# Breaking State-of-the-Art Binary Code Obfuscation

## A Program Synthesis-based Approach

Toulouse Hacking Convention

March 9, 2018

Tim Blazytko, `@mr_phrazer`
`http://synthesis.to`

Moritz Contag, `@dwuid`
`https://dwuid.com`

Chair for Systems Security
Ruhr-Universität Bochum
`<firstname.lastname>@rub.de`

# Syntia: Synthesizing the Semantics of Obfuscated Code

Tim Blazytko, Moritz Contag, Cornelius Aschermann, and Thorsten Holz, *Ruhr-Universität Bochum*

https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/blazytko

This paper is included in the Proceedings of the
26th USENIX Security Symposium
August 16–18, 2017 • Vancouver, BC, Canada
ISBN 978-1-931971-40-9

## Setting the Scene

- Obfuscated code, semantics?
- Traditional deobfuscation techniques
- **Orthogonal approach**

## Motivation

~~Prevent~~ **Complicate** reverse engineering attempts.

- Intellectual Property
- Malicious Payloads
- Digital Rights Management

## Motivation

~~Prevent~~ **Complicate** reverse engineering attempts.

- Intellectual Property
- Malicious Payloads
- Digital Rights Management

"We achieved our goals. We were uncracked for **13 whole days**."

– Martin Slater, 2K Australia, on *BioShock* (2007).

## How to protect software?

## Approaches

Abuse shortcomings of file parsers and other tools of the trade.

- `fld tbyte ptr [__bad_values]` crashing OllyDbg 1.10.
- Fake `SizeOfImage` crashing process dumpers.

## Approaches

Abuse shortcomings of file parsers and other tools of the trade.

- `fld tbyte ptr [__bad_values]` crashing OllyDbg 1.10.
- Fake `SizeOfImage` crashing process dumpers.

Detect artifacts of the debugging process.

- `PEB.BeingDebugged` bit being set.
- `int 2D` and exception handling in debuggers.

## Approaches

Abus

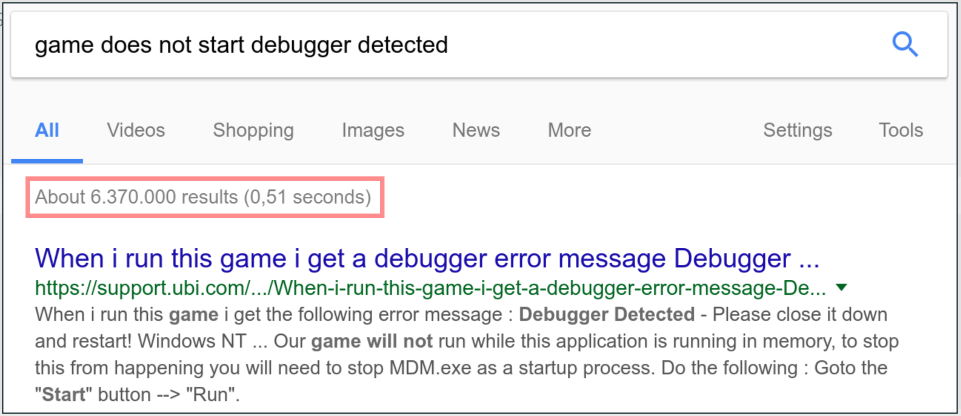
•

•

Dete

•

•



game does not start debugger detected

About 6.370.000 results (0,51 seconds)

When i run this game i get a debugger error message Debugger ...

https://support.ubi.com/.../When-i-run-this-game-i-get-a-debugger-error-message-De...

When i run this **game** i get the following error message : **Debugger Detected** - Please close it down and restart! Windows NT ... Our **game will not** run while this application is running in memory, to stop this from happening you will need to stop MDM.exe as a startup process. Do the following : Goto the "**Start**" button --> "Run".

## Requirements

1. We want the technique to be *semantics-preserving*.

Preserve the observable behavior of the application.

## Requirements

1. We want the technique to be *semantics-preserving*.
2. We want to avoid external dependencies, focus on code only.

Assume white-box attack scenario.

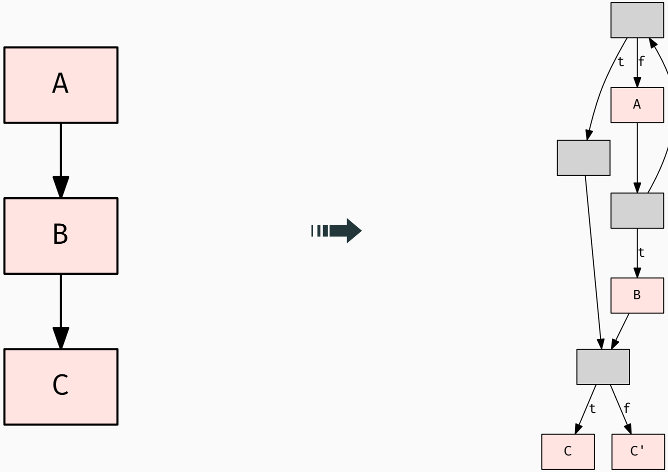
## Requirements

1. We want the technique to be *semantics-preserving*.
2. We want to avoid external dependencies, focus on code only.
3. We want techniques where **effort(deploy)** $\ll$ **effort(attack)**.

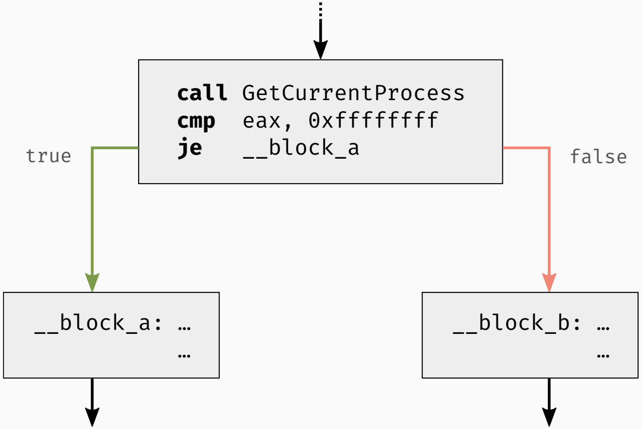
Anti-Debugging tricks are effort **1:1**.

# Code Obfuscation Techniques

Opaque Predicates

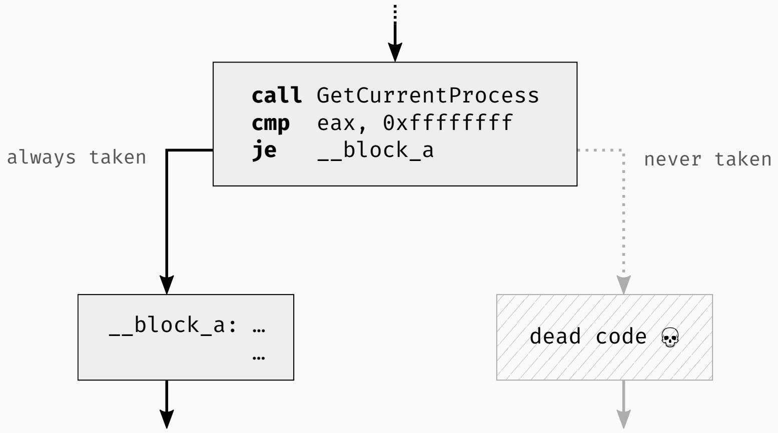

A
B
C
t
f
A
f
t
B
t
f
C
C'

## Opaque Predicates



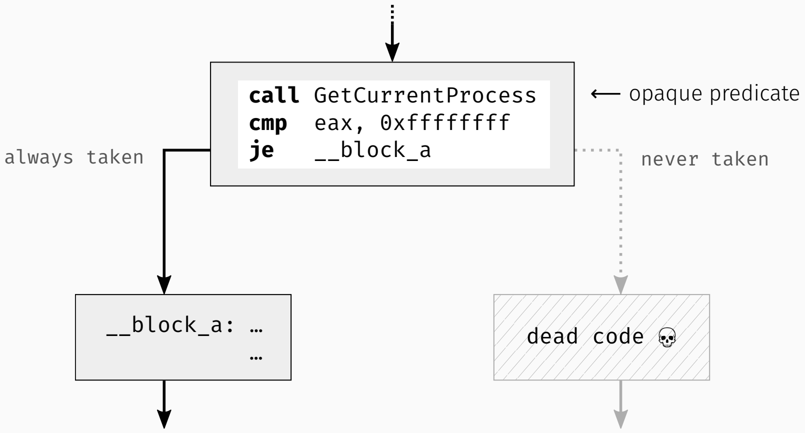
```
call GetCurrentProcess
cmp  eax, 0xffffffff
je   __block_a
```

## Opaque Predicates


```
call GetCurrentProcess
cmp  eax, 0xffffffff
je   __block_a
```

Opaque True Predicate

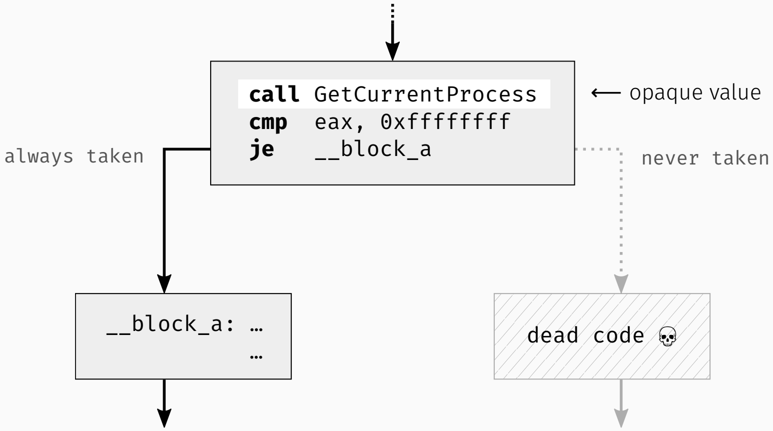
## Opaque Predicates



```
call GetCurrentProcess
cmp  eax, 0xffffffff
je   __block_a
```

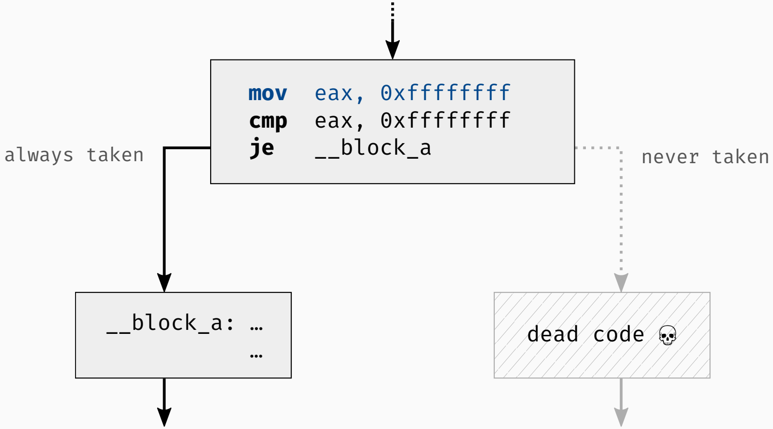
Opaque True Predicate

## Opaque Predicates


```
call GetCurrentProcess
cmp  eax, 0xffffffff
je   __block_a
```

Opaque True Predicate

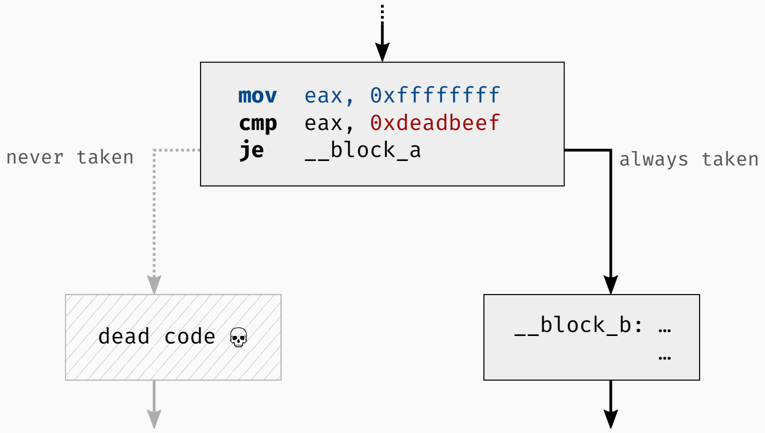
# Opaque Predicates



```
mov  eax, 0xffffffff
cmp  eax, 0xffffffff
je   __block_a
```

Opaque True Predicate

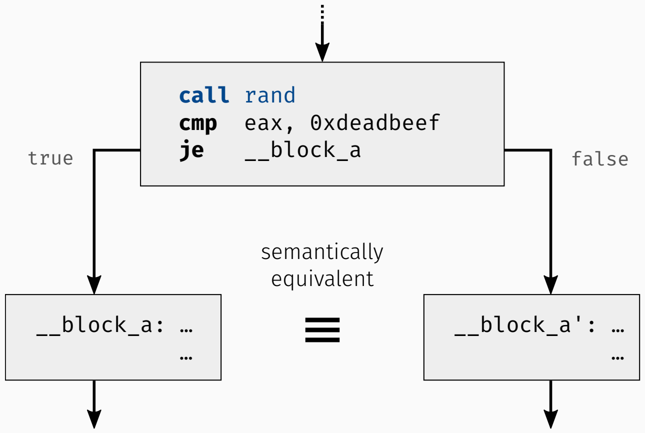
## Opaque Predicates


```
mov   eax, 0xffffffff
cmp   eax, 0xdeadbeef
je    __block_a
```

```
__block_b: …
           …
```

Opaque False Predicate

# Opaque Predicates



```
call rand
cmp  eax, 0xdeadbeef
je   __block_a
```

Random Opaque Predicate

duplicated block

## Opaque Predicates

- ⊕ Increase in complexity (branch count, McCabe)
- ⊕ Can be built on hard problems (e. g., aliasing)
- ⊕ Forces analyst to **encode additional knowledge**
- ⊕ Hard to solve statically

⚠ Examples

- `GetCurrentProcess()` $\Rightarrow -1$
- `fldpi`[1] $\Rightarrow$ `st(0)` $= \pi$
- $x^2 \geq 0 \;\; \forall x$
- $x + 1 \neq x \;\; \forall x$
- pointer A *must-alias* pointer B
- $\mathrm{checksum}(code) =$ `0x1c43b5cf`

## Opaque Predicates

- ⊕ Increase in complexity (branch count, McCabe)
- ⊕ Can be built on hard problems (e. g., aliasing)
- ⊕ Forces analyst to encode additional knowledge
- ⊕ Hard to solve statically

- ⊖ Solved for free using **concrete execution traces**

⚠ Examples

- `GetCurrentProcess()` $\Rightarrow -1$
- `fldpi`[1] $\Rightarrow$ `st(0)` $= \pi$
- $x^2 \geq 0 \quad \forall x$
- $x + 1 \neq x \quad \forall x$
- pointer A *must-alias* pointer B
- checksum(*code*) $=$ `0x1c43b5cf`

## Code Obfuscation Techniques

Virtual Machines

## Virtual Machines

```
mov ecx, [esp+4]
xor eax, eax
mov ebx, 1

__secret_ip:
  mov  edx, eax
  add  edx, ebx
  mov  eax, ebx
  mov  ebx, edx
  loop __secret_ip

mov eax, ebx
ret
```

# Virtual Machines

```
mov ecx, [esp+4]
xor eax, eax
mov ebx, 1

__secret_ip:
  mov  edx, eax
  add  edx, ebx
  mov  eax, ebx
  mov  ebx, edx
  loop __secret_ip

mov eax, ebx
ret
```

## Virtual Machines

```
mov ecx, [esp+4]
xor eax, eax
mov ebx, 1

__secret_ip:
  mov  edx, eax
  add  edx, ebx
  mov  eax, ebx
  mov  ebx, edx
  loop __secret_ip

mov eax, ebx
ret
```

# Virtual Machines

```
mov ecx, [esp+4]
xor eax, eax
mov ebx, 1

__secret_ip:
   mov   edx, eax
   add   edx, ebx
   mov   eax, ebx
   mov   ebx, edx
   loop  __secret_ip

mov eax, ebx
ret
```

made-up instruction set

```
__bytecode:    vld  r1
   vld  r0     vpop r2
   vpop r1     vldi #1
   vld  r2     vld  r3
   vld  r1     vsub r3
   vadd r1     vld  #0
   vld  r2     veq  r3
   vpop r0     vbr0 #-0E
```

# Virtual Machines

```
mov ecx, [esp+4]
xor eax, eax
mov ebx, 1

__secret_ip:
  push __bytecode
  call vm_entry



mov eax, ebx
ret
```

made-up instruction set

```
__bytecode:
  db 54 68 69 73 20 64 6f
  db 65 73 6e 27 74 20 6c
  db 6f 6f 6b 20 6c 69 6b
  db 65 20 61 6e 79 74 68
  db 69 6e 67 20 74 6f 20
  db 6d 65 2e de ad be ef
```

## Virtual Machines

```
mov ecx, [esp+4]
xor eax, eax
mov ebx, 1

__secret_ip:
  push __bytecode
  call vm_entry

mov eax, ebx
ret
```

made-up instruction set

```
__bytecode:
  db 54 68 69 73 20 64 6f
  db 65 73 6e 27 74 20 6c
  db 6f 6f 6b 20 6c 69 6b
  db 65 20 61 6e 79 74 68
  [illegible]9 6e 67 20 74 6f 20
  [illegible] 65 2e de ad be ef
```

## Virtual Machines

**Core Components**

| | |
|---|---|
| VM Entry/Exit | Context Switch: native context ⇔ virtual context |
| **VM Dispatcher** | Fetch–Decode–Execute loop |
| **Handler Table** | Individual VM ISA instruction semantics |

- **Entry** Copy native context (registers, flags) to VM context.
- **Exit** Copy VM context back to native context.

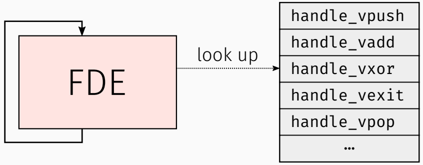
- Mapping from native to virtual registers is often 1:1.

## Virtual Machines

**Core Components**

| | |
|---|---|
| **VM Entry/Exit** | Context Switch: native context ⇔ virtual context |
| **VM Dispatcher** | Fetch–Decode–Execute loop |
| **Handler Table** | Individual VM ISA instruction semantics |

1. Fetch and decode instruction
2. Forward virtual instruction pointer
3. Look up handler for opcode in handler table
4. Invoke handler

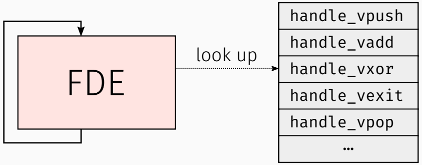


# Virtual Machines

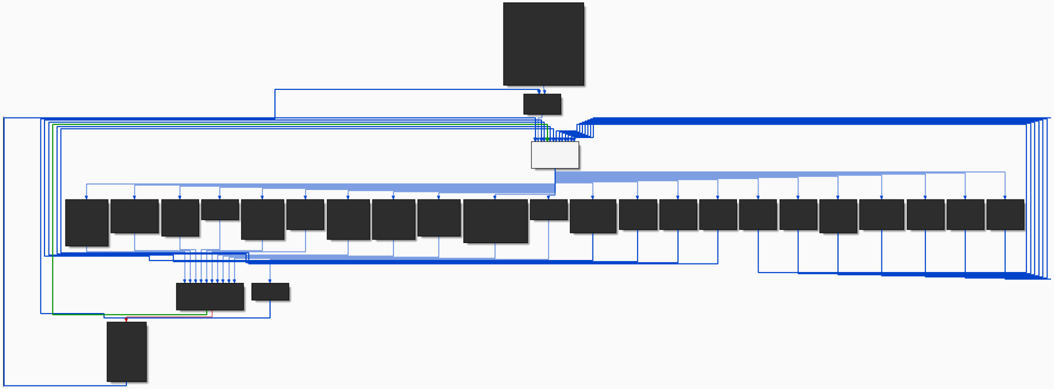
**Core Components**

| | |
|---|---|
| **VM Entry/Exit** | Context Switch: native context ⇔ virtual context |
| **VM Dispatcher** | Fetch–Decode–Execute loop |
| **Handler Table** | Individual VM ISA instruction semantics |

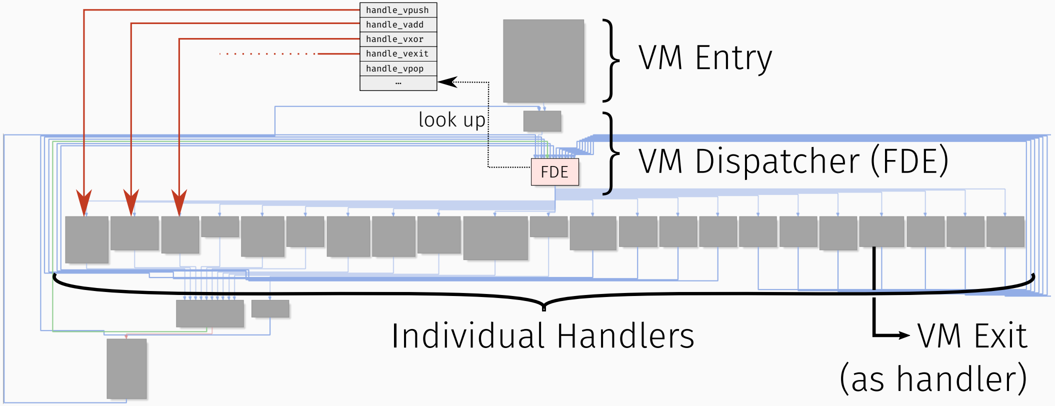
- Table of function pointers indexed by opcode
- One handler per virtual instruction
- Each handler decodes operands and updates VM context



## Virtual Machines



# Virtual Machines



## Virtual Machines

```
__vm_dispatcher:
  mov   bl, [rsi]
  inc   rsi
  movzx rax, bl
  jmp   __handler_table[rax * 8]
```

VM Dispatcher

`rsi` – virtual instruction pointer
`rbp` – VM context

## Virtual Machines

```
__vm_dispatcher:
  mov   bl, [rsi]
  inc   rsi
  movzx rax, bl
  jmp   __handler_table[rax * 8]
```

VM Dispatcher

`rsi` – virtual instruction pointer
`rbp` – VM context

```
__handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
  and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

Handler performing `nor`
(with flag side-effects)

## Virtual Machine Hardening

## Virtual Machines

**Hardening Technique #1** – Obfuscating individual VM components.

- Handlers are *conceptually simple*.

# Virtual Machines

**Hardening Technique #1** – Obfuscating individual VM components.

- Handlers are *conceptually simple*.
- Apply traditional code obfuscation transformations:
    - Substitution (`mov rax, rbx` ➟ `push rbx; pop rax`)
    - Opaque Predicates
    - Junk Code
    - ...

