

Analysis of virtual machine based obfuscators

Vladislav Hrčka
RNDr. Jaroslav Janáček, PhD
Mgr. Peter Košinár

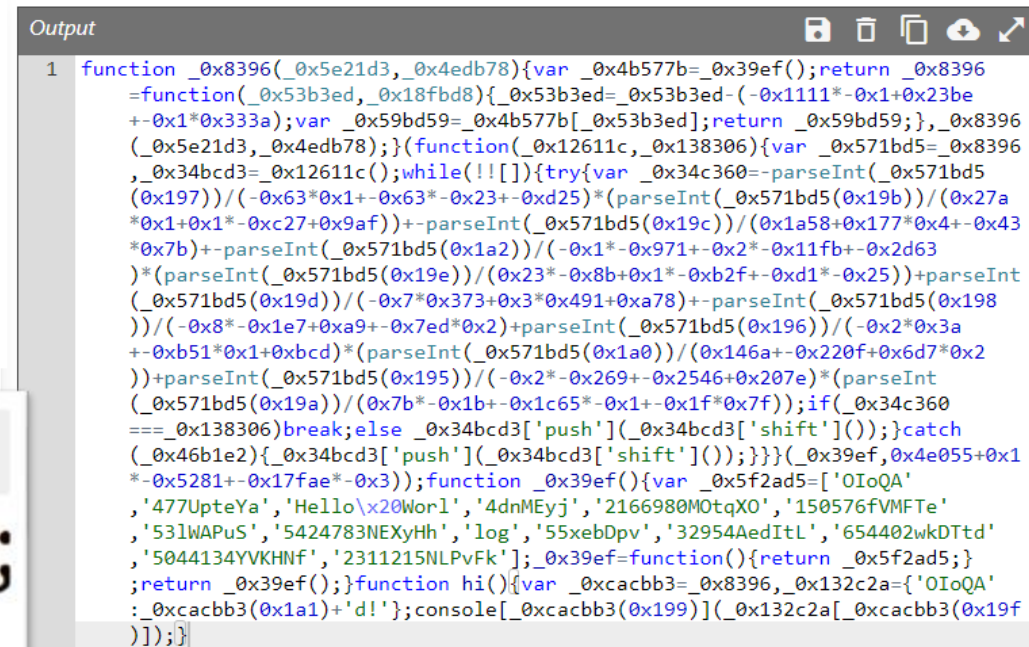
Introduction – obfuscation

- Obfuscated javascript:



```
1 function hi() {  
2   console.log("Hello World!");  
3 }
```

```
function hi() {  
  console.log("Hello World!");  
}
```



```
1 function _0x8396(_0x5e21d3,_0x4edb78){var _0x4b577b=_0x39ef();return _0x8396  
=function(_0x53b3ed,_0x18fbd8){_0x53b3ed=_0x53b3ed-(-0x1111*-0x1+0x23be  
+-0x1*0x333a);var _0x59bd59=_0x4b577b[_0x53b3ed];return _0x59bd59;},_0x8396  
(_0x5e21d3,_0x4edb78);}(  
function(_0x12611c,_0x138306){var _0x571bd5=_0x8396  
,_0x34bcd3=_0x12611c();while(!![]){try{var _0x34c360=-parseInt(_0x571bd5  
(0x197))/(-0x63*0x1+-0x63*-0x23+-0xd25)*(parseInt(_0x571bd5(0x19b)))/(0x27a  
*0x1+0x1*-0xc27+0x9af))+parseInt(_0x571bd5(0x19c))/(0x1a58+0x177*0x4+-0x43  
*0x7b)+parseInt(_0x571bd5(0x1a2))/(-0x1*-0x971+-0x2*-0x11fb+-0x2d63  
)*(parseInt(_0x571bd5(0x19e))/(0x23*-0x8b+0x1*-0xb2f+-0xd1*-0x25))+parseInt  
(_0x571bd5(0x19d))/(-0x7*0x373+0x3*0x491+0xa78)+parseInt(_0x571bd5(0x198  
))/(-0x8*-0x1e7+0xa9+-0x7ed*0x2)+parseInt(_0x571bd5(0x196))/(-0x2*0x3a  
+-0xb51*0x1+0xbcd)*(parseInt(_0x571bd5(0x1a0))/(0x146a+-0x220f+0x6d7*0x2  
(_0x571bd5(0x19a))/(0x7b*-0x1b+-0x1c65*-0x1+-0x1f*0x7f));if(_0x34c360  
===_0x138306)break;else _0x34bcd3['push'](_0x34bcd3['shift']());}catch  
(_0x46b1e2){_0x34bcd3['push'](_0x34bcd3['shift']());}})(_0x39ef,0x4e055+0x1  
*-0x5281+-0x17fae*-0x3);function _0x39ef(){var _0x5f2ad5=['0IoQA'  
, '477UpteYa', 'Hello\x20Worl', '4dnMEyj', '2166980M0tqX0', '150576fVMFTe'  
, '531WAPuS', '5424783NEXyHh', 'log', '55xebDpv', '32954AedItL', '654402wkDTtd'  
, '5044134YVKHNF', '2311215NLPvFk'];_0x39ef=function(){return _0x5f2ad5;}  
;return _0x39ef();}function hi(){var _0xcacbb3=_0x8396,_0x132c2a={'0IoQA'  
:_0xcacbb3(0x1a1)+'d!'};console[_0xcacbb3(0x199)](_0x132c2a[_0xcacbb3(0x19f  
)]);}
```

Aim

- describe virtual machine based obfuscators in general
- summarize methods to deal with such kind of obfuscators
- analyze a specific virtual machine
- design and implement methods to deobfuscate code protected with the specific machine

CodeVirtualizer

- *“Code Virtualizer is a powerful code obfuscation system for Windows applications that helps developers to protect their sensitive code areas against Reverse Engineering with very strong obfuscation code, based on code virtualization.”*

