



MAKING ROP BETTER WITH CPU EMULATION AND REALTIME ANALYSIS

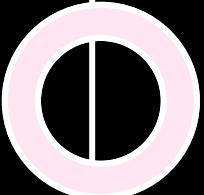


DYLAN KNOFF (ELBEE_EZ@PROTONMAIL.COM)

[HTTPS://FAULTPOINT.COM/POST/2024-06-02-BINJA-PLUGIN-ROPVIEW/](https://FAULTPOINT.COM/POST/2024-06-02-BINJA-PLUGIN-ROPVIEW/)

DISTRICTCON YEAR ZERO





Agenda

What is ROPView?

Components

Unicorn Engine

Gadget Emulation and Analysis

Data Analysis and Semantic Search

Demos

Pitfalls



Who am I

Junior at GMU (CompSci)

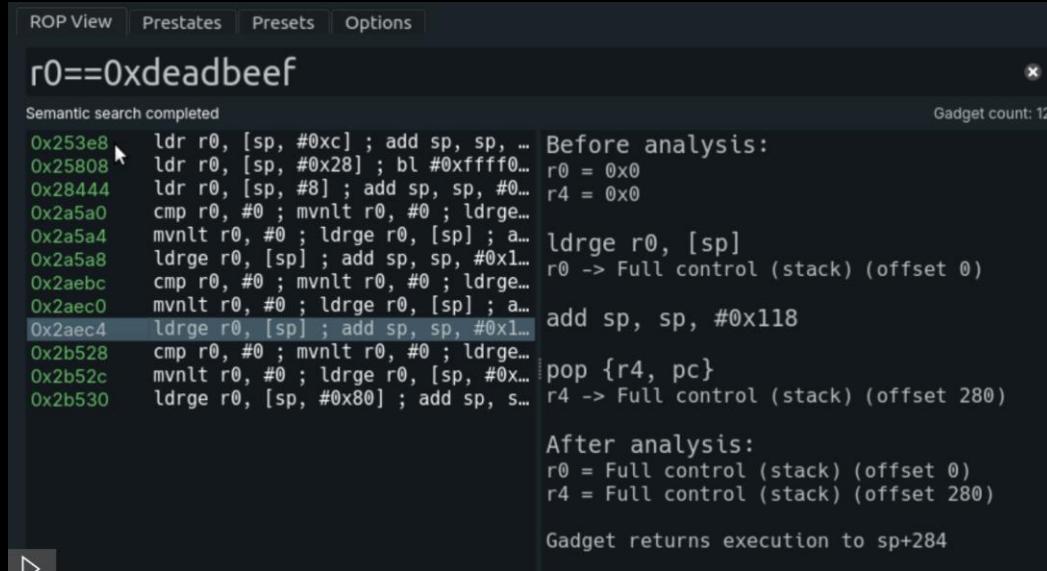
Experience with RE/VR in embedded devices
(but interested in other areas of RE)

Mason Competitive Cyber, U.S. Cyber Team
(previously), CTF player, Co-founder @
bhartee.ai

Interested in emulation, automating
weaponization, and tooling



Introducing ROPView



The screenshot shows the ROPView plugin interface within Binary Ninja. The title bar says 'r0==0xdeadbeef'. The menu bar includes 'ROP View', 'Prestates', 'Presets', and 'Options'. The main area displays assembly code with annotations. A tooltip 'Before analysis:' shows register values: r0 = 0x0, r4 = 0x0. It then lists instructions: ldr r0, [sp, #0x28]; bl #0xfffff0...; ldr r0, [sp, #8]; add sp, sp, #0...; cmp r0, #0; mvnlt r0, #0; ldrge...; mvnlt r0, #0; ldrge r0, [sp]; a...; ldrge r0, [sp]; add sp, sp, #0x1...; cmp r0, #0; mvnlt r0, #0; ldrge...; mvnlt r0, #0; ldrge r0, [sp]; a...; ldrge r0, [sp]; add sp, sp, #0x1...; cmp r0, #0; mvnlt r0, #0; ldrge...; mvnlt r0, #0; ldrge r0, [sp, #0x...; ldrge r0, [sp, #0x80]; add sp, s...'. A tooltip 'After analysis:' shows r0 = Full control (stack) (offset 0), r4 = Full control (stack) (offset 280). A note at the bottom says 'Gadget returns execution to sp+284'. The status bar at the bottom right shows 'Gadget count: 12'.

Plugin for BinaryNinja, featureful Gadget Searcher/Analyzer

Per-instruction effect analysis (backed by Unicorn)

Dataframe aggregation for pandas-based search queries

Caching, usual options, search presets, analysis prestates

armv7, thumb2, aarch64, mipsel32+64, mips32+64, i386, and amd64 (extendable constants)



Presets

Create

cet

translates to:

disasm.str.contains("endbr64") and inst_cnt < 10

Add

Current Presets

- stackfinder
- system
- tails
- lia0
- registers
- sleep_a0

\$a0 > 0 and \$a0 < 600



Prestates (/w Corefile support!)

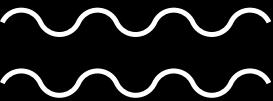
ROP View | Prestates | Presets | Options

Analysis Prestate

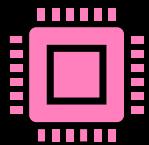
rax=	0xfffffffffffffe00	r9=	0x0
rbx=	0x0	r10=	0x4021c4
rcx=	0x7ffff7d147e2	r11=	0x246
rdx=	0x32	r12=	0x7fffffff0c8
rsi=	0x7fffffffde20	r13=	0x40177f
rdi=	0x0	r14=	0x403e18
rbp=	0x7fffffffde90	r15=	0x7ffff7ffd040
r8=	0x3		

Warning: The corefile feature is currently in beta and may be incomplete.





Components



Discovery

Gadget discovery handled in-house using Binja API (cached)



Analyzer

Handled via Unicorn emulations and hooks (cached)



Aggregator

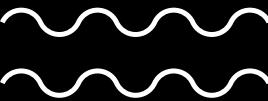
Handled via pandas, sorted by attributes and by register effects (uses Analysis States)



Renderer

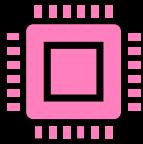
Handles gadget repooling events and cache coherency





Components

Coolest!



