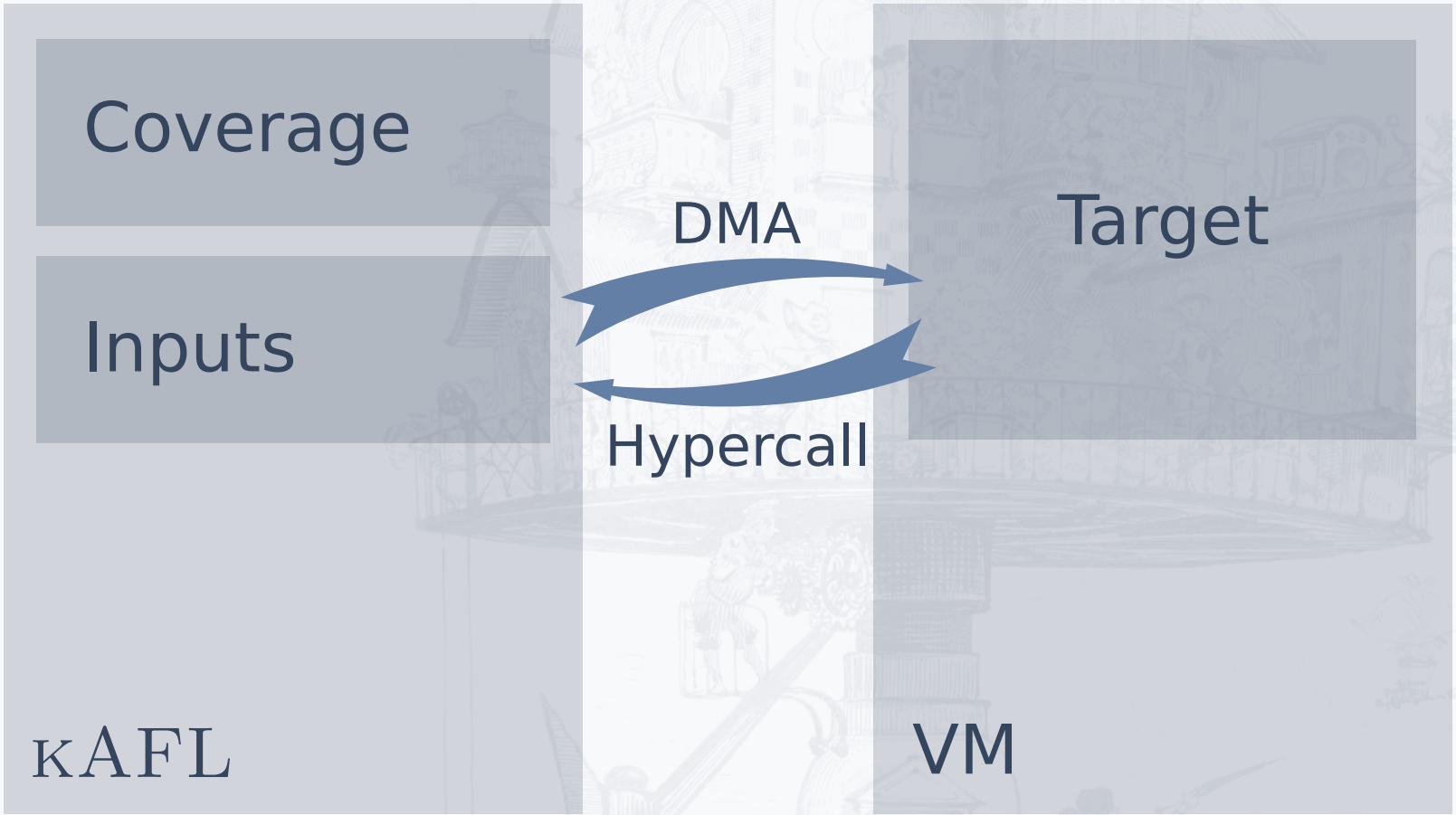
Inputs

kAFL

Target

VM


Coverage
Inputs
DMA
Target
kAFL
VM


Coverage
Inputs
kAFL
DMA
Hypercall
Target
VM

Coverage

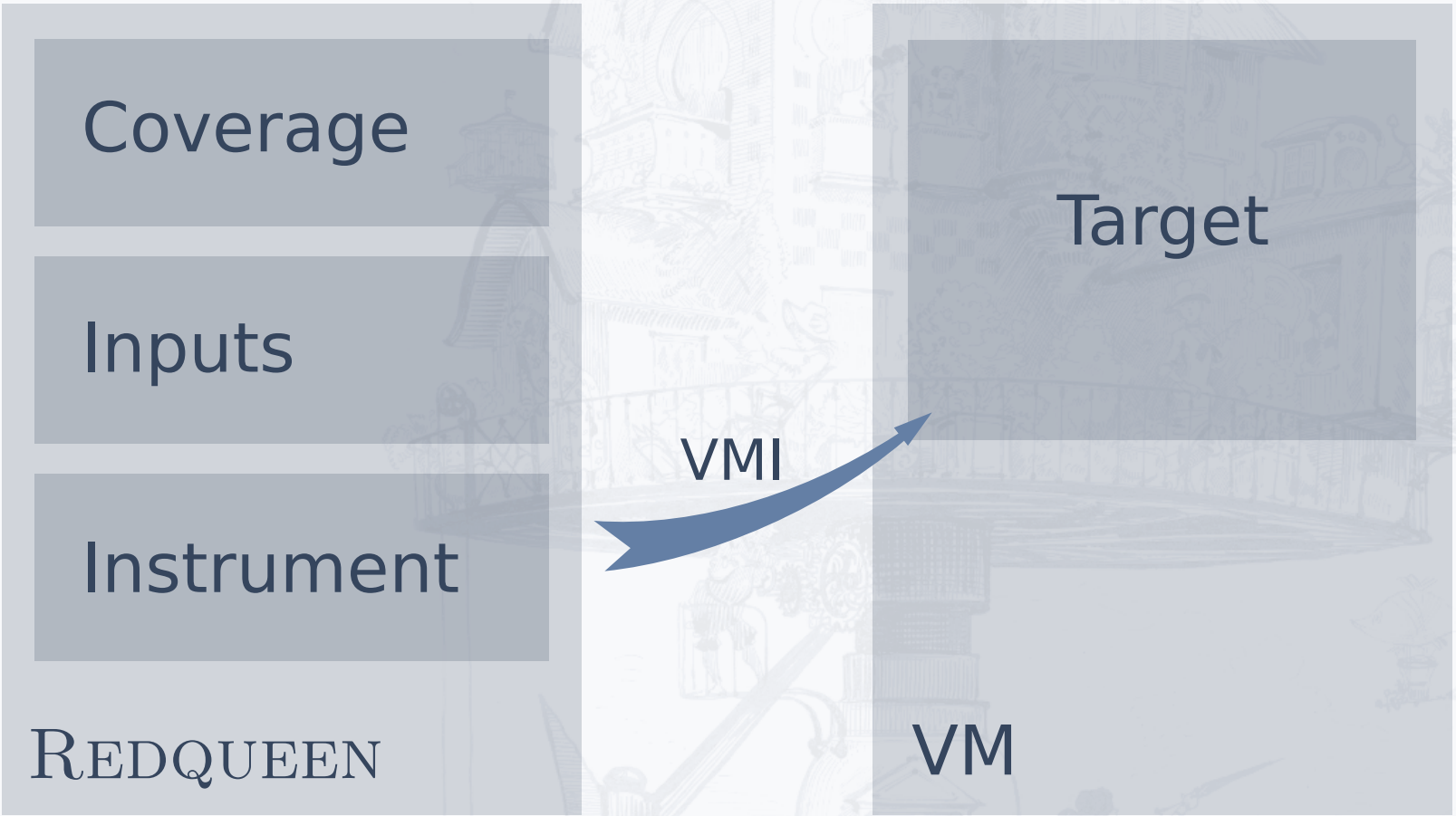
Inputs

Instrument

REDQUEEN

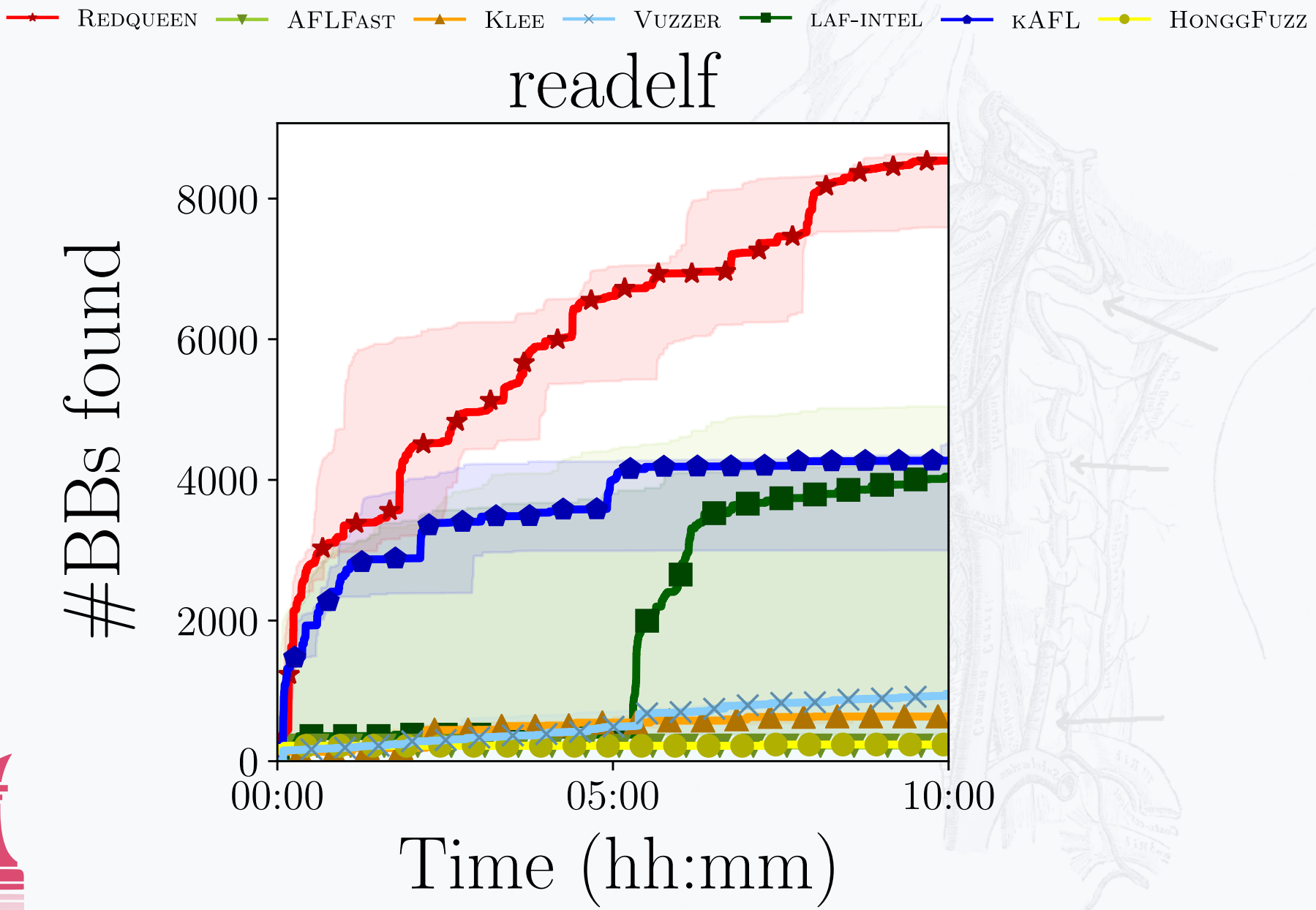
Target