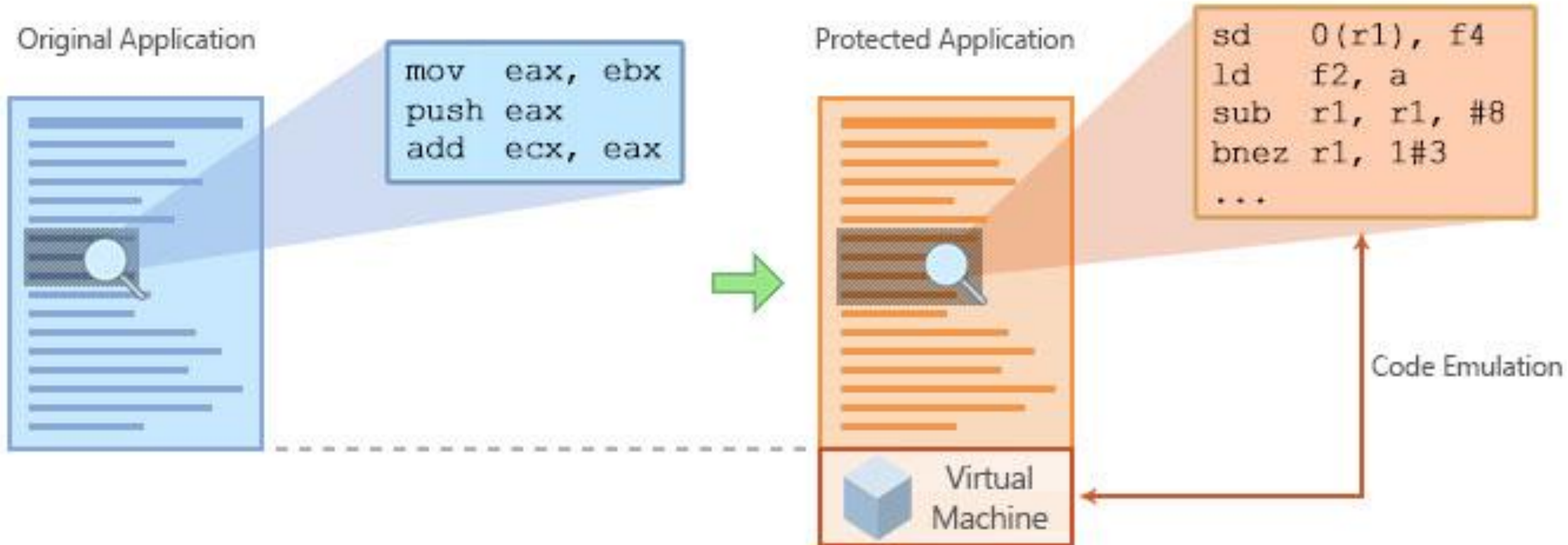
```
#include <stdio.h>
#include "VirtualizerSDK.h"

void main()
{
    VIRTUALIZER_START           // the area to protect starts here