Discovery

Gadget discovery handled in-house using Binja API (cached)



Analyzer

Handled via Unicorn emulations and hooks (cached)

- Deterministic and self-resolving
- Lightweight, even with analysis steps + hooks!



Aggregator

Handled via pandas, sorted by attributes and by register effects (uses Analysis States)



Renderer

Handles gadget repooling events and cache coherency





Unicorn Engine

```
1 # i386
2 mu = Uc(UC_ARCH_X86, UC_MODE_32)
3
4 # write machine code to be emulated to memory
5 mu.mem_map(CODE_BASE, 0x1000)
6 mu.mem_write(CODE_BASE, X86_CODE32)
7 mu.mem_protect(CODE_BASE, 0x1000, (UC_PROT_READ+UC_PROT_EXEC))
8
9 # stack
10 mu.mem_map(STACK_BASE, 0x1000)
11 mu.mem_write(STACK_BASE+0x100, STACK_DATA)
12 mu.mem_protect(STACK_BASE, 0x1000, (UC_PROT_READ+UC_PROT_WRITE))
13 mu.reg_write(UC_X86_REG_ESP, STACK_BASE+0x100)
14
15 # emulate code in infinite time & unlimited instructions
16 mu.emu_start(CODE_BASE, CODE_BASE + len(X86_CODE32))
<
```

<https://www.unicorn-engine.org>

- Lightweight and contextless CPU emulation framework backed by Qemu
- Easy to use (mapping, write/read, register access) (+ python bindings)
- Lots of (mostly implemented) hooks including:
 - Before a insn is executed
 - When a memory violation occurs

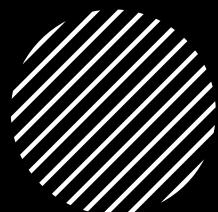
Drawbacks

- Hooks slow emulation
- Space-complex due to mass instancing (especially during semantic searches)





```
Before analysis:  
No registers to list (gadget doesn't clobber)  
  
mov qword ptr [r14], r15  
0x600df0 -> b'DCBA\x00\x00\x00\x00' (.init_array)  
  
ret  
  
After analysis:  
*0x600df0 = b'DCBA\x00\x00\x00\x00' (.init_array)  
  
Gadget returns execution to sp+0
```

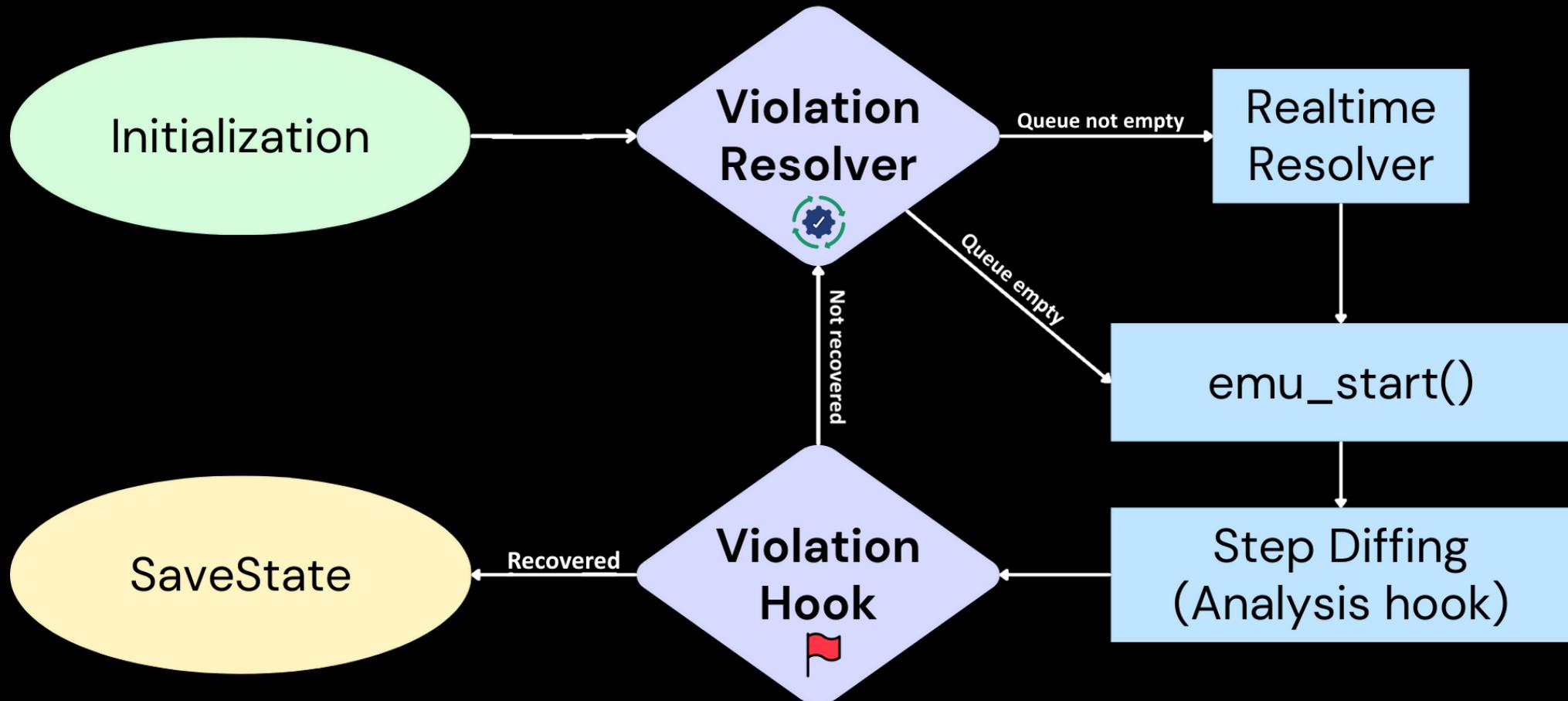


Analyzer

- Emulates and caches effects of selected item.
- The issue: Gadget emulations should be small, but context is large
- Resolver/Analysis Steps
 - Analysis hooks
 - Hook executes in-between fetch and execute and diffs clobbered registers/memory (populates Analysis pane + Dataframe columns).
 - CPU Exception Hooks
 - Hook exceptions to correct emu as needed and track errors. (ie unmapped memory is fetched and marked executable only when an access violation occurs).

