R2PIPE

AUTOMATING BINARY ANALYSIS WITH RADARE

@NighterMan / #NNC5ED

DISCLAIMER

Habemus Curso @HabemusC... 3h
Radare2 No se utiliza en ninguna empresa multinacional seria ni lo admite como herramienta, se licencian las buenas Taller radare=no sirve

Habemus Curso @HabemusC... 5m
@javiespejo De acuerdo !!! pero si no te dejan usarla en empresas, que quieres que diga , que si y engañamos a todos? bueno ya veo q si..

Habemus Curso @HabemusC... 8m
@ivan_eguiguren Es q si no se licencia, para q lo vas a usar y pagar pasta? APRENDER SOLOS si es para 'CASA'@belky318 @javiespejo @trufae

Habemus Curso @HabemusC... 6m
@virtualminds_es Gilipoplleces? die UNA EMPRESA DE RENOMBRE Q USE RADARE2 , A VER, EL LISTO !!! #AscoDeAmiguismos

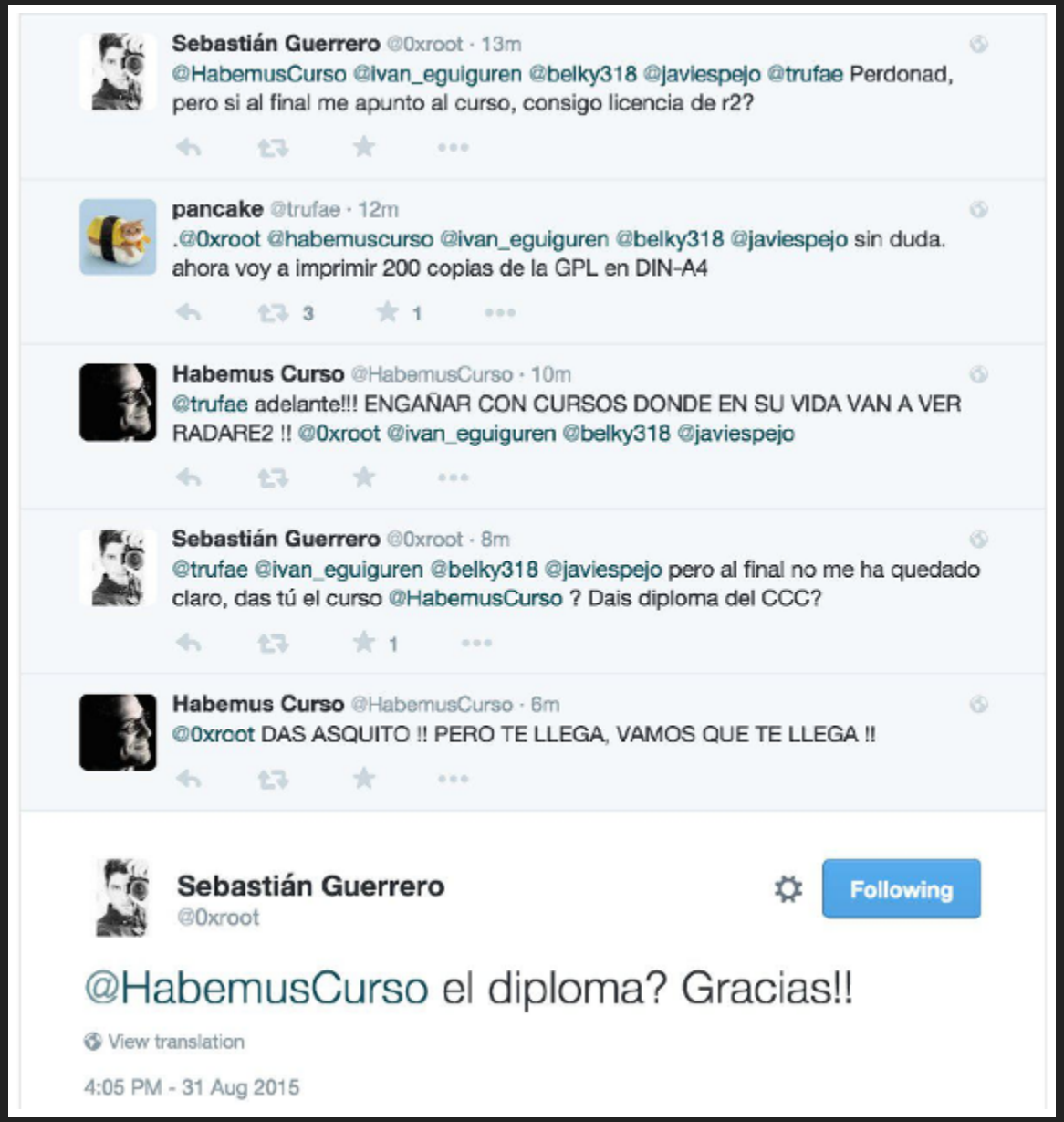
Habemus Curso @HabemusC... 33s
@0xroot CARA A CARA !! A VER CUAL ES TU REACCIÓN !! TU EMPRESA LICENCIA RADARE2 LISTO? @ivan_eguiguren @belky318 @javiespejo @trufae