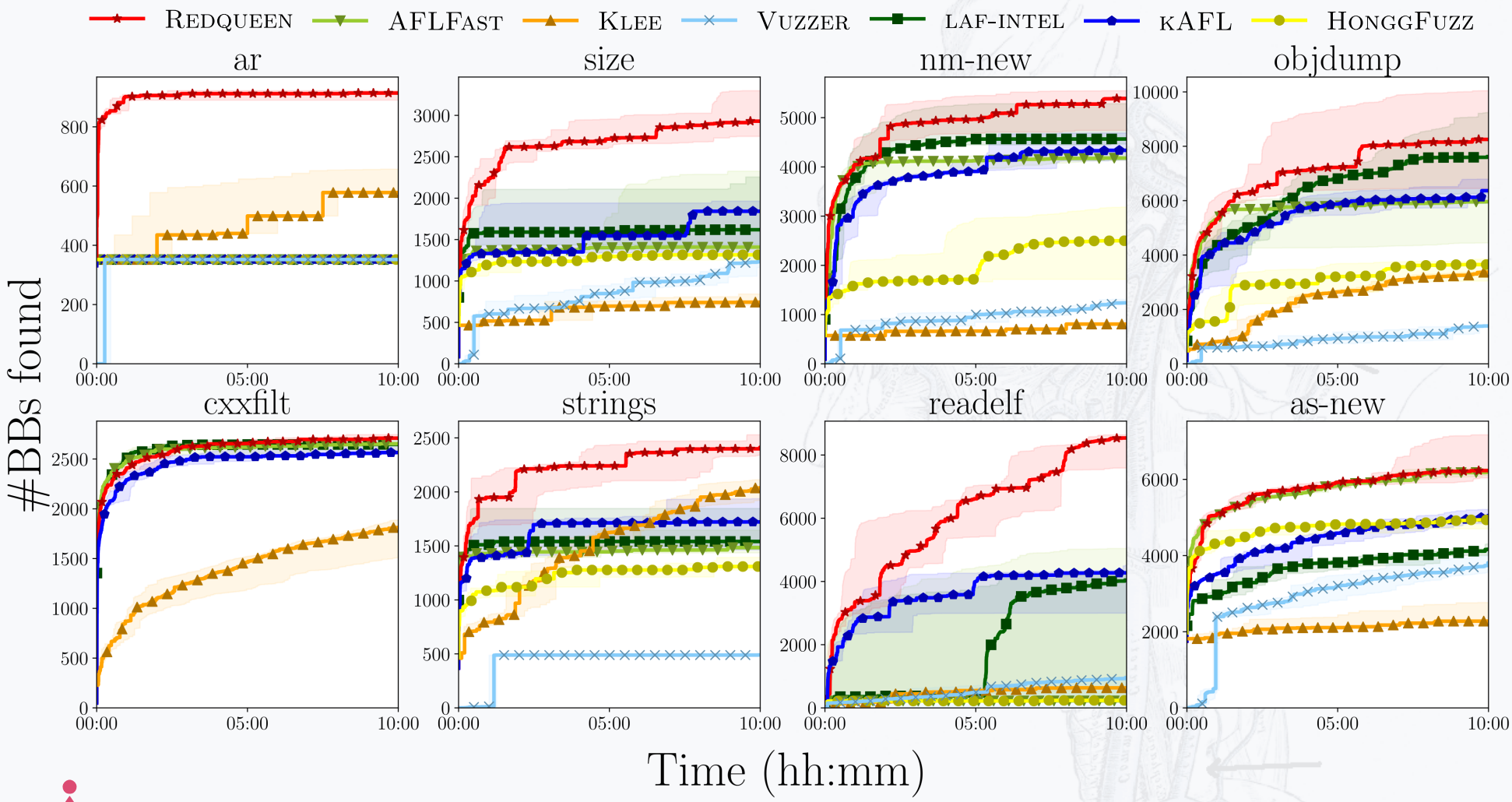
VM

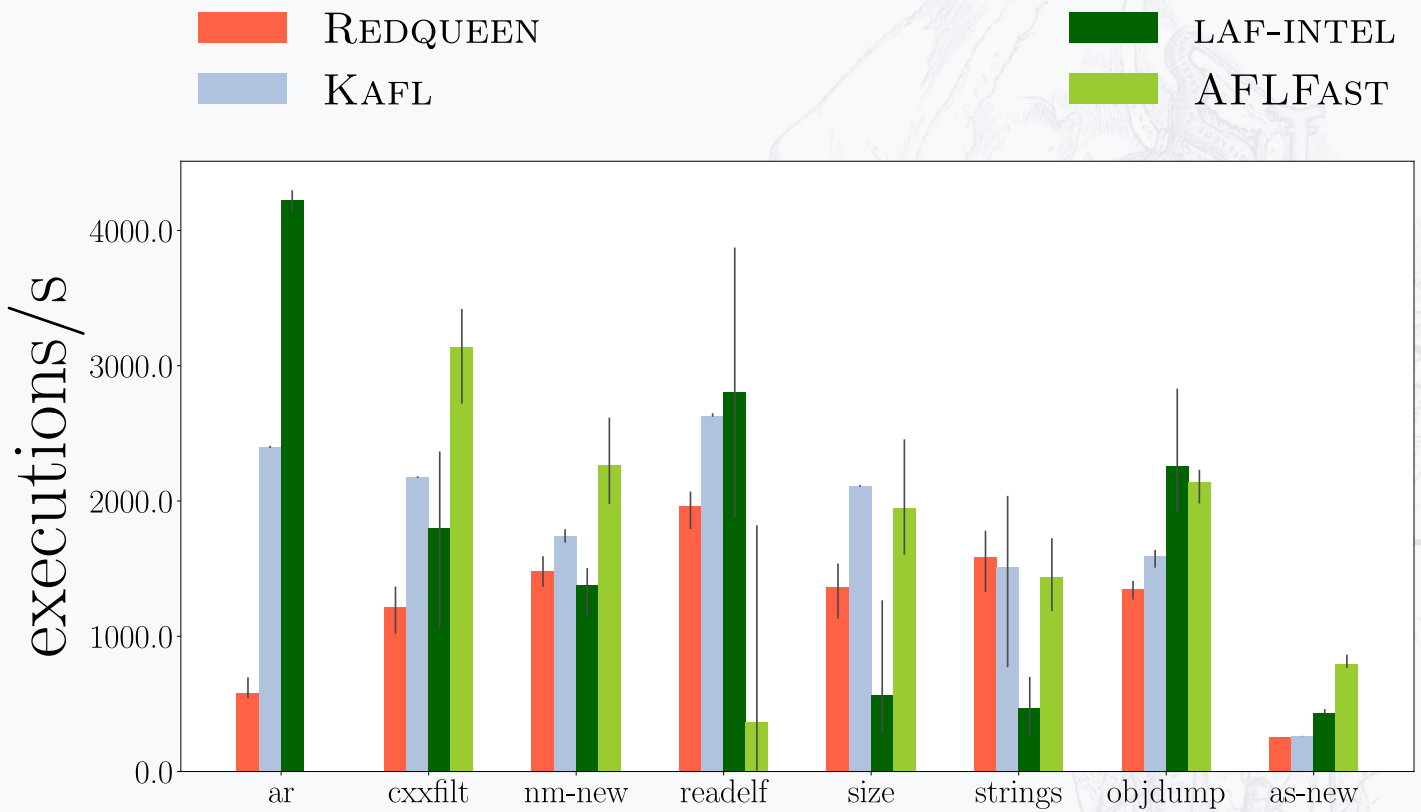

Coverage
Inputs
Instrument
REDQUEEN
VMI
Target
VM

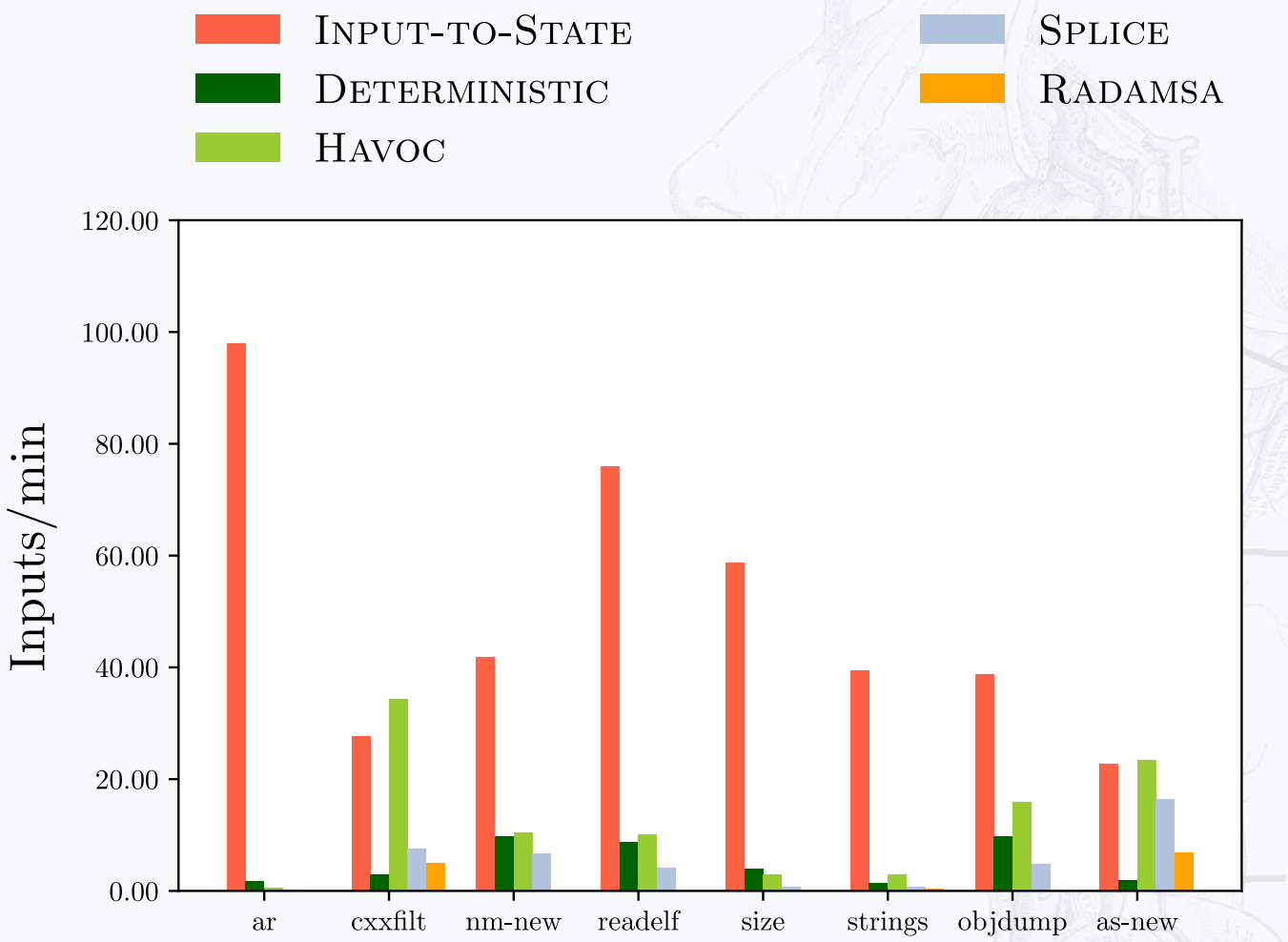
EvAL

Cornelius Aschermann

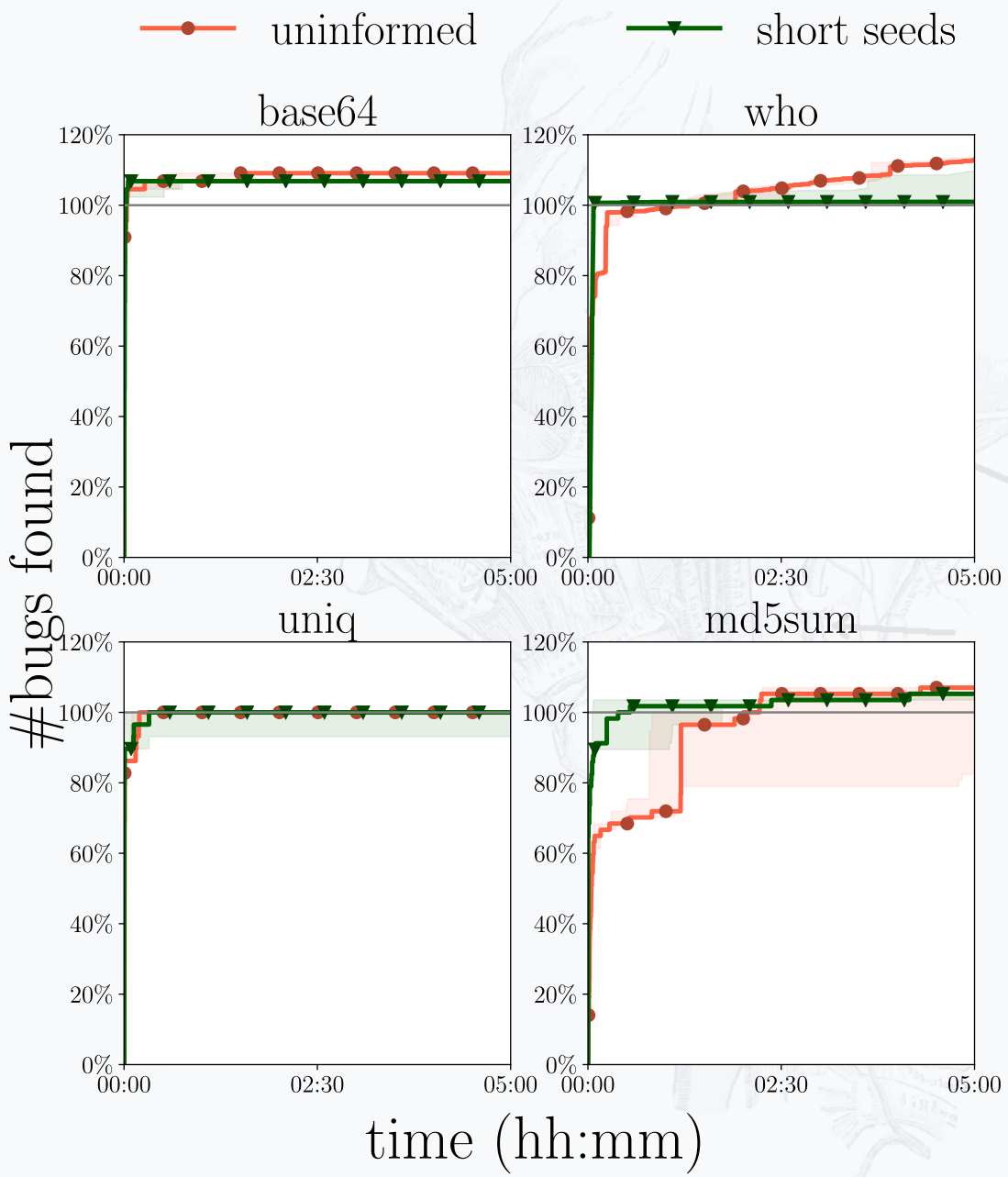

Redqueen
AFLFast
Klee
Vuzzer
laf-intel
kAFL
HonggFuzz
readelf