Gadget Emulation Framework

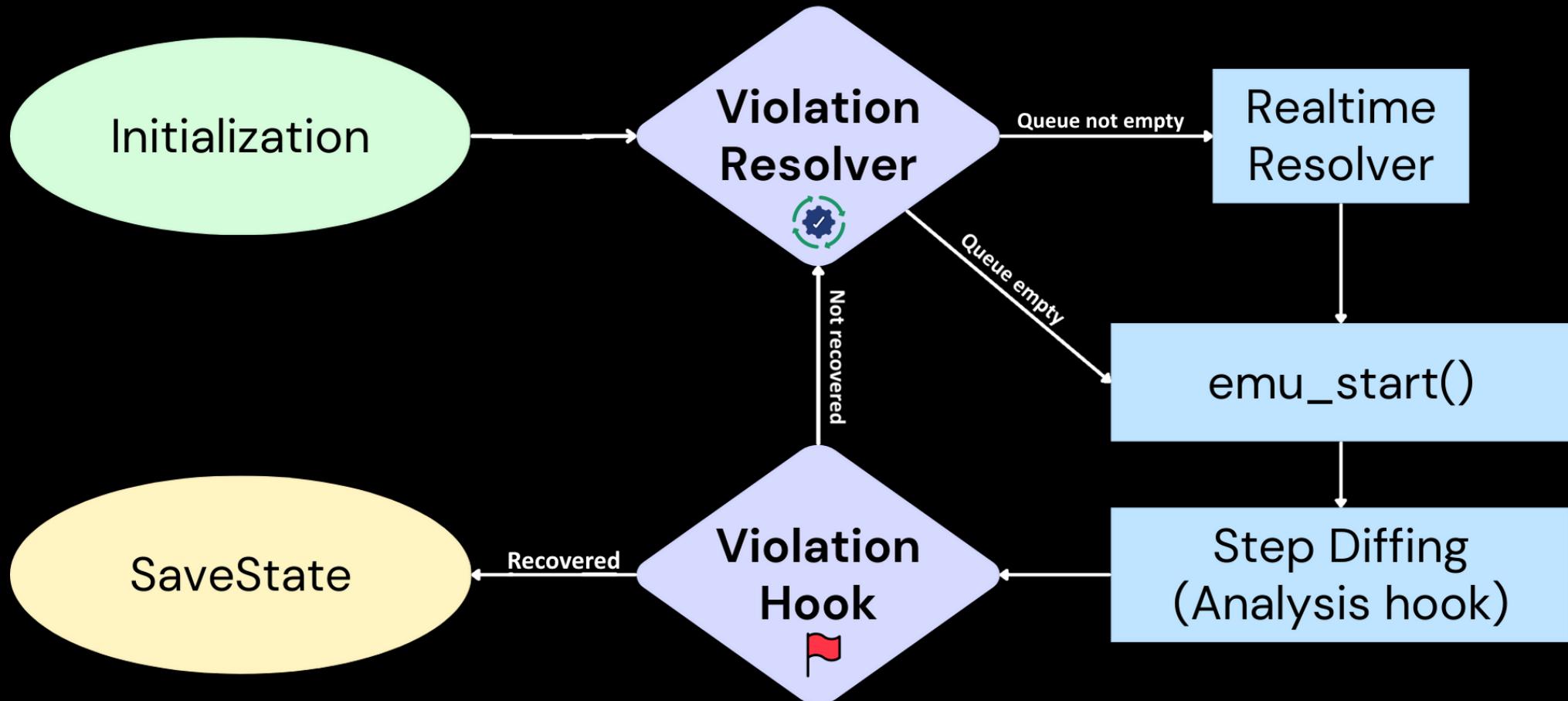
```
mov [r14], r15; ret;  
r14==0x600df0, r15=0x1234
```



Gadget Emulation Framework

```
mov [r14], r15; ret;  
r14==0x600df0, r15=0x1234
```