Habemus Curso @HabemusC... 1m
@0xroot YA TE DIGO QUE EL MAS TONTO TIENE QUE SALIR.. QUE GANAS TENEMOS TODOS DE CAZARTE @ivan_eguiguren @belky318 @javiespejo @trufae

Habemus Curso @HabemusC... 1m
@trufae adelante!!! ENGAÑAR CON CURSOS DONDE EN SU VIDA VAN A VER RADARE2 !! @0xroot @ivan_eguiguren @belky318 @javiespejo

Habemus Curso @HabemusC... 2m
@trufae lA GENTE QUIERES TRABAJAR Y RADARE2 NO LES VA A YUDAR Y PUNTO, Q QUEREIS ENGAÑAR? @0xroot @ivan_eguiguren @belky318 @javiespejo

Habemus Curso @HabemusC... 3m
@trufae Anda , iros todos a LA MIERDA, así de fácil !! las cosas claras SIN ENGAÑAR @0xroot @ivan_eguiguren @belky318 @javiespejo



Sebastián Guerrero @0xroot · 13m
@HabemusCurso @ivan_eguiguren @belky318 @javiespejo @trufae Perdonad, pero si al final me apunto al curso, consigo licencia de r2?

pancake @trufae · 12m
.@0xroot @habemuscurso @ivan_eguiguren @belky318 @javiespejo sin duda. ahora voy a imprimir 200 copias de la GPL en DIN-A4
3 1

Habemus Curso @HabemusCurso · 10m
@trufae adelante!!! ENGAÑAR CON CURSOS DONDE EN SU VIDA VAN A VER RADARE2 !! @0xroot @ivan_eguiguren @belky318 @javiespejo

Sebastián Guerrero @0xroot · 8m
@trufae @ivan_eguiguren @belky318 @javiespejo pero al final no me ha quedado claro, das tú el curso @HabemusCurso ? Dais diploma del CCC?
1

Habemus Curso @HabemusCurso · 6m
@0xroot DAS ASQUITO !! PERO TE LLEGA, VAMOS QUE TE LLEGA !!

Sebastián Guerrero
@0xroot
Following

@HabemusCurso el diploma? Gracias!!

View translation

4:05 PM - 31 Aug 2015

¿QUÉ ES R2PIPE?

The r2pipe APIs are based on a single r2 primitive found behind r_core_cmd_str() which is a function that accepts a string parameter describing the r2 command to run and returns a string with the result.

https://github.com/radare/radare2-bindings/tree/master/r2pipe

¿QUÉ ES R2PIPE?

Ahora en cristiano

r2pipe es un API para diferentes lenguajes que nos permite enviar comandos a radare y obtener el resultado de la ejecución de estos de forma sencilla.

¿POR QUÉ R2PIPE?

