Introduction to r2pipe-api

Quim Aguado

whoami

Quim Aguado

CS student in UAB

Twitter: @___quim Telegram: @qui_m

Currently working on my final degree project in BSC

Interested in reversing, exploiting, parallel programming all kind of low level stuff

r2pipe?

a magical pipe where you throw radare2 commands at, and it'll answer you their results[1]

Allows to script r2 commands, in many programming languages.

```
>>> import r2pipe
>>> r = r2pipe.open('/bin/ls')
>>> r.cmd('aa')
''
>>> r.cmdj('aflj')[0]['name']
'sub.strcoll_f20'
```

```
>>> r.cmd('p8 3')
'554889'
```

https://github.com/radare/radare2-r2pipe

1. http://beta.rada.re/en/latest/scripting.html

r2pipe-api?

High level API on top of r2pipe

Main goals:

- Easy to {use,learn}
- Clean interface to expose r2 **functionality**
- Automatize common and repetitive stuff
- Useful for scripting
- Be well documented
- Simple

https://github.com/radare/radare2-r2pipe-api

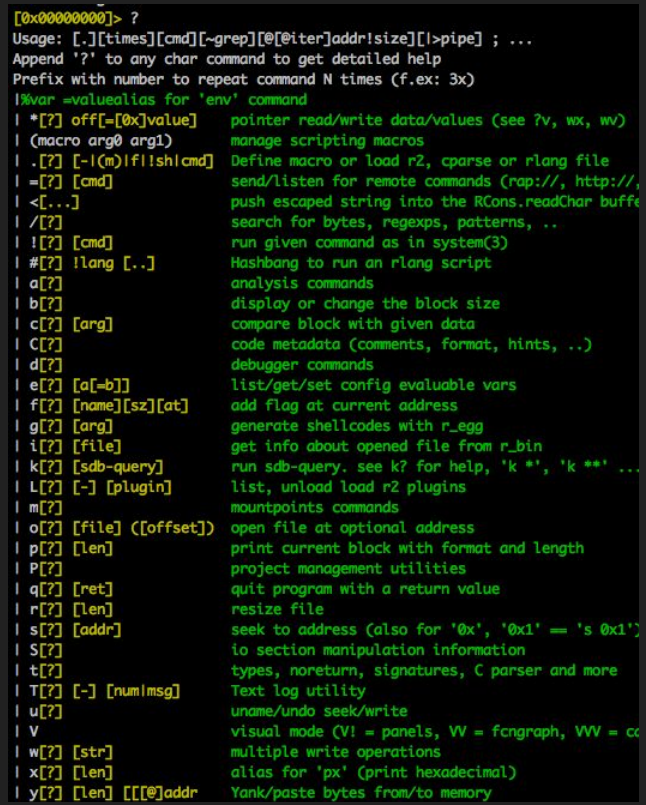
!r2pipe-api

Replace for r2pipe

r2pipe-api organization

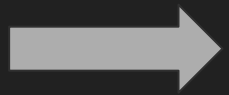
Different classes representing r2 functionality

- Debugger
- Esil
- Graph
- Write
- Files
- Analysis
- ...



r2pipe-api example

```
import r2pipe
r = r2pipe.open('bin_file')
r.cmd('doo')
r.cmd('db main')
r.cmd('dc')
reg = r.cmdj('drj')['rax']
r.cmd('dr rax=0x100')
```



```
from r2api import R2Api
r = R2Api('bin_file')
d = r.debugger
d.start()
d.at('main').setBreakpoint()
d.cont()
reg = d.cpu.rax
d.cpu.rax = 0x100
```

Install

Functions

```
>>> f = r.functionByName('sym._test')[0]
>>> f.analyze()
>>> print(f.info())
...
>>> f.name
'sym._test'
>>> f.name = 'sym.foo'
>>> f.name
'sym.foo'
```

Files

```
>>> r.files
[<r2api.file.File object at 0x10f4cda90>]
>>> r.open('/bin/ls')
4
>>> r.files[0].filename
'hello'
>>> r.files[1].filename
'/bin/ls'
>>> r.files[1].size
38688
>>> r.files[1].offset
0
>>> r.files[1].iomaps
[<r2api.iomap.IOMap object at 0x10f4cd9e8>, ...]
```

ESIL

```
>>> r = R2Api('hello')
>>> r.esil.eval('2,3,+')
5
>>> r.esil.vm.init()
>>> print(r.esil.vm.cpu)
rax       0x00000000000000
rbx       0x00000000000000
...
rip       0x00000100000ed0
rbp       0x00000000178000
rflags    0x00000000000000
rsp       0x00000000178000
```

ESIL

```
>>> r.esil.vm.cpu.rax
0
>>> r.esil.vm.cpu.rax = 0xdeadbeef
>>> hex(r.esil.vm.cpu.rax)
'0xdeadbeef'
```

```
>>> curr_pc = r.esil.vm.cpu.rip
>>> r.esil.vm.untilAddr(curr_pc + 1).cont()
```

Print

```
>>> r.print.at('main').bytes(1)
b'U'
```

```
>>> r.print.at('main').disassemble(2)
[<r2api.base.Result object at 0x1081536d8>,
 <r2api.base.Result object at 0x108153860>]
>>> r.print.at('main').disassemble(2)[0].opcode
'push rbp'
```

```
>>> r.print.at('main').hexdump(3)
'554889'
```

Config

```
>>> r.config.asm.bits
32
>>> r.config.asm.bits = 64
>>> r.config.asm.bits
64
```

Write

```
>>> r.print.hexdump(1)
'55'
>>> r.write.hex('90')
''
>>> r.print.hexdump(1)
'90'
```

```
>>> r.write.string('hello world!')
>>> r.write.nop()
>>> r.write.bytes(b'\x90\x90\x90')
>>> r.write.assembly('nop')
```

Debugger

```
>>> r.debugger.start()
File dbg:///Users/quim/r2con/r2api/demo/files/hello  reopened in
read-write mode
= attach 22755 22755
>>> r.debugger.at('main').setBreakpoint()
>>> r.debugger.cont()
>>> hex(r.debugger.cpu.rip)
'0x100000ed0'
>>> r.debugger.cpu.rax = 0x100
```

Testing

halp

We need contributors!

Only knowledge required:

- Python
- r2

A good and easy way to get involved with r2 community

Just send a PR

Ping me or pancake on {telegram,irc}

halp (TODO)

- **Analysis stuff**
- **Improve debugger and ESIL support**
- Graph stuff
- Types
- Search
- r2pm
- ...
- **Documentation**
  - Sphinx (?)
- **Improve and extend testing**
- Logging

Questions?