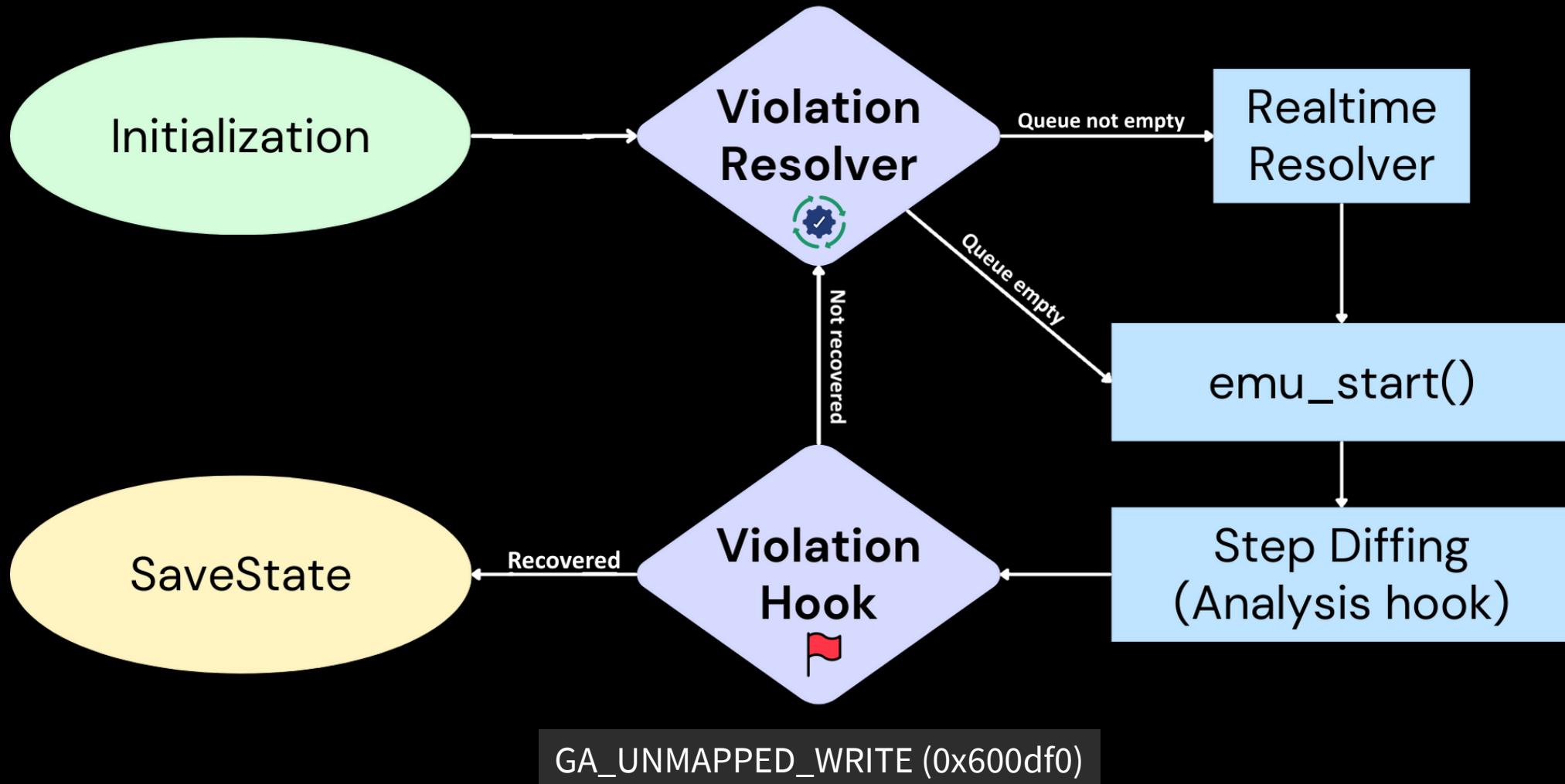
Resolver Queue:
Empty



Gadget Emulation Framework

```
mov [r14], r15; ret;  
r14==0x600df0, r15=0x1234
```

