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
Setting the Scene

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

Prevent **Complicate** reverse engineering attempts.

- Intellectual Property
- Malicious Payloads
- Digital Rights Management

“We achieved our goals. We were uncracked for 13 whole days.”
Motivation

Prevent **Complicate** reverse engineering attempts.

- Intellectual Property
- Malicious Payloads
- Digital Rights Management

“We achieved our goals. We were uncracked for 13 whole days.”

How to protect software?
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

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.

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

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

Preserve the observable behavior of the application.
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.
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 $\text{effort(deploy)} \ll \text{effort(attack)}$.

Anti-Debugging tricks are effort 1:1.
Code Obfuscation Techniques

Opaque Predicates
Opaque Predicates

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

true

__block_a: ...
...

false

__block_b: ...
...
```

...
Opaque Predicates

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

__block_a: ...
... 

dead code 💀
```

Opaque True Predicate
Opaque Predicates

call GetCurrentProcess
cmp eax, 0xffffffff
je __block_a

__block_a: ...
...

dead code 💀
always taken never taken

Opaque True Predicate

opaque predicate

never taken
Opaque Predicates

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

Opaque True Predicate

```
__block_a: ...
...
```

dead code 💀

always taken

never taken

opaque value
Opaque True Predicate

```assembly
mov    eax, 0xffffffff
cmp    eax, 0xffffffff
je     __block_a
__block_a: ...
...
```

always taken

never taken

dead code 💀

Opaque True Predicate
Opaque Predicates

```assembly
mov    eax, 0xffffffff
cmp    eax, 0xdeadbeef
je     __block_a
__block_b: ...

...dead code

always taken
never taken
```

Opaque False Predicate

dead code 💀

__block_b: ...

...
Opaque Predicates

Random Opaque Predicate
duplicated block

call rand
cmp eax, 0xdeadbeef
je __block_a

__block_a: ...
...

__block_a': ...
...

semantically equivalent

≡

true false
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() ⇒ \(-1\)`
- `fldpi \Rightarrow st(0) = \pi`  
- `x^2 \geq 0 \ \forall x`  
- `x + 1 \neq x \ \forall x`  
- Pointer A **must-alias** pointer B  
- `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` $\Rightarrow st(\emptyset) = \pi$
  - $x^2 \geq 0 \ \forall x$
  - $x + 1 \neq x \ \forall x$
  - pointer A must-alias pointer B
  - `checksum(code) = 0x1c43b5cf`
Code Obfuscation Techniques

Virtual Machines
Virtual Machines

```assembly
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

```assembly
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

made-up instruction set

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

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

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

```
#...#...
```
Virtual Machines

made-up instruction set

__secret_ip:
  push __bytecode
  call vm_entry

__bytecode:
  db 54 68 69 73 20 64 6f
  ... 68
  db 69 6e 67 20 74 6f 20
  db 6d 65 2e de ad be ef

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

mov eax, ebx
ret
Virtual Machines

made-up instruction set

```
__secret_ip:
  push __bytecode
  call vm_entry

__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
```

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

Core Components

<table>
<thead>
<tr>
<th>VM Entry/Exit</th>
<th>Context Switch: native context ↔ virtual context</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM Dispatcher</td>
<td>Fetch–Decode–Execute loop</td>
</tr>
<tr>
<td>Handler Table</td>
<td>Individual VM ISA instruction semantics</td>
</tr>
</tbody>
</table>

- **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

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM Entry/Exit</td>
<td>Context Switch: native context ↔ virtual context</td>
</tr>
<tr>
<td>VM Dispatcher</td>
<td>Fetch–Decode–Execute loop</td>
</tr>
<tr>
<td>Handler Table</td>
<td>Individual VM ISA instruction semantics</td>
</tr>
</tbody>
</table>

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

```
// FDE
look up
handle_vpush
handle_vadd
handle_vxor
handle_vexit
handle_vpop
...
```
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

VM Entry
VM Dispatcher (FDE)
Individual Handlers
VM Exit
(as handler)
__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
  popf  [rbp]
  jmp   __vm_dispatcher