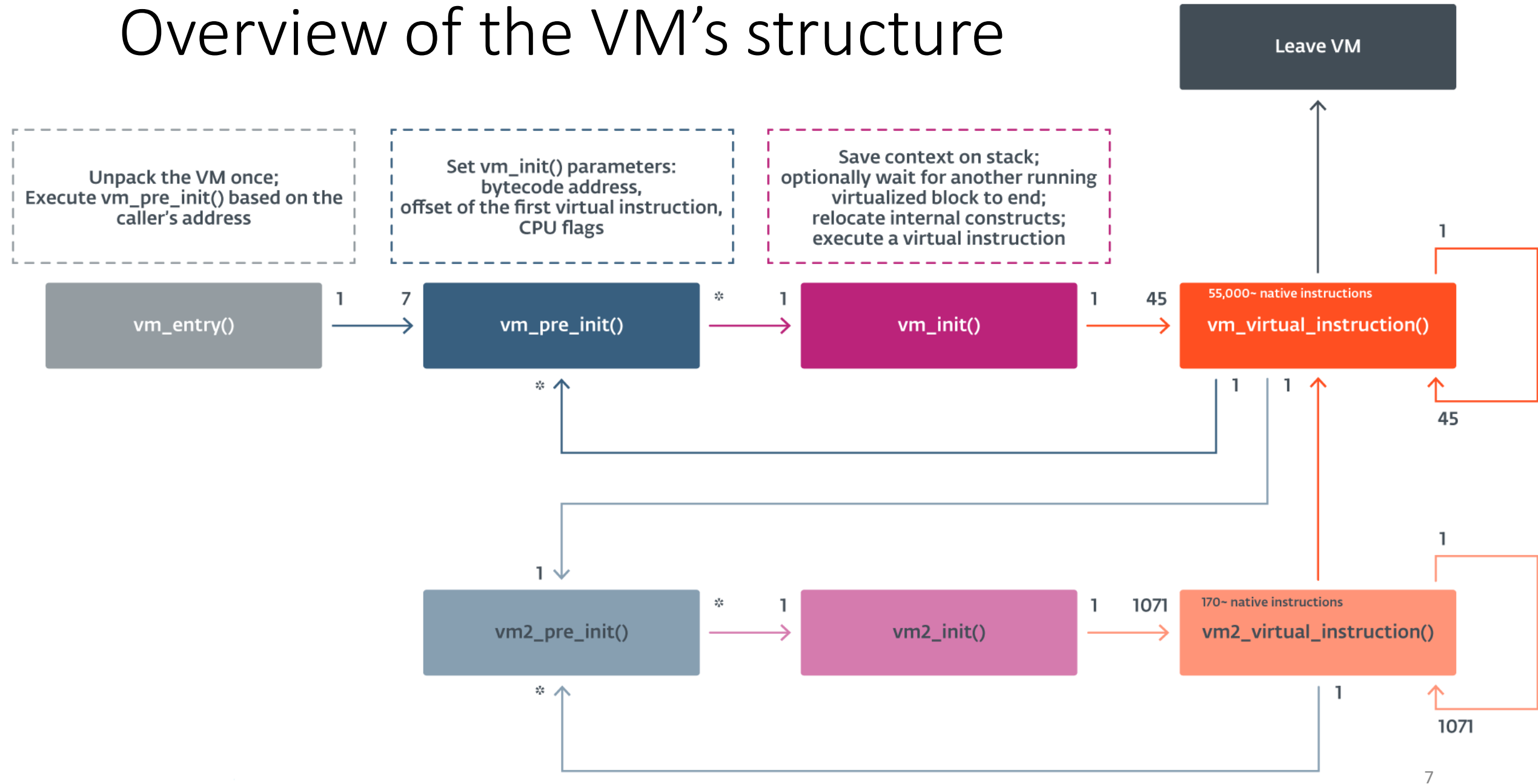
    printf("Hello World");

    VIRTUALIZER_END           // end of area to protect
}
```