```
mov eax, dword [rbp]
mov ecx, dword [rbp+4]
cmp r11w, r13w
sub rbp, 4
not eax
clc
cmc
cmp rdx, 0x28b105fa
not ecx
cmp r12b, r9b
```

## Virtual Machines

**Hardening Technique #2** – Duplicating VM handlers.

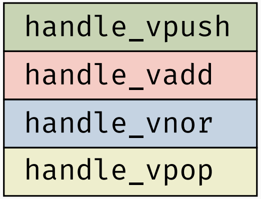
- Handler table is typically indexed using one byte (= 256 entries).

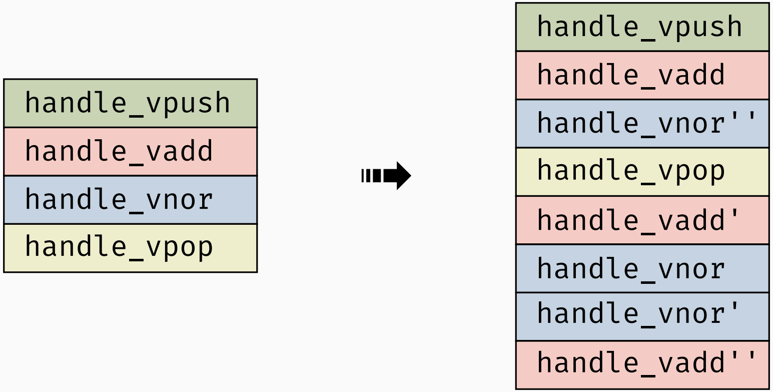
## Virtual Machines

**Hardening Technique #2** – Duplicating VM handlers.

- Handler table is typically indexed using one byte (= 256 entries).
- **Idea:** *Duplicate* existing handlers to populate full table.
- Use traditional obfuscation techniques to impede *code similarity* analyses.

**Goal:** Increase workload of reverse engineer.


handle_vpush
handle_vadd
handle_vnor
handle_vpop


handle_vpush
handle_vadd
handle_vnor
handle_vpop
handle_vpush
handle_vadd
handle_vnor''
handle_vpop
handle_vadd'
handle_vnor
handle_vnor'
handle_vadd''

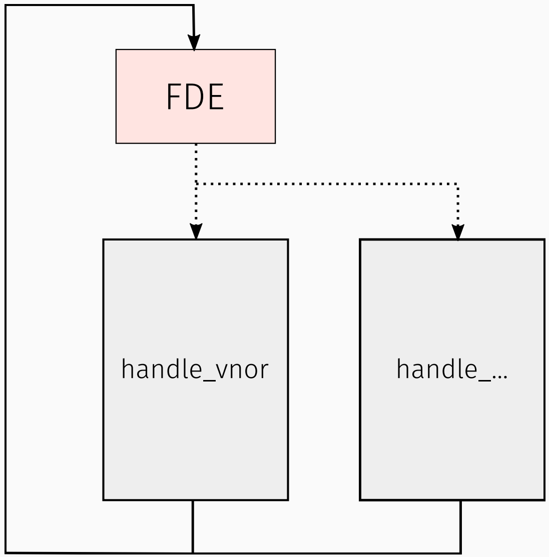
## Virtual Machines

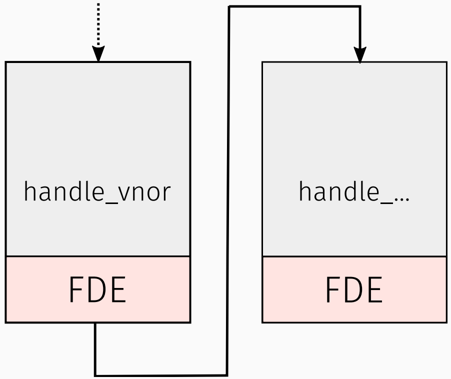
**Hardening Technique #3** – No central VM dispatcher.

- A *central* VM dispatcher allows attacker to easily observe VM execution.
- **Idea:** Instead of branching to the central dispatcher, *inline* it into each handler.

**Goal:** No “single point of failure”.

(Themida, VMProtect Demo)


FDE
handle_vnor
handle_…


handle_vnor
FDE
handle_...
FDE

Programming Techniques

R. Morris Editor

# Threaded Code

James R. Bell
Digital Equipment Corporation

The concept of "threaded code" is presented as an alternative to machine language code. Hardware and software realizations of it are given. In software it is realized as interpretive code not needing an interpreter. Extensions and optimizations are mentioned.

Key Words and Phrases: interpreter, machine code, time tradeoff, space tradeoff, compiled code, subroutine calls, threaded code

CR Categories: 4.12, 4.13, 6.33

Fig. 2 Flow of control: interpretive code.

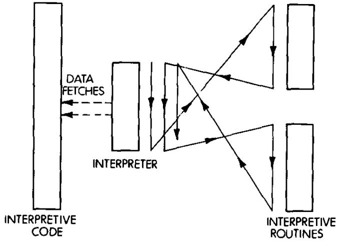
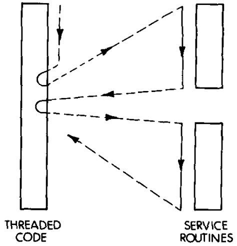


Fig. 3. Flow of control: threaded code.



## Virtual Machines

**Hardening Technique #4** – No explicit handler table.

- An *explicit* handler table easily reveals all VM handlers.

## Virtual Machines

**Hardening Technique #4** – No explicit handler table.

- An *explicit* handler table easily reveals all VM handlers.
- **Idea:** Instead of querying an explicit handler table,
  *encode* the next handler address in the VM instruction itself.

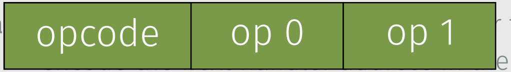
**Goal:** Hide location of handlers that have not been executed yet.

(VMProtect Full, SolidShield)

## Virtual Machines

Hardening Technique #4 – No explicit handler table.

- An *explicit* handler table easily reveals all VM handlers.
- **Idea** table,
  e VM instruction itself.



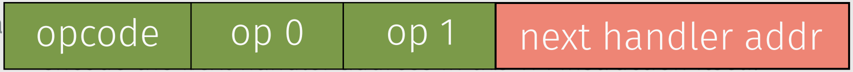
**Goal:** Hide location of handlers that have not been executed yet.

(VMProtect Full, SolidShield)

# Virtual Machines

**Hardening Technique #4** – No explicit handler table.

- An *explicit* handler table easily reveals all VM handlers.
- **Idea**



**Goal:** Hide location of handlers that have not been executed yet.

(VMProtect Full, SolidShield)

SOFTWARE–PRACTICE AND EXPERIENCE, VOL. 11, 963–973 (1981)

# Interpretation Techniques*

PAUL KLINT

*Mathematical Centre, P.O. Box 4079, 1009AB Amsterdam, The Netherlands*

## SUMMARY

**The relative merits of implementing high level programming languages by means of interpretation or compilation are discussed. The properties and the applicability of interpretation techniques known as classical interpretation, direct threaded code and indirect threaded code are described and compared.**

KEY WORDS Interpretation versus compilation Interpretation techniques Instruction encoding Code generation Direct threaded code Indirect threaded code.

## Virtual Machines

**Hardening Technique #5** – Blinding VM bytecode.

- *Global analyses* on the bytecode possible, easy to patch instructions.

## Virtual Machines

**Hardening Technique #5** – Blinding VM bytecode.

- *Global analyses* on the bytecode possible, easy to patch instructions.
- **Idea:**
    - *Flow-sensitive* instruction decoding (“decryption” based on key register).
    - Custom decryption routine per handler, diversification.
    - Patching requires re-encryption of subsequent bytecode.

**Goal:** Hinder global analyses of bytecode and patching.

```
$\mathit{operand} \leftarrow [\mathbf{vIP} + 0]$



$\mathit{context} \leftarrow \mathrm{semantics}(\mathit{context}, \mathit{operand})$
$\mathit{next\_handler} \leftarrow [\mathbf{vIP} + 4]$



$\mathbf{vIP} \leftarrow \mathbf{vIP} + 8$
$\mathbf{jmp}$ $\mathit{next\_handler}$
```

```
   operand        ← [vIP + 0]

🔑 operand        ← unmangle(operand, key)
🔑 key            ← unmangle′(key, operand)

   context        ← semantics(context, operand)
   next_handler   ← [vIP + 4]

🔑 next_handler   ← unmangle″(next_handler, key)
🔑 key            ← unmangle‴(key, next_handler)

   vIP ← vIP + 8
   jmp next_handler
```

## Code Obfuscation Techniques

Mixed Boolean-Arithmetic

## Mixed Boolean-Arithmetic

What does this expression compute?

$$(x \oplus y) + 2 \cdot (x \wedge y)$$

## Mixed Boolean-Arithmetic

What does this expression compute?

$$(x \oplus y) + 2 \cdot (x \wedge y)$$
$$= x + y$$

## Mixed Boolean-Arithmetic

What does this expression compute?

$$(((x \oplus y) + ((x \wedge y) \ll 1)) \vee z) + (((x \oplus y) + ((x \wedge y) \ll 1)) \wedge z)$$

# Mixed Boolean-Arithmetic

What does this expression compute?

$$(((x \oplus y) + ((x \wedge y) \ll 1)) \vee z) + (((x \oplus y) + ((x \wedge y) \ll 1)) \wedge z)$$

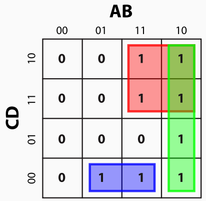
$$= \mathbf{x + y + z}$$

- Boolean identities?
- Arithmetic identities?
- Karnaugh-Veitch maps?

$$A \cdot 0 = 0$$

$$A + B = \overline{\overline{A} \cdot \overline{B}}$$

$$x^2 - y^2 = (x + y)(x - y)$$



| CD \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 10 | 0 | 0 | 1 | 1 |
| 11 | 0 | 0 | 1 | 1 |
| 01 | 0 | 0 | 0 | 1 |
| 00 | 0 | 1 | 1 | 1 |

# Mixed Boolean-Arithmetic

**Boolean-arithmetic algebra BA[$n$]**

$$(B^n, \wedge, \vee, \oplus, \neg, \leq, \geq, >, <, \leq^s, \geq^s, >^s, <^s, \neq, =, \gg^s, \gg, \ll, +, -, \cdot)$$
is a Boolean-arithmetic algebra BA[$n$], for $n > 0$, $B = \{0, 1\}$.

BA[$n$] includes, amongst others, both:

- Boolean algebra $(B^n, \wedge, \vee, \neg)$,
- Integer modular ring $\mathbb{Z}/(2^n)$.

**No techniques to simplify such expressions easily!**

## Deobfuscation

## Symbolic Execution

```
__handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
  and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

Handler performing `nor`
(with flag side-effects)

# Symbolic Execution

```
__handle_vnor:
• mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
  and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

Handler performing `nor`
(with flag side-effects)

rcx $\leftarrow$ [rbp]

## Symbolic Execution

```
__handle_vnor:
  mov   rcx, [rbp]
• mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
  and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)

$$\begin{aligned} rcx &\leftarrow [rbp] \\ rbx &\leftarrow [rbp + 4] \end{aligned}$$

## Symbolic Execution

```
__handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
• not   rcx
  not   rbx
  and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)

$$\begin{aligned} \mathrm{rcx} &\leftarrow [\mathrm{rbp}] \\ \mathrm{rbx} &\leftarrow [\mathrm{rbp}+4] \\ \mathrm{rcx} &\leftarrow \neg\,\mathrm{rcx} = \neg\,[\mathrm{rbp}] \end{aligned}$$

# Symbolic Execution

```
__handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
• not   rbx
  and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

Handler performing `nor`
(with flag side-effects)

$$\begin{aligned}
\text{rcx} &\leftarrow [\text{rbp}] \\
\text{rbx} &\leftarrow [\text{rbp} + 4] \\
\text{rcx} &\leftarrow \neg\,\text{rcx} = \neg\,[\text{rbp}] \\
\text{rbx} &\leftarrow \neg\,\text{rbx} = \neg\,[\text{rbp} + 4]
\end{aligned}$$

# Symbolic Execution

```
__handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
• and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)

$$
\begin{aligned}
\mathrm{rcx} &\leftarrow [\mathrm{rbp}] \\
\mathrm{rbx} &\leftarrow [\mathrm{rbp}+4] \\
\mathrm{rcx} &\leftarrow \neg\,\mathrm{rcx} = \neg\,[\mathrm{rbp}] \\
\mathrm{rbx} &\leftarrow \neg\,\mathrm{rbx} = \neg\,[\mathrm{rbp}+4] \\
\mathrm{rcx} &\leftarrow \mathrm{rcx} \wedge \mathrm{rbx} \\
&\qquad = (\neg\,[\mathrm{rbp}]) \wedge (\neg\,[\mathrm{rbp}+4])
\end{aligned}
$$

## Symbolic Execution

```
__handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
• and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)

$$
\begin{aligned}
\text{rcx} &\leftarrow [\text{rbp}] \\
\text{rbx} &\leftarrow [\text{rbp} + 4] \\
\text{rcx} &\leftarrow \neg\,\text{rcx} = \neg\,[\text{rbp}] \\
\text{rbx} &\leftarrow \neg\,\text{rbx} = \neg\,[\text{rbp} + 4] \\
\text{rcx} &\leftarrow \text{rcx} \wedge \text{rbx} \\
&\quad = (\neg\,[\text{rbp}]) \wedge (\neg\,[\text{rbp} + 4]) \\
&\quad = [\text{rbp}] \downarrow [\text{rbp} + 4]
\end{aligned}
$$

## Symbolic Execution

```
__handle_vnor:
    mov   rcx, [rbp]
    mov   rbx, [rbp + 4]
    not   rcx
    not   rbx
    and   rcx, rbx
  • mov   [rbp + 4], rcx
    pushf
    pop   [rbp]
    jmp   __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)

$$
\begin{aligned}
\text{rcx} &\leftarrow [\text{rbp}] \\
\text{rbx} &\leftarrow [\text{rbp}+4] \\
\text{rcx} &\leftarrow \neg\,\text{rcx} = \neg\,[\text{rbp}] \\
\text{rbx} &\leftarrow \neg\,\text{rbx} = \neg\,[\text{rbp}+4] \\
\text{rcx} &\leftarrow \text{rcx} \wedge \text{rbx} \\
&\qquad = (\neg\,[\text{rbp}]) \wedge (\neg\,[\text{rbp}+4]) \\
&\qquad = [\text{rbp}] \downarrow [\text{rbp}+4] \\
[\text{rbp}+4] &\leftarrow \text{rcx} = [\text{rbp}] \downarrow [\text{rbp}+4]
\end{aligned}
$$

## Symbolic Execution

```
__handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
  and   rcx, rbx
  mov   [rbp + 4], rcx
• pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)

$$\begin{aligned}
\text{rcx} &\leftarrow [\text{rbp}] \\
\text{rbx} &\leftarrow [\text{rbp}+4] \\
\text{rcx} &\leftarrow \neg\,\text{rcx} = \neg\,[\text{rbp}] \\
\text{rbx} &\leftarrow \neg\,\text{rbx} = \neg\,[\text{rbp}+4] \\
\text{rcx} &\leftarrow \text{rcx} \wedge \text{rbx} \\
&\qquad = (\neg\,[\text{rbp}]) \wedge (\neg\,[\text{rbp}+4]) \\
&\qquad = [\text{rbp}] \downarrow [\text{rbp}+4] \\
[\text{rbp}+4] &\leftarrow \text{rcx} = [\text{rbp}] \downarrow [\text{rbp}+4] \\
\\
\text{rsp} &\leftarrow \text{rsp} - 4 \\
[\text{rsp}] &\leftarrow \text{flags}
\end{aligned}$$

## Symbolic Execution

```
__handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
  and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
• pop   [rbp]
  jmp   __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)

$$\begin{aligned}
\text{rcx} &\leftarrow [\text{rbp}] \\
\text{rbx} &\leftarrow [\text{rbp}+4] \\
\text{rcx} &\leftarrow \neg\,\text{rcx} = \neg\,[\text{rbp}] \\
\text{rbx} &\leftarrow \neg\,\text{rbx} = \neg\,[\text{rbp}+4] \\
\text{rcx} &\leftarrow \text{rcx} \wedge \text{rbx} \\
&\qquad = (\neg\,[\text{rbp}]) \wedge (\neg\,[\text{rbp}+4]) \\
&\qquad = [\text{rbp}] \downarrow [\text{rbp}+4] \\
[\text{rbp}+4] &\leftarrow \text{rcx} = [\text{rbp}] \downarrow [\text{rbp}+4] \\
\\
\text{rsp} &\leftarrow \text{rsp} - 4 \\
[\text{rsp}] &\leftarrow \text{flags} \\
[\text{rbp}] &\leftarrow [\text{rsp}] = \text{flags} \\
\text{rsp} &\leftarrow \text{rsp} + 4
\end{aligned}$$

## Symbolic Execution

```
__handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not
  and
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
• jmp   __vm_dispatcher
```

$$
\begin{aligned}
\text{rcx} &\leftarrow [\text{rbp}] \\
\text{rbx} &\leftarrow [\text{rbp} + 4] \\
\text{rcx} &\leftarrow \neg\,\text{rcx} = \neg\,[\text{rbp}] \\
\text{rbx} &\leftarrow \neg\,\text{rbx} = \neg\,[\text{rbp} + 4]
\end{aligned}
$$

$$\boxed{[\text{rbp} + 4] \leftarrow ([\text{rbp}] \downarrow [\text{rbp} + 4])}$$

$$
\begin{aligned}
&= [\text{rbp}] \downarrow [\text{rbp} + 4] \\
[\text{rbp} + 4] &\leftarrow \text{rcx} = [\text{rbp}] \downarrow [\text{rbp} + 4]
\end{aligned}
$$

$$
\begin{aligned}
\text{rsp} &\leftarrow \text{rsp} - 4 \\
[\text{rsp}] &\leftarrow \text{flags} \\
[\text{rbp}] &\leftarrow [\text{rsp}] = \text{flags} \\
\text{rsp} &\leftarrow \text{rsp} + 4
\end{aligned}
$$

Handler performing **nor**
(with flag side-effects)

## Virtual Machine Handler

```
mov     eax, dword [rbp]
mov     ecx, dword [rbp + 4]
cmp     r11w, r13w
sub     rbp, 4
not     eax
clc
cmc
cmp     rdx, 0x28b105fa
not     ecx
cmp     r12b, r9b
cmc
and     eax, ecx
jmp     0xc239
mov     word [rbp + 8], eax
pushfq
movzx   eax, r10w
and     ax, di
pop     qword [rbp]
sub     rsi, 4
shld    rax, rdx, 0x1b
xor     ah, 0x4d
mov     eax, dword [rsi]
cmp     ecx, r11d
test    r10, 0x179708d5
xor     eax, ebx
jmp     0xfffffffffff63380
dec     eax
stc
ror     eax, 1
jmp     0xffffffffffff2a70
dec     eax
clc
bswap   eax
test    bp, 0x5124
neg     eax
test    dil, 0xe9
cmp     bx, r14w
cmc
push    rbx
sub     bx, 0x49f8
xor     dword [rsp], eax
and     bh, 0xaf
pop     rbx
movsxd  rax, eax
test    r13b, 0x94
add     rdi, rax
jmp     0xfffffffffffc67c7
lea     rax, [rsp + 0x140]
cmp     rbp, rax
ja      0x6557b
jmp     rdi
```

# Virtual Machine Handler

```
IRDst = ((- {((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0 8, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init
+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[8:16] 16 24, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32})[31:32]?((RDI_init+{(- {((((@32[(RSI_init
+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0 8, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init
[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[8:16] 16 24, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32}) 0 32, 0xFFFFFFFF 32 64}),(RDI_init+{(- {((((@32[(RSI_init
+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0 8, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init
[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[8:16] 16 24, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32}) 0 32, 0x0 32 64}))) pf = parity(((RBP_init+(- RSP_init)+0xFFFFFFFFFFFFFEBC)&0xFF))
cf = ((((RBP_init+(- RSP_init)+0xFFFFFFFFFFFFFEBC)^(RBP_init+0xFFFFFFFFFFFFFFFC))&((RBP_init+0xFFFFFFFFFFFFFFFC)^(RSP_init+0x140)))^(RBP_init+(- RSP_init)+0xFFFFFFFFFFFFFEBC)^(RBP_init+0xFFFFFFFFFFFFFFFC)^(RSP_init+0x140))[63:64] af =
((RBP_init+(- RSP_init)+0xFFFFFFFFFFFFFEBC)^(RBP_init+0xFFFFFFFFFFFFFFFC)^(RSP_init+0x140))[4:5] @64[(RBP_init+0xFFFFFFFFFFFFFFFC)] = (((@32[RBP_init]^0xFFFFFFFF)&(@32[(RBP_init+0x4)]^0xFFFFFFFF))?({0x2 0 2, parity(((@32
[RBP_init]^0xFFFFFFFF)&(@32[(RBP_init+0x4)]^0xFFFFFFFF)&0xFF)) 2 3, 0x0 3 4, (((- R9_init[0:8])+R12_init[0:8])^R12_init[0:8]^R9_init[0:8])[4:5] 4 5, 0x0 5 7, ((@32[RBP_init]^0xFFFFFFFF)&(@32[(RBP_init+0x4)]^0xFFFFFFFF))[31:32] 7 8, tf_init 8 9, i_f_init
9 10, df_init 10 11, 0x0 11 12, iopl_f_init 12 14, nt_init 14 15, 0x0 15 16, rf_init 16 17, vm_init 17 18, ac_init 18 19, vif_init 19 20, vip_init 20 21, i_d_init 21 22, 0x0 22 64},{0x2 0 2, parity(((@32[RBP_init]^0xFFFFFFFF)&(@32[(RBP_init
+0x4)]^0xFFFFFFFF)&0xFF)) 2 3, 0x0 3 4, (((- R9_init[0:8])+R12_init[0:8])^R12_init[0:8]^R9_init[0:8])[4:5] 4 5, 0x2 5 7, ((@32[RBP_init]^0xFFFFFFFF)&(@32[(RBP_init+0x4)]^0xFFFFFFFF))[31:32] 7 8, tf_init 8 9, i_f_init 9 10, df_init 10 11, 0x0 11 12,
iopl_f_init 12 14, nt_init 14 15, 0x0 15 16, rf_init 16 17, vm_init 17 18, ac_init 18 19, vif_init 19 20, vip_init 20 21, i_d_init 21 22, 0x0 22 64})) RAX = (RSP_init+0x140) of = (((RBP_init+(- RSP_init)+0xFFFFFFFFFFFFFEBC)^(RBP_init
+0xFFFFFFFFFFFFFFFC))&((RBP_init+0xFFFFFFFFFFFFFFFC)^(RSP_init+0x140)))[63:64] RCX = {(@32[(RBP_init+0x4)]^0xFFFFFFFF) 0 32, 0x0 32 64} @32[(RBP_init+0x4)] = ((@32[RBP_init]^0xFFFFFFFF)&(@32[(RBP_init+0x4)]^0xFFFFFFFF)) RBX =
{((- {((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0 8, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init
+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[8:16] 16 24, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32})^RBX_init[0:32]) 0 32, RBX_init[32:64] 32 64} RBP =
(RBP_init+0xFFFFFFFFFFFFFFFC) zf = ((RBP_init+(- RSP_init)+0xFFFFFFFFFFFFFEBC)?(0x0,0x1)) RSI = (RSI_init+0xFFFFFFFFFFFFFFFC) RDI = ((- {((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0
8, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[8:16] 16 24, ((((@32[(RSI_init
+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32})[31:32]?((RDI_init+{(- {((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0 8, ((((@32[(RSI_init
+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[8:16] 16 24, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init
[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32}) 0 32, 0xFFFFFFFF 32 64}),(RDI_init+{(- {((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0 8, ((((@32[(RSI_init
+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[8:16] 16 24, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init
[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32}) 0 32, 0x0 32 64}))) nf = (RBP_init+(- RSP_init)+0xFFFFFFFFFFFFFEBC)[63:64] @32[(RSP_init+0xFFFFFFFFFFFFFFF8)] = ((- {((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])
+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0 8, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)
[8:16] 16 24, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32})^RBX_init[0:32]) @32[(RSP_init+0xFFFFFFFFFFFFFFFC)] = RBX_init[32:64]
```

# Virtual Machine Handler

