# Reverse Debugging with radare2

Ren Kimura (@RKX1209)

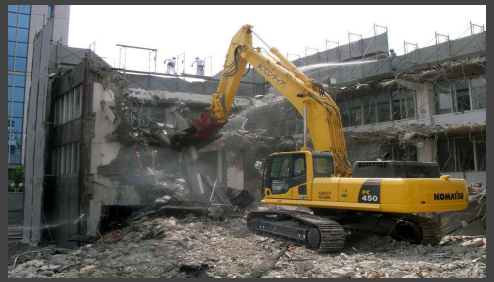
# whoami? - @RKX1209

- University student in Japan
- Mainly focused on Kernel Exploitation and Jailbreak

BTW: There are some cool **Japanese words** in r2-related projects:)

***居合刀 (Iaito)***

***解体 (Kaitai)***

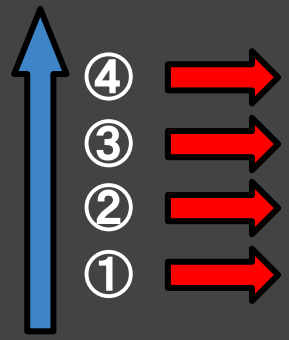


# GSoC works

## “Add Reverse Debugging support to r2”

What’s Reverse Debugging?

In short, Enable to seek program counter backward.




```
0x00400536    push rbp
0x00400537    mov rbp, rsp
0x0040053a    mov edi, str.Hello_World
0x0040053f    call sym.imp.puts
```

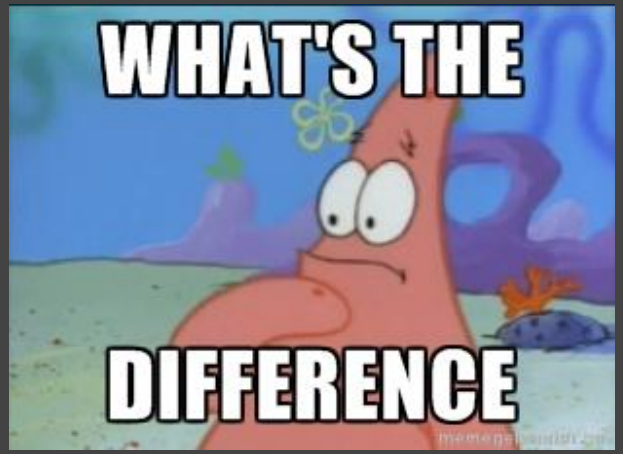
**Need to restore %edi and %rbp to previous value. And also stack state.**

***Step back, Step back, Step back, Step back…..***

# Approaches

There are some approaches to implement Reverse Debugging.

- **Timeless Debugging**
  - Original GSoC Project title is “Timeless Debugging support”.
- **Record and Replay**

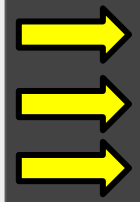


# Timeless Debugging

Records **all operations** like, load/store memory, regsiters…

geohot's **qira** uses QEMU for recording.

```
0x00400536    push rbp
0x00400537    mov rbp, rsp
0x0040053a    mov edi, str.Hello_World
0x0040053f    call sym.imp.puts
```



**[stack_addr] <= %rbp**
**%rbp <= %rsp**
**%edi <= [str_addr]**

***Records per operations***

**This approach is not suitable for radare2...**

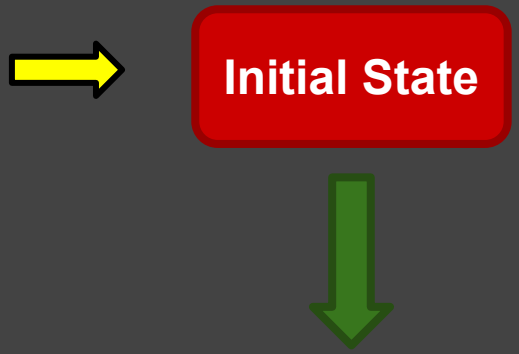
# Record and Replay(RnR)

Record **Initial program state** and **some events**,

then replay from it.

```
0x00400536    push rbp
0x00400537    mov rbp, rsp
0x0040053a    mov edi, str.Hello_World
0x0040053f    call sym.imp.puts
```

**Save Initial program state by ptrace(2)**




**Replay until desired point**

**It looks nice for r2 architecture!**

# r2 recorder

In r2, program record is called as “Trace Session”.