CodeVirtualizer



Overview of the VM's structure



Attacks

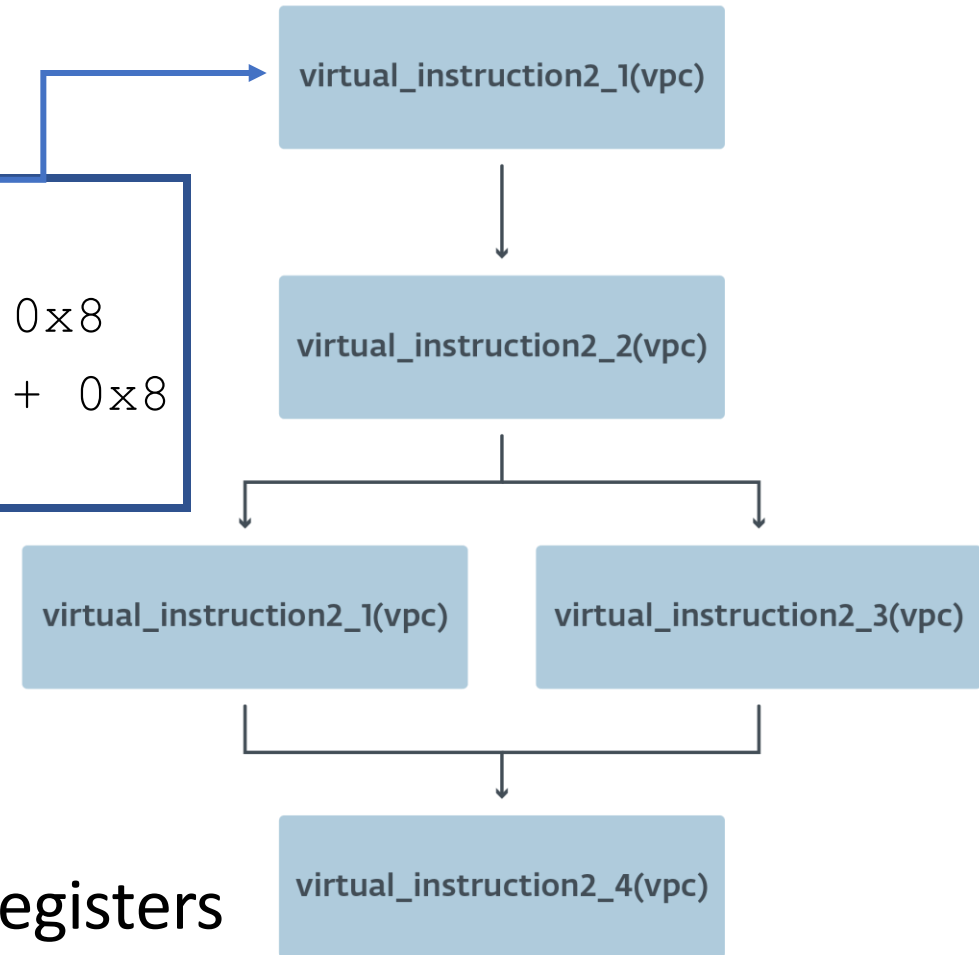
- **Manually build a disassembler**
 - Rolles R. Unpacking virtualization obfuscators, 3rd USENIX Workshop on Offensive Technologies, 2009
- **Program synthesis of handlers – slow**
 - Blazytko T, Contag M, Aschermann C, Holz T, Syntia: Synthesizing the semantics of obfuscated code, 26th USENIX Security Symposium, pages 643-659, 2017
- **Path coverage and trace merging – limitations and slow**
 - Salwan J, Bardin S, Potet ML, Symbolic deobfuscation: From virtualized code back to the original, International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pages 372-392, 2018