- Idea sencilla pero funcional
- A todo el mundo le gusta el JSON
- Mejor rendimiento que bindings tipo ffi
- Fácil de usar. No es necesario conocer el API de radare
- Sencillo de mantener ya que no depende del API interno de r2 y no le afectan sus cambios
- Variedad de metodos de conexion (http, socket, rlangpipe, pipe)

PROGRESO DE R2PIPE

connect() & launch() con r2.cmd()

pipe()

r2.cmdj()

rlangpipe() / lpipe()

r2.syscmd() & r2.syscmdj()

~~promises~~

ioplugin

MAS PROGRESO DE R2PIPE

pipeSync() & lpipeSync()

open() & openSync()

cmd.esil.intr

cmd.esil.trap

DISPONIBLE PARA DIFERENTES LENGUAJES (MAYO 2015 - 9 LENGUAJES)

- node.js (@pancake / @NighterMan)
- Go (@nibble)
- ruby (@crowell)
- C# / .NET (@masterofjellyfish)
- python (@pancake)
- dlang (@pancake)
- perl (@pancake)
- java (@pancake)
- newlisp (@pancake)

DISPONIBLE PARA DIFERENTES LENGUAJES (OCTUBRE 2015 - 14 LENGUAJES)

	pipe	spawn	async	http	tcp	rap	json
nodejs	x	x	x	x	x	-	x
python	x	x	-	x	x	x	x
swift	x	x	x	x	-	-	x
dotnet	x	x	x	x	-	-	-
haskell	x	x	-	x	-	-	x
java	-	x	-	x	-	-	-
golang	x	x	-	-	-	-	x
ruby	x	x	-	-	-	-	x
rust	x	x	-	-	-	-	x
vala	-	x	x	-	-	-	-
erlang	x	x	-	-	-	-	-
newlisp	x	-	-	-	-	-	-
dlang	x	-	-	-	-	-	x
perl	x	-	-	-	-	-	-

¿COMO FUNCIONA R2PIPE?

LA MAYORIA DE LOS COMANDOS SOPORTAN SALIDA EN JSON AGREGANDO LA LETRA 'J' AL FINAL DE ESTE

```
[0x004048c5]> pd 1
;-- entry0:
0x004048c5    31ed           xor ebp, ebp
[0x004048c5]> pdj 1
[{"offset":4212933,"fcn_addr":0,"fcn_last":0,"size":2,"opcode":"xo
```

MULTIPLES METODOS DE CONEXIÓN

ASYNC	SYNC
connect()	pipeSync()
launch()	lpipeSync()
pipe()	
lpipe() / rlangpipe()	

CONNECT()

- Usa el servidor http de radare para enviar comandos
- Permite conectar con instancias de r2 remotas

Se levanta un servidor web de radare

```
$ r2 -c "=h 8080" /bin/ls
```

El API envia los comandos a:

http://host:8080/cmd/

http://cloud.radare.org/cmd/pdj%204

LAUNCH()

El API ejecuta radare y escucha comandos en un puerto tcp

```
[0x004048c5]> .?
|Usage: .[r2cmd] | [file] | [!command] | [(macro)] # define macro
| .:8080              listen for commands on given tcp port
| ./ ELF              interpret output of command /m ELF as r. comm
[0x004048c5]> .:8084
Listening for commands on port 8084
```

El API envia comandos al puerto a la escucha

```
$ nc -v 127.0.0.1 8084
localhost [127.0.0.1] 8084 (?) open
pdj 4
[{"offset":4212933,"fcn_addr":0,"fcn_last":0,"size":2,"opcode":"xo
```

PIPE()

- El API ejecuta r2 como un proceso hijo
- Se envian comandos y se lee el resultado por stdin/stdout
- Radare envia un 0x00 cuando esta listo para recibir comandos
- Se envia 0x00 al final de la respuesta de cada comando

```
$ echo pdj 1 | r2 -q0 /bin/ls | hexdump -C
00000000  00 5b 7b 22 6f 66 66 73  65 74 22 3a 34 32 31 32  |.[{"offset":4212|
00000010  39 33 33 2c 22 66 63 6e  5f 61 64 64 72 22 3a 30  |933,"fcn_addr":0|
00000020  2c 22 66 63 6e 5f 6c 61  73 74 22 3a 30 2c 22 73  |,"fcn_last":0,"s|
00000030  69 7a 65 22 3a 32 2c 22  6f 70 63 6f 64 65 22 3a  |ize":2,"opcode":|
00000040  22 78 6f 72 20 65 62 70  2c 20 65 62 70 22 2c 22  |"xor ebp, ebp","|
00000050  62 79 74 65 73 22 3a 22  33 31 65 64 22 2c 22 74  |bytes":"31ed","t|
00000060  79 70 65 22 3a 22 78 6f  72 22 2c 22 74 79 70 65  |ype":"xor","type|
00000070  5f 6e 75 6d 22 3a 32 38  2c 22 74 79 70 65 32 5f  |_num":28,"type2_|
00000080  6e 75 6d 22 3a 30 2c 22  66 6c 61 67 73 22 3a 5b  |num":0,"flags":[|
00000090  22 65 6e 74 72 79 30 22  5d 7d 5d 0a 0a 00        |"entry0"]}]....|
```