```
IRDst = ((- {((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0 8, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init
+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[8:16] 16 24, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32})[31:32]?((RDI_init+{(- {((((@32[(RSI_init
+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0 8, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init
[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[8:16] 16 24, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32}) 0 32, 0xFFFFFFFF 32 64}),(RDI_init+{(- {((((@32[(RSI_init
+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0 8, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init
[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[8:16] 16 24, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32}) 0 32, 0x0 32 64})))    pf = parity(((RBP_init+(- RSP_init)+0xFFFFFFFFFFFFFEBC)&0xFF))
cf = ((((RBP_init+(- RSP_init)+0xFFFFFFFFFFFFFEBC)^(RBP_init+0xFFFFFFFFFFFFFFFC))&((RBP_init+0xFFFFFFFFFFFFFFFC)^(RSP_init+0x140)))^(RBP_init+(- RSP_init)+0xFFFFFFFFFFFFFEBC)^(RBP_init+0xFFFFFFFFFFFFFFFC)^(RSP_init+0x140))[63:64]    af =
((RBP_init+(- RSP_init)+0xFFFFFFFFFFFFFEBC)^(RBP_init+0xFFFFFFFFFFFFFFFC)^(RSP_init+0x140))[4:5]    @64[(RBP_init+0xFFFFFFFFFFFFFFFC)] = ((@32[RBP_init]^0xFFFFFFFF)&(@32[(RBP_init+0x4)]^0xFFFFFFFF))?({0x2 0 2, parity(((@32
[RBP_init]^0xFFFFFFFF)&(@32[(RBP_init+0x4)]^0xFFFFFFFF[illegible]xF[illegible] 3, 0x0 3 4, (((- R9_init[0:8])+[illegible]_init[0:8])^[illegible]ir[illegible]8]^R9_i[illegible]:8])[4:5] 4 5, 0x0 5 7 [illegible]@32[RBP_in[illegible]0x[illegible]FFFF)&([illegible](RBP_init+0x4)]^0xFFFFFFFF))[31:32] 7 8, tf_init 8 9, i_f_init
9 10, df_init 10 11, 0x0 11 12, iopl_f_init 12 14, nt_in[illegible]0x0 15 16, rf_init 16 17, [illegible]nit 17 18, [illegible]19 20, [illegible]init 20 [illegible]i_d_init 2[illegible] 0 2, parity(((@32[RBP_init]^0xFFFFFFFF)&(@32[(RBP_init
+0x4)]^0xFFFFFFFF)&0xFF)) 2 3, 0x0 3 4, (((- R9_init[0:8[illegible]R12_i[illegible]_ini[illegible]RBI[illegible]]^0xF[illegible]FF)&([illegible] 7 8, tf_init 8 9, i_f_init 9 10, df_init 10 11, 0x0 11 12,
iopl_f_init 12 14, nt_init 14 15, 0x0 15 16, rf_init 16 1[illegible] ac_in[illegible]vif_i[illegible]t 2[illegible]2, 0x[illegible]64}))    [illegible] (((RBP_init+(- RSP_init)+0xFFFFFFFFFFFFFEBC)^(RBP_init
+0xFFFFFFFFFFFFFFFC))&((RBP_init+0xFFFFFFFFFFFFFFFC)[illegible]0x[illegible])))[63:[illegible] = [illegible]2[(RBP_[illegible]FF[illegible]F) ( [illegible], 0x0 [illegible]@[illegible](RBP_init[illegible]2[RB[illegible]it[illegible]xFFFFFFFF)&(@32[(RBP_init+0x4)]^0xFFFFFFFF))    RBX =
{((- {((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[[illegible]FF[illegible]) >>[illegible]0xFF[illegible]FF)[24:[illegible]@[illegible]RS[illegible]t+0x[illegible]FFFF[illegible])]^RBX[illegible]F) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init
+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>[illegible]1)+[illegible])[8:16] 16 24, ((((@3[illegible]SI_init+0xF[illegible]FFF[illegible]_init[[illegible]2[illegible]xFF[illegible]F) >>> 0x1[illegible]0xFF[illegible]32})^RBX_init[0:32]) 0 32, RBX_init[32:64] 32 64}    RBP =
(RBP_init+0xFFFFFFFFFFFFFFFC)    zf = ((RBP_init+(- RSP_init)+0xFFFF[illegible]FFEBC)?(0x0,0x1))    [illegible] = (RSI_init+0xFFFFFF[illegible]    RDI = ((- {((((@32[illegible]SI_init+0xFFFFFFFFFF[illegible]X_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0
8, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFF[illegible] >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xF[illegible]) >>> 0x1)+0xFFFFFFFF)[8:16] 16 24, ((((@32[(RSI_init
+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32})[31:32]?((RDI_init+{(- {((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0 8, ((((@32[(RSI_init
+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[8:16] 16 24, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init
[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32}) 0 32, 0xFFFFFFFF 32 64}),(RDI_init+{(- {((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0 8, ((((@32[(RSI_init
+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[8:16] 16 24, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init
[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32}) 0 32, 0x0 32 64})))    nf = (RBP_init+(- RSP_init)+0xFFFFFFFFFFFFFEBC)[63:64]    @32[(RSP_init+0xFFFFFFFFFFFFFFF8)] = ((- {((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])
+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[24:32] 0 8, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[16:24] 8 16, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)
[8:16] 16 24, ((((@32[(RSI_init+0xFFFFFFFFFFFFFFFC)]^RBX_init[0:32])+0xFFFFFFFF) >>> 0x1)+0xFFFFFFFF)[0:8] 24 32})^RBX_init[0:32])    @32[(RSP_init+0xFFFFFFFFFFFFFFFC)] = RBX_init[32:64]
```

$$M_1 = (\neg M_1) \wedge (\neg M_2)$$

## Mixed Boolean-Arithmetic Expression

```
int mixed_boolean(int A, int B, int C) {
    int result;

    result = (((1438524315 + ((((1438524315 + C) + 1438524315 * ((2956783114 - -1478456685 * C) |
    (-1478456685 * (1668620215 - A) - 2956783115))) + A) - 1553572265)) + 1438524315 * ((2956783114 -
    -1478456685 * ((((1438524315 + C) + 1438524315 * ((2956783114 - -1478456685 * C) | (-1478456685 *
    (1668620215 - A) - 2956783115))) + A) - 1553572265)) | (-1478456685 * (1668620215 - B) -
    2956783115))) - ((1438524315 + (1668620215 - ((((1438524315 + C) + 1438524315 * ((2956783114 -
    -1478456685 * C) | (-1478456685 * (1668620215 - A) - 2956783115))) + A) - 1553572265))) +
    1438524315 * ((2956783114 - -1478456685 * (1668620215 - ((((1438524315 + C) + 1438524315 *
    ((2956783114 - -1478456685 * C) | (-1478456685 * (1668620215 - A) - 2956783115))) + A) -
    1553572265))) | (-1478456685 * B - 2956783115)))) + 1553572265;

    return -1478456685 * result - 2956783115;
}
```

# Mixed Boolean-Arithmetic Expression

```
RAX = {((((- ((((- (((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+(@32[(RBP_init
+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+((((- (((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32
[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init
+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}*0xFFFFFFFFA7E08A93),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)]
[31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0x0 32 64}*0xFFFFFFFFA7E08A93)))[0:32])+0xB03CEE0A)|
((((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init
+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}*0x581F756D),({(((@32
[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0x0 32 64}*0x581F756D)))
[0:32]+0xB03CEE0A))[31:32]?(({((((- (((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]
+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32,
0xFFFFFFFF 32 64}*0xFFFFFFFFA7E08A93),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]
+0x5C99A1A9) 0 32, 0x0 32 64}*0xFFFFFFFFA7E08A93)))[0:32])+0xB03CEE0A)|((((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32,
0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}
*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}*0x581F756D),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32,
0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0x0 32 64}*0x581F756D)))[0:32]+0xB03CEE0A)) 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({((((- (((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32,
0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}
*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}*0xFFFFFFFFA7E08A93),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32,
0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0x0 32 64}*0xFFFFFFFFA7E08A93)))[0:32])+0xB03CEE0A)|((((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32
[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init
+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}*0x581F756D),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32
[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0x0 32 64}*0x581F756D)))[0:32]+0xB03CEE0A)) 0 32, 0x0 32 64}
*0x55BE239B)))[0:32]+0xB9334352)[31:32]?(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))
[0:32]+(@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+((((- (((@32[(RBP_init
+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init
+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}
*0xFFFFFFFFA7E08A93),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32,
0x0 32 64}*0xFFFFFFFFA7E08A93)))[0:32])+0xB03CEE0A)|((((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}
*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))
[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}*0x581F756D),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}
*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0x0 32 64}*0x581F756D)))[0:32]+0xB03CEE0A))[31:32]?(({((((- (((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init
+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init
+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}*0xFFFFFFFFA7E08A93),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}
*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0x0 32 64}*0xFFFFFFFFA7E08A93)))[0:32])+0xB03CEE0A)|((((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init
+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init
+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}*0x581F756D),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32
[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0x0 32 64}*0x581F756D)))[0:32]+0xB03CEE0A)) 0 32, 0xFFFFFFFF 32 64}
*0x55BE239B),({((((- (((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?
(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}
*0xFFFFFFFFA7E08A93),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32,
0x0 32 64}*0xFFFFFFFFA7E08A93)))[0:32])+0xB03CEE0A)|((((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}
*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))
[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}*0x581F756D),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}
*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0x0 32 64}*0x581F756D)))[0:32]+0xB03CEE0A)) 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0xB9334352) 0 32, 0xFFFFFFFF 32 64}*0x581F756D),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init
+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+(@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32
64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+((((- (((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init
+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init
+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}*0xFFFFFFFFA7E08A93),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}
*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0x0 32 64}*0xFFFFFFFFA7E08A93)))[0:32])+0xB03CEE0A)|((((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init
+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init
+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}*0x581F756D),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32
[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0x0 32 64}*0x581F756D)))[0:32]+0xB03CEE0A))[31:32]?(({((((- (((@32[(RBP_init
+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init
+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}
*0xFFFFFFFFA7E08A93),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32,
0x0 32 64}*0xFFFFFFFFA7E08A93)))[0:32])+0xB03CEE0A)|((((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}
*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}*0x55BE239B)))
[0:32]+0x5C99A1A9) 0 32, 0xFFFFFFFF 32 64}*0x581F756D),({(((@32[(RBP_init+0xFFFFFFFFFFFFFFFC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFFC)] 0 32, 0x0 32 64}
*0x55BE239B)))[0:32]+0x5C99A1A9) 0 32, 0x0 32 64}*0x581F756D)))[0:32]+0xB03CEE0A)) 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({((((- (((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}
*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+0x5C99A1A9)[31:32]?(({(((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),
```

## Mixed Boolean-Arithmetic Expression