Our approach

- **Virtual instruction summary:**

```
RIP = @64[@64[RBP_init + 0xA4] + 0x5A8]
@64[RBP_init + 0x28] = @64[RBP_init + 0x28] + 0x8
@64[RBP_init + 0x141] = @64[RBP_init + 0x141] + 0x8
@64[RBP_init + 0x12A] = @64[RSP_init]
```

- **Build a graph from summaries**
- **Treat some values as concrete:**
 - Rolling decryption registers
 - Memory accesses relative to the bytecode pointer
- **Between blocks preserve only the decryption registers**



Virtual instruction summary

```
IRDst = ({@16[RBP_init + 0xB] + -(@32[RBP_init + 0x70] ^ {@16[@64[RBP_i  
@16[RBP_init + 0xB] = @16[RBP_init + 0xB] + -(@32[RBP_init + 0x70] ^ {@1  
@32[RBP_init + 0x70] = @32[RBP_init + 0x70] & (@32[RBP_init + 0x70] ^ {@  
@64[RBP_init + {@16[RBP_init + 0xB] + -(@32[RBP_init + 0x70] ^ {@16[@64
```

```
RIP = @64[@64[RBP_init + 0xA4] + 0x5A8]  
@64[RBP_init + 0x28] = @64[RBP_init + 0x28] + 0x8  
@64[RBP_init + 0x141] = @64[RBP_init + 0x141] + 0x8  
@64[RBP_init + 0x12A] = @64[RSP_init]
```

```
RIP = @64[instruction_table + 0x5A8]  
IEIP = IEIP + 0x8  
IRSP = IRSP + 0x8  
REG1 = @64[RSP_init]
```

```
POP REG1
```

Source code init

```
10 class VM1(Wslink):
11     def __init__(self, file_path, reloc=0):
12         super().__init__(file_path, reloc)
13         self.instr_table_addr = 0x11de70
14         self.obf_reg_offsets = {(0x8, 8), (0x11, 32), (0x26, 32), (0x60, 16), (0xbe, 32),
15                                 (0xd4, 32), (0x13b, 32), (0x147, 32), (0x107, 64),
16                                 (0xc2, 64), (0xae, 64), (0x3e, 8), (0x34, 8), (0x102, 8),
17                                 (0x35, 8), (0x3f, 64), (0x92, 64), (0x57, 8),
18                                 (0x1d, 8)}
19
19         self.vm_base = 0xf9206
20         self.vm_pc_off = 0x2c
21         self.instr_table_off = 0xee
22
23
24 reloc = -0xf33f5
25 vma = VM1('extracted_vm.dmp', reloc)
26 first_executed_virt_instr = 0x21dbea + reloc
27 first_virt_instr_in_table = 0x21B4E0 + reloc
28 vma.process(first_executed_virt_instr, 0x19, opt=True)
```