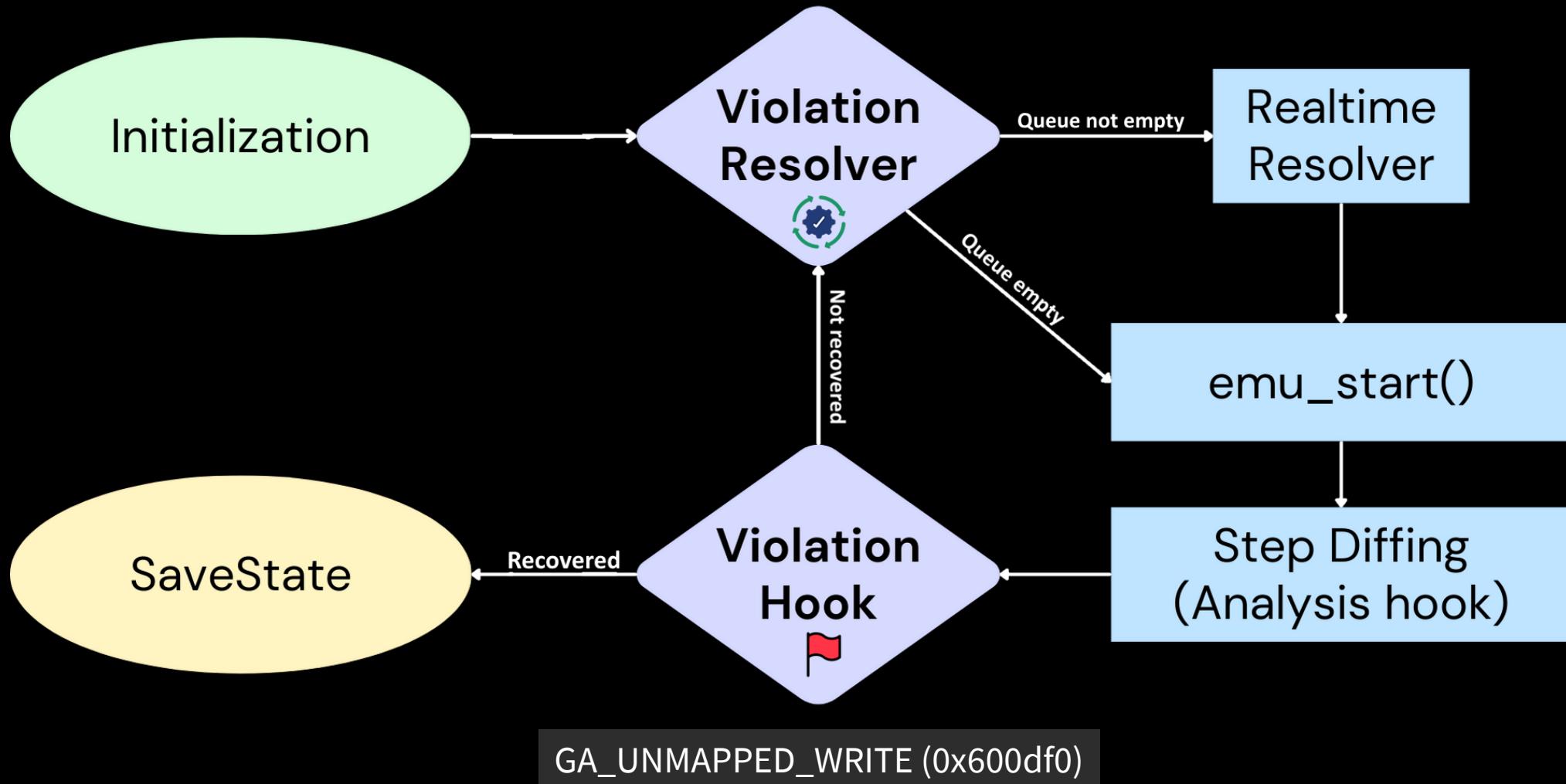
Resolver Queue:
Empty



Gadget Emulation Framework

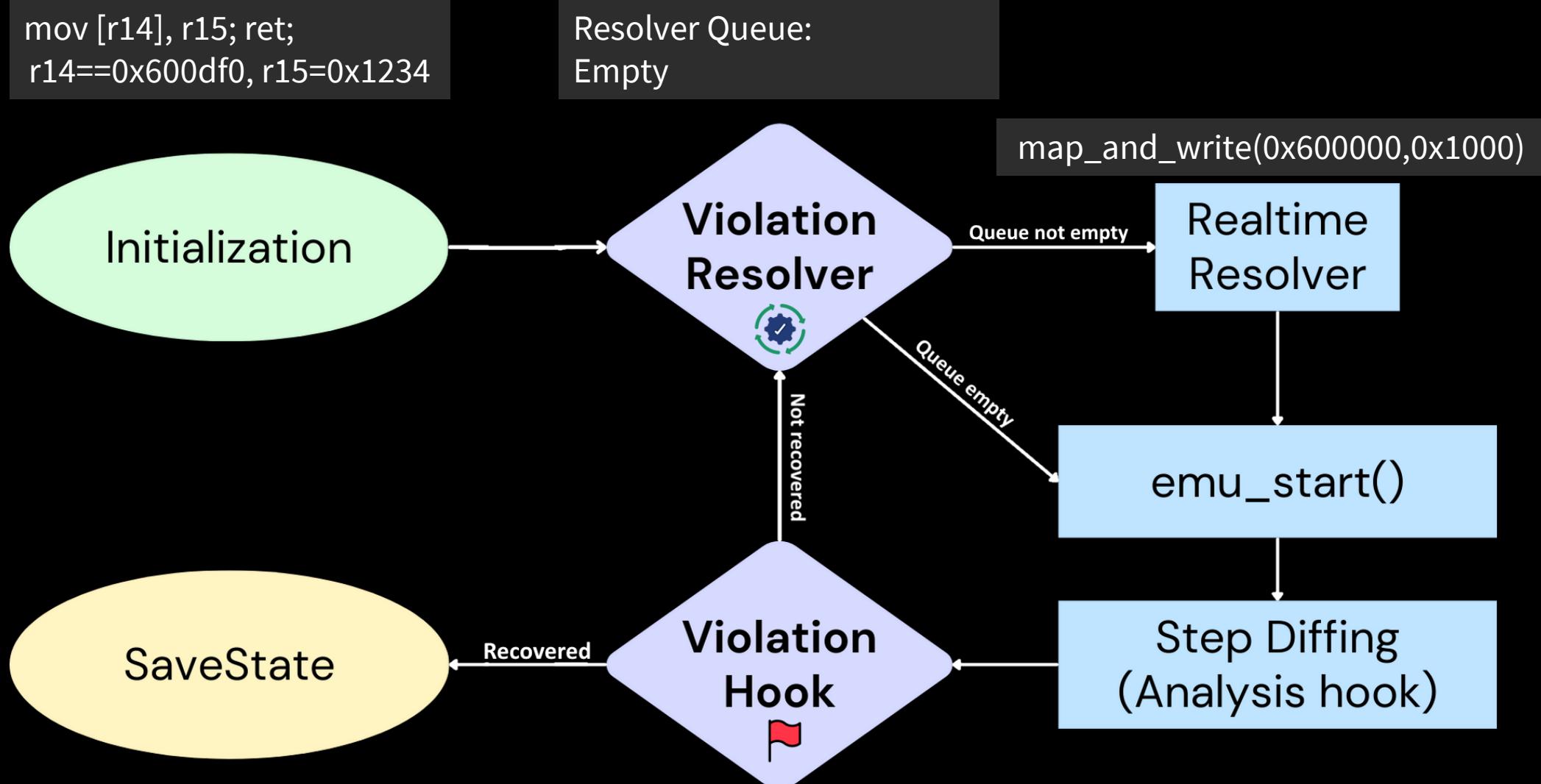
```
mov [r14], r15; ret;  
r14==0x600df0, r15=0x1234
```

Resolver Queue:
(0x600df0)