```
RAX = {((((- ((((- (((@32[(RBP_init+0xFFFFFFFFFFFFFFEC)][31:32]?(({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0xFFFFFFFF 32 64}*0x55BE239B),({@32[(RBP_init+0xFFFFFFFFFFFFFFEC)] 0 32, 0x0 32 64}*0x55BE239B)))[0:32]+(@32[(RBP_init
[illegible]
```

$$RAX = (M_4 \mid M_0) - M_2$$

## Symbolic Execution

- ⊕ Captures full semantics of executed code
- ⊕ Computer algebra system, some degree of simplification
- ⊖ Usability decreases with increasing *syntactic* complexity
  - Artificial complexity (substitution, ...)
  - Algebraic complexity (MBA)

## Symbolic Execution

- ⊕ Captures full semantics of executed code
- ⊕ Computer algebra system, some degree of simplification
- ⊖ Usability decreases with increasing *syntactic* complexity
  - Artificial complexity (substitution, ...)
  - Algebraic complexity (MBA)

What if we could reason about *semantics* only instead of *syntax*?

## Program Synthesis

## Program Synthesis: A Semantic Approach

We use $f$ as a black-box:

$$f(x, y, z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$

# Program Synthesis: A Semantic Approach

We use $f$ as a black-box:

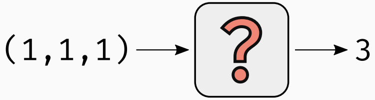
$$f(x,y,z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$

$$\begin{tikzcd}
\texttt{(1,1,1)} \arrow[r] & \boxed{?} \arrow[r] & 3
\end{tikzcd}$$

## Program Synthesis: A Semantic Approach

We use $f$ as a black-box:

$$f(x, y, z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$

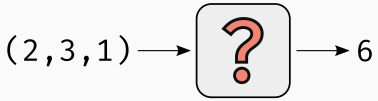


$(1, 1, 1) \rightarrow 3$

# Program Synthesis: A Semantic Approach

We use $f$ as a black-box:

$$f(x,y,z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$

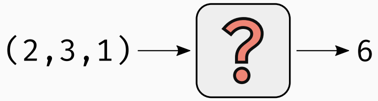


$(1,1,1) \rightarrow 3$

## Program Synthesis: A Semantic Approach

We use *f* as a black-box:

$$f(x,y,z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$



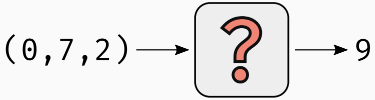
$(1,1,1) \rightarrow 3$

$(2,3,1) \rightarrow 6$

## Program Synthesis: A Semantic Approach

We use $f$ as a black-box:

$$f(x, y, z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$



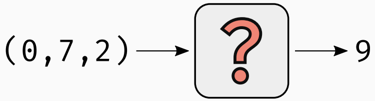
$(1, 1, 1) \rightarrow 3$

$(2, 3, 1) \rightarrow 6$

## Program Synthesis: A Semantic Approach

We use $f$ as a black-box:

$$f(x, y, z) := (((x \oplus y) + ((x \land y) \cdot 2)) \lor z) + (((x \oplus y) + ((x \land y) \cdot 2)) \land z)$$



$(1, 1, 1) \rightarrow 3$

$(2, 3, 1) \rightarrow 6$

$(0, 7, 2) \rightarrow 9$

## Program Synthesis: A Semantic Approach

We use $f$ as a black-box:

$$f(x, y, z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$

$$(1, 1, 1) \rightarrow 3$$
$$(2, 3, 1) \rightarrow 6$$
$$(0, 7, 2) \rightarrow 9$$

We **learn** a function that has the same I/O behavior:

## Program Synthesis: A Semantic Approach

RUHR UNIVERSITÄT BOCHUM RUB

We use $f$ as a black-box:

$$f(x, y, z) := (((x \oplus y) + ((x \wedge y) \cdot 2)) \vee z) + (((x \oplus y) + ((x \wedge y) \cdot 2)) \wedge z)$$

$$(1, 1, 1) \rightarrow 3$$
$$(2, 3, 1) \rightarrow 6$$
$$(0, 7, 2) \rightarrow 9$$

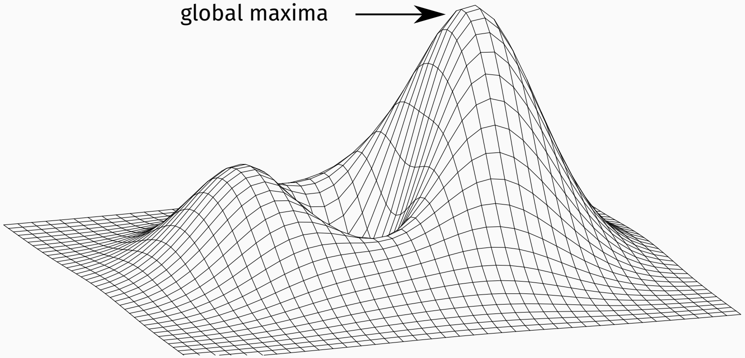
We **learn** a function that has the same I/O behavior:

$$h(x, y, z) := x + y + z$$

## How to synthesize programs?

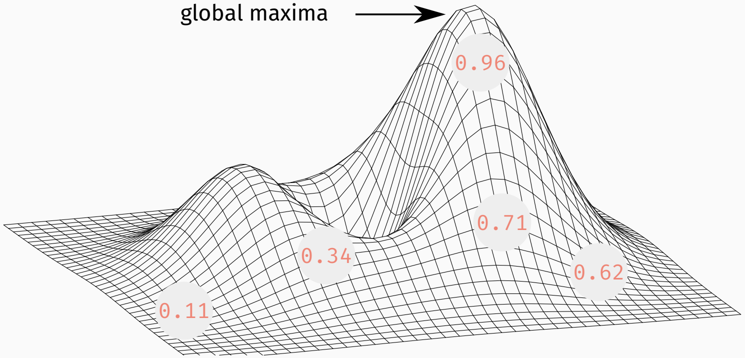
## Stochastic Program Synthesis

- probabilistic optimization problem



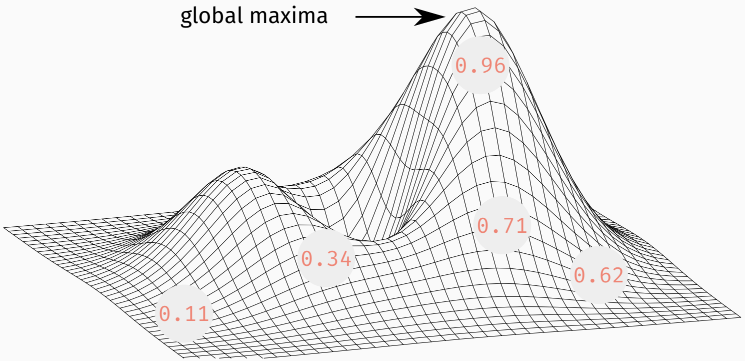
## Stochastic Program Synthesis

- probabilistic optimization problem



# Stochastic Program Synthesis

- probabilistic optimization problem
- based on Monte Carlo Tree Search (MCTS)



Let's synthesize: $a + b \mod 8$

## Program Generation

$$U \to U + U \mid U * U \mid a \mid b$$

## Program Generation

$$U \rightarrow U + U \mid U * U \mid a \mid b$$

- non-terminal symbol: $U$

## Program Generation

$$U \rightarrow U + U \mid U * U \mid a \mid b$$

- non-terminal symbol: $U$
- input variables: $\{a, b\}$

## Program Generation

$$U \rightarrow U + U \mid U * U \mid a \mid b$$

- non-terminal symbol: $U$
- input variables: $\{a, b\}$
- candidate programs: $a, \quad b, \quad a * b, \quad a + b, \quad \ldots$

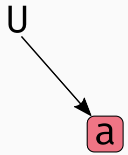
## Program Generation

$$U \to U + U \mid U * U \mid a \mid b$$

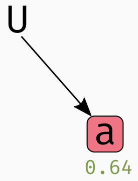
- non-terminal symbol: $U$
- input variables: $\{a, b\}$
- candidate programs: $a, \quad b, \quad a * b, \quad a + b, \quad \ldots$
- intermediate programs: $U + U, \quad U * U, \quad U + b, \quad \ldots$

## Monte Carlo Tree Search

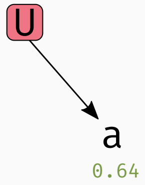
## Monte Carlo Tree Search



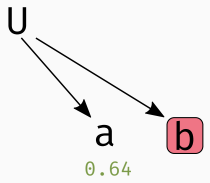
## Monte Carlo Tree Search



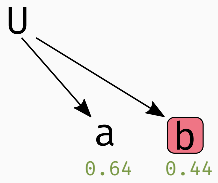
## Monte Carlo Tree Search



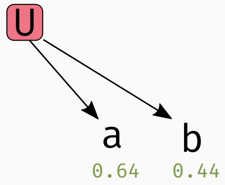
## Monte Carlo Tree Search



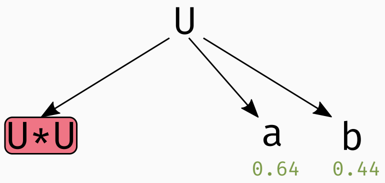
## Monte Carlo Tree Search



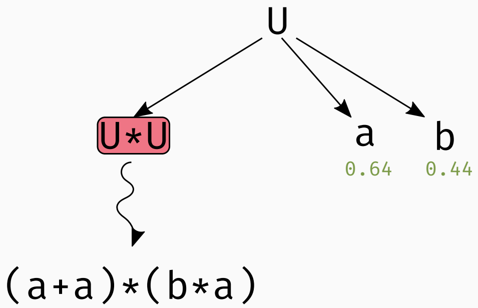
## Monte Carlo Tree Search



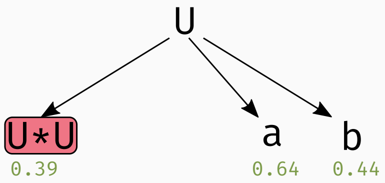
## Monte Carlo Tree Search



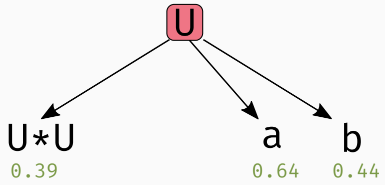
## Monte Carlo Tree Search



## Monte Carlo Tree Search



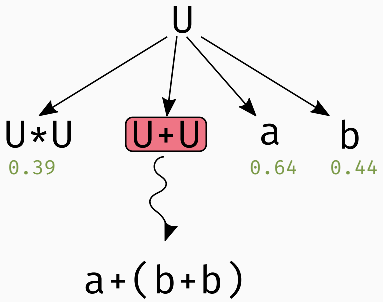
## Monte Carlo Tree Search



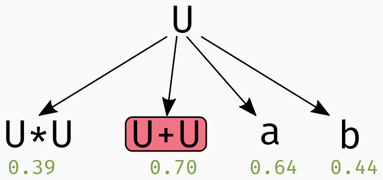
## Monte Carlo Tree Search

$$\begin{tikzcd}
 & U \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
U*U & \boxed{U+U} & a & b \\
0.39 & & 0.64 & 0.44
\end{tikzcd}$$

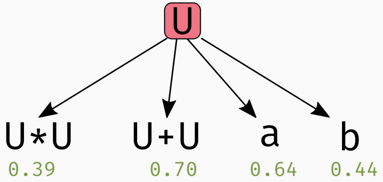
## Monte Carlo Tree Search



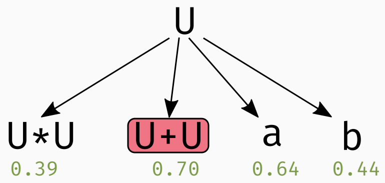
## Monte Carlo Tree Search



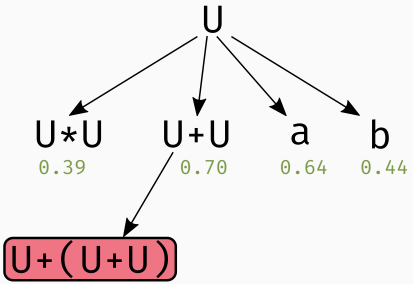
## Monte Carlo Tree Search



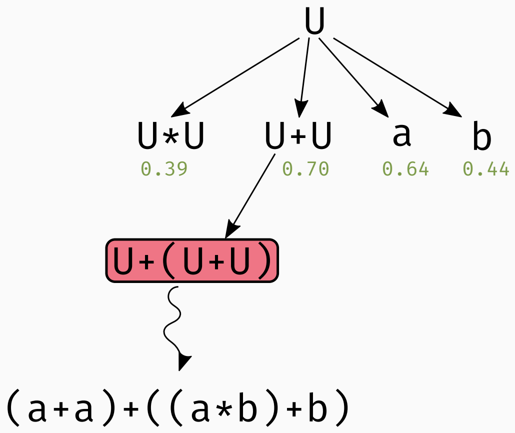
## Monte Carlo Tree Search



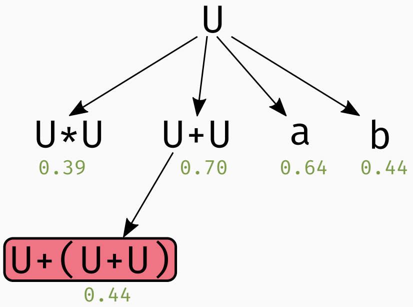
# Monte Carlo Tree Search



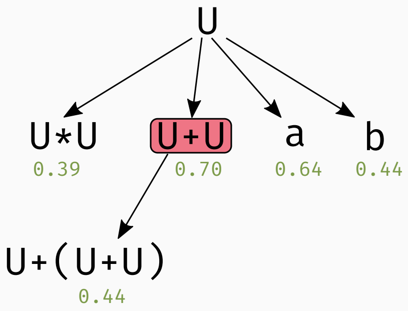
# Monte Carlo Tree Search



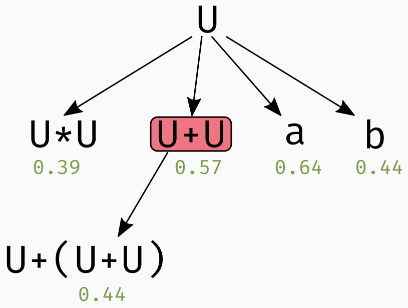
## Monte Carlo Tree Search



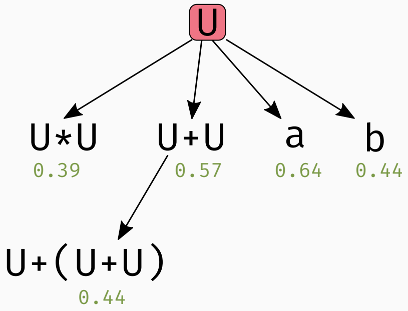
## Monte Carlo Tree Search



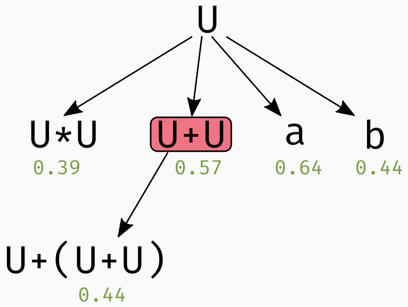
## Monte Carlo Tree Search



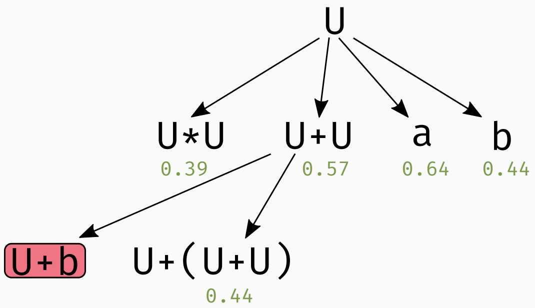
## Monte Carlo Tree Search



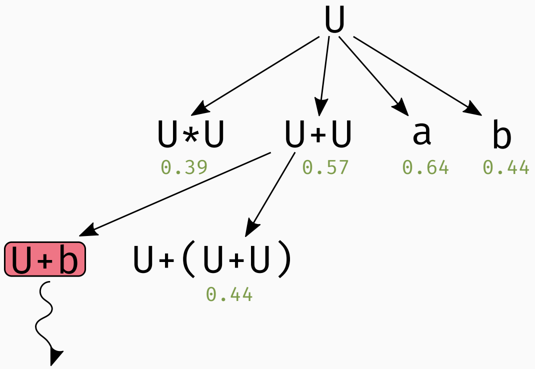
# Monte Carlo Tree Search



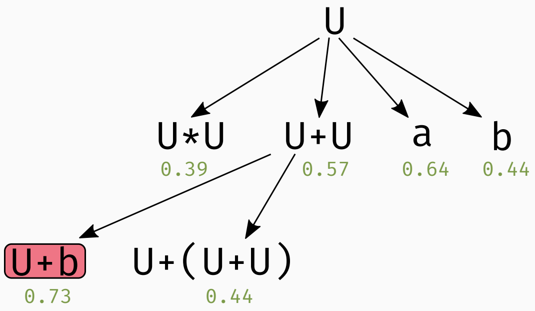
# Monte Carlo Tree Search



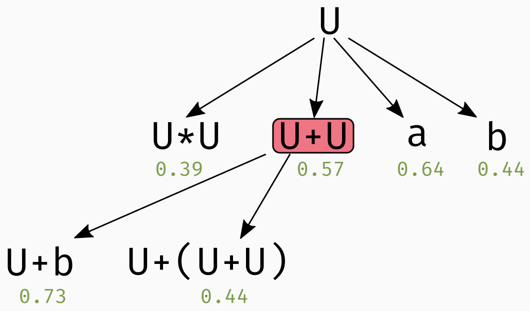
## Monte Carlo Tree Search



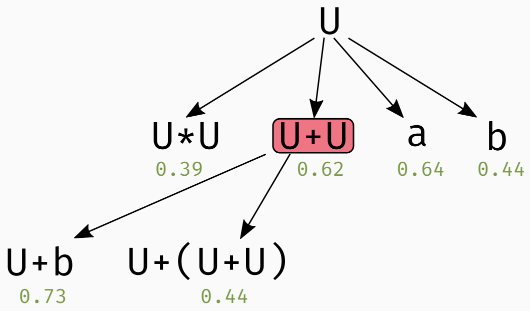
## Monte Carlo Tree Search



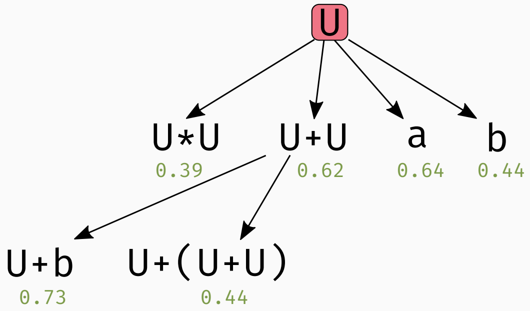
## Monte Carlo Tree Search



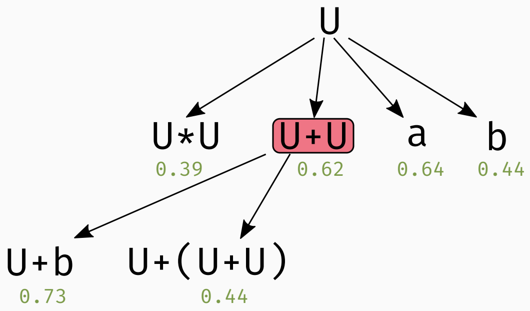
# Monte Carlo Tree Search



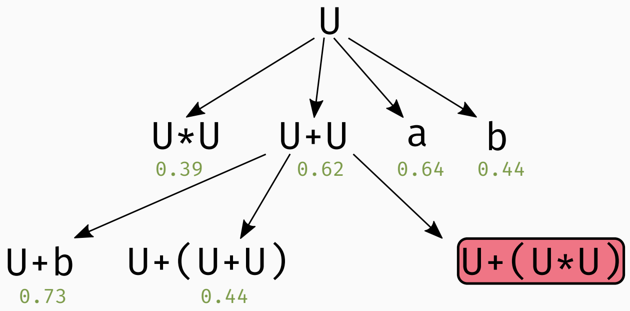
## Monte Carlo Tree Search



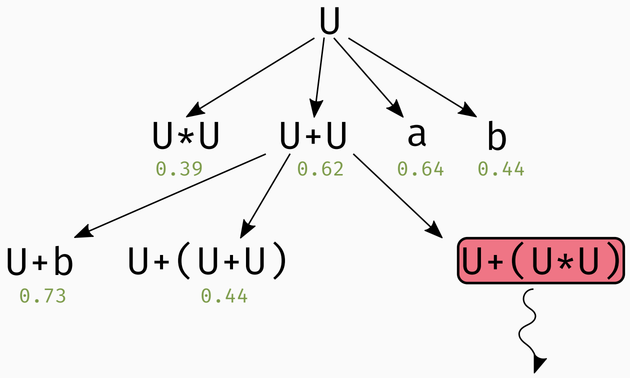
## Monte Carlo Tree Search



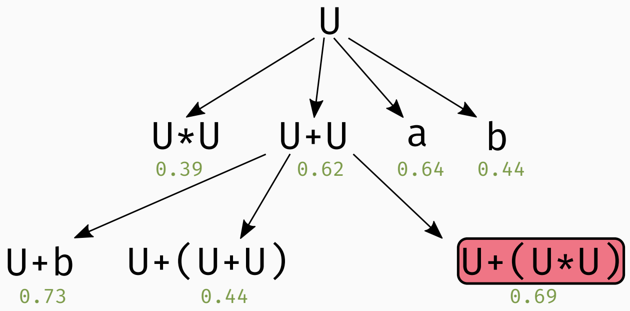
## Monte Carlo Tree Search



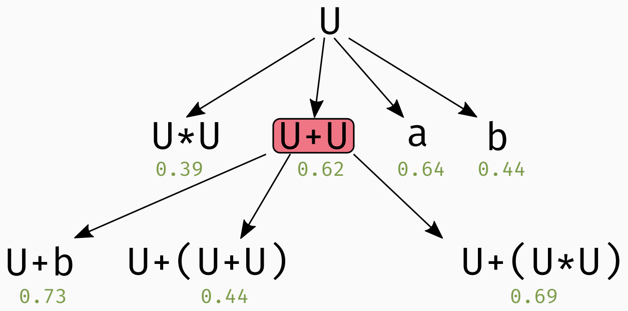
## Monte Carlo Tree Search



## Monte Carlo Tree Search



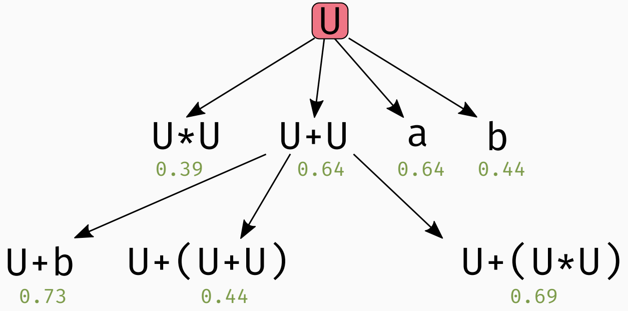
## Monte Carlo Tree Search



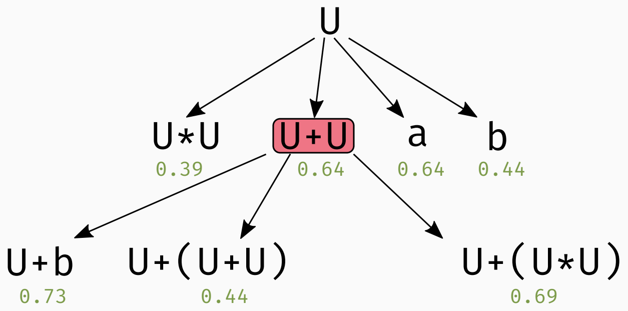
# Monte Carlo Tree Search

$$\begin{tikzcd}
 & & & \mathtt{U} \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & \\
 & & \mathtt{U*U}\ {\scriptstyle 0.39} & \boxed{\mathtt{U+U}}\ {\scriptstyle 0.64} \arrow[dll] \arrow[dl] \arrow[drr] & \mathtt{a}\ {\scriptstyle 0.64} & \mathtt{b}\ {\scriptstyle 0.44} \\
 \mathtt{U+b}\ {\scriptstyle 0.73} & \mathtt{U+(U+U)}\ {\scriptstyle 0.44} & & & & \mathtt{U+(U*U)}\ {\scriptstyle 0.69}
\end{tikzcd}$$

## Monte Carlo Tree Search



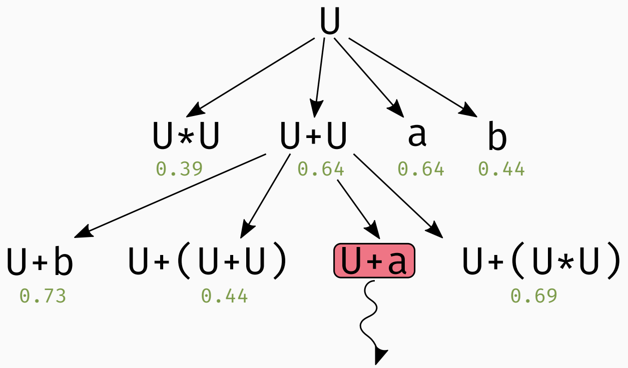
## Monte Carlo Tree Search



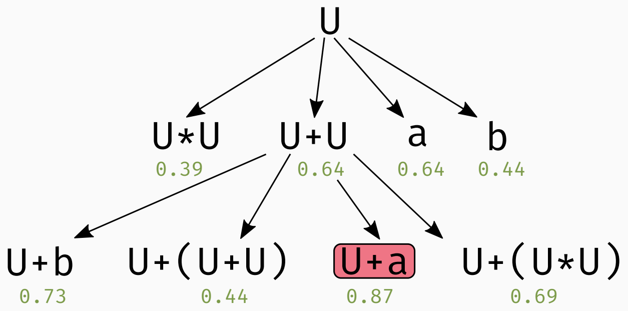
# Monte Carlo Tree Search

$$\begin{tikzcd}
 & & & U \arrow[dll] \arrow[d] \arrow[drr] \arrow[drrr] & & & \\
 & \underset{0.39}{U*U} & & \underset{0.64}{U+U} \arrow[dlll] \arrow[dl] \arrow[dr] \arrow[drrr] & & \underset{0.64}{a} & \underset{0.44}{b} \\
\underset{0.73}{U+b} & & \underset{0.44}{U+(U+U)} & & \boxed{U+a} & & \underset{0.69}{U+(U*U)}
\end{tikzcd}$$

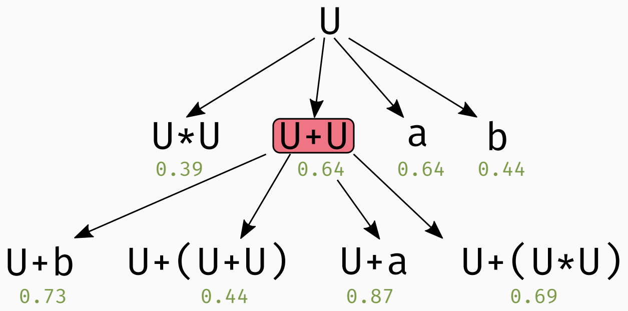
## Monte Carlo Tree Search



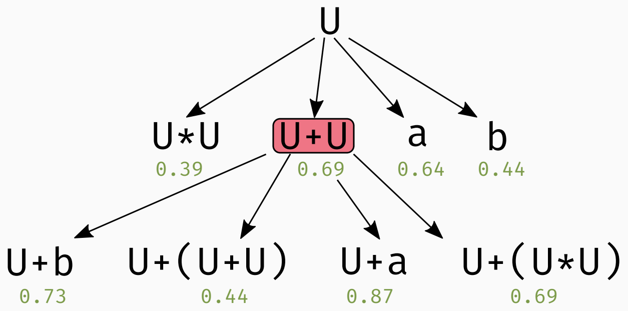
## Monte Carlo Tree Search



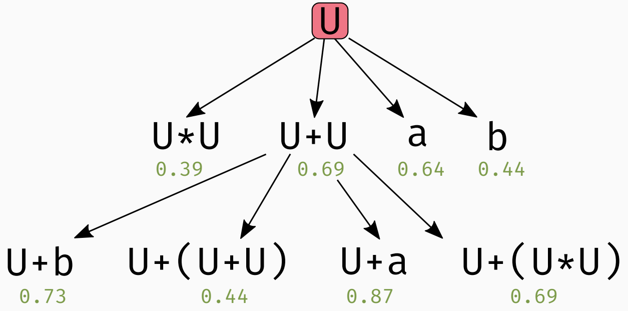
## Monte Carlo Tree Search



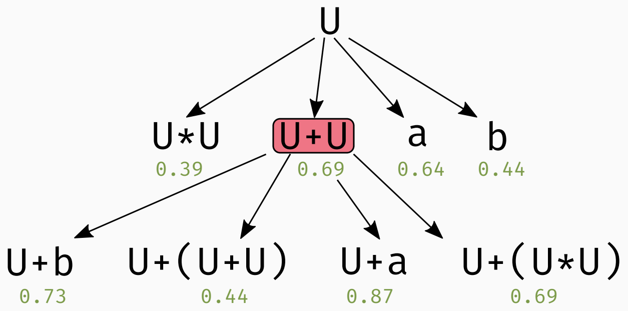
## Monte Carlo Tree Search



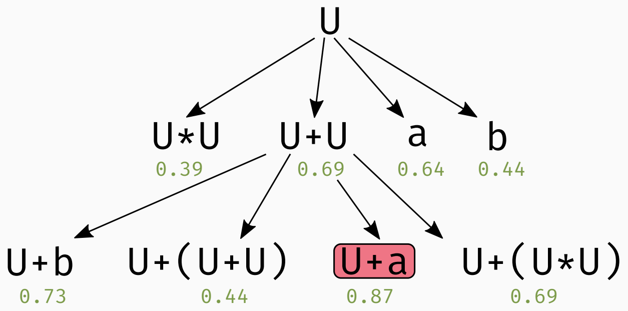
## Monte Carlo Tree Search



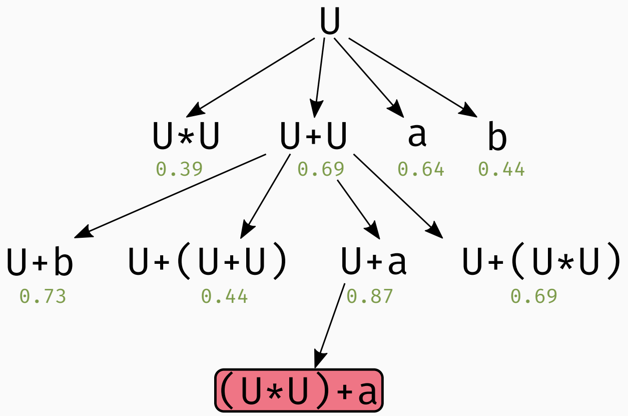
# Monte Carlo Tree Search



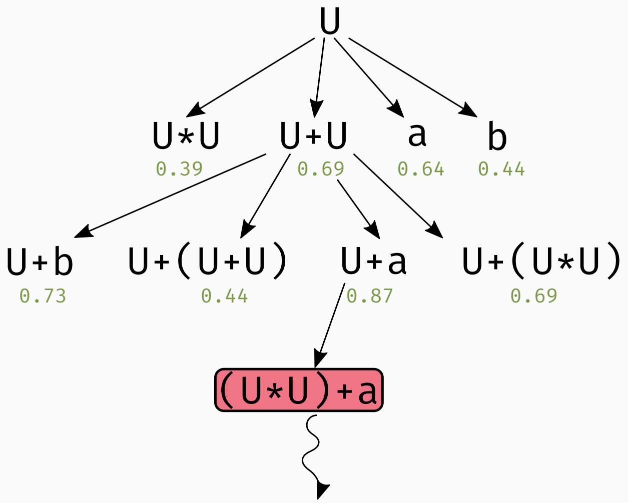
## Monte Carlo Tree Search



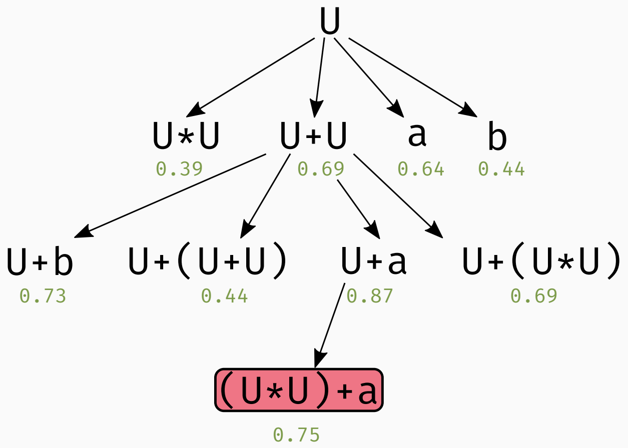
## Monte Carlo Tree Search



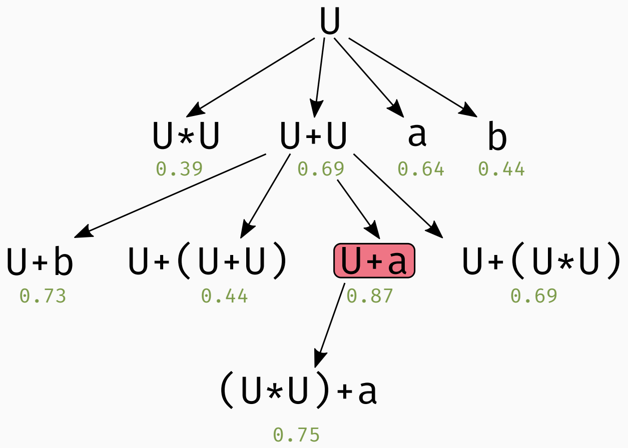
## Monte Carlo Tree Search



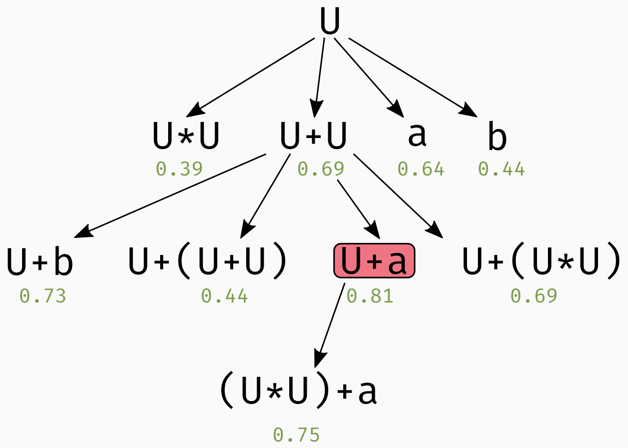
# Monte Carlo Tree Search



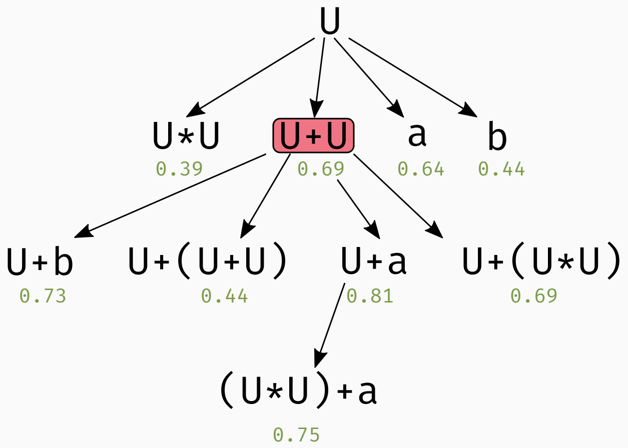
## Monte Carlo Tree Search



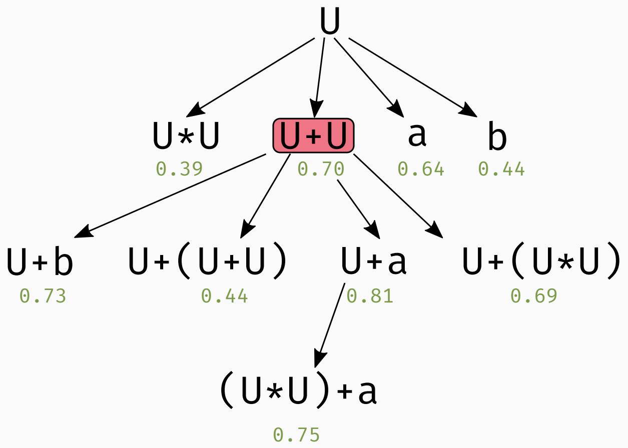
## Monte Carlo Tree Search



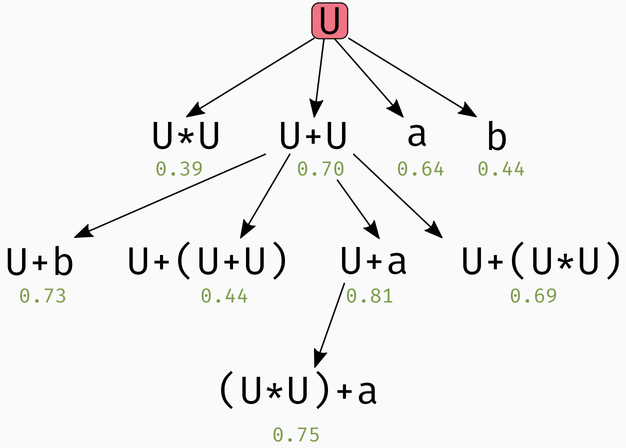
## Monte Carlo Tree Search



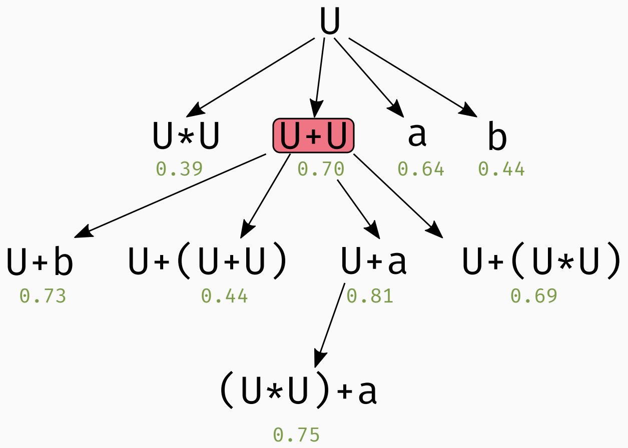
## Monte Carlo Tree Search



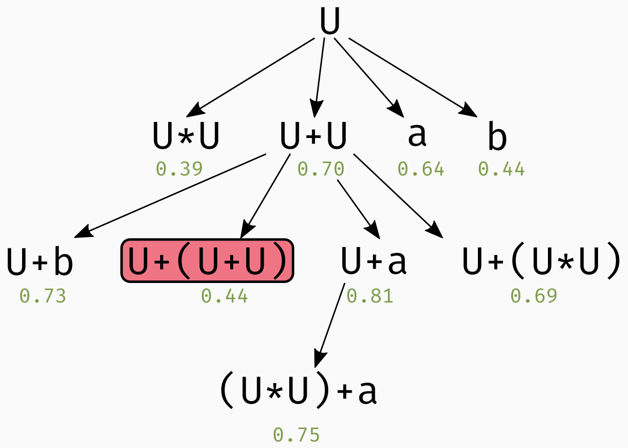
## Monte Carlo Tree Search



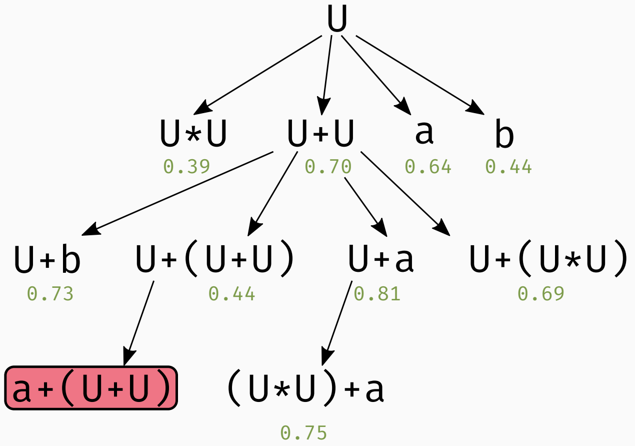
# Monte Carlo Tree Search



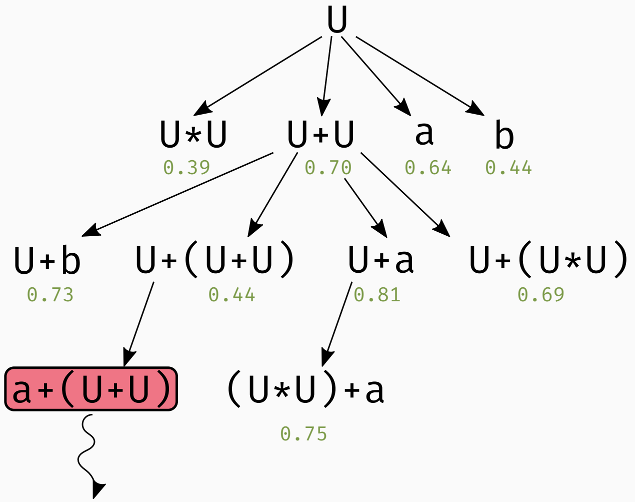
## Monte Carlo Tree Search



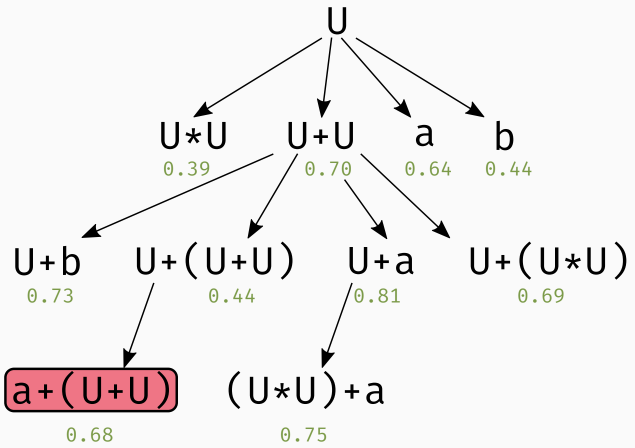
## Monte Carlo Tree Search



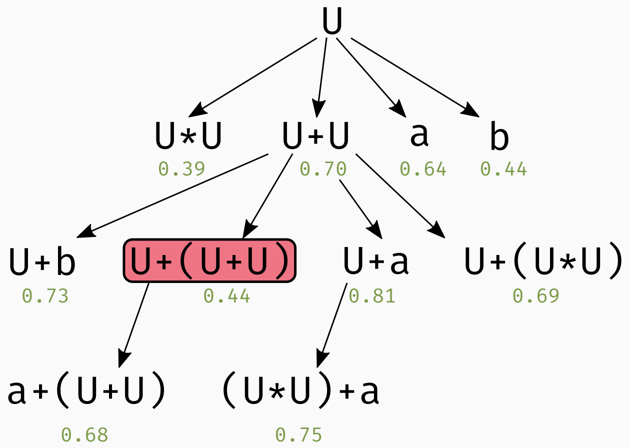
## Monte Carlo Tree Search



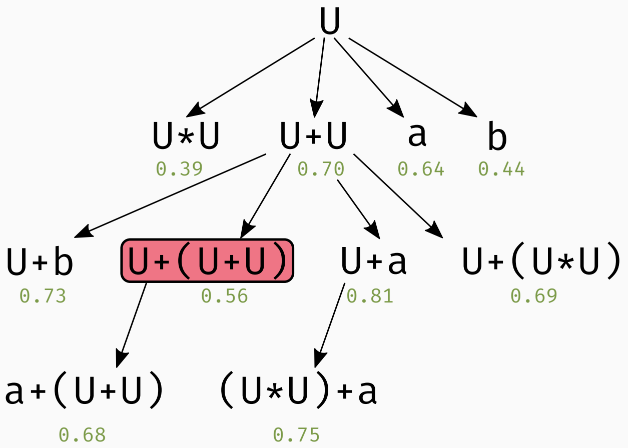
## Monte Carlo Tree Search



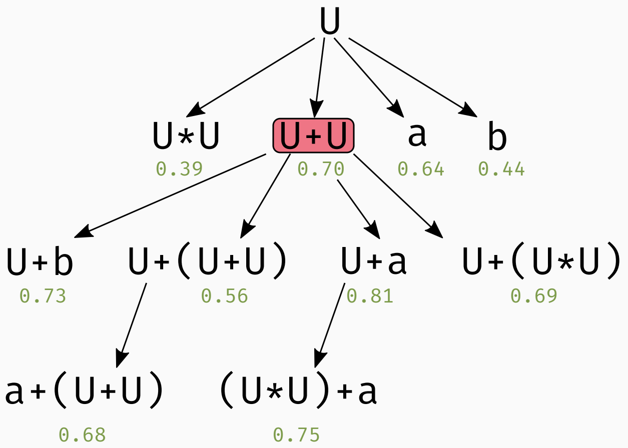
## Monte Carlo Tree Search



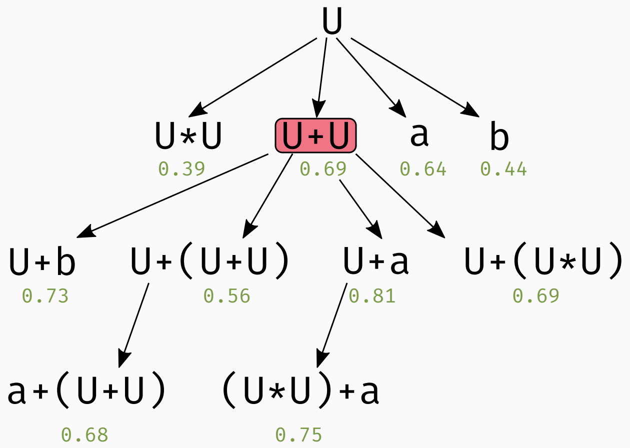
# Monte Carlo Tree Search



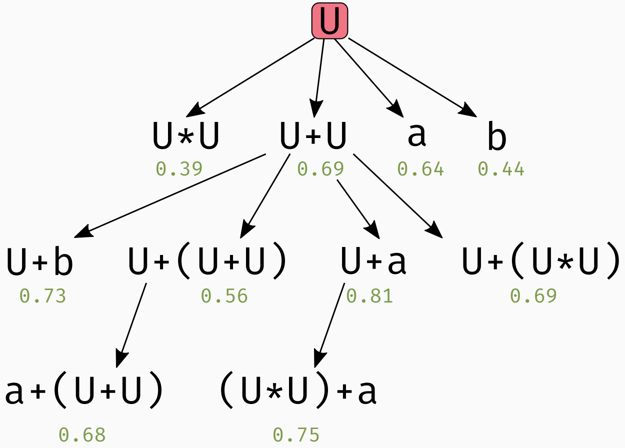
## Monte Carlo Tree Search



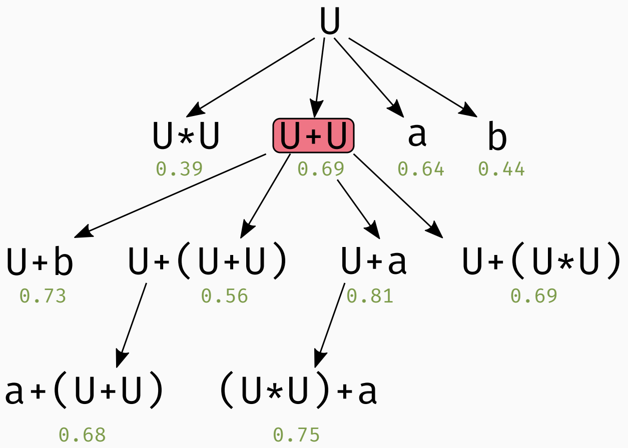
## Monte Carlo Tree Search



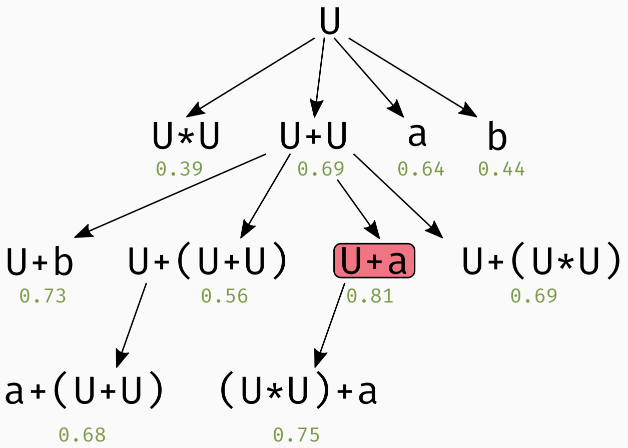
## Monte Carlo Tree Search



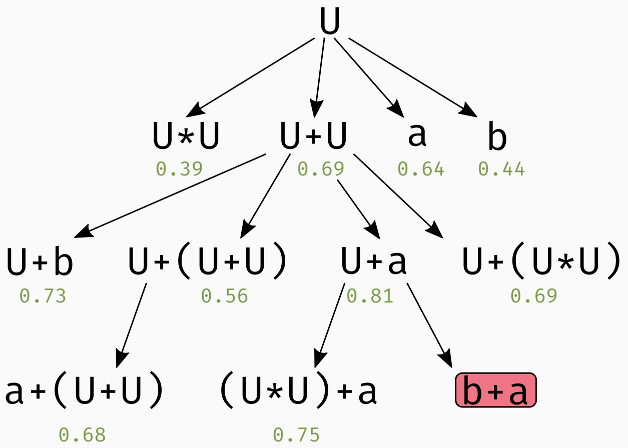
## Monte Carlo Tree Search



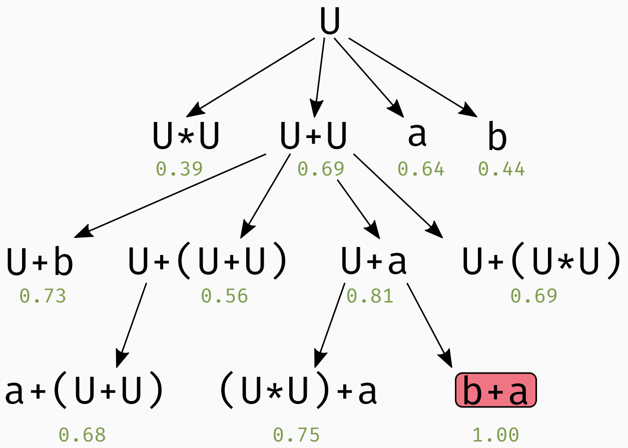
## Monte Carlo Tree Search



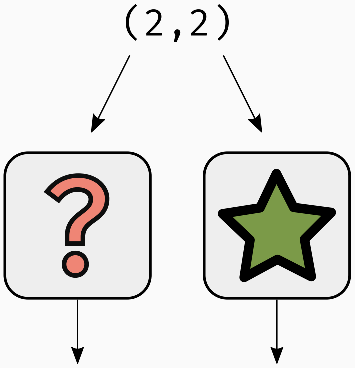
## Monte Carlo Tree Search



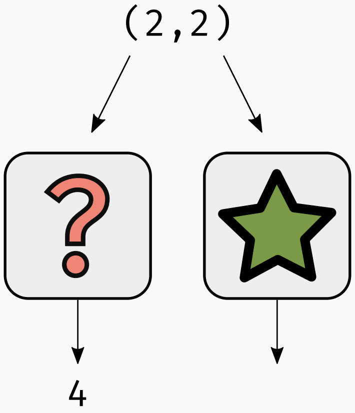
## Monte Carlo Tree Search



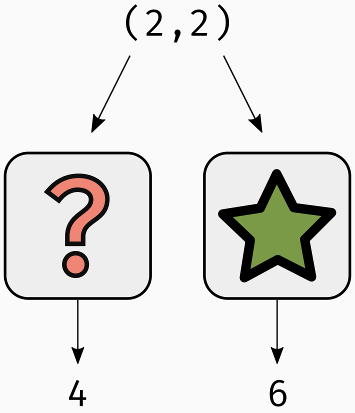
## Score Calculation



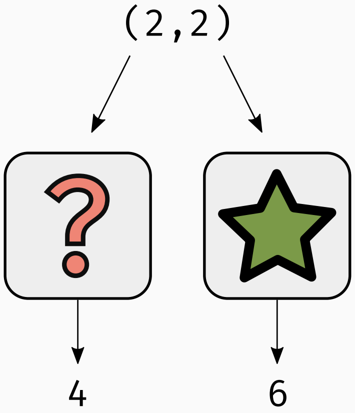
## Score Calculation



## Score Calculation

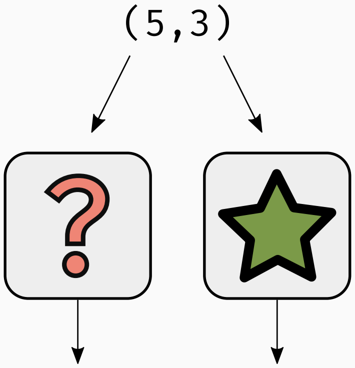


## Score Calculation



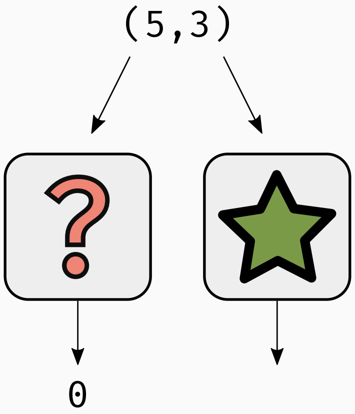
$$\text{similarity}(4, 6) \quad = \quad 0.78$$

## Score Calculation



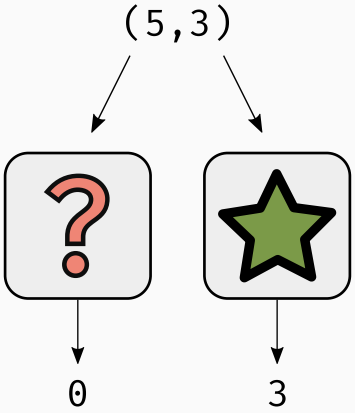
$$\text{similarity}(4, 6) \quad = \quad 0.78$$

# Score Calculation



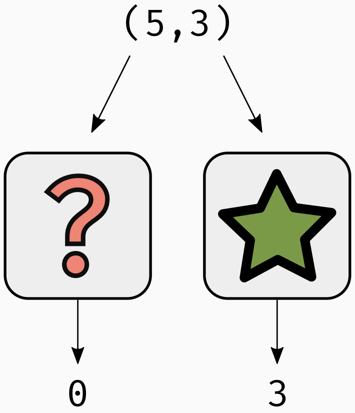
$$\text{similarity}(4,6) \quad = \quad 0.78$$

## Score Calculation



$$\text{similarity}(4,6) = 0.78$$

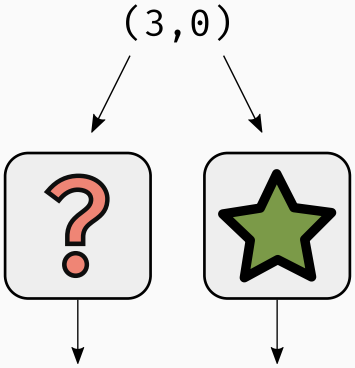
## Score Calculation



$$\text{similarity}(4,6) = 0.78$$

$$\text{similarity}(0,3) = 0.33$$

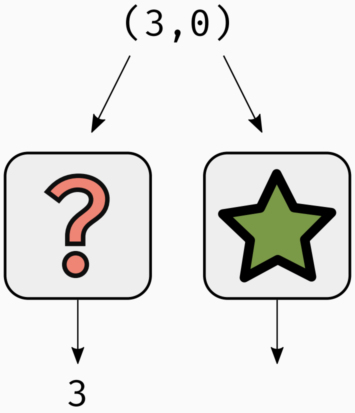
## Score Calculation



$$\text{similarity}(4, 6) = 0.78$$

$$\text{similarity}(0, 3) = 0.33$$

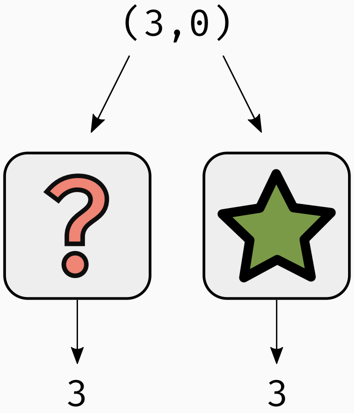
## Score Calculation



$$\text{similarity}(4,6) = 0.78$$

$$\text{similarity}(0,3) = 0.33$$

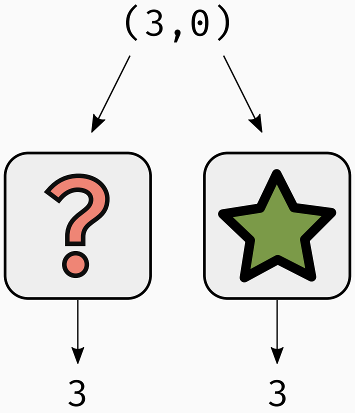
## Score Calculation



$$\text{similarity}(4,6) = 0.78$$

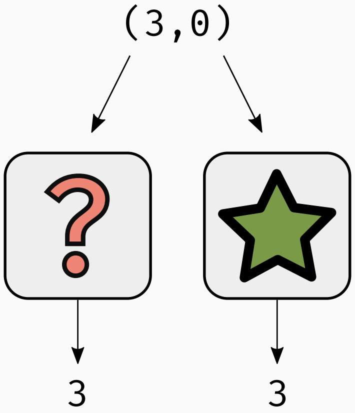
$$\text{similarity}(0,3) = 0.33$$

## Score Calculation



| | | |
|---|---|---|
| similarity(4, 6) | = | 0.78 |
| similarity(0, 3) | = | 0.33 |
| similarity(3, 3) | = | 1.0 |

## Score Calculation



| | | |
|---|---|---|
| similarity(4, 6) | = | 0.78 |
| similarity(0, 3) | = | 0.33 |
| similarity(3, 3) | = | 1.0 |

average score: 0.70

## Output Similarity: similarity$(O, O')$