You can use **dts (debug trace session)** command.

| | |
|---|---|
| **dts** | **List all trace sessions** |
| **dts+/-** | **Add/Delete trace session** |
| **dtst/f [file]** | **Read/Save trace session** |
| **dtsC <id> <comment>** | **Add comment for given trace session** |

More detail. Let’s type “**dts?**” in your own r2 debugger console.

# Record and Replay for r2

Firstly you need to record **Initial program state** by “dts+”.

**Save current program state by “dts+”**

**Current PC**

```
0x00400536    push rbp
0x00400537    mov rbp, rsp
0x0040053a    mov edi, str.Hello_World
0x0040053f    call sym.imp.puts
```

**Trace Session**

# Record and Replay for r2

Then, you can step out or continue as usual.

**Go forward by dso, dc or dcu….**

```
0x00400536    push rbp
0x00400537    mov rbp, rsp
0x0040053a    mov edi, str.Hello_World
0x0040053f    call sym.imp.puts
```

**Current PC**

**Trace Session**

# Record and Replay for r2

OK. Let's back one step by "**dsb" (debug step back)** command.

**Currently, pc is at 40053f and you want to step back to 40053a.**

**Trace Session**

```
0x00400536    push rbp
0x00400537    mov rbp, rsp
0x0040053a    mov edi, str.Hello_World
0x0040053f    call sym.imp.puts
```

**Current PC**

# Record and Replay for r2

Reverse debugging commands firstly, restore program state to **previous Trace Session**.

**Current PC**

```
0x00400536    push rbp
0x00400537    mov rbp, rsp
0x0040053a    mov edi, str.Hello_World
0x0040053f    call sym.imp.puts
```

**Trace Session**

**Restore state**

# Record and Replay for r2

Then, replay until previous address.(i.e. 0x40053a)

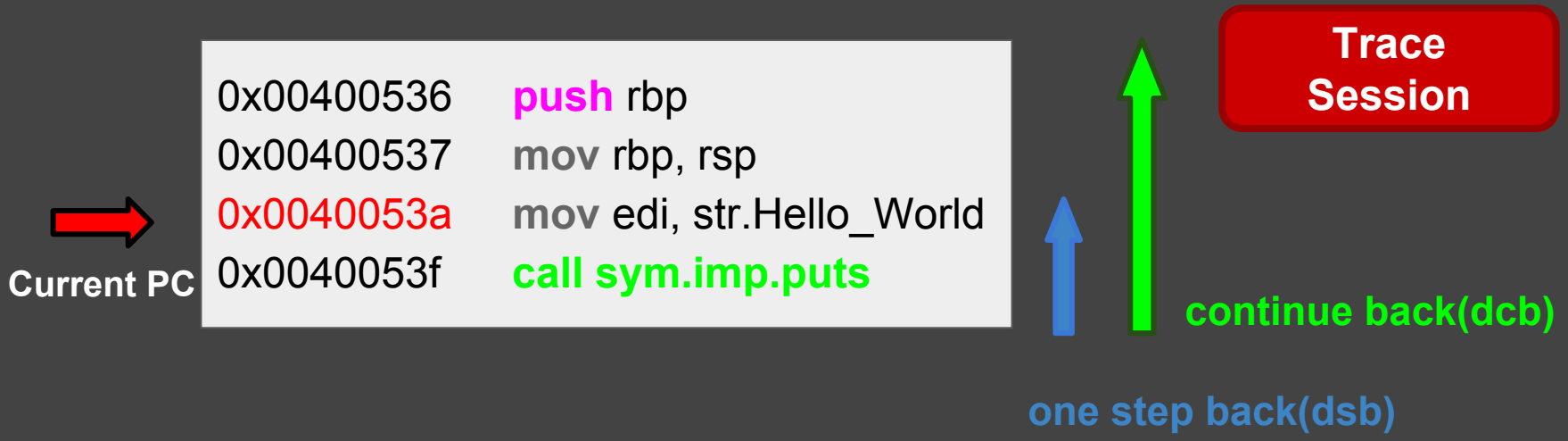
Current PC

```
0x00400536    push rbp
0x00400537    mov rbp, rsp
0x0040053a    mov edi, str.Hello_World
0x0040053f    call sym.imp.puts
```


Replay

# Reverse Debugging for r2

You can also **continue back(dcb)** that seeks program counter backward until hit the breakpoint.