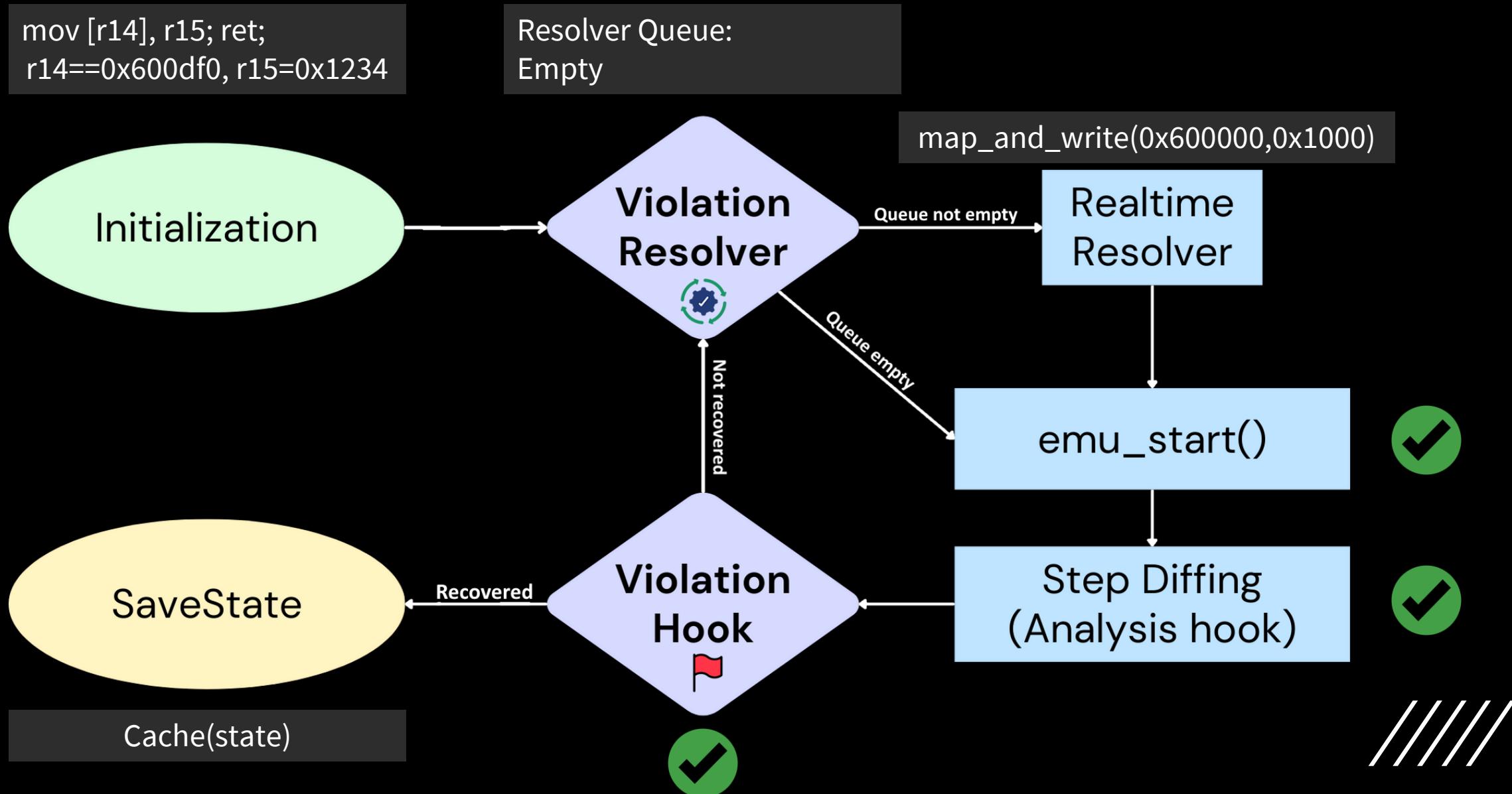


GA_UNMAPPED_WRITE (0x600df0)

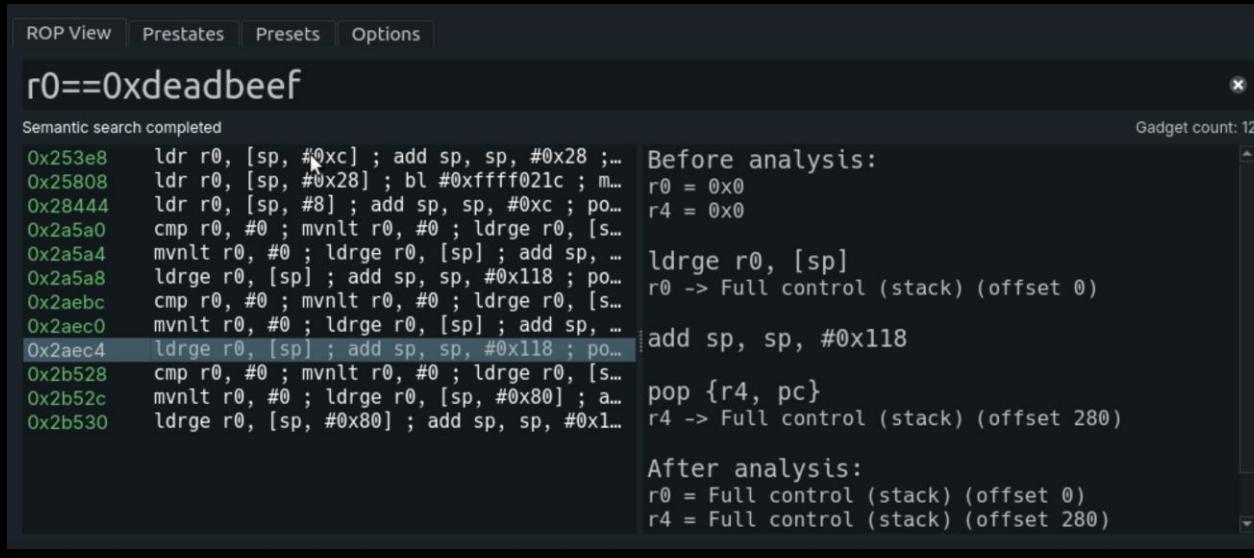
Gadget Emulation Framework



Gadget Emulation Framework



Data Analysis



ROP View Prestates Presets Options

r0==0xdeadbeef

Semantic search completed

Before analysis:

```
0x253e8 ldr r0, [sp, #0xc] ; add sp, sp, #0x28 ;...
0x25808 ldr r0, [sp, #0x28] ; bl #0xfffff021c ; m...
0x28444 ldr r0, [sp, #8] ; add sp, sp, #0xc ; po...
0x2a5a0 cmp r0, #0 ; mvnlt r0, #0 ; ldrge r0, [s...
0x2a5a4 mvnlt r0, #0 ; ldrge r0, [sp] ; add sp, ...
0x2a5a8 ldrge r0, [sp] ; add sp, sp, #0x118 ; po...
0x2aebc cmp r0, #0 ; mvnlt r0, #0 ; ldrge r0, [s...
0x2aec0 mvnlt r0, #0 ; ldrge r0, [sp] ; add sp, ...
0x2aec4 ldrge r0, [sp] ; add sp, sp, #0x118 ; po...
0x2b528 cmp r0, #0 ; mvnlt r0, #0 ; ldrge r0, [s...
0x2b52c mvnlt r0, #0 ; ldrge r0, [sp, #0x80] ; a...
0x2b530 ldrge r0, [sp, #0x80] ; add sp, sp, #0x1...
```

add sp, sp, #0x118

pop {r4, pc}

r4 -> Full control (stack) (offset 280)

After analysis:

```
r0 = Full control (stack) (offset 0)
r4 = Full control (stack) (offset 280)
```



```
df.query("((reg==0xdeadbeef or reg==FULL_CONTROL) and not reg==NOT_ANALYZED)")
```

Dataframe objects includes:

- Addr (int) – Address literal
- Loc (str) – Address as hex string
- Bytes (str) – Raw asm
- Inst_cnt (int) – Instruction count
- Disasm (str) – Gadget disassembly
- Registers (int[]) – Register values (analysis-based)
- Presets