DEOBFUSCATED

```
@64[RBP_init+ 0x15] = @64[RSP_init]
@64[RBP_init+ 0x11F] = @64[RSP_init]
@64[RBP_init+ 0x1E] = RSP_init+ 0x78
@64[RBP_init+ 0x4F] = @64[RSP_init]
@64[RBP_init+ 0x1E] = RSP_init+ 0x80
@64[RBP_init+ 0xCC] = @64[RSP_init]
@64[RBP_init+ 0x1E] = RSP_init+ 0x88
@64[RBP_init+ 0x1E] = RSP_init+ 0x98
@64[RBP_init+ 0x13F] = @64[@64[RBP_init+ 0x13F]]
@32[RBP_init+ 0x53] = 0x0
@32[RBP_init+ 0x4F] = 0x1C
@64[RBP_init+ 0x133] = 0x3092
@64[RBP_init+ 0x133] = @64[RBP_init+ 0x80] + 0x3092
@64[RBP_init+ 0x133] = @64[RBP_init+ 0x80] + 0xE3808
@64[RBP_init+ 0x74] = @64[RBP_init+ 0x80] + 0xE3808
IRDst= loc_key_291
```

loc_key_291

```
== 0x1, 6, 7, (@64[RBP_init+ 0x4F] + 0xFFFFFFFFFFFFFFFF)[63:64], 7, 8, (@32[RBP_init+ 0xCC])[8:11], 8, 11, (@64[RBP_init+ 0x4F] ^ (@64[RBP_init+ 0x4F] + 0xFFFFFFFFFFFFFFFF)) & (@64[RBP_init+ 0x4F] ^ 0x1))[63:64], 11, 12, (@32[RBP_init+ 0xCC])[12:15], 12, 15, 0x0, 15, 16, (@
```

```
9, vif_init, 19, 20, vip_init, 20, 21, i_d_init, 21, 22, 0x0, 22, 32) & 0x40, {0x2, 0, 2, parity(@32[RBP_init+ 0xCC] & 0x40), 2, 3, 0x8, 3, 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,
```

loc_key_318

```
@64[RBP_init+ 0x133] = 0x30A2
@64[RBP_init+ 0x133] = @64[RBP_init+ 0x80] + 0x30A2
@64[RBP_init+ 0x133] = @64[RBP_init+ 0x80] + 0x2FB0
@64[RBP_init+ 0x15] = @64[RBP_init+ 0x80] + 0x2FB0
@64[RBP_init+ 0x11F] = @64[RBP_init+ 0x13F]
@64[RBP_init+ 0x133] = 0x30AB
@64[RBP_init+ 0x133] = @64[RBP_init+ 0x80] + 0x30AB
@64[RBP_init+ 0x133] = @64[RBP_init+ 0x80] + 0x8C038
@64[RSP_init+ 0xFFFFFFFFFFFFFFF8] = @64[RBP_init+ 0x4F]
@64[RBP_init+ 0x1E] = @64[RBP_init+ 0x1E] + 0xFFFFFFFFFFFFFFF8
@64[RSP_init+ 0xFFFFFFFFFFFFFFF8] = @64[RBP_init+ 0x4F]
@64[RBP_init+ 0x1E] = @64[RBP_init+ 0x1E] + 0xFFFFFFFFFFFFFFF8
@64[RSP_init+ 0xFFFFFFFFFFFFFFF8] = @64[RBP_init+ 0xCC]
@64[RBP_init+ 0x1E] = @64[RBP_init+ 0x1E] + 0xFFFFFFFFFFFFFFF8
@64[RSP_init+ 0xFFFFFFFFFFFFFFF8] = @64[RBP_init+ 0x4F]
@64[RBP_init+ 0x1E] = @64[RBP_init+ 0x1E] + 0xFFFFFFFFFFFFFFF8
@64[RSP_init+ 0xFFFFFFFFFFFFFFF8] = @64[RBP_init+ 0x11F]
@64[RBP_init+ 0x1E] = @64[RBP_init+ 0x1E] + 0xFFFFFFFFFFFFFFF8
@64[RSP_init+ 0xFFFFFFFFFFFFFFF8] = @64[RBP_init+ 0x15]
```

```

@64[RBP_init+ 0x13F] = @64[RSP_init]
@64[RBP_init+ 0x1E] = RSP_init+ 0x60
@64[RBP_init+ 0x13F] = @64[RSP_init]
@64[RBP_init+ 0x1E] = RSP_init+ 0x68
@64[RBP_init+ 0x1E] = RSP_init+ 0x70
@64[RBP_init+ 0x15] = @64[RSP_init]