Handler performing nor
(with flag side-effects)
Virtual Machine Hardening
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 ($\text{mov rax, rbx} \rightarrow \text{push rbx; pop rax}$)
  - Opaque Predicates
  - Junk Code
  - ...

```assembly
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_vnor'
handle_vadd'
handle_vadd''
handle_vnor

handle_vpop
handle_vnor''
handle_vadd'
handle_vnor'
handle_vnor''
handle_vadd''
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
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
Hardening Technique #4 – No explicit handler table.

- An explicit handler table easily reveals all VM handlers.
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)
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)
Hardening Technique #4 – No explicit handler table.

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

- Idea:

  
<table>
<thead>
<tr>
<th>opcode</th>
<th>op 0</th>
<th>op 1</th>
<th>next handler addr</th>
</tr>
</thead>
</table>

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

(VMProtect Full, SolidShield)
Interpretation Techniques

PAUL KLIJT

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.
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.
operand \leftarrow [\text{vIP} + 0]

c\text{ontext} \leftarrow \text{semantics}(\text{context}, \text{operand})

\text{next}\_\text{handler} \leftarrow [\text{vIP} + 4]

\text{vIP} \leftarrow \text{vIP} + 8

\text{jmp next}\_\text{handler}
operand $\leftarrow [\text{vIP} + 0]$

operand $\leftarrow \text{unmangle}(\text{operand}, \text{key})$

key $\leftarrow \text{unmangle}'(\text{key}, \text{operand})$

context $\leftarrow \text{semantics}(\text{context}, \text{operand})$

next_handler $\leftarrow [\text{vIP} + 4]$

next_handler $\leftarrow \text{unmangle}''(\text{next_handler}, \text{key})$

key $\leftarrow \text{unmangle}'''(\text{key}, \text{next_handler})$

\text{vIP} \leftarrow \text{vIP} + 8

\text{jmp} \text{ next_handler}
Code Obfuscation Techniques

Mixed Boolean-Arithmetic
What does this expression compute?

\[(x \oplus y) + 2 \cdot (x \land y)\]
What does this expression compute?

\[(x \oplus y) + 2 \cdot (x \land y) = x + y\]
What does this expression compute?

$$(((x \oplus y) + ((x \land y) \ll 1)) \lor z) + (((x \oplus y) + ((x \land y) \ll 1)) \land z)$$
Mixed Boolean-Arithmetic

What does this expression compute?

\[
((x \oplus y) + ((x \land y) \ll 1)) \lor z) + (((x \oplus y) + ((x \land y) \ll 1)) \land z) \\
= x + y + z
\]

• Boolean identities?
  \[A \cdot 0 = 0\]

• Arithmetic identities?
  \[A + B = \overline{A} \cdot \overline{B}\]

• Karnaugh-Veitch maps?
  \[x^2 - y^2 = (x + y)(x - y)\]
Mixed Boolean-Arithmetic

Boolean-arithmetic algebra $BA[n]$

$$(B^n, \land, \lor, \oplus, \neg, \leq, \geq, >, <, \leq^s, \geq^s, >^s, <^s, \neq, =, >^s, >^s, \ll, +, -, \cdot)$$

is a Boolean-arithmetic algebra $BA[n]$, for $n > 0$, $B = \{0, 1\}$.

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

- Boolean algebra $(B^n, \land, \lor, \neg)$,
- 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
    popf  [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

rcx ← [rbp]

Handler performing nor
(with flag side-effects)
__handle_vnor:

- mov rcx, [rbp]
- mov rbx, [rbp + 4]
- not rcx
- not rbx
- and rcx, rbx
- mov [rbp + 4], rcx
- pushf
- popf
- pop [rbp]
- jmp __vm_dispatcher

rcx ← [rbp]
rbx ← [rbp + 4]

Handler performing nor (with flag side-effects)
Symbolic Execution

_handler_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

rcx ← [rbp]
r bx ← [rbp + 4]
rcx ← ¬rcx = ¬[rbp]

Handler performing nor
(with flag side-effects)
Symbolic Execution

__handle_vnor:

\[
\begin{align*}
\text{mov} & \quad \text{rcx, [rbp]} \\
\text{mov} & \quad \text{rbx, [rbp + 4]} \\
\text{not} & \quad \text{rcx} \\
\text{not} & \quad \text{rbx} \\
\text{and} & \quad \text{rcx, rbx} \\
\text{mov} & \quad \text{[rbp + 4], rcx} \\
\text{pushf} & \\
\text{pop} & \quad \text{[rbp]} \\
\text{jmp} & \quad \text{__vm_dispatcher}
\end{align*}
\]

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

\[
\begin{align*}
\text{rcx} & \leftarrow \text{[rbp]} \\
\text{rbx} & \leftarrow \text{[rbp + 4]} \\
\text{rcx} & \leftarrow \neg\text{rcx} = \neg\text{[rbp]} \\
\text{rbx} & \leftarrow \neg\text{rbx} = \neg\text{[rbp + 4]}
\end{align*}
\]
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

rcx  ← [rbp]
rbx  ← [rbp + 4]
rcx  ← ¬rcx = ¬[rbp]
rbx  ← ¬rbx = ¬[rbp + 4]
rcx  ← rcx ∧ rbx

= (¬[rbp]) ∧ (¬[rbp + 4])

Handler performing nor
(with flag side-effects)
Symbolic Execution

_Handler_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 ← [rbp]
rbx ← [rbp + 4]
rcx ← ¬ rcx = ¬[rbp]
rbx ← ¬ rbx = ¬[rbp + 4]
rcx ← rcx ∧ rbx
      = (¬[rbp]) ∧ (¬[rbp + 4])
      = [rbp] ↓ [rbp + 4]
```
Symbolic Execution

```assembly
__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

rcx  ←  [rbp]
rbx  ←  [rbp + 4]
rcx  ←  ¬rcx = ¬[rbp]
rbx  ←  ¬rbx = ¬[rbp + 4]
rcx  ←  rcx ∧ rbx
         = (¬[rbp]) ∧ (¬[rbp + 4])
         = [rbp] ↓ [rbp + 4]

[rbp + 4]  ←  rcx = [rbp] ↓ [rbp + 4]
```