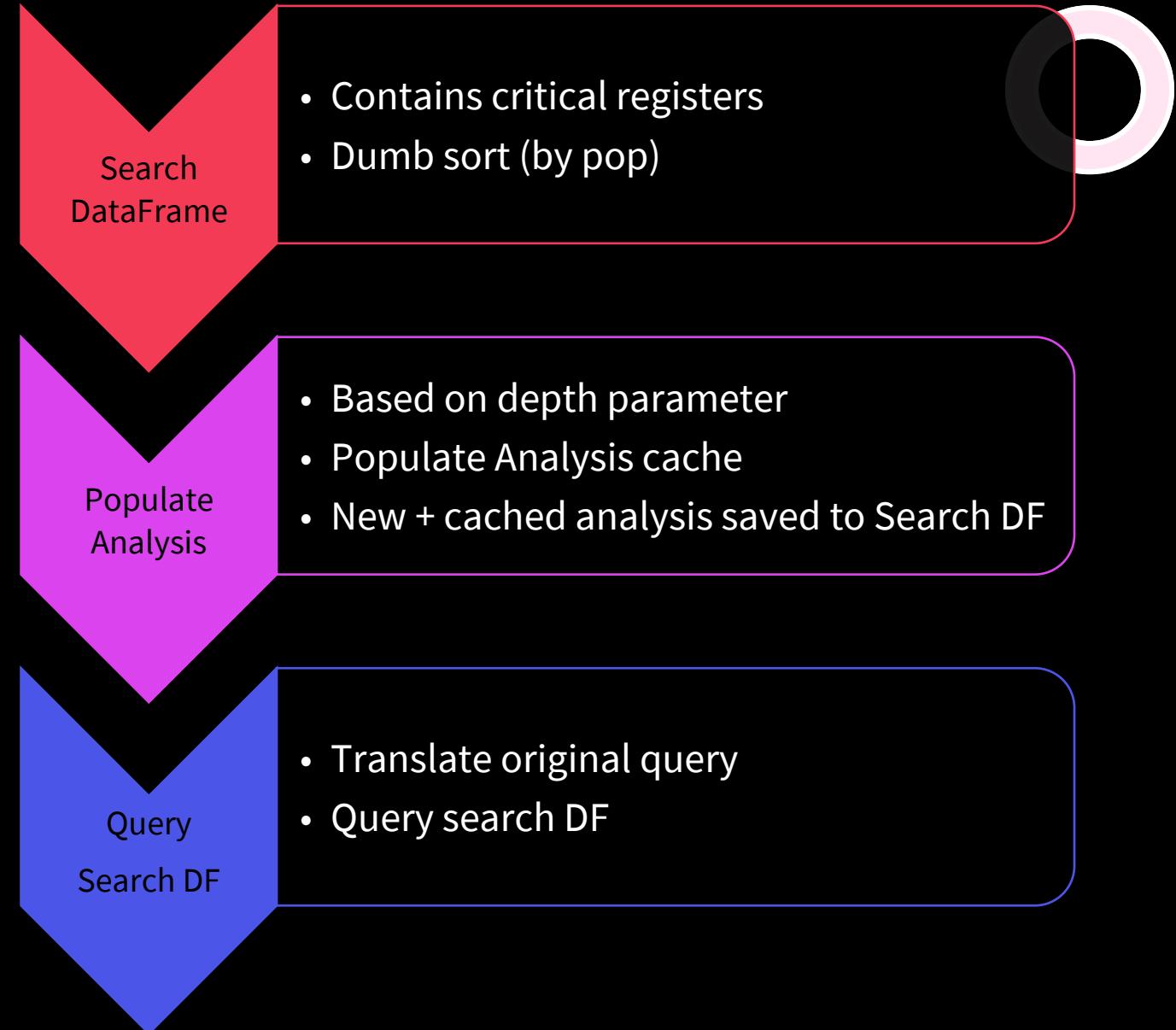
All objects can be queried with the respective pandas functions available for that object.

pandas.Series.str.cat	pandas.Series.str.casefold
pandas.Series.str.center	pandas.Series.str.extractall
pandas.Series.str.contains	pandas.Series.str.find
pandas.Series.str.count	pandas.Series.str.findall
pandas.Series.str.decode	pandas.Series.str.fullmatch
pandas.Series.str.encode	pandas.Series.str.get
pandas.Series.str.endswith	pandas.Series.str.index
pandas.Series.str.extract	pandas.Series.str.join

<https://pandas.pydata.org/docs/reference/api/pandas.Series.str.html>



SemanticSearch





Usage



All bytes except the 2 MSB must be in ASCII range + extra constraints

ROP View **Prestates** Presets Options

addr > 0x10000 and loc.match("0x[00-ff]{2}[20-7f]{2,}") and not loc.contains('00')

Gadget count: 4076

```
0x9840      pop {sl, pc} ;
0x9890      pop {r4, r5, pc} ;
0x988c      strb r3, [r5] ; pop {r4, r5, pc} ;
0x9888      mov r3, #1 ; strb r3, [r5] ; pop {r4, r5, pc} ;
0x9884      bne #0xfe4 ; mov r3, #1 ; strb r3, [r5] ; pop {r4, r5, pc} ;
0x9880      cmp r2, #0 ; bne #0xfe8 ; mov r3, #1 ; strb r3, [r5] ; pop {r4, r5, pc} ;
0x987c      ldr r2, [r3] ; cmp r2, #0 ; bne #0xfec ; mov r3, #1 ; strb r3, [r5] ; pop {r4, r5, pc} ;
0x9878      ldr r3, [r4] ; ldr r2, [r3] ; cmp r2, #0 ; bne #0xfec ; mov r3, #1 ; strb r3, [r5] ; pop {r4, r5, pc} ;
0x995c      str r3, [r4] ; pop {r4, r5, pc} ;
0x9958      ldr r3, [pc, #0x68] ; str r3, [r4] ; pop {r4, r5, pc} ;
0x9954      mov r0, #0 ; ldr r3, [pc, #0x68] ; str r3, [r4] ; pop {r4, r5, pc} ;
0x996c      mov r0, #0 ; pop {r4, r5, pc} ;
0x9968      str r3, [r0] ; mov r0, #0 ; pop {r4, r5, pc} ;
0x9964      ldr r3, [pc, #0x5c] ; str r3, [r0] ; mov r0, #0 ; pop {r4, r5, pc} ;
0x99bc      mov r0, #2 ; str r3, [r4] ; pop {r4, r5, pc} ;
0x99b8      ldr r3, [pc, #0x14] ; mov r0, #2 ; str r3, [r4] ; pop {r4, r5, pc} ;
```

```
Before analysis:  
No registers to list (gadget doesn't clobber)  
  
ble #0x10c4  
Control flow changed -> Branch to 0x10c4  
  
add r4, r4, #2  
:  
cmp r4, r5  
  
bne #0xf94  
  
mvn r0, #0  
  
pop {r4, r5, r6, r7, r8, pc}
```

https://faultpoint.com/assets/ropview_demos/ascii.gif

sleep_a0 -> (a0 > 0 and a0 < 600)