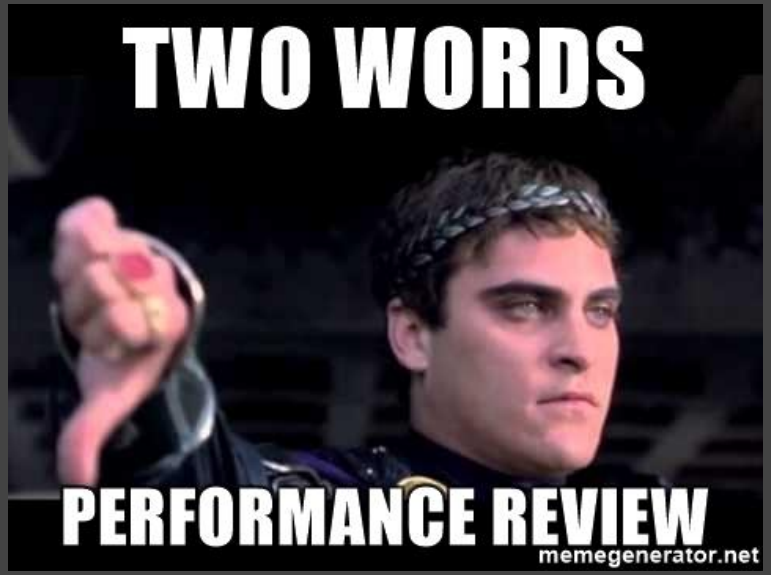
# DEMO

Reverse Debugging with radare2

# Performance problem(Execution time)

When you run reverse debug commands at several time, r2 **always replay** from previous Trace Session.

ex. Long loop iterations,
Heavy memory operations...



# Checkpoint optimization

Reverse Debugger puts some **checkpoints** automatically at first replaying time. Then, replayer can use nearest one.

Current PC

```
(very long operations)
.......
0x00400536    push rbp
0x00400537    mov rbp, rsp
0x0040053a    mov edi, str.Hello_World
0x0040053f    call sym.imp.puts
```

Trace Session — checkpoint 1

Trace Session — checkpoint 2

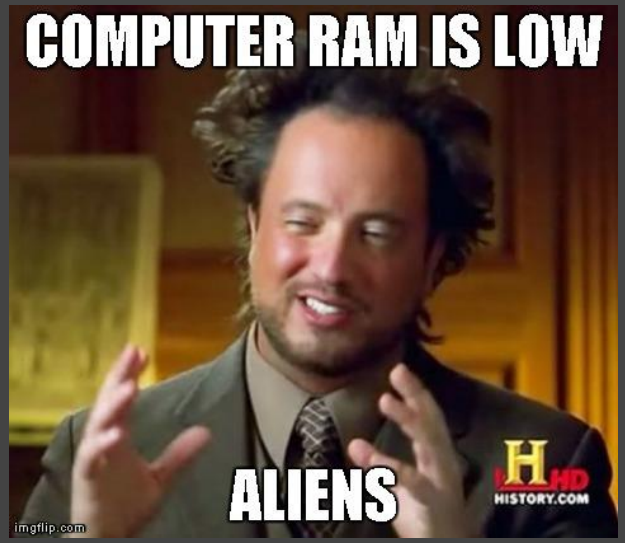
Trace Session — checkpoint 3

Save sessions at replaying time

# Memory size problem

There are many trace sessions(by checkpoint system or ‘dts+’s by user)

Each trace session has **entire program state**, like all memory and register dump. XD



# Trace Session optimization

Trace session should have only **changed parts** in memory from a previous trace session. (like diff snapshot)

Current PC

```
(very long operations)
.......
0x00400536    push rbp
0x00400537    mov rbp, rsp
0x0040053a    mov edi, str.Hello_World
0x0040053f    call sym.imp.puts
```

**Trace Session (base)** — base session has entire dump

**Trace Session (diff1)** — only changed parts from base

**Trace Session (diff2)** — only changed parts from diff1

**Each session has only diff pages**

# Trace Session optimization

## Entire dump(**before**)

**Session 1**

**0x40000-0x40100** s -r-x /bin/ls /bin/ls .r_w_

**0x60000-0x60100** s -r-- /bin/ls /bin/ls .r_w_

**0xfe800-0x109000** s -rw [heap] [heap]

**0x7fb000-0x7ff100** s -r-x /lib/libc-2.23.so

.
.
.
.