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 ← [rbp]
rbx ← [rbp + 4]
rcx ← ¬rcx = ¬[rbp]
rbx ← ¬rbx = ¬[rbp + 4]
rcx ← rcx ∧ rbx
    = (¬[rbp]) ∧ (¬[rbp + 4])
    = [ebp] ↓ [ebp + 4]

[rpb + 4] ← rcx = [ebp] ↓ [ebp + 4]

rsp ← rsp − 4
[rsp] ← flags
```
Symbolic Execution

__handle_vnor:

```c
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)

```c
rcx ← [rbp]
rbx ← [rbp + 4]
rcx ← ¬ rcx = ¬ [rbp]
rbx ← ¬ rbx = ¬ [rbp + 4]
rcx ← rcx ∧ rbx
    = (¬[rbp]) ∧ (¬[rbp + 4])
    = [rbp] ↓ [rbp + 4]

[rbp + 4] ← rcx = [rbp] ↓ [rbp + 4]

rsp ← rsp − 4
[rsp] ← flags
[rbp] ← [rsp] = flags
rsp ← rsp + 4
```
Symbolic Execution

_Handler_vnor:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov rcx, [rbp]</td>
<td>rcx ← [rbp]</td>
</tr>
<tr>
<td>mov rbx, [rbp + 4]</td>
<td>rbx ← [rbp + 4]</td>
</tr>
<tr>
<td>not rcx</td>
<td>rcx ← ¬rcx = ¬[rbp]</td>
</tr>
<tr>
<td>not rbx</td>
<td>rbx ← ¬rbx = ¬[rbp + 4]</td>
</tr>
<tr>
<td>and rcx</td>
<td>rcx ← rcx ∧ rbx</td>
</tr>
<tr>
<td>mov [rbp + 4], rcx</td>
<td>[rbp + 4] ← rcx = [rbp] ↓ [rbp + 4]</td>
</tr>
<tr>
<td>pushf [rbp]</td>
<td></td>
</tr>
<tr>
<td>pop [rbp]</td>
<td></td>
</tr>
<tr>
<td>jmp __vm_dispatcher</td>
<td></td>
</tr>
</tbody>
</table>

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

```assembly
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    rs1, 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  
cmc  
cmp    ...    rdi, rax  
jmp    0xffffffffffff2a70  
dec    eax  
cmc  
lea    rax, [rsp + 0x140]  
cmp    rbp, rax  
ja     0x6557b  
jmp    rdi
```
Virtual Machine Handler
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 * (1668620215 - ((((1438524315 + C) + 1438524315 * ((2956783114 - -1478456685 * C) | (-1478456685 * (1668620215 - A) - 2956783115))) + A) - 1553572265)) | (-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
Mixed Boolean-Arithmetic Expression

\[ RAX = (M4 | M0) - M2 \]