```
11110111100100001000110010000000

11100010000110011110101100000000
```

Let’s compare:

## Output Similarity: similarity$(O, O')$

```
1111011110010000100011001000000
1110001000011001111010110000000
```

Are they in the same range?

## Output Similarity: similarity$(O, O')$

```
11110111100100001000110010000000
11100001000011001111010110000000
```

How many bits are different?

## Output Similarity: similarity$(O, O')$

```
11110111100100001000110010000000
00010101011101101010000110000000
11100010000110011110101100000000
```

How close are they numerically?

DEMO

## How to synthesize obfuscated code?

## Obtaining Code

static disassembly

## Obtaining Code

```
54 68 69 73 20 64 6f
65 73 6e 27 74 20 6c
6f 6f 6b 20 6c 69 6b
65 20 61 6e 79 74 68
69 6e 67 20 74 6f 20
6d 65 2e de ad be ef
```

static disassembly

memory dump

# Obtaining Code

```
54 68 69 73 20 64 6f
65 73 6e 27 74 20 6c
6f 6f 6b 20 6c 69 6b
65 20 61 6e 79 74 68
69 6e 67 20 74 6f 20
6d 65 2e de ad be ef
```

```
mov    r15, 0x200               mov    r15, rdx
xor    r15, 0x800               xor    r10d, dword ptr [r12]
mov    rbx, rbp                 sub    r15, 0x800
add    rbx, 0xc0                or     rdx, 0x400
mov    rbx, qword ptr [rbx]     mov    rsi, 0x200
mov    r13, 1                   mov    r14, rbp
mov    rcx, 0                   sub    rsi, rsi
mov    r15, rbp                 mov    rdi, rbp
add    r15, 0xc0                mov    r8, 0x400
or     rcx, 0x88                sub    rsi, r9
add    rbx, 0xb                 sub    r8, rsi
mov    r15, qword ptr [r15]     add    r14, 0
or     r12, 0xffffffff80000000  add    rsi, rax
sub    rcx, 0x78                and    r8, 0x88
movzx  r10, word ptr [rbx]      xor    rsi, r14
xor    r12, r13                 mov    rsi, rbp
add    r12, 0xffff              add    rdi, 0xc0
add    r15, 0                   sub    r8, rdi
mov    r8, rbp                  add    r8, 0x78
sub    rcx, 0x10                add    rsi, 4
or     r12, r12                 mov    rcx, 0x200
or     rcx, 0x800               mov    rdi, qword ptr [rdi]
movzx  r11, word ptr [r15]      add    dword ptr [rsi], 0x254
xor    rcx, 0x800               xor    rcx, 0xf0
mov    r12, r15                 add    rcx, r10
add    r8, 0                    add    rdi, 6
xor    r12, 0xf0                mov    r8, 0x400
mov    rbx, 0x58                mov    ax, word ptr [rdi]
add    r11, rbp                 mov    r8, 1
```

static disassembly

memory dump

instruction trace

## Learning Code Semantics

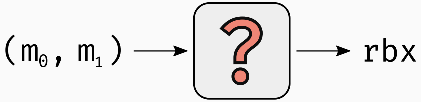
```
__handle_vnor:
mov   rcx, [rbp]
mov   rbx, [rbp + 4]
not   rcx
not   rbx
and   rcx, rbx
mov   [rbp + 4], rcx
pushf
pop   [rbp]
jmp   __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)

## Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
• not   rbx
  and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

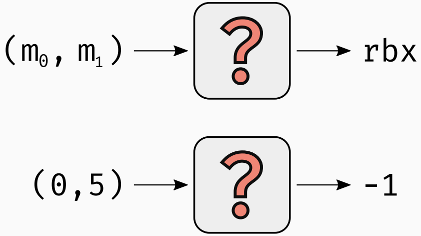
Handler performing **nor**
(with flag side-effects)



# Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
• not   rbx
  and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

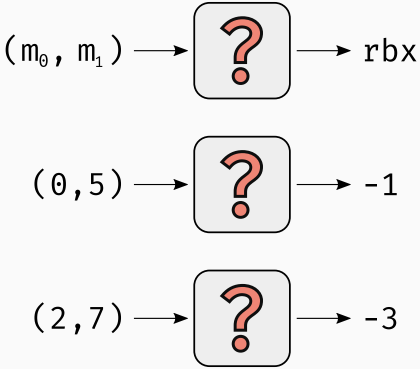
Handler performing **nor**
(with flag side-effects)



# Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
• not   rbx
  and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

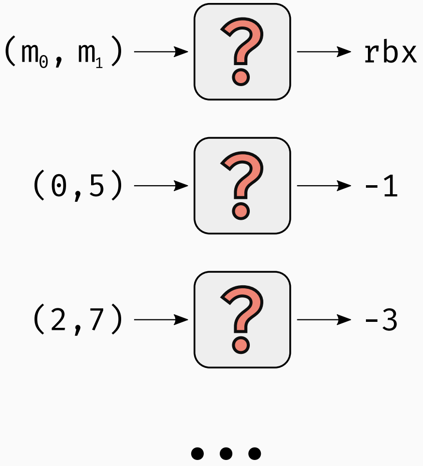
Handler performing **nor**
(with flag side-effects)



# Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
• not   rbx
  and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

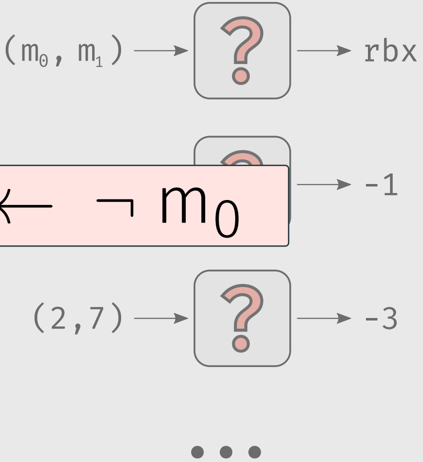
Handler performing **nor**
(with flag side-effects)



# Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
• not   rbx
  and   rcx, rbx
  mov   [rbp + 4], rc
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)



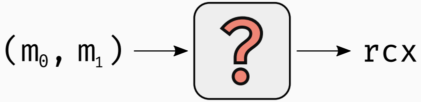
$(m_0, m_1) \rightarrow ? \rightarrow$ rbx
? $\rightarrow$ -1
$(2,7) \rightarrow ? \rightarrow$ -3
• • •

$$rbx \leftarrow \neg\, m_0$$

## Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
• and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

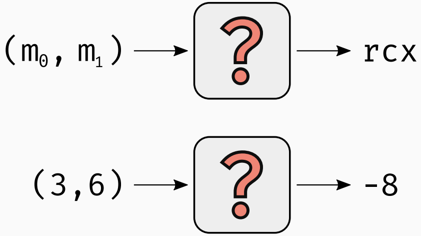
Handler performing **nor**
(with flag side-effects)



# Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
• and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

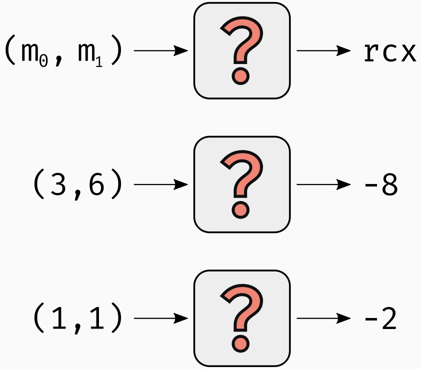
Handler performing **nor**
(with flag side-effects)



## Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
• and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

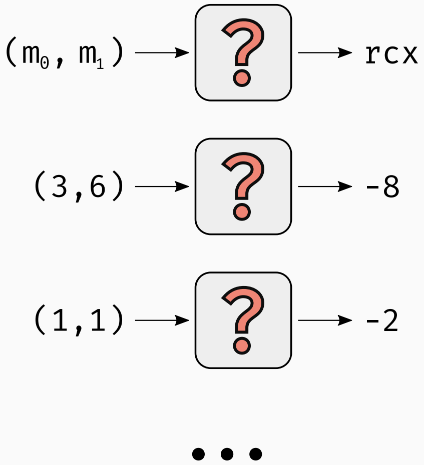
Handler performing **nor**
(with flag side-effects)



# Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
• and   rcx, rbx
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

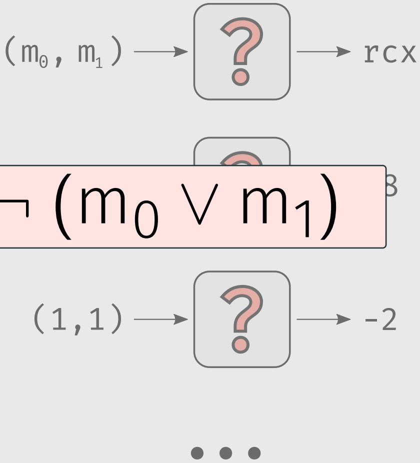
Handler performing `nor`
(with flag side-effects)



# Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
• and   rcx, rb
  mov   [rbp +
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

Handler performing nor
(with flag side-effects)

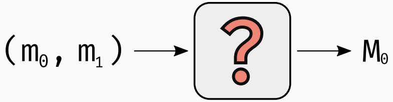


$$rcx \leftarrow \neg(m_0 \lor m_1)$$

## Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
  and   rcx, rbx
• mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

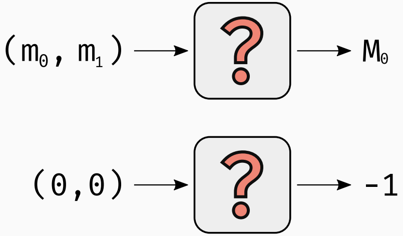
Handler performing **nor**
(with flag side-effects)



# Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
  and   rcx, rbx
• mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

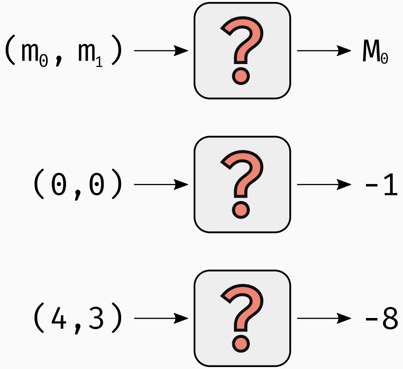
Handler performing **nor**
(with flag side-effects)



## Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
  and   rcx, rbx
• mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

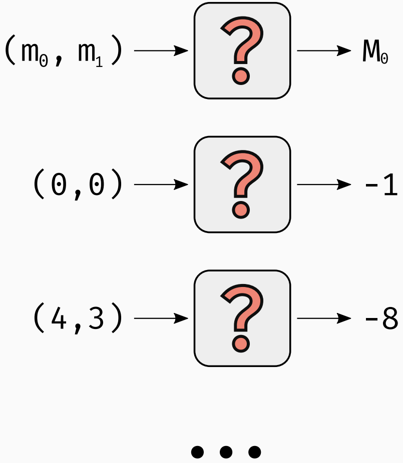
Handler performing **nor**
(with flag side-effects)


$$\begin{tikzcd}
(m_0, m_1) \arrow[r] & \boxed{?} \arrow[r] & M_0 \\
(0,0) \arrow[r] & \boxed{?} \arrow[r] & -1 \\
(4,3) \arrow[r] & \boxed{?} \arrow[r] & -8
\end{tikzcd}$$

## Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
  and   rcx, rbx
• mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

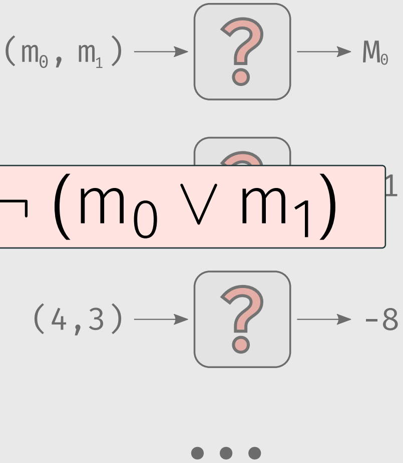
Handler performing nor
(with flag side-effects)


$(m_0, m_1) \rightarrow ? \rightarrow M_0$
$(0,0) \rightarrow ? \rightarrow -1$
$(4,3) \rightarrow ? \rightarrow -8$
• • •

## Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not   rbx
  and   rcx, rb
• mov   [rbp +
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

Handler performing nor
(with flag side-effects)



$$M_0 \leftarrow \neg(m_0 \vee m_1)$$

## Learning Code Semantics

```
  __handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
• not   rbx
• and   rcx, rbx
• mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
  jmp   __vm_dispatcher
```

Handler performing **nor**
(with flag side-effects)

$$\text{rbx} \leftarrow \neg\, m_0$$

$$\text{rcx} \leftarrow \neg\, (m_0 \vee m_1)$$

$$M_0 \leftarrow \neg\, (m_0 \vee m_1)$$

## I/O Sampling

RUHR UNIVERSITÄT BOCHUM RUB

WinDbg

Valgrind

x64dbg

Pin

DynamoRIO

bochs
think inside the bochs.

Unicorn

Miasm

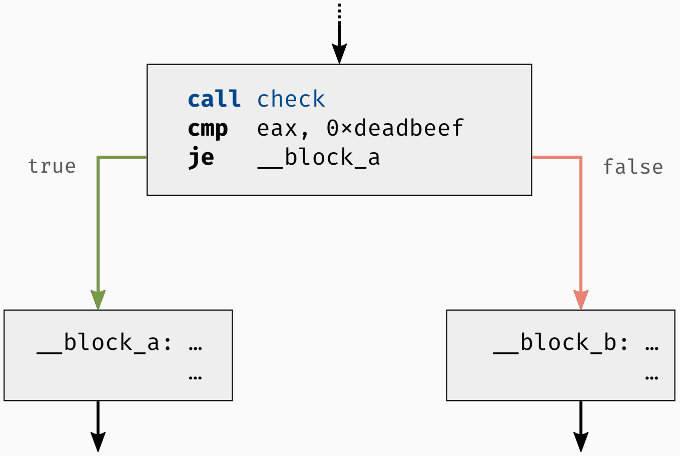
angr

TRITON
Dynamic Binary Analysis

<your tool here>

Metasm

## Instruction Trace: Forced Execution

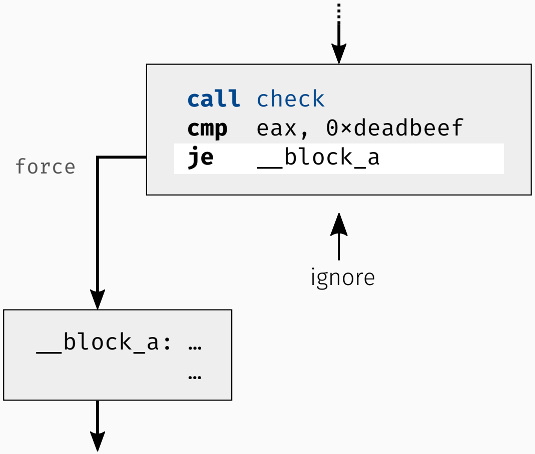


```
call check
cmp  eax, 0×deadbeef
je   __block_a
```

```
__block_a: …
           …
```

```
__block_b: …
           …
```

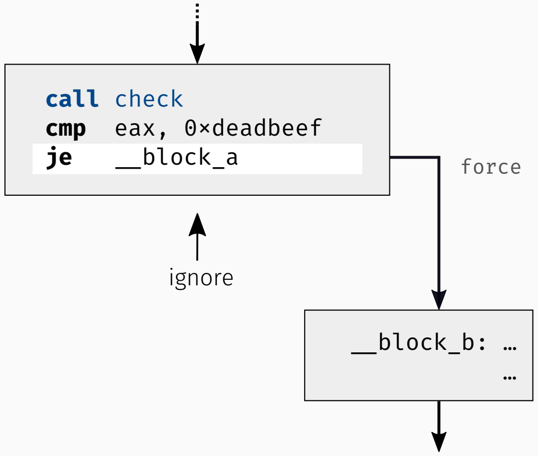
## Instruction Trace: Forced Execution



```
call check
cmp  eax, 0×deadbeef
je   __block_a
```

```
__block_a: …
           …
```

# Instruction Trace: Forced Execution



```
call check
cmp  eax, 0×deadbeef
je   __block_a
```

```
__block_b: …
           …
```

## Syntia

- program synthesis framework for code deobfuscation
- written in Python
- random I/O sampling for assembly code
- MCTS-based program synthesis

`https://github.com/RUB-SysSec/syntia`

DEMO

## Breaking Virtual Machine Obfuscation

## Reminder: Virtual Machine Hardening

**Hardening Technique #1** – Obfuscating individual VM components.

**Hardening Technique #2** – Duplicating VM handlers.

**Hardening Technique #3** – No central VM dispatcher.

**Hardening Technique #4** – No explicit handler table.

**Hardening Technique #5** – Blinding VM bytecode.

# #1: Obfuscating Individual VM Components

```
mov    r15, 0x200
xor    r15, 0x800
mov    rbx, rbp
add    rbx, 0xc0
mov    rbx, qword ptr [rbx]
mov    r13, 1
mov    rcx, 0
mov    r15, rbp
add    r15, 0xc0
or     rcx, 0x88
add    rbx, 0xb
mov    r15, qword ptr [r15]
or     r12, 0xffffffff80000000
sub    rcx, 0x78
movzx  r10, word ptr [rbx]
xor    r12, r13
add    r12, 0xffff
add    r15, 0
mov    r8, rbp
sub    rcx, 0x10
or     r12, r12
or     rcx, 0x800
movzx  r11, word ptr [r15]
xor    rcx, 0x800
mov    r12, r15
add    r8, 0
xor    r12, 0xf0
mov    rbx, 0x58
add    r11, rbp
xor    rbx, 0x800
and    r12, 0x20
add    rbx, 0x800
mov    r11, qword ptr [r11]
add    rbx, 1
and    r12, r9
mov    rdx, 1
xor    r10d, dword ptr [r8]
sub    r9, r11
pushfq
xor    rbx, 0xf0
xor    rbx, 0x800
and    rdx, r8
mov    r12, rbp
xor    rdx, 0x20
sub    rbx, 4
add    r11, 0x2549b044
or     rbx, 0x78
and    rdx, r10
mov    rax, 0
add    r12, 0x42
```

```
mov    r15, rdx
xor    r10d, dword ptr [r12]
sub    r15, 0x800
or     rdx, 0x400
mov    rsi, 0x200
mov    r14, rbp
sub    rsi, rsi
mov    rdi, rbp
mov    r8, 0x400
sub    rsi, r9
sub    r8, rsi
add    r14, 0
add    rsi, rax
and    r8, 0x88
xor    rsi, r14
mov    rsi, rbp
add    rdi, 0xc0
sub    r8, rdi
add    r8, 0x78
add    rsi, 4
mov    rcx, 0x200
mov    rdi, qword ptr [rdi]
add    dword ptr [rsi], 0x2549b044
xor    rcx, 0xf0
add    rcx, r10
add    rdi, 6
mov    r8, 0x400
mov    ax, word ptr [rdi]
mov    r8, 1
mov    rsi, rbp
and    rcx, 8
sub    rcx, 1
mov    rcx, rdi
add    rsi, 0x29
or     rcx, 8
mov    r8, rsi
add    rcx, 4
mov    r13b, byte ptr [rsi]
cmp    r13b, 0xd2
jbe    0x204
and    r8, r13
or     rcx, r13
or     rcx, 4
mov    rbx, rbp
or     rcx, 4
sub    rcx, 0x400
add    rax, rbp
or     rcx, 0x80
add    rcx, 0x80
add    rbx, 0x5a
```

```
add    r8, 1
or     r8, 0x78
add    word ptr [rbx], r10w
mov    r15, rax
sub    r15, rax
pop    r9
mov    rcx, rbp
add    rcx, 0xc0
mov    rcx, qword ptr [rcx]
add    rcx, 8
movzx  r10, word ptr [rcx]
mov    r9, rbp
add    r9, 0
xor    r10d, dword ptr [r9]
and    rdi, 0xffffffff80000000
sub    r13, 0xf0
mov    rsi, 0
sub    r13, 0x20
mov    rbx, rbp
or     r13, 0x88
and    rcx, 8
mov    r8, 0x58
add    rbx, 0xc0
mov    rbx, qword ptr [rbx]
sub    rcx, 0x20
add    rdi, 0x80
sub    r13, 0x10
add    rbx, 8
mov    si, word ptr [rbx]
or     r9, 0xffff
sub    r9, 1
mov    r9, rbp
mov    r12, 0x58
add    r9, 0
sub    r13, 0x80
mov    r15, r13
or     rcx, r12
xor    esi, dword ptr [r9]
mov    r10, rbp
add    r10, 0xcc
sub    r15, 0x20
xor    esi, dword ptr [r10]
xor    r13, 0x90
add    rdi, 0x10
mov    r14, rsi
mov    rdx, rbp
add    rdx, 0
add    dword ptr [rdx], esi
xor    r12, 1
mov    r13, r15
```

```
or     r14, r14
mov    rax, rbp
and    rcx, r13
add    rax, 4
sub    r8, -0x80000000
add    r13, 0xffff
and    rcx, 0x20
mov    r10, rbp
add    r13, r15
add    r14, r8
add    r10, 0x89
xor    word ptr [r10], si
xor    rdx, r11
mov    rsi, rbp
sub    rdx, rbx
and    rax, 0x40
or     rbx, 0xf0
add    rsi, 0x5a
mov    r8, rcx
movzx  rsi, word ptr [rsi]
mov    rax, 0x200
mov    r14, rbp
and    rax, rdx
and    rcx, 0x20
add    r14, 0x89
or     rax, 0x40
xor    si, 0x7a28
add    rdx, 0x78
add    rdx, 0x20
movzx  r14, word ptr [r14]
mov    rcx, 0x58
add    rsi, rbp
xor    rax, rdx
add    r8, 0x80
mov    r15, rsi
add    r14, rbp
add    r8, r15
mov    rbx, 0
and    rdx, 0x10
mov    r14, qword ptr [r14]
add    qword ptr [rsi], r14
pushfq
xor    r11, r14
add    r15, r14
mov    r13, 0x12
mov    r8, 0
and    r14, 0x88
and    r13, 0x40
add    r13, 1
mov    rdx, rbp
```

```
mov    r14, 0x200
add    rdx, 0xc0
add    r11, r14
or     r15, 0x88
mov    rdx, qword ptr [rdx]
add    rdx, 0xa
add    r11, 0x78
mov    r8b, byte ptr [rdx]
cmp    r8b, 0
je     0x49e
mov    rdx, rbp
or     r11, 0x40
and    r15, 1
xor    r11, 0x10
add    rdx, 0xc0
or     r14, 4
mov    r15, 0x12
mov    rdx, qword ptr [rdx]
sub    r11, r8
add    rdx, 4
or     r11, 0x80
mov    r8w, word ptr [rdx]
mov    r14, r8
add    r8, rbp
xor    r13, 4
pop    r10
mov    qword ptr [r8], r10
jmp    0x4ae
xor    rsi, 0x88
xor    rbx, 0xffffffff80000000
add    rsi, 0x78
mov    r10b, 0x68
mov    r9, 0x12
or     rbx, r10
and    r15, 0x78
mov    r14, rbp
or     r9, 8
add    r14, 0x29
xor    rbx, rdi
and    r15, 0x3f
or     byte ptr [r14], r10b
mov    rax, 0x58
mov    r8, rbp
sub    rsi, 0x78
add    r8, 0x127
mov    rdi, rbx
xor    rbx, 0x3f
mov    r8, qword ptr [r8]
xor    rsi, 1
mov    rax, rbp
```

```
add    r15, 0x3f
or     r15, 0xffffffff80000000
and    rsi, r9
add    rax, 0xc0
add    rdi, r14
or     rsi, 1
mov    rax, qword ptr [rax]
and    rdi, 0x7fffffff
add    rax, 2
sub    rsi, 4
or     rbx, rsi
movzx  rax, word ptr [rax]
mov    r9, rbp
mov    r13, 0x200
mov    r10, 0x58
add    r9, 0
or     r10, 0x20
add    eax, dword ptr [r9]
xor    r10, 0x40
add    eax, 0x3f505c07
add    r15, 0x88
mov    r12, rbp
or     rdi, 0x90
add    r12, 0
or     rbx, 0x80
add    rdi, 0xf0
mov    r13, 0x400
add    dword ptr [r12], eax
and    rsi, r8
or     r10, 8
and    rbx, 0x20
and    rax, 0xffff
mov    r11, 0
add    r13, r8
or     rbx, 1
shl    rax, 3
add    r8, rax
or     rbx, r15
sub    r15, 0x10
or     r11, r13
mov    rbx, qword ptr [r8]
mov    rdx, rbp
sub    r13, 0x80
add    rdx, 0xc0
add    qword ptr [rdx], 0xd
jmp    rbx
```

## #1: Obfuscating Individual VM Components

```
mov    r15, 0x200
xor    r15, 0x800
mov    rbx, rbp
add    rbx, 0xc0
mov    rbx, qword ptr [rbx]
mov    r13, 1
mov    rcx, 0
mov    r15, rbp
add    r15, 0xc0
or     rcx, 0x88
add    rbx, 0xb
mov    r15, qword ptr [r15]
or     r12, 0xffffffff80000000
sub    rcx, 0x78
movzx  r10, word ptr [rbx]
xor    r12, r13
add    r12, 0xffff
add    r15, 0
mov    r8, rbp
sub    rcx, 0x10
or     r12, r12
or     rcx, 0x800
movzx  r11, word ptr [r15]
xor    rcx, 0x800
mov    r12, r15
add    r8, 0
xor    r12, 0xf0
mov    rbx, 0x58
add    r11, rbp
xor    rbx, 0x800
and    r12, 0x20
add    rbx, 0x800
mov    r11, qword ptr [r11]
add    rbx, 1
and    r12, r9
mov    rdx, 1
xor    r10d, dword ptr [r8]
sub    r9, r11
pushfq
xor    rbx, 0xf0
xor    rbx, 0x800
and    rdx, r8
mov    r12, rbp
xor    rdx, 0x20
sub    rbx, 4
add    r11, 0x2549b044
or     rbx, 0x78
and    rdx, r10
mov    rax, 0
add    r12, 0x42
```

```
mov    r15, rdx
xor    r10d, dword ptr [r12]
sub    r15, 0x800
or     rdx, 0x400
mov    rsi, 0x200
mov    r14, rbp
sub    rsi, rsi
mov    rdi, rbp
mov    r8, 0x400
sub    rsi, r9
sub    r8, rsi
add    r14, 0
add    rsi, rax
and    r8, 0x88
xor    rsi, r14
mov    rsi, rbp
add    rdi, 0xc0
sub    r8, rdi
add    r8, 0x78
add    rsi, 4
mov    rcx, 0x200
mov    rdi, qword [illegible]
add    dw[illegible] b04
xor    r[illegible]
add    r[illegible]
add    r[illegible]
mov    r8, [illegible]
mov    ax, word ptr [rdi]
mov    r8, 1
mov    rsi, rbp
and    rcx, 8
sub    rcx, 1
mov    rcx, rdi
add    rsi, 0x29
or     rcx, 8
mov    r8, rsi
add    rcx, 4
mov    r13b, byte ptr [rsi]
cmp    r13b, 0xd2
jbe    0x204
and    r8, r13
or     rcx, r13
or     rcx, 4
mov    rbx, rbp
or     rcx, 4
sub    rcx, 0x400
add    rax, rbp
or     rcx, 0x80
add    rcx, 0x80
add    rbx, 0x5a
```

```
add    r8, 1
or     r8, 0x78
add    word ptr [rbx], r10w
mov    r15, rax
sub    r15, rax
pop    r9
mov    rcx, rbp
add    rcx, 0xc0
mov    rcx, qword ptr [rcx]
add    rcx, 8
movzx  r10, word ptr [rcx]
mov    r9, rbp
add    r9, 0
xor    r10d, dword ptr [r9]
and    rdi, 0xffffffff80000000
sub    r13, 0xf0
mov    rsi, 0
sub    r13, 0x20
mov    rbx, rbp
or     r13, 0x88
and    rcx, 8
mov    r8, 0x58
[illegible]
[illegible] p[illegible]
[illegible]
[illegible]
[illegible]
add    rbx, 8
mov    si, word ptr [rbx]
or     r9, 0xffff
sub    r9, 1
mov    r9, rbp
mov    r12, 0x58
add    r9, 0
sub    r13, 0x80
mov    r15, r13
or     rcx, r12
xor    esi, dword ptr [r9]
mov    r10, rbp
add    r10, 0xcc
sub    r15, 0x20
xor    esi, dword ptr [r10]
xor    r13, 0x90
add    rdi, 0x10
mov    r14, rsi
mov    rdx, rbp
add    rdx, 0
add    dword ptr [rdx], esi
xor    r12, 1
mov    r13, r15
```