#BBs found
0
2000
4000
6000
8000
00:00
05:00
10:00
Time (hh:mm)


REDQUEEN
AFLFAST
KLEE
VUZZER
LAF-INTEL
KAFL
HONGGFUZZ
ar
size
nm-new
objdump
cxxfilt
strings
readelf
as-new
#BBs found
Time (hh:mm)
00:00
05:00
10:00


REDQUEEN
KAFL
LAF-INTEL
AFLFAST
executions/s
4000.0
3000.0
2000.0
1000.0
0.0
ar
cxxfilt
nm-new
readelf
size
strings
objdump
as-new


Input-to-State
Deterministic
Havoc
Splice
Radamsa
Inputs/min
120.00
100.00
80.00
60.00
40.00
20.00
0.00
ar
cxxfilt
nm-new
readelf
size
strings
objdump
as-new

LAVA-M


uninformed
short seeds
base64
who
uniq
md5sum
#bugs found
time (hh:mm)
0%
20%
40%
60%
80%
100%
120%
00:00
02:30
05:00

Bugs

CVE-2018-12641 (binutils nm-new)
CVE-2018-12697 (binutils libiberty)
CVE-2018-12698 (binutils libiberty)
CVE-2018-12699 (binutils objdump)
CVE-2018-12700 (binutils objdump)
CVE-2018-12934 (binutils cxxfilt)
CVE-2018-12928 (Linux kernel 4.15.0 hfs.ko)
CVE-2018-12929 (Linux kernel 4.15.0 ntfs.ko)
CVE-2018-12930 (Linux kernel 4.15.0 ntfs.ko)
CVE-2018-12931 (Linux kernel 4.15.0 ntfs.ko)
CVE-2018-12932 (Wine)
CVE-2018-12933 (Wine)

CVE-2018-14337 (mruby)
CVE-2018-14566 (bash)
CVE-2018-14567 (xml2)
CVE-2018-16747 (fdk-aac)
CVE-2018-16748 (fdk-aac)
CVE-2018-16749 (ImageMagick)
CVE-2018-16750 (ImageMagick)
CVE-2018-12935 (ImageMagick)
CVE-2018-20116 (tcpdump)
CVE-2018-20117 (tcpdump)
CVE-2018-20118 (tcpdump)
CVE-2018-20119 (tcpdump)

Overview

- Input-to-State
- Encodings
- Colorization
- Checksum Patches
- Intel PT
- VM - Introspection