This formula represents a mixed Boolean-arithmetic expression. The expression combines both arithmetic and logical operations, where \( RAX \) is the result, \( M4 \) and \( M0 \) are operands, and \( M2 \) is another operand subtracted from the result of the logical OR operation between \( M4 \) and \( M0 \).
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
We use $f$ as a black-box:

$$f(x, y, z) := (((x \oplus y) + ((x \land y) \cdot 2)) \lor z) + (((x \oplus y) + ((x \land y) \cdot 2)) \land z)$$
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) → [?] → 3
We use $f$ as a black-box:

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

$$(1, 1, 1) \rightarrow 3$$
We use \( f \) as a black-box:

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

\((2, 3, 1) \to 6\)

\((1, 1, 1) \to 3\)
We use $f$ as a black-box:

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

(2,3,1) → 6

(1,1,1) → 3

(2,3,1) → 6
We use $f$ as a black-box:

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

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

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

$(2, 3, 1) \rightarrow 6$
We use $f$ as a black-box:

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

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 7, 2</td>
<td>9</td>
</tr>
<tr>
<td>1, 1, 1</td>
<td>3</td>
</tr>
<tr>
<td>2, 3, 1</td>
<td>6</td>
</tr>
<tr>
<td>0, 7, 2</td>
<td>9</td>
</tr>
</tbody>
</table>
We use $f$ as a black-box:

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

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

- $(1, 1, 1) \rightarrow 3$
- $(2, 3, 1) \rightarrow 6$
- $(0, 7, 2) \rightarrow 9$
We use $f$ as a black-box:

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

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

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

(1, 1, 1) $\rightarrow$ 3
(2, 3, 1) $\rightarrow$ 6
(0, 7, 2) $\rightarrow$ 9
How to synthesize programs?
Stochastic Program Synthesis

- probabilistic optimization problem

global maxima
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

\[ U \rightarrow U + U \mid U \times U \mid a \mid b \]
Program Generation

\[ U \rightarrow U + U \mid U \times U \mid a \mid b \]

- non-terminal symbol: \( U \)
Program Generation

\[ U \rightarrow U + U \mid U \times U \mid a \mid b \]

- non-terminal symbol: \( U \)
- input variables: \( \{a, b\} \)
Program Generation

\[ U \rightarrow U + U \mid U \times U \mid a \mid b \]

- non-terminal symbol: \( U \)
- input variables: \( \{a, b\} \)
- candidate programs: \( a, b, a \times b, a + b, \ldots \)
Program Generation

\[ U \rightarrow U + U \mid U \ast U \mid a \mid b \]

- non-terminal symbol: \( U \)
- input variables: \( \{a, b\} \)
- candidate programs: \( a, b, a \ast b, a + b, \ldots \)
- intermediate programs: \( U + U, U \ast U, U + b, \ldots \)
Monte Carlo Tree Search
Monte Carlo Tree Search

U

0.64
Monte Carlo Tree Search
Monte Carlo Tree Search
Monte Carlo Tree Search

U

U*U  a  b

0.64  0.44
Monte Carlo Tree Search

\[(a+a) \times (b \times a)\]

\[0.64 \ 0.44\]
Monte Carlo Tree Search
Monte Carlo Tree Search

U*U

0.39

U

0.64

0.44

a

b
Monte Carlo Tree Search

The diagram illustrates a decision tree with the root node labeled as $U$. The branches from the root node lead to subnodes labeled as $U^*U$, $U+U$, $a$, and $b$. Each subnode is annotated with probabilities: $0.39$, $0.64$, and $0.44$. The tree structure and probabilities suggest a probabilistic decision-making process in Monte Carlo Tree Search.
Monte Carlo Tree Search

\[ a + (b + b) \]

\[ U \]

\[ a \]

\[ b \]

\[ U \]

\[ U \times U \]

\[ U + U \]

\[ 0.39 \]

\[ 0.64 \]

\[ 0.44 \]
Monte Carlo Tree Search
Monte Carlo Tree Search

U

U*U  U+U  a  b

0.39  0.70  0.64  0.44
Monte Carlo Tree Search
Monte Carlo Tree Search

\[(a+a) + ((a*b) + b)\]

\[U\]

- \[U*U\] \(0.39\)
- \[U+U\] \(0.70\)
- \[a\] \(0.64\)
- \[b\] \(0.44\)

\[U+(U+U)\]

\[(a+a) + ((a*b) + b)\]
Monte Carlo Tree Search

\[ U \]
- \[ U \times U \]
- \[ U + U \]
- \[ a \]
- \[ b \]

\[ 0.39 \quad 0.70 \quad 0.64 \quad 0.44 \]

\[ U + (U + U) \]

\[ 0.44 \]
Monte Carlo Tree Search
Monte Carlo Tree Search

U

U*U

U+U

U+(U+U)