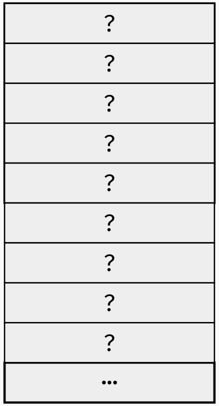
```
or     r14, r14
mov    rax, rbp
and    rcx, r13
add    rax, 4
sub    r8, -0x80000000
add    r13, 0xffff
and    rcx, 0x20
mov    r10, rbp
add    r13, r15
add    r14, r8
add    r10, 0x89
xor    word ptr [r10], si
xor    rdx, r11
mov    rsi, rbp
sub    rdx, rbx
and    rax, 0x40
or     rbx, 0xf0
add    rsi, 0x5a
mov    r8, rcx
movzx  rsi, word ptr [rsi]
mov    rax, 0x200
m[illegible] rbp
a[illegible] rdx
a[illegible] 0x20
a[illegible] 0x80
o[illegible]
x[illegible] s[illegible]
add    rdx, [illegible]
add    rdx, 0x20
movzx  r14, word ptr [r14]
mov    rcx, 0x58
add    rsi, rbp
xor    rax, rdx
add    r8, 0x80
mov    r15, rsi
add    r14, rbp
add    r8, r15
mov    rbx, 0
and    rdx, 0x10
mov    r14, qword ptr [r14]
add    qword ptr [rsi], r14
pushfq
xor    r11, r14
add    r15, r14
mov    r13, 0x12
mov    r8, 0
and    r14, 0x88
and    r13, 0x40
add    r13, 1
mov    rdx, rbp
```

```
mov    r14, 0x200
add    rdx, 0xc0
add    r11, r14
or     r15, 0x88
mov    rdx, qword ptr [rdx]
add    rdx, 0xa
add    r11, 0x78
mov    r8b, byte ptr [rdx]
cmp    r8b, 0
je     0x49e
mov    rdx, rbp
or     r11, 0x40
and    r15, 1
xor    r11, 0x10
add    rdx, 0xc0
or     r14, 4
mov    r15, 0x12
mov    rdx, qword ptr [rdx]
sub    r11, r8
add    rdx, 4
or     r11, 0x80
mo[illegible] r[illegible]word ptr [rdx]
mo[illegible] [illegible]8
ad[illegible] [illegible]p
xo[illegible]
po[illegible]
mo[illegible] qw[illegible]], r10
jmp    0x4a[illegible]
xor    rsi, 0x88
xor    rbx, 0xffffffff80000000
add    rsi, 0x78
mov    r10b, 0x68
mov    r9, 0x12
or     rbx, r10
and    r15, 0x78
mov    r14, rbp
or     r9, 8
add    r14, 0x29
xor    rbx, rdi
and    r15, 0x3f
or     byte ptr [r14], r10b
mov    rax, 0x58
mov    r8, rbp
sub    rsi, 0x78
add    r8, 0x127
mov    rdi, rbx
xor    rbx, 0x3f
mov    r8, qword ptr [r8]
xor    rsi, 1
mov    rax, rbp
```

```
add    r15, 0x3f
or     r15, 0xffffffff80000000
and    rsi, r9
add    rax, 0xc0
add    rdi, r14
or     rsi, 1
mov    rax, qword ptr [rax]
and    rdi, 0x7fffffff
add    rax, 2
sub    rsi, 4
or     rbx, rsi
movzx  rax, word ptr [rax]
mov    r9, rbp
mov    r13, 0x200
mov    r10, 0x58
add    r9, 0
or     r10, 0x20
add    eax, dword ptr [r9]
xor    r10, 0x40
add    eax, 0x3f505c07
add    r15, 0x88
mov    r12, rbp
or     rdi, 0x90
add    r12, 0
or     rbx, 0x80
add    rdi, 0xf0
mov    r13, 0x400
add    dword ptr [r12], eax
and    rsi, r8
or     r10, 8
and    rbx, 0x20
and    rax, 0xffff
mov    r11, 0
add    r13, r8
or     rbx, 1
shl    rax, 3
add    r8, rax
or     rbx, r15
sub    r15, 0x10
or     r11, r13
mov    rbx, qword ptr [r8]
mov    rdx, rbp
sub    r13, 0x80
add    rdx, 0xc0
add    qword ptr [rdx], 0xd
jmp    rbx
```

$$\text{u64 res} = M_{13} + M_{14}$$

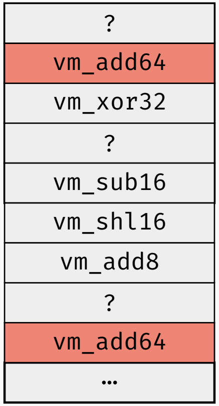
## #2: Duplicating VM Handlers



## #2: Duplicating VM Handlers

| |
|---|
| ? |
| vm_add64 |
| vm_xor32 |
| ? |
| vm_sub16 |
| vm_shl16 |
| vm_add8 |
| ? |
| vm_add64 |
| ... |

## #2: Duplicating VM Handlers



# #5: Blinding VM Bytecode

```
mov    r15, 0x200
xor    r15, 0x800
mov    rbx, rbp
add    rbx, 0xc0
mov    rbx, qword ptr [rbx]
mov    r13, 1
mov    rcx, 0
mov    r15, rbp
add    r15, 0xc0
or     rcx, 0x88
add    rbx, 0xb
mov    r15, qword ptr [r15]
or     r12, 0xffffffff80000000
sub    rcx, 0x78
movzx  r10, word ptr [rbx]
xor    r12, r13
add    r12, 0xffff
add    r15, 0
mov    r8, rbp
sub    rcx, 0x10
or     r12, r12
or     rcx, 0x800
movzx  r11, word ptr [r15]
xor    rcx, 0x800
mov    r12, r15
add    r8, 0
xor    r12, 0xf0
mov    rbx, 0x58
add    r11, rbp
xor    rbx, 0x800
and    r12, 0x20
add    rbx, 0x800
mov    r11, qword ptr [r11]
add    rbx, 1
and    r12, r9
mov    rdx, 1
xor    r10d, dword ptr [r8]
sub    r9, r11
pushfq
xor    rbx, 0xf0
xor    rbx, 0x800
and    rdx, r8
mov    r12, rbp
xor    rdx, 0x20
sub    rbx, 4
add    r11, 0x2549b044
or     rbx, 0x78
and    rdx, r10
mov    rax, 0
add    r12, 0x42
mov    r15, rdx
xor    r10d, dword ptr [r12]
sub    r15, 0x800
or     rdx, 0x400
mov    rsi, 0x200
mov    r14, rbp
sub    rsi, rsi
mov    rdi, rbp
mov    r8, 0x400
sub    rsi, r9
sub    r8, rsi
add    r14, 0
add    rsi, rax
and    r8, 0x88
xor    rsi, r14
mov    rsi, rbp
add    rdi, 0xc0
sub    r8, rdi
add    r8, 0x78
add    rsi, 4
mov    rcx, 0x200
mov    rdi, qword ptr [rdi]
add    dword ptr [rsi], 0x2549b044
xor    rcx, 0xf0
add    rcx, r10
add    rdi, 6
mov    r8, 0x400
mov    ax, word ptr [rdi]
mov    r8, 1
mov    rsi, rbp
and    rcx, 8
sub    rcx, 1
mov    rcx, rdi
add    rsi, 0x29
or     rcx, 8
mov    r8, rsi
add    rcx, 4
mov    r13b, byte ptr [rsi]
cmp    r13b, 0xd2
jbe    0x204
and    r8, r13
or     rcx, r13
or     rcx, 4
mov    rbx, rbp
or     rcx, 4
sub    rcx, 0x400
add    rax, rbp
or     rcx, 0x80
add    rcx, 0x80
add    rbx, 0x5a
add    r8, 1
or     r8, 0x78
add    word ptr [rbx], r10w
mov    r15, rax
sub    r15, rax
pop    r9
mov    rcx, rbp
add    rcx, 0xc0
mov    rcx, qword ptr [rcx]
add    rcx, 8
movzx  r10, word ptr [rcx]
mov    r9, rbp
add    r9, 0
xor    r10d, dword ptr [r9]
and    rdi, 0xffffffff80000000
sub    r13, 0xf0
mov    rsi, 0
sub    r13, 0x20
mov    rbx, rbp
or     r13, 0x88
and    rcx, 8
mov    r8, 0x58
add    rbx, 0xc0
mov    rbx, qword ptr [rbx]
sub    rcx, 0x20
add    rdi, 0x80
sub    r13, 0x10
add    rbx, 8
mov    si, word ptr [rbx]
or     r9, 0xffff
sub    r9, 1
mov    r9, rbp
mov    r12, 0x58
add    r9, 0
sub    r13, 0x80
mov    r15, r13
or     rcx, r12
xor    esi, dword ptr [r9]
mov    r10, rbp
add    r10, 0xcc
sub    r15, 0x20
xor    esi, dword ptr [r10]
xor    r13, 0x90
add    rdi, 0x10
mov    r14, rsi
mov    rdx, rbp
add    rdx, 0
add    dword ptr [rdx], esi
xor    r12, 1
mov    r13, r15
or     r14, r14
mov    rax, rbp
and    rcx, r13
add    rax, 4
sub    r8, -0x80000000
add    r13, 0xffff
and    rcx, 0x20
mov    r10, rbp
add    r13, r15
add    r14, r8
add    r10, 0x89
xor    word ptr [r10], si
xor    rdx, r11
mov    rsi, rbp
sub    rdx, rbx
and    rax, 0x40
or     rbx, 0xf0
add    rsi, 0x5a
mov    r8, rcx
movzx  rsi, word ptr [rsi]
mov    rax, 0x200
mov    r14, rbp
and    rax, rdx
and    rcx, 0x20
add    r14, 0x89
or     rax, 0x40
xor    si, 0x7a28
add    rdx, 0x78
add    rdx, 0x20
movzx  r14, word ptr [r14]
mov    rcx, 0x58
add    rsi, rbp
xor    rax, rdx
add    r8, 0x80
mov    r15, rsi
add    r14, rbp
add    r8, r15
mov    rbx, 0
and    rdx, 0x10
mov    r14, qword ptr [r14]
add    qword ptr [rsi], r14
pushfq
xor    r11, r14
add    r15, r14
mov    r13, 0x12
mov    r8, 0
and    r14, 0x88
and    r13, 0x40
add    r13, 1
mov    rdx, rbp
mov    r14, 0x200
add    rdx, 0xc0
add    r11, r14
or     r15, 0x88
mov    rdx, qword ptr [rdx]
add    rdx, 0xa
add    r11, 0x78
mov    r8b, byte ptr [rdx]
cmp    r8b, 0
je     0x49e
mov    rdx, rbp
or     r11, 0x40
and    r15, 1
xor    r11, 0x10
add    rdx, 0xc0
or     r14, 4
mov    r15, 0x12
mov    rdx, qword ptr [rdx]
sub    r11, r8
add    rdx, 4
or     r11, 0x80
mov    r8w, word ptr [rdx]
mov    r14, r8
add    r8, rbp
xor    r13, 4
pop    r10
mov    qword ptr [r8], r10
jmp    0x4ae
xor    rsi, 0x88
xor    rbx, 0xffffffff80000000
add    rsi, 0x78
mov    r10b, 0x68
mov    r9, 0x12
or     rbx, r10
and    r15, 0x78
mov    r14, rbp
or     r9, 8
add    r14, 0x29
xor    rbx, rdi
and    r15, 0x3f
or     byte ptr [r14], r10b
mov    rax, 0x58
mov    r8, rbp
sub    rsi, 0x78
add    r8, 0x127
mov    rdi, rbx
xor    rbx, 0x3f
mov    r8, qword ptr [r8]
xor    rsi, 1
mov    rax, rbp
add    r15, 0x3f
or     r15, 0xffffffff80000000
and    rsi, r9
add    rax, 0xc0
add    rdi, r14
or     rsi, 1
mov    rax, qword ptr [rax]
and    rdi, 0x7fffffff
add    rax, 2
sub    rsi, 4
or     rbx, rsi
movzx  rax, word ptr [rax]
mov    r9, rbp
mov    r13, 0x200
mov    r10, 0x58
add    r9, 0
or     r10, 0x20
add    eax, dword ptr [r9]
xor    r10, 0x40
add    eax, 0x3f505c07
add    r15, 0x88
mov    r12, rbp
or     rdi, 0x90
add    r12, 0
or     rbx, 0x80
add    rdi, 0xf0
mov    r13, 0x400
add    dword ptr [r12], eax
and    rsi, r8
or     r10, 8
and    rbx, 0x20
and    rax, 0xffff
mov    r11, 0
add    r13, r8
or     rbx, 1
shl    rax, 3
add    r8, rax
or     rbx, r15
sub    r15, 0x10
or     r11, r13
mov    rbx, qword ptr [r8]
mov    rdx, rbp
sub    r13, 0x80
add    rdx, 0xc0
add    qword ptr [rdx], 0xd
jmp    rbx
```

# #5: Blinding VM Bytecode

RUHR UNIVERSITÄT BOCHUM | RUB

```
mov    r15, 0x200
xor    r15, 0x800
mov    rbx, rbp
add    rbx, 0xc0
mov    rbx, qword ptr [rbx]
mov    r13, 1
mov    rcx, 0
mov    r15, rbp
add    r15, 0xc0
or     rcx, 0x88
add    rbx, 0xb
mov    r15, qword ptr [r15]
or     r12, 0xffffffff80000000
sub    rcx, 0x78
movzx  r10, word ptr [rbx]
xor    r12, r13
add    r12, 0xffff
add    r15, 0
mov    r8, rbp
sub    rcx, 0x10
or     r12, r12
or     rcx, 0x800
movzx  r11, word ptr [r15]
xor    rcx, 0x800
mov    r12, r15
add    r8, 0
xor    r12, 0xf0
mov    rbx, 0x58
add    r11, rbp
xor    rbx, 0x800
and    r12, 0x20
add    rbx, 0x800
mov    r11, qword ptr [r11]
add    rbx, 1
and    r12, r9
mov    rdx, 1
xor    r10d, dword ptr [r8]
sub    r9, r11
pushfq
xor    rbx, 0xf0
xor    rbx, 0x800
and    rdx, r8
mov    r12, rbp
xor    rdx, 0x20
sub    rbx, 4
add    r11, 0x2549b044
or     rbx, 0x78
and    rdx, r10
mov    rax, 0
add    r12, 0x42
```

```
mov    r15, rdx
xor    r10d, dword ptr [r12]
sub    r15, 0x800
or     rdx, 0x400
mov    rsi, 0x200
mov    r14, rbp
sub    rsi, rsi
mov    rdi, rbp
mov    r8, 0x400
sub    rsi, r9
sub    r8, rsi
add    r14, 0
add    rsi, rax
and    r8, 0x88
xor    rsi, r14
mov    rsi, rbp
add    rdi, 0xc0
sub    r8, rdi
add    r8, 0x78
add    rsi, 4
mov    rcx, 0x200
mov    rdi, qword ptr [rdi]
add    dword ptr [rsi], 0x2549
xor    rcx, 0xf0
add    rcx, r10
add    rdi, 6
mov    r8, 0x400
mov    ax, word ptr [rdi]
mov    r8, 1
mov    rsi, rbp
and    rcx, 8
sub    rcx, 1
mov    rcx, rdi
add    rsi, 0x29
or     rcx, 8
mov    r8, rsi
add    rcx, 4
mov    r13b, byte ptr [rsi]
cmp    r13b, 0xd2
jbe    0x204
and    r8, r13
or     rcx, r13
or     rcx, 4
mov    rbx, rbp
or     rcx, 4
sub    rcx, 0x400
add    rax, rbp
or     rcx, 0x80
add    rcx, 0x80
add    rbx, 0x5a
```

```
add    r8, 1
or     r8, 0x78
add    word ptr [rbx], r10w
mov    r15, rax
sub    r15, rax
pop    r9
mov    rcx, rbp
```

```
add    r9, 0
sub    r13, 0x80
mov    r15, r13
or     rcx, r12
xor    esi, dword ptr [r9]
mov    r10, rbp
add    r10, 0xcc
sub    r15, 0x20
xor    esi, dword ptr [r10]
xor    r13, 0x90
add    rdi, 0x10
mov    r14, rsi
mov    rdx, rbp
add    rdx, 0
add    dword ptr [rdx], esi
xor    r12, 1
mov    r13, r15
```

```
or     r14, r14
mov    rax, rbp
and    rcx, r13
add    rax, 4
sub    r8, -0x80000000
add    r13, 0xffff
and    rcx, 0x20
```

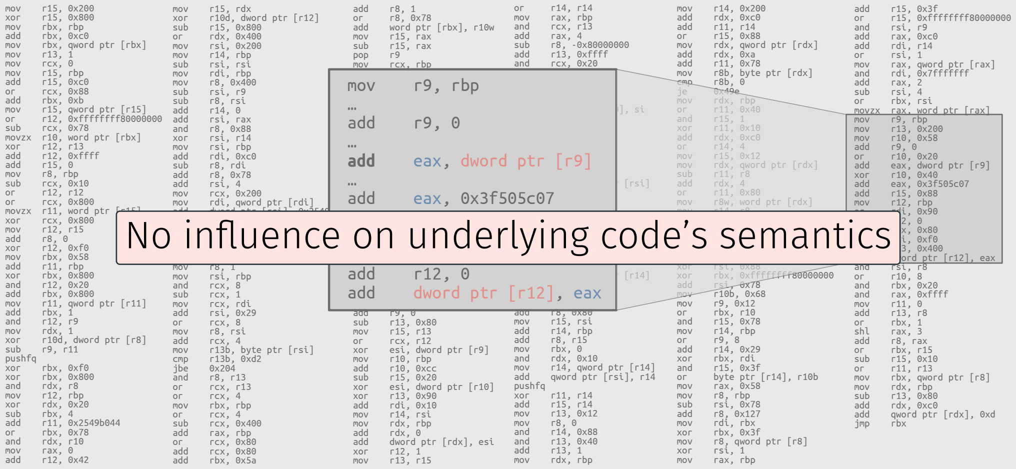
```
add    r8, 0x80
mov    r15, rsi
add    r14, rbp
add    r8, r15
mov    rbx, 0
and    rdx, 0x10
mov    r14, qword ptr [r14]
add    qword ptr [rsi], r14
pushfq
xor    r11, r14
add    r15, r14
mov    r13, 0x12
mov    r8, 0
and    r14, 0x88
and    r13, 0x40
add    r13, 1
mov    rdx, rbp
```

```
mov    r14, 0x200
add    rdx, 0xc0
add    r11, r14
or     r15, 0x88
mov    rdx, qword ptr [rdx]
add    rdx, 0xa
add    r11, 0x78
mov    r8b, byte ptr [rdx]
cmp    r8b, 0
je     0x49e
mov    rdx, rbp
or     r11, 0x40
and    r15, 1
xor    r11, 0x10
add    rdx, 0xc0
or     r14, 4
mov    r15, 0x12
mov    rdx, qword ptr [rdx]
sub    r11, r8
add    rdx, 4
or     r11, 0x80
mov    r8w, word ptr [rdx]
mov    r14, r8
add    r8, rbp
xor    r13, 4
pop    r10
mov    qword ptr [r8], r10
jmp    0x4ae
xor    rsi, 0x88
xor    rbx, 0xffffffff80000000
add    rsi, 0x78
mov    r10b, 0x68
mov    r9, 0x12
or     rbx, r10
and    r15, 0x78
mov    r14, rbp
or     r9, 8
add    r14, 0x29
xor    rbx, rdi
and    r15, 0x3f
or     byte ptr [r14], r10b
mov    rax, 0x58
mov    r8, rbp
sub    rsi, 0x78
add    r8, 0x127
mov    rdi, rbx
xor    rbx, 0x3f
mov    r8, qword ptr [r8]
xor    rsi, 1
mov    rax, rbp
```

```
add    r15, 0x3f
or     r15, 0xffffffff80000000
and    rsi, r9
add    rax, 0xc0
add    rdi, r14
or     rsi, 1
mov    rax, qword ptr [rax]
and    rdi, 0x7fffffff
add    rax, 2
sub    rsi, 4
or     rbx, rsi
movzx  rax, word ptr [rax]
mov    r9, rbp
mov    r13, 0x200
mov    r10, 0x58
add    r9, 0
or     r10, 0x20
add    eax, dword ptr [r9]
xor    r10, 0x40
add    eax, 0x3f505c07
add    r15, 0x88
mov    r12, rbp
or     rdi, 0x90
add    r12, 0
or     rbx, 0x80
add    rdi, 0xf0
mov    r13, 0x400
add    dword ptr [r12], eax
and    rsi, r8
or     r10, 8
and    rbx, 0x20
and    rax, 0xffff
mov    r11, 0
add    r13, r8
or     rbx, 1
shl    rax, 3
add    r8, rax
or     rbx, r15
sub    r15, 0x10
or     r11, r13
mov    rbx, qword ptr [r8]
mov    rdx, rbp
sub    r13, 0x80
add    rdx, 0xc0
add    qword ptr [rdx], 0xd
jmp    rbx
```

```
mov     r9, rbp
…
add     r9, 0
…
add     eax, dword ptr [r9]
…
add     eax, 0x3f505c07
…
mov     r12, rbp
…
add     r12, 0
add     dword ptr [r12], eax
```

# #5: Blinding VM Bytecode



```
mov    r15, 0x200
xor    r15, 0x800
mov    rbx, rbp
add    rbx, 0xc0
mov    rbx, qword ptr [rbx]
mov    r13, 1
mov    rcx, 0
mov    r15, rbp
add    r15, 0xc0
or     rcx, 0x88
add    rbx, 0xb
mov    r15, qword ptr [r15]
or     r12, 0xffffffff80000000
sub    rcx, 0x78
movzx  r10, word ptr [rbx]
xor    r12, r13
add    r12, 0xffff
add    r15, 0
mov    r8, rbp
sub    rcx, 0x10
or     r12, r12
or     rcx, 0x800
movzx  r11, word ptr [r15]
xor    rcx, 0x800
mov    r12, r15
add    r8, 0
xor    r12, 0xf0
mov    rbx, 0x58
add    r11, rbp
xor    rbx, 0x800
and    r12, 0x20
add    rbx, 0x800
mov    r11, qword ptr [r11]
add    rbx, 1
and    r12, r9
mov    rdx, 1
xor    r10d, dword ptr [r8]
sub    r9, r11
pushfq
xor    rbx, 0xf0
xor    rbx, 0x800
and    rdx, r8
mov    r12, rbp
xor    rdx, 0x20
sub    rbx, 4
add    r11, 0x2549b044
or     rbx, 0x78
and    rdx, r10
mov    rax, 0
add    r12, 0x42
```

```
mov    r15, rdx
xor    r10d, dword ptr [r12]
sub    r15, 0x800
or     rdx, 0x400
mov    rsi, 0x200
mov    r14, rbp
sub    rsi, rsi
mov    rdi, rbp
mov    r8, 0x400
sub    rsi, r9
sub    r8, rsi
add    r14, 0
add    rsi, rax
and    r8, 0x88
xor    rsi, r14
mov    rsi, rbp
add    rdi, 0xc0
sub    r8, rdi
add    r8, 0x78
add    rsi, 4
mov    rcx, 0x200
mov    rdi, qword ptr [rdi]

mov    r8, 1
mov    rsi, rbp
and    rcx, 8
sub    rcx, 1
mov    rcx, rdi
add    rsi, 0x29
or     rcx, 8
mov    r8, rsi
add    rcx, 4
mov    r13b, byte ptr [rsi]
cmp    r13b, 0xd2
jbe    0x204
and    r8, r13
or     rcx, r13
or     rcx, 4
mov    rbx, rbp
or     rcx, 4
sub    rcx, 0x400
add    rax, rbp
or     rcx, 0x80
add    rcx, 0x80
add    rbx, 0x5a
```

```
add    r8, 1
or     r8, 0x78
add    word ptr [rbx], r10w
mov    r15, rax
sub    r15, rax
pop    r9
mov    rcx, rbp

add    r9, 0
sub    r13, 0x80
mov    r15, r13
or     rcx, r12
xor    esi, dword ptr [r9]
mov    r10, rbp
add    r10, 0xcc
sub    r15, 0x20
xor    esi, dword ptr [r10]
xor    r13, 0x90
add    rdi, 0x10
mov    r14, rsi
mov    rdx, rbp
add    rdx, 0
add    dword ptr [rdx], esi
xor    r12, 1
mov    r13, r15
```

```
or     r14, r14
mov    rax, rbp
and    rcx, r13
add    rax, 4
sub    r8, -0x80000000
add    r13, 0xffff
and    rcx, 0x20

add    r8, 0x80
mov    r15, rsi
add    r14, rbp
add    r8, r15
mov    rbx, 0
and    rdx, 0x10
mov    r14, qword ptr [r14]
add    qword ptr [rsi], r14
pushfq
xor    r11, r14
add    r15, r14
mov    r13, 0x12
mov    r8, 0
and    r14, 0x88
and    r13, 0x40
add    r13, 1
mov    rdx, rbp
```

```
mov    r14, 0x200
add    rdx, 0xc0
add    r11, r14
or     r15, 0x88
mov    rdx, qword ptr [rdx]
add    rdx, 0xa
add    r11, 0x78
mov    r8b, byte ptr [rdx]
cmp    r8b, 0
je     0x49e
mov    rdx, rbp
or     r11, 0x40
and    r15, 1
xor    r11, 0x10
add    rdx, 0xc0
or     r14, 4
mov    r15, 0x12
mov    rdx, qword ptr [rdx]
sub    r11, r8
add    rdx, 4
or     r11, 0x80
mov    r8w, word ptr [rdx]

xor    rsi, 0x88
xor    rbx, 0xffffffff80000000
add    rsi, 0x78
mov    r10b, 0x68
mov    r9, 0x12
or     rbx, r10
and    r15, 0x78
mov    r14, rbp
or     r9, 8
add    r14, 0x29
xor    rbx, rdi
and    r15, 0x3f
or     byte ptr [r14], r10b
mov    rax, 0x58
mov    r8, rbp
sub    rsi, 0x78
add    r8, 0x127
mov    rdi, rbx
xor    rbx, 0x3f
mov    r8, qword ptr [r8]
xor    rsi, 1
mov    rax, rbp
```

```
add    r15, 0x3f
or     r15, 0xffffffff80000000
and    rsi, r9
add    rax, 0xc0
add    rdi, r14
or     rsi, 1
mov    rax, qword ptr [rax]
and    rdi, 0x7fffffff
add    rax, 2
sub    rsi, 4
or     rbx, rsi
movzx  rax, word ptr [rax]
mov    r9, rbp
mov    r13, 0x200
mov    r10, 0x58
add    r9, 0
or     r10, 0x20
add    eax, dword ptr [r9]
xor    r10, 0x40
add    eax, 0x3f505c07
add    r15, 0x88
mov    r12, rbp

and    rsi, r8
or     r10, 8
and    rbx, 0x20
and    rax, 0xffff
mov    r11, 0
add    r13, r8
or     rbx, 1
shl    rax, 3
add    r8, rax
or     rbx, r15
sub    r15, 0x10
or     r11, r13
mov    rbx, qword ptr [r8]
mov    rdx, rbp
sub    r13, 0x80
add    rdx, 0xc0
add    qword ptr [rdx], 0xd
jmp    rbx
```

```
mov    r9, rbp
…
add    r9, 0
…
add    eax, dword ptr [r9]
…
add    eax, 0x3f505c07

add    r12, 0
add    dword ptr [r12], eax
```