```

```

> 48 89 5C 24 10      mov     [rsp+10h], rbx
> 57                  push   rdi
> 48 83 EC 20         sub    rsp, 20h
> 48 8B 1A           mov    rbx, [rdx]
> B8 1C 00 00 00     mov    eax, 28
> 48 8D 3D AF 07 0E 00 lea   rdi, ServiceStatus
> 0F 1F 80 00 00 00 00 nop    dword ptr [rax+00000000h]

```

```
@64[RBP_init + 0xCC] = {032[RBP_init + 0xCC][0:1] 0 1, 0x1 1 2, p
```

```
@64[RBP_init + 0x4F] = @64[RBP_init + 0x4F] + 0xFFFFFFFFFFFFFFFF
```

REG1 -= 1

```
@64[RBP_init + 0x133] = @64[RBP_init + 0x4F]
```

```
@64[RBP_init + 0x133] = @64[RBP_init + 0x4F] + @64[RBP_init + 0x74]
```

```
@8[@64[RBP_init + 0x4F] + @64[RBP_init + 0x74]] = 0x0
```

[REG1+REG2] = 0

```
IRDst = ((@32[RBP_init + 0xCC] & 0x40)?({0x2 0 2, parity(@32[RBP_init
```

```

@64[RBP_init+ 0x133] = @64[RBP_init+ 0x80] + 0xE3808
@64[RBP_init+ 0x74] = @64[RBP_init+ 0x80] + 0xE3808
IRDst= loc_key_291

```

```

72 48 11 C3
>28 C6 04 38 00
>28 75 F7

```

```

loc_180003060:
dec     rax
mov     byte ptr [rax+rdi], 0
jnz    short loc_180003060

```

```

FF
lea    rdx, HandlerProc ; lpHandlerPr
mov    rcx, rbx ; lpServiceNam
call   cs:RegisterServiceCtrlHandlerK
mov    cs:hServiceStatus, rax
test   rax, rax
jz     short loc_1800030FD

```

loc_key_291

```
[RBP_init+ 0x4F] + 0xFFFFFFFFFFFFFFFF)[63:64], 7, 8, (@32[RBP_init+ 0xCC])[8:11], 8, 11, ((@64[RBP_init+ 0x4F] ^ (@64[RBP_init+ 0x4F] + 0xFFFFFFFFFFFFFFFF)) & (@64[RBP_init+ 0x4F] ^ 0x1))[63:64], 11, 12, (@32[RBP
```

```
20, vip_init, 20, 21, i_d_init, 21, 22, 0x0, 22, 32) & 0x40, {0x2, 0, 2, parity(@32[RBP_init+ 0xCC] & 0x40), 2, 3, 0x8, 3, 8, tf_init, 8, 9, i_f_init, 9, 10, df_init, 10, 11, 0x0, 11, 12, iopl_f_init, 12, 14, nt_init, 1
```

DEOBFUSCATED

ORIGINAL SAMPLE

Analyzing results

Bytecode block	VM1	VM2
Size in bytes	695	1,145
Total number of processed virtual instructions	62	109
Total number of underlying native instructions	3,536,427	17,406
Total number of resulting IR instructions (including IRDstS)	192	307
Execution time in seconds	382	10

Thank you