0.39 0.57 0.64 0.44

0.44
Monte Carlo Tree Search

U

U*U
0.39

U+U
0.57

a
0.64

b
0.44

U+(U+U)
0.44
Monte Carlo Tree Search

U

U*U
U+U
U+U
U+b
U+(U+U)
0.39 0.57 0.64 0.44 0.44
Monte Carlo Tree Search

- $U \times U$
- $U + U$
- $a$
- $b$
- $U + (U + U)$
- $U + b$

Values:
- $U \times U$: 0.39
- $U + U$: 0.57
- $a$: 0.64
- $b$: 0.44
- $U + (U + U)$: 0.44
Monte Carlo Tree Search
Monte Carlo Tree Search

The diagram illustrates a decision tree with the root node labeled 'U'. The tree branches out to four nodes labeled 'U*U', 'U+U', 'a', and 'b'. The probabilities associated with these branches are as follows:

- 'U*U': 0.39
- 'U+U': 0.57
- 'a': 0.64
- 'b': 0.44
- 'U+b': 0.73
- 'U+(U+U)': 0.44
Monte Carlo Tree Search

- $U \ast U$
- $U \ast U$
- $U + U$
- $a$
- $b$
- $U + b$
- $U + (U + U)$

Values:
- $0.39$
- $0.62$
- $0.64$
- $0.44$
- $0.73$
- $0.44$
Monte Carlo Tree Search

- $U * U$
- $U + U$
- $a$
- $b$
- $U + b$
- $U + (U + U)$

Values:
- $0.39$
- $0.62$
- $0.64$
- $0.44$
- $0.73$
- $0.44$
Monte Carlo Tree Search

\[
\begin{align*}
U & \\
U & \quad U+U & \quad a & \quad b \\
& \quad U*U & & \\
& \quad U+b & & \quad U+(U*U) \\
& & \quad U+(U+U) & \quad U+(U*U) \quad U+(U+U)
\end{align*}
\]

0.64 0.44 0.39 0.62 0.73 0.44
Monte Carlo Tree Search

\[
\begin{align*}
U & \quad \text{U} \\
U \ast U & \quad \text{U} \ast \text{U} \\
U + U & \quad \text{U} + \text{U} \\
b & \quad \text{b} \\
U + b & \quad \text{U} + \text{b} \\
U + (U + U) & \quad \text{U} + (\text{U} + \text{U}) \\
\end{align*}
\]

\[
\begin{align*}
0.39 & \quad 0.62 \\
0.73 & \quad 0.44 \\
0.39 & \quad 0.62 \\
0.73 & \quad 0.44 \\
0.64 & \quad 0.44 \\
\end{align*}
\]
Monte Carlo Tree Search

U

U*U  U+U  a  b

0.39  0.62  0.64  0.44

U+b  U+(U+U)  U+(U*U)

0.73  0.44  0.69
Monte Carlo Tree Search

U

U*U → 0.39  
U+U → 0.62

U+b → 0.73

U+(U+U) → 0.44

U+(U*U) → 0.69

a

b
Monte Carlo Tree Search
Monte Carlo Tree Search

\[ \begin{align*}
    U & \quad 0.39 \quad 0.64 \quad 0.64 \quad 0.44 \\
    U*U & \quad \quad \quad \quad U+U & \quad a & \quad b \\
    U+b & \quad U+(U+U) & \quad U+(U*U) \quad 0.73 \quad 0.44 \quad 0.69
\end{align*} \]
Monte Carlo Tree Search
Monte Carlo Tree Search
Monte Carlo Tree Search

- U
  - U*U
    - 0.39
  - U+U
    - 0.64
  - a
    - 0.64
  - b
    - 0.44

- U+b
  - 0.73

- U+(U+U)
  - 0.44

- U+a
  - 0.87

- U+(U*U)
  - 0.69
Monte Carlo Tree Search

\[ U + (U \times U) \]

\[ U + (U + U) \]

\[ U + b \]

\[ U + (U + U) \]

\[ U + a \]

\[ U + (U \times U) \]

Values:

- 0.73
- 0.44
- 0.69
- 0.64
Monte Carlo Tree Search
Monte Carlo Tree Search
Monte Carlo Tree Search

```
U
  /\  
U*U U+U
  /\  /  
U+b U+(U*U) U+a U+(U+U)
```