ROP View Prestates Presets Options

sleep_a0

Gadget count: 82940

```
0xf4294    beqz $v0, 0x1044 ; addiu $s6, $s6, 4 ; ...
0xf4298    addiu $s6, $s6, 4 ; lw $a1, ($s6) ; lw ...
0xf429c    lw $a1, ($s6) ; lw $a0, 0x1c($s3) ; sll...
0xf42a0    lw $a0, 0x1c($s3) ; sll $v0, $a1, 1 ; a...
0xf42a4    sll $v0, $a1, 1 ; addu $v0, $v0, $a1 ; ...
0xf42a8    addu $v0, $v0, $a1 ; sll $v0, $v0, 2 ; ...
0xf42ac    sll $v0, $v0, 2 ; lw ...
0xf42b0    lw $a1, 0x90($sp) ;
0xf42b4    lw $t9, 0x94($sp) ;
0xf42b8    jalr $t9 ; addu $a0,
0xf43e0    beqz $v0, 0x1218 ; s
0xf43e4    sw $v0, 0x24($s0) ;
0xf43e8    lb $v1, 0x34($v0) ; ...
0xf43ec    bltz $v1, 0x1198 ; lw $s3, 0x98($sp) ; ...
0xf43f0    lw $s3, 0x98($sp) ; sw $v0, 0x30($s0) ;...
0xf43f4    sw $v0, 0x30($s0) ; sw $v0, 0x2c($s0) ;...
```

Performing semantic search

78%

Cancel

https://faultpoint.com/assets/ropview_demos/sleepa0.gif



Pop-pop-ret, control r13, avoid pivots, constrain range

ROP View

Prestates

Presets

Options

ppr and disasm.contains('r13') and not disasm.contains('rsp') and addr < 0x44d44| 

Gadget count: 19115

		Before analysis:
0xf4244	pop rbx ; pop rbp ; pop r12 ; pop r13 ; pop r1...	rbx = 0x0
0xf4245	pop rbp ; pop r12 ; pop r13 ; pop r14 ; pop r1...	rbp = 0x0
0xf4246	pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret ;	r12 = 0x0
0xf4247	pop rsp ; pop r13 ; pop r14 ; pop r15 ; ret ;	r13 = 0x0
0xf4248	pop r13 ; pop r14 ; pop r15 ; ret ;	r14 = 0x0
0xf4249	pop rbp ; pop r14 ; pop r15 ; ret ;	r15 = 0x0
0xf424a	pop r14 ; pop r15 ; ret ;	
0xf424b	pop rsi ; pop r15 ; ret ;	
0xf424c	pop r15 ; ret ;	pop rbx
0xf424d	pop rdi ; ret ;	rbx -> Full control (stack) (offset 0)
0xf424e	ret ;	
0xf4333	clc ; pop rbp ; pop r12 ; pop r13 ; pop r14 ; ...	pop rbp
0xf43d9	neg eax ; pop r12 ; pop r13 ; pop r14 ; pop r1...	rbp -> Full control (stack) (offset 8)
0xf43da	fadd dword ptr [rcx + 0x5c] ; pop r13 ; pop r1...	
0xf45a5	add rsp, 8 ; pop rbx ; pop rbp ; pop r12 ; pop...	pop r12
0xf45a6	add esp, 8 ; pop rbx ; pop rbp ; pop r12 ; pop...	

https://faultpoint.com/assets/ropview_demos/constraints.gif



execve -> (rax==0x3b or rdi==CONTROL or rdx==0 or ... or sys)

ROP View

Prestates

Presets

Options

execve



Searching...

Gadget count: 19115

```
0xf4244    pop rbx ; pop rbp ; pop r12 ; pop r13 ;...
0xf4245    pop rbp ; pop r12 ; pop r13 ; pop r14 ;...
0xf4246    pop r12 ; pop r13 ; pop r14 ; pop r15 ;...
0xf4247    pop rsp ; pop r13 ; pop r14 ; pop r15 ;...
0xf4248    pop r13 ; pop r14 ; pop r15 ; ret ;
0xf4249    pop rbp ; pop r14 ; pop r15 ; ret ;
0xf424a    pop r14 ; pop r15 ; ret ;
0xf424b    pop rsi ; pop r15 ; ret ;
0xf424c    pop r15 ; ret ;
0xf424d    pop rdi ; ret ;
0xf424e    ret ;
0xf4333    clc ; pop rbp ; pop r15 ; pop r14 ;...
0xf43d9    neg eax ; pop r12 ; pop r13 ; pop r14 ;...
0xf43da    fadd dword ptr [rcx + 0x5c] ; pop r13 ;...
0xf45a5    add rsp, 8 ; pop rbx ; pop rbp ; pop r1...
0xf45a6    add esp, 8 ; pop rbx ; pop rbp ; pop r1...
```

Performing semantic search

0%

Cancel

https://faultpoint.com/assets/ropview_demos/execve.gif





Pitfalls

Tuning SemanticSearch parameters

SemanticMatches?

SemanticDepth

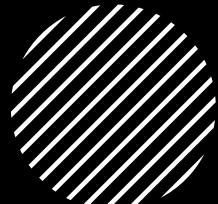
When do we give up?

Stack accuracy

Emulated stack uses cyclic
data, lacks actual stack
context

Limited by dependencies

Emulation issues /w unicorn
possible





ROPVIEW

EXPLOIT DEVELOPMENT FRAMEWORK

Version 2 released!

<https://github.com/elbee-cyber/RopView>





Questions?