No influence on underlying code's semantics

# #3: No Central VM Dispatcher

RUHR UNIVERSITÄT BOCHUM RUB

```
mov    r15, 0x200
xor    r15, 0x800
mov    rbx, rbp
add    rbx, 0xc0
mov    rbx, qword ptr [rbx]
mov    r13, 1
mov    rcx, 0
mov    r15, rbp
add    r15, 0xc0
or     rcx, 0x88
add    rbx, 0xb
mov    r15, qword ptr [r15]
or     r12, 0xffffffff80000000
sub    rcx, 0x78
movzx  r10, word ptr [rbx]
xor    r12, r13
add    r12, 0xffff
add    r15, 0
mov    r8, rbp
sub    rcx, 0x10
or     r12, r12
or     rcx, 0x800
movzx  r11, word ptr [r15]
xor    rcx, 0x800
mov    r12, r15
add    r8, 0
xor    r12, 0xf0
mov    rbx, 0x58
add    r11, rbp
xor    rbx, 0x800
and    r12, 0x20
add    rbx, 0x800
mov    r11, qword ptr [r11]
add    rbx, 1
and    r12, r9
mov    rdx, 1
xor    r10d, dword ptr [r8]
sub    r9, r11
pushfq
xor    rbx, 0xf0
xor    rbx, 0x800
and    rdx, r8
mov    r12, rbp
xor    rdx, 0x20
sub    rbx, 4
add    r11, 0x2549b044
or     rbx, 0x78
and    rdx, r10
mov    rax, 0
add    r12, 0x42
```

```
mov    r15, rdx
xor    r10d, dword ptr [r12]
sub    r15, 0x800
or     rdx, 0x400
mov    rsi, 0x200
mov    r14, rbp
sub    rsi, rsi
mov    rdi, rbp
mov    r8, 0x400
sub    rsi, r9
sub    r8, rsi
add    r14, 0
add    rsi, rax
and    r8, 0x88
xor    rsi, r14
mov    rsi, rbp
add    rdi, 0xc0
sub    r8, rdi
add    r8, 0x78
add    rsi, 4
mov    rcx, 0x200
mov    rdi, qword ptr [rdi]
add    dword ptr [rsi], 0x2549b044
xor    rcx, 0xf0
add    rcx, r10
add    rdi, 6
mov    r8, 0x400
mov    ax, word ptr [rdi]
mov    r8, 1
mov    rsi, rbp
and    rcx, 8
sub    rcx, 1
mov    rcx, rdi
add    rsi, 0x29
or     rcx, 8
mov    r8, rsi
add    rcx, 4
mov    r13b, byte ptr [rsi]
cmp    r13b, 0xd2
jbe    0x204
and    r8, r13
or     rcx, r13
or     rcx, 4
mov    rbx, rbp
or     rcx, 4
sub    rcx, 0x400
add    rax, rbp
or     rcx, 0x80
add    rcx, 0x80
add    rbx, 0x5a
```

```
add    r8, 1
or     r8, 0x78
add    word ptr [rbx], r10w
mov    r15, rax
sub    r15, rax
pop    r9
mov    rcx, rbp
add    rcx, 0xc0
mov    rcx, qword ptr [rcx]
add    rcx, 8
movzx  r10, word ptr [rcx]
mov    r9, rbp
add    r9, 0
xor    r10d, dword ptr [r9]
and    rdi, 0xffffffff80000000
sub    r13, 0xf0
mov    rsi, 0
sub    r13, 0x20
mov    rbx, rbp
or     r13, 0x88
and    rcx, 8
mov    r8, 0x58
add    rbx, 0xc0
mov    rbx, qword ptr [rbx]
sub    rcx, 0x20
add    rdi, 0x80
sub    r13, 0x10
add    rbx, 8
mov    si, word ptr [rbx]
or     r9, 0xffff
sub    r9, 1
mov    r9, rbp
mov    r12, 0x58
add    r9, 0
sub    r13, 0x80
mov    r15, r13
or     rcx, r12
xor    esi, dword ptr [r9]
mov    r10, rbp
add    r10, 0xcc
sub    r15, 0x20
xor    esi, dword ptr [r10]
xor    r13, 0x90
add    rdi, 0x10
mov    r14, rsi
mov    rdx, rbp
add    rdx, 0
add    dword ptr [rdx], esi
xor    r12, 1
mov    r13, r15
```

```
or     r14, r14
mov    rax, rbp
and    rcx, r13
add    rax, 4
sub    r8, -0x80000000
add    r13, 0xffff
and    rcx, 0x20
mov    r10, rbp
add    r13, r15
add    r14, r8
add    r10, 0x89
xor    word ptr [r10], si
xor    rdx, r11
mov    rsi, rbp
sub    rdx, rbx
and    rax, 0x40
or     rbx, 0xf0
add    rsi, 0x5a
mov    r8, rcx
movzx  rsi, word ptr [rsi]
mov    rax, 0x200
mov    r14, rbp
and    rax, rdx
and    rcx, 0x20
add    r14, 0x89
or     rax, 0x40
xor    si, 0x7a28
add    rdx, 0x78
add    rdx, 0x20
movzx  r14, word ptr [r14]
mov    rcx, 0x58
add    rsi, rbp
xor    rax, rdx
add    r8, 0x80
mov    r15, rsi
add    r14, rbp
add    r8, r15
mov    rbx, 0
and    rdx, 0x10
mov    r14, qword ptr [r14]
add    qword ptr [rsi], r14
pushfq
xor    r11, r14
add    r15, r14
mov    r13, 0x12
mov    r8, 0
and    r14, 0x88
and    r13, 0x40
add    r13, 1
mov    rdx, rbp
```

```
mov    r14, 0x200
add    rdx, 0xc0
add    r11, r14
or     r15, 0x88
mov    rdx, qword ptr [rdx]
add    rdx, 0xa
add    r11, 0x78
mov    r8b, byte ptr [rdx]
cmp    r8b, 0
je     0x49e
mov    rdx, rbp
or     r11, 0x40
and    r15, 1
xor    r11, 0x10
add    rdx, 0xc0
or     r14, 4
mov    r15, 0x12
mov    rdx, qword ptr [rdx]
sub    r11, r8
add    rdx, 4
or     r11, 0x80
mov    r8w, word ptr [rdx]
mov    r14, r8
add    r8, rbp
xor    r13, 4
pop    r10
mov    qword ptr [r8], r10
jmp    0x4ae
xor    rsi, 0x88
xor    rbx, 0xffffffff80000000
add    rsi, 0x78
mov    r10b, 0x68
mov    r9, 0x12
or     rbx, r10
and    r15, 0x78
mov    r14, rbp
or     r9, 8
add    r14, 0x29
xor    rbx, rdi
and    r15, 0x3f
or     byte ptr [r14], r10b
mov    rax, 0x58
mov    r8, rbp
sub    rsi, 0x78
add    r8, 0x127
mov    rdi, rbx
xor    rbx, 0x3f
mov    r8, qword ptr [r8]
xor    rsi, 1
mov    rax, rbp
```

```
add    r15, 0x3f
or     r15, 0xffffffff80000000
and    rsi, r9
add    rax, 0xc0
add    rdi, r14
or     rsi, 1
mov    rax, qword ptr [rax]
and    rdi, 0x7fffffff
add    rax, 2
sub    rsi, 4
or     rbx, rsi
movzx  rax, word ptr [rax]
mov    r9, rbp
mov    r13, 0x200
mov    r10, 0x58
add    r9, 0
or     r10, 0x20
add    eax, dword ptr [r9]
xor    r10, 0x40
add    eax, 0x3f505c07
add    r15, 0x88
mov    r12, rbp
or     rdi, 0x90
add    r12, 0
or     rbx, 0x80
add    rdi, 0xf0
mov    r13, 0x400
add    dword ptr [r12], eax
and    rsi, r8
or     r10, 8
and    rbx, 0x20
and    rax, 0xffff
mov    r11, 0
add    r13, r8
```

```
or     rbx, 1
shl    rax, 3
add    r8, rax
or     rbx, r15
sub    r15, 0x10
or     r11, r13
mov    rbx, qword ptr [r8]
mov    rdx, rbp
sub    r13, 0x80
add    rdx, 0xc0
add    qword ptr [rdx], 0xd
jmp    rbx
```

# #3: No Central VM Dispatcher

```
mov    r15, 0x200
xor    r15, 0x800
mov    rbx, rbp
add    rbx, 0xc0
mov    rbx, qword ptr [rbx]
mov    r13, 1
mov    rcx, 0
mov    r15, rbp
add    r15, 0xc0
or     rcx, 0x88
add    rbx, 0xb
mov    r15, qword ptr [r15]
or     r12, 0xffffffff80000000
sub    rcx, 0x78
movzx  r10, word ptr [rbx]
xor    r12, r13
add    r12, 0xffff
add    r15, 0
mov    r8, rbp
sub    rcx, 0x10
or     r12, r12
or     rcx, 0x800
movzx  r11, word ptr [r15]
xor    rcx, 0x800
mov    r12, r15
add    r8, 0
xor    r12, 0xf0
mov    rbx, 0x58
add    r11, rbp
xor    rbx, 0x800
and    r12, 0x20
add    rbx, 0x800
mov    r11, qword ptr [r11]
add    rbx, 1
and    r12, r9
mov    rdx, 1
xor    r10d, dword ptr [r8]
sub    r9, r11
pushfq
xor    rbx, 0xf0
xor    rbx, 0x800
and    rdx, r8
mov    r12, rbp
xor    rdx, 0x20
sub    rbx, 4
add    r11, 0x2549b044
or     rbx, 0x78
and    rdx, r10
mov    rax, 0
add    r12, 0x42
```

```
mov    r15, rdx
xor    r10d, dword ptr [r12]
sub    r15, 0x800
or     rdx, 0x400
mov    rsi, 0x200
mov    r14, rbp
sub    rsi, rsi
mov    rdi, rbp
mov    r8, 0x400
sub    rsi, r9
sub    r8, rsi
add    r14, 0
add    rsi, rax
and    r8, 0x88
xor    rsi, r14
mov    rsi, rbp
add    rdi, 0xc0
sub    r8, rdi
add    r8, 0x78
add    rsi, 4
mov    rcx, 0x200
mov    rdi, qword ptr [rdi]
add    dword ptr [rsi], 0x2549
xor    rcx, 0xf0
add    rcx, r10
add    rdi, 6
mov    r8, 0x400
mov    ax, word ptr [rdi]
mov    r8, 1
mov    rsi, rbp
and    rcx, 8
sub    rcx, 1
mov    rcx, rdi
add    rsi, 0x29
or     rcx, 8
mov    r8, rsi
add    rcx, 4
mov    r13b, byte ptr [rsi]
cmp    r13b, 0xd2
jbe    0x204
and    r8, r13
or     rcx, r13
or     rcx, 4
mov    rbx, rbp
or     rcx, 4
sub    rcx, 0x400
add    rax, rbp
or     rcx, 0x80
add    rcx, 0x80
add    rbx, 0x5a
```

```
add    r8, 1
or     r8, 0x78
add    word ptr [rbx], r10w
mov    r15, rax
sub    r15, rax
pop    r9
mov    rcx, rbp
```

```
or     rbx, 1
shl    rax, 3
add    r8, rax
or     rbx, r15
sub    r15, 0x10
or     r11, r13
mov    rbx, qword ptr [r8]
mov    rdx, rbp
sub    r13, 0x80
add    rdx, 0xc0
add    qword ptr [rdx], 0xd
jmp    rbx
```

```
add    r9, 0
sub    r13, 0x80
mov    r15, r13
or     rcx, r12
xor    esi, dword ptr [r9]
mov    r10, rbp
add    r10, 0xcc
sub    r15, 0x20
xor    esi, dword ptr [r10]
xor    r13, 0x90
add    rdi, 0x10
mov    r14, rsi
mov    rdx, rbp
add    rdx, 0
add    dword ptr [rdx], esi
xor    r12, 1
mov    r13, r15
```

```
or     r14, r14
mov    rax, rbp
and    rcx, r13
add    rax, 4
sub    r8, -0x80000000
add    r13, 0xffff
and    rcx, 0x20
```

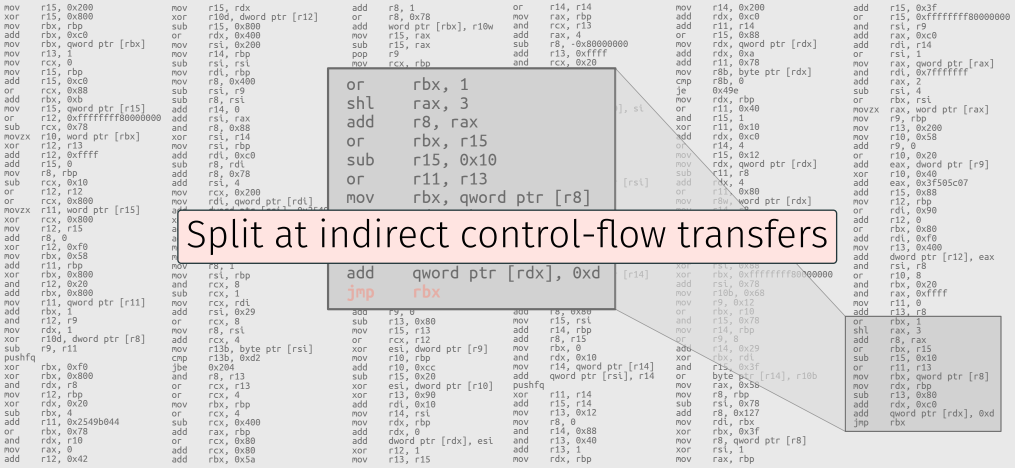
```
add    r8, 0x80
mov    r15, rsi
add    r14, rbp
add    r8, r15
mov    rbx, 0
and    rdx, 0x10
mov    r14, qword ptr [r14]
add    qword ptr [rsi], r14
pushfq
xor    r11, r14
add    r15, r14
mov    r13, 0x12
mov    r8, 0
and    r14, 0x88
and    r13, 0x40
add    r13, 1
mov    rdx, rbp
```

```
mov    r14, 0x200
add    rdx, 0xc0
add    r11, r14
or     r15, 0x88
mov    rdx, qword ptr [rdx]
add    rdx, 0xa
add    r11, 0x78
mov    r8b, byte ptr [rdx]
cmp    r8b, 0
je     0x49e
mov    rdx, rbp
or     r11, 0x40
and    r15, 1
xor    r11, 0x10
add    rdx, 0xc0
or     r14, 4
mov    r15, 0x12
mov    rdx, qword ptr [rdx]
sub    r11, r8
add    rdx, 4
or     r11, 0x80
mov    r8w, word ptr [rdx]
mov    r14, r8
add    r8, rbp
xor    r13, 4
pop    r10
mov    qword ptr [r8], r10
jmp    0x4ae
xor    rsi, 0x88
xor    rbx, 0xffffffff80000000
add    rsi, 0x78
mov    r10b, 0x68
mov    r9, 0x12
or     rbx, r10
and    r15, 0x78
mov    r14, rbp
or     r9, 8
add    r14, 0x29
xor    rbx, rdi
and    r15, 0x3f
or     byte ptr [r14], r10b
mov    rax, 0x58
mov    r8, rbp
sub    rsi, 0x78
add    r8, 0x127
mov    rdi, rbx
xor    rbx, 0x3f
mov    r8, qword ptr [r8]
xor    rsi, 1
mov    rax, rbp
```

```
add    r15, 0x3f
or     r15, 0xffffffff80000000
and    rsi, r9
add    rax, 0xc0
add    rdi, r14
or     rsi, 1
mov    rax, qword ptr [rax]
and    rdi, 0x7fffffff
add    rax, 2
sub    rsi, 4
or     rbx, rsi
movzx  rax, word ptr [rax]
mov    r9, rbp
mov    r13, 0x200
mov    r10, 0x58
add    r9, 0
or     r10, 0x20
add    eax, dword ptr [r9]
xor    r10, 0x40
add    eax, 0x3f505c07
add    r15, 0x88
mov    r12, rbp
or     rdi, 0x90
add    r12, 0
or     rbx, 0x80
add    rdi, 0xf0
mov    r13, 0x400
add    dword ptr [r12], eax
and    rsi, r8
or     r10, 8
and    rbx, 0x20
and    rax, 0xffff
mov    r11, 0
add    r13, r8
or     rbx, 1
shl    rax, 3
add    r8, rax
or     rbx, r15
sub    r15, 0x10
or     r11, r13
mov    rbx, qword ptr [r8]
mov    rdx, rbp
sub    r13, 0x80
add    rdx, 0xc0
add    qword ptr [rdx], 0xd
jmp    rbx
```

# #3: No Central VM Dispatcher

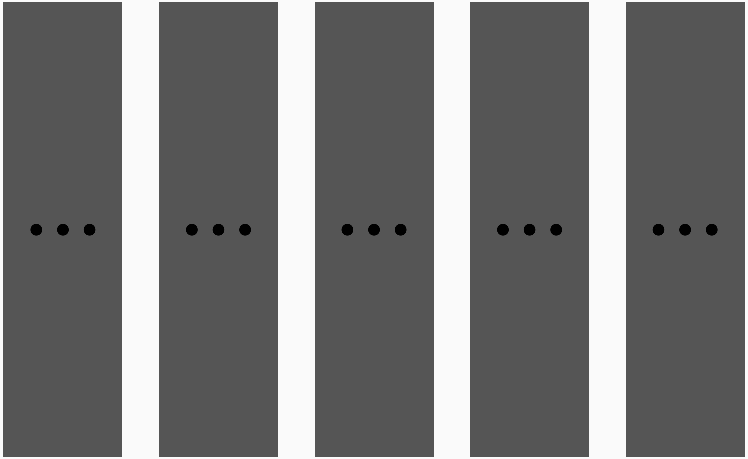
RUHR UNIVERSITÄT BOCHUM RUB



```
or      rbx, 1
shl     rax, 3
add     r8, rax
or      rbx, r15
sub     r15, 0x10
or      r11, r13
mov     rbx, qword ptr [r8]
mov     rdx, rbp
sub     r13, 0x80
add     rdx, 0xc0
add     qword ptr [rdx], 0xd
jmp     rbx
```

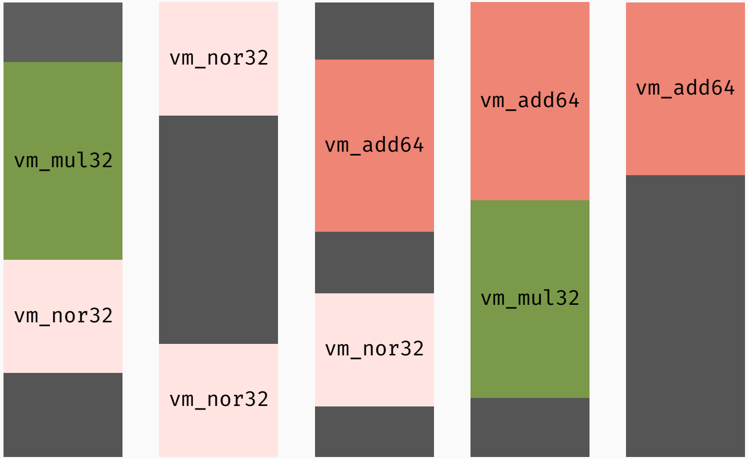
Split at indirect control-flow transfers

## #4: No Explicit Handler Table



## #4: No Explicit Handler Table

## #4: No Explicit Handler Table



## Conclusion

## Take Aways

1. syntactic complexity insignificant

## Take Aways

1. syntactic complexity insignificant
2. semantic complexity low within specified boundaries

## Take Aways

1. syntactic complexity insignificant
2. semantic complexity low within specified boundaries
3. learn underlying code’s semantics despite obfuscation

## Take Aways

1. syntactic complexity insignificant
2. semantic complexity low within specified boundaries
3. learn underlying code's semantics despite obfuscation

Program Synthesis as an orthogonal approach to traditional techniques

## Limitations

## Implementation Shortcomings

choosing *meaningful* code window boundaries

$$(x \oplus y) + 2 \cdot (x \wedge y) \quad \text{vs.} \quad (x \oplus y) + 2$$

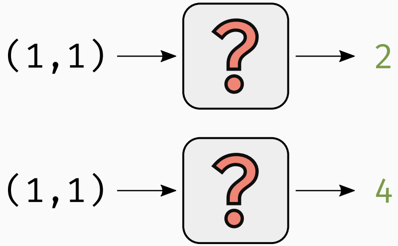
constants

$$x + 15324326921$$

control-flow operations

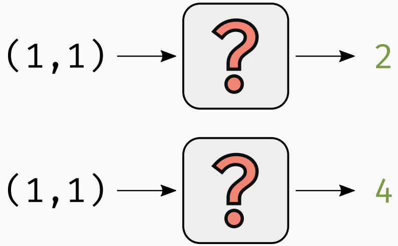
$$x \ ? \ y \ : \ z$$

## Limitations



non-determinism

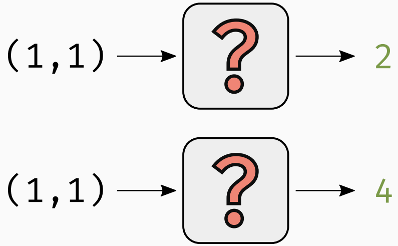
## Limitations



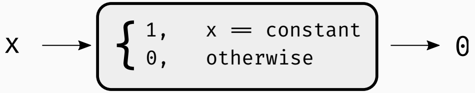
non-determinism

semantic complexity

## Limitations



non-determinism

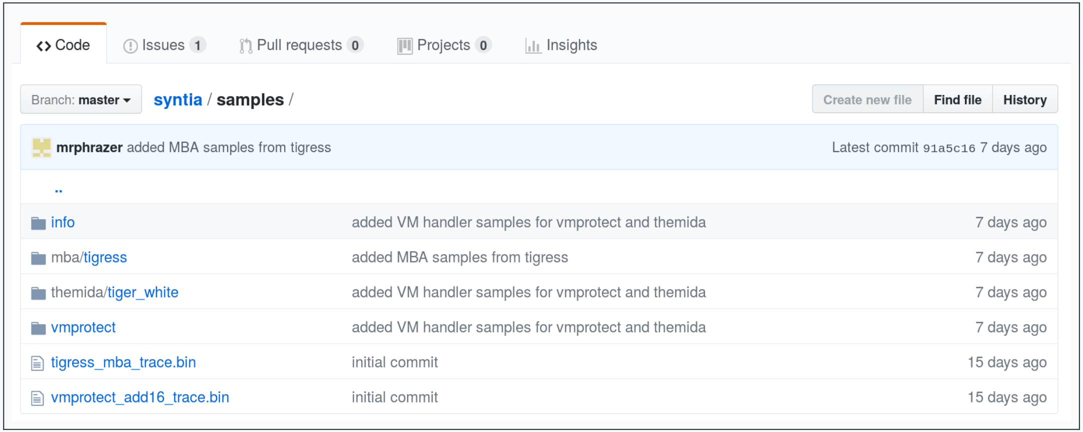


x
$\begin{cases} 1, & x == \text{constant} \\ 0, & \text{otherwise} \end{cases}$
0

point functions

semantic complexity

## *Do* try it at home!



Branch: master

syntia / samples /

mrphrazer added MBA samples from tigress Latest commit `91a5c16` 7 days ago

..

| | | |
|---|---|---|
| info | added VM handler samples for vmprotect and themida | 7 days ago |
| mba/tigress | added MBA samples from tigress | 7 days ago |
| themida/tiger_white | added VM handler samples for vmprotect and themida | 7 days ago |
| vmprotect | added VM handler samples for vmprotect and themida | 7 days ago |
| tigress_mba_trace.bin | initial commit | 15 days ago |
| vmprotect_add16_trace.bin | initial commit | 15 days ago |

## Summary

- obfuscation techniques (opaque predicates, VM, MBA)
- symbolic execution for syntactic deobfuscation
- program synthesis for semantic deobfuscation

https://github.com/RUB-SysSec/syntia