Values:
- U: 0.64, 0.44, 0.39, 0.69
- U+b: 0.73, 0.44, 0.69
- U+(U+U): 0.69
- U+(U*U): 0.87

Monte Carlo Tree Search

- $U$
  - $U \times U$
  - $U + U$
  - $a$
  - $b$
- $U + b$
- $U + (U + U)$
- $U + a$
- $U + (U \times U)$
Monte Carlo Tree Search
Monte Carlo Tree Search

\[ U * U \quad U + U \quad a \quad b \]
\[ U + b \quad U + (U + U) \quad U + a \quad U + (U * U) \]
\[ 0.73 \quad 0.44 \quad 0.87 \quad 0.69 \]
Monte Carlo Tree Search

U

 U*U  U+U  a  b
0.39  0.69  0.64  0.44

 U+b  U+(U+U)  U+a  U+(U*U)
0.73  0.44  0.87  0.69

(U*U)+a
0.75
Monte Carlo Tree Search

- $U$  
  - $U \times U$  
  - $U + U$  
  - $U + b$  
  - $U + (U + U)$  
  - $U + a$  
  - $U + (U \times U)$  
  - $(U \times U) + a$

Values:
- $U \times U$: 0.39
- $U + U$: 0.69
- $U + b$: 0.73
- $U + (U + U)$: 0.44
- $U + a$: 0.87
- $U + (U \times U)$: 0.69
- $(U \times U) + a$: 0.75
Monte Carlo Tree Search

U

U*U  U+U  a  b

0.39  0.69  0.64  0.44

U+b  U+(U+U)  U+a  U+(U*U)

0.73  0.44  0.81  0.69

(U*U)+a

0.75
Monte Carlo Tree Search

\[ U \]

\[ U*U \]  \[ U+U \]  \[ a \]  \[ b \]

\[ U+b \]  \[ U+(U+U) \]  \[ U+a \]  \[ U+(U*U) \]

\[ 0.39 \]  \[ 0.69 \]  \[ 0.64 \]  \[ 0.44 \]

\[ 0.73 \]  \[ 0.44 \]  \[ 0.81 \]  \[ 0.69 \]

\[ (U*U)+a \]

\[ 0.75 \]
Monte Carlo Tree Search

U

- U*U
  - U+U
    - U+b
    - U+(U+U)
      - U+a
      - U+(U*U)
        - (U*U)+a
          - 0.75
    - 0.73
- 0.70
- 0.64
- 0.44
- 0.69
- 0.81
- 0.69
Monte Carlo Tree Search

U

U*U  U+U  a  b

0.39  0.70  0.64  0.44

U+b  U+U+U  U+a  U+U*U

0.73  0.44  0.81  0.69

(U*U)+a

0.75
Monte Carlo Tree Search

U

- $U \times U$: 0.39
- $U + U$: 0.70
- a: 0.64
- b: 0.44

- $U + b$: 0.73
- $U + (U + U)$: 0.44
- $U + a$: 0.81
- $U + (U \times U)$: 0.69

- $(U \times U) + a$: 0.75
Monte Carlo Tree Search

![Monte Carlo Tree Search Diagram]

The diagram illustrates a tree structure with nodes labeled as $U$, $U*U$, $U+U$, $U+b$, $U+(U*U)$, $U+a$, and $U+(U+U)$. The tree nodes are connected with arrows indicating the flow of decisions. The diagram also includes numerical values associated with each node, likely representing probabilities or values in the context of Monte Carlo Tree Search.
Monte Carlo Tree Search

U

U*U

U+U

a

b

U+b

U+(U+U)

U+a

U+(U*U)

a+(U+U)

(U*U)+a

0.64

0.39

0.70

0.64

0.44

0.73

0.44

0.69

0.81

0.69

0.75
Monte Carlo Tree Search

\[ U \]

- \[ U \times U \]
- \[ U + U \]
- \[ a \]
- \[ b \]
- \[ U + b \]
- \[ U + (U + U) \]
- \[ U + a \]
- \[ U + (U \times U) \]

Values:
- \[ U \times U \]: 0.39
- \[ U + U \]: 0.70
- \[ a \]: 0.64
- \[ b \]: 0.44
- \[ U + b \]: 0.73
- \[ U + (U + U) \]: 0.44
- \[ U + a \]: 0.81
- \[ U + (U \times U) \]: 0.69