**Session 2**

**0x40000-0x40100** s -r-x /bin/ls /bin/ls .r_w_

**0x60000-0x60100** s -r-- /bin/ls /bin/ls .r_w_

**0xfe800-0x109000** s -rw [heap] [heap]

**0x7fb000-0x7ff100** s -r-x /lib/libc-2.23.so

.
.
.
.

**Session 3**

**0x40000-0x40100** s -r-x /bin/ls /bin/ls .r_w_

**0x60000-0x60100** s -r-- /bin/ls /bin/ls .r_w_

**0xfe800-0x109000** s -rw [heap] [heap]

**0x7fb000-0x7ff100** s -r-x /lib/libc-2.23.so

.
.
.
.

.....

# Trace Session optimization

## Diff style session chain(**after**)

**Session 1**

**0x40000-0x42000** s -rw /bin/ls /bin/ls .r_w_

**0x60000-0x61000** s -r-- /bin/ls /bin/ls .r_w_

**0xfe800-0x109000** s -rw [heap] [heap]

**0x7fb000-0x7ff100** s -rw /lib/libc-2.23.so

.
.
.
.

**Session 2**

**0x40000-0x40100**(page 0)

**0x40400-0x40500**(page 4)

**0xfea00-0xfeb00**(page 2)

**Session 3**

**0x7fb00-0x7fc00**(page 0)

.....

**Each session have only changed pages**

# DEMO

List diff sessions

# Reverse Debugging for ESIL

Not only debugger mode but, you can also do reverse debugging for **ESIL mode**.

What is ESIL?

**E**valuable **S**trings **I**ntermediate **L**anguage

sub rsp, 0x648

1608,rsp,-=,$c,cf,=,$z,zf,=,$s,sf,=,$o,of,=

**Application:** Code Emulation, Decompile, VM Emulation….

# Reverse Debugging for ESIL

Not only debugger mode but, you can also reverse debugging for **ESIL mode**.

Save current ESIL state by “aets+”

**Trace Session**

**Current PC**

```
0x00400536    ebp,4,esp,-=,esp,=[4]
0x00400537    esp,ebp,=
0x0040053a
0x0040053f    call sym.imp.puts
```

one step back(aesb)

***Architecture independent Reverse Debugging!***

# DEMO

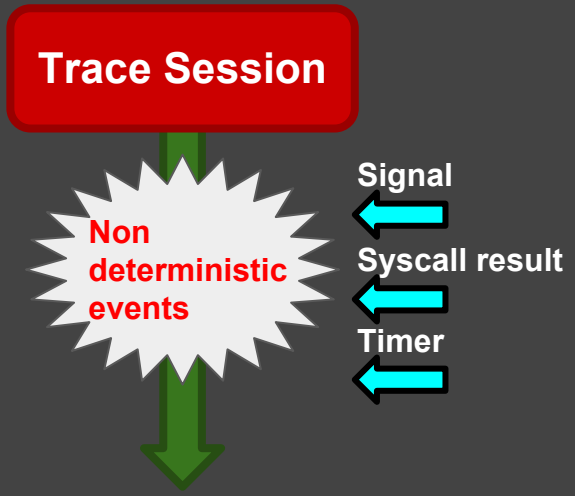
Reverse Debugging for ESIL

# Future work

r2 Reverse Debugger is not supporting **non deterministic events**.(like syscall results, signal….)

Current PC

```
(very long operations)
…….
0x00400536    push rbp
0x00400537    mov rbp, rsp
0x0040053a    mov edi, str.Hello_World
0x0040053f    call sym.imp.puts
```




These events should be replayed!

# Thank you!

**r2 reverse debugger document:**

https://radare.gitbooks.io/radare2book/content/debugger/revdebug.html

**My blog post:**

https://rkx1209.github.io/