- \[ a + (U + U) \]: 0.68
- \[ (U \times U) + a \]: 0.75
Monte Carlo Tree Search

![Diagram of Monte Carlo Tree Search]
Monte Carlo Tree Search
Monte Carlo Tree Search

\[
\begin{array}{c}
U \\
U \times U & U + U & a & b \\
0.39 & 0.69 & 0.64 & 0.44 \\
U + b & U + (U + U) & U + a & U + (U \times U) \\
0.73 & 0.56 & 0.81 & 0.69 \\
a + (U + U) & (U \times U) + a \\
0.68 & 0.75
\end{array}
\]
Monte Carlo Tree Search

- $U * U$
- $U + U$
- $a$
- $b$
- $U + b$
- $U + (U + U)$
- $U + a$
- $U + (U * U)$
- $a + (U + U)$
- $(U * U) + a$

Values:
- $0.39$
- $0.69$
- $0.64$
- $0.44$
- $0.73$
- $0.56$
- $0.81$
- $0.69$
- $0.68$
- $0.75$
Monte Carlo Tree Search

\[ U \]

- \[ U \ast U \]
  - \[ U + b \] (0.73)
  - \[ U + (U + U) \] (0.56)
- \[ U + U \]
  - \[ U + a \] (0.81)
  - \[ U + (U \ast U) \] (0.69)
- \[ a \]
  - \[ a + (U + U) \] (0.68)
  - \[ (U \ast U) + a \] (0.75)
- \[ b \]
  - \[ b + a \]
Monte Carlo Tree Search

U

U*U   U+U   a   b

0.39   0.69   0.64   0.44

U+b   U+(U+U)   U+a   U+(U*U)

0.73   0.56   0.81   0.69

a+(U+U)   (U*U)+a   b+a

0.68   0.75   1.00
Score Calculation

(2, 2)

similarity(4; 6) = 0.78
similarity(0; 3) = 0.33
similarity(3; 3) = 1.00

average score: 0.70
Score Calculation

(2, 2)

similarity(4; 6) = 0.78
similarity(0; 3) = 0.33
similarity(3; 3) = 1.0

average score: 0.70
Score Calculation

\[(2,2)\]

\[\text{similarity}(4;6) = \frac{78}{2} = 39\]

\[\text{similarity}(0;3) = \frac{33}{2} = 16.5\]

\[\text{similarity}(3;3) = \frac{1}{2} = 0.5\]

Average score: \[\frac{39 + 16.5 + 0.5}{3} = 22\]
(2, 2)

\[
similarity(4, 6) = 0.78
\]
Score Calculation

\[(5,3)\]

\[
\text{similarity}(4,6) = 0.78
\]
Score Calculation

\[
\text{similarity}(4, 6) = 0.78
\]
Score Calculation

\[ \text{similarity}(4, 6) = 0.78 \]
Score Calculation

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

\[ \text{similarity}(0, 3) = 0.33 \]
Score Calculation

\[(3, 0)\]

\[
similarity(4, 6) = 0.78
\]

\[
similarity(0, 3) = 0.33
\]
Score Calculation

similarity(4, 6) = 0.78
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

(3, 0)

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

average score: 0.70
Output Similarity: similarity($O, O'$)

Let’s compare:
Output Similarity: $\text{similarity}(O, O')$

Are they in the same range?
Output Similarity: similarity($O, O'$)

How many bits are different?
Output Similarity: similarity($O, O'$)

How close are they numerically?
DEMO
How to synthesize obfuscated code?
static disassembly
Obtaining Code

static disassembly

memory dump
Obtaining Code

memory dump

指令追踪

static disassembly

指令反汇编

```
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
```
Learning Code Semantics

```assembly
__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)
__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)
__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)
Handler performing **nor** (with flag side-effects)
__handle_vnor:

```assembly
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)
Handler performing \texttt{nor} (with flag side-effects)
__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)
__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)
Handler performing **nor** (with flag side-effects)
Learning Code Semantics

```assembly
__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)
__handle_vnor:

```assembly
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)
__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

_Handler_vnor:
\[
\text{mov} \quad \text{rcx}, \ [\text{rbp}] \\
\text{mov} \quad \text{rbx}, \ [\text{rbp} + 4] \\
\text{not} \quad \text{rcx} \\
\text{not} \quad \text{rbx} \\
\text{and} \quad \text{rcx}, \ \text{rbx} \\
\text{- mov} \quad [\text{rbp} + 4], \ \text{rcx} \\
\text{- pushf} \\
\text{- pop} \quad [\text{rbp}] \\
\text{jmp} \quad \_\_\_\_\text{vm\_dispatcher}
\]

Handler performing nor (with flag side-effects)
Learning Code Semantics

_Handler_vnor:

```assembly
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)

\[ M_0 \leftarrow \neg (m_0 \lor m_1) \]
__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

rbx ← ¬ m₀
rcx ← ¬ (m₀ ∨ m₁)
M₀ ← ¬ (m₀ ∨ m₁)

Handler performing **nor**
(with flag side-effects)
I/O Sampling

Tools:
- WinDbg
- Pin
- bochs
- Unicorn
- Valgrind
- Dynamorio
- angr
- Metasm
- x64dbg
- <your tool here>
Instruction Trace: Forced Execution

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

true

```
__block_a: ...
...
```

false

```
__block_b: ...
...
```

Instruction Trace: Forced Execution

```
call check
cmp eax, 0xdeadbeef
je __block_a
ignore
force
```

```
__block_a: ...
...
```
Instruction Trace: Forced Execution

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

ignore

__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
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.
#2: Duplicating VM Handlers

<table>
<thead>
<tr>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
</tr>
<tr>
<td>?</td>
</tr>
<tr>
<td>?</td>
</tr>
<tr>
<td>?</td>
</tr>
<tr>
<td>?</td>
</tr>
<tr>
<td>?</td>
</tr>
<tr>
<td>?</td>
</tr>
<tr>
<td>?</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
#2: Duplicating VM Handlers

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>?</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
#2: Duplicating VM Handlers

<table>
<thead>
<tr>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>vm_add64</td>
</tr>
<tr>
<td>vm_xor32</td>
</tr>
<tr>
<td>?</td>
</tr>
<tr>
<td>vm_sub16</td>
</tr>
<tr>
<td>vm_shl16</td>
</tr>
<tr>
<td>vm_add8</td>
</tr>
<tr>
<td>?</td>
</tr>
<tr>
<td>vm_add64</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
No influence on underlying code's semantics
#3: No Central VM Dispatcher

### Split at indirect control-flow transfers
#4: No Explicit Handler Table
#4: No Explicit Handler Table
#4: No Explicit Handler Table

```
vm_mul32
vm_nor32
vm_nor32
vm_nor32
vm_add64
vm_add64
vm_add64
vm_nor32
vm_nor32
vm_nor32
vm_nor32
vm_add64
vm_nor32
vm_nor32
vm_add64
vm_mul32
```
Conclusion
1. syntactic complexity insignificant
1. syntactic complexity insignificant
2. semantic complexity low within specified boundaries
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

\[(x \oplus y) + 2 \cdot (x \land y) \quad \text{vs.} \quad (x \oplus y) + 2\]

constants

\[x + 15324326921\]

control-flow operations

\[x \ ? \ y : z\]
Limitations

(1,1) → 2

(1,1) → 4

non-determinism
Limitations

non-determinism

(1,1) 2

(1,1) 4

semantic complexity

AES
Limitations

- AES
- Semantic complexity
- Non-determinism

\[ x \rightarrow \begin{cases} 1, & x = \text{constant} \\ 0, & \text{otherwise} \end{cases} \rightarrow 0 \]

Point functions

Non-determinism

AES

Semantic complexity
<table>
<thead>
<tr>
<th>Branch: master</th>
<th>syntia / samples /</th>
<th>Latest commit 91a5c10 7 days ago</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrphrazer added MBA samples from tigress</td>
<td>info</td>
<td>added VM handler samples for vmprotect and themida 7 days ago</td>
</tr>
<tr>
<td></td>
<td>mba/tigress</td>
<td>added MBA samples from tigress 7 days ago</td>
</tr>
<tr>
<td></td>
<td>themida/tiger_white</td>
<td>added VM handler samples for vmprotect and themida 7 days ago</td>
</tr>
<tr>
<td></td>
<td>vmprotect</td>
<td>added VM handler samples for vmprotect and themida 7 days ago</td>
</tr>
<tr>
<td></td>
<td>tigress_mba_trace.bin</td>
<td>initial commit 15 days ago</td>
</tr>
<tr>
<td></td>
<td>vmprotect_add16_trace.bin</td>
<td>initial commit 15 days ago</td>
</tr>
</tbody>
</table>
Summary

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

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