LPIPE() / RLANGPIPE()

- R2 se forkea y ejecuta el comando deseado
- Los scripts se ejecutan dentro de la misma instancia
- Pasa un par de pipes a traves de las variables de entorno
  - R2PIPE_IN
  - R2PIPE_OUT
- Se invoca a traves del comando #!pipe
  - #!pipe node script.js
  - #!pipe ls -al

```
[0x004048c5]> #!pipe env | grep R2PIPE
R2PIPE_OUT=13
R2PIPE_IN=10
```

LPIPE() / RLANGPIPE()

Script de ejemplo /tmp/test.js

```
console.log("El argumento es: " + process.argv[2]);
```

Ejecución

```
$ r2 /bin/ls
[0x004048c5]> #!pipe node /tmp/test.js "ojete moreno"
El argumento es: ojete moreno
[0x004048c5]>
```

Usando Alias

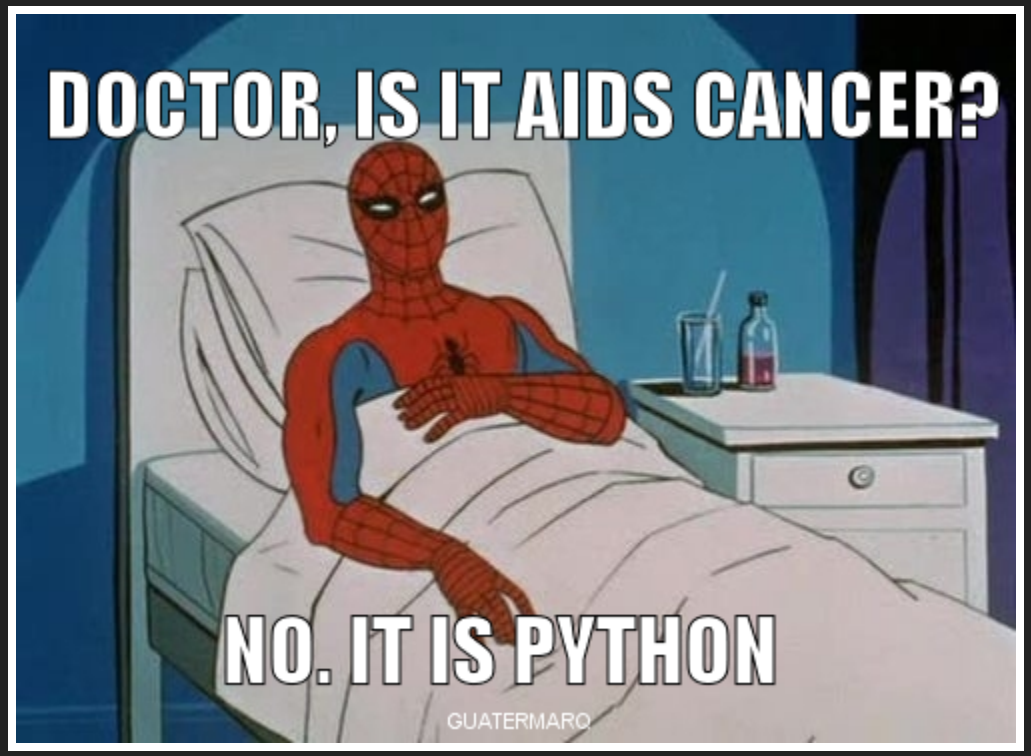
```
$ r2 /bin/ls
[0x004048c5]> $test=#!pipe node /tmp/test.js
[0x004048c5]> $test "ola k ase" otro
El argumento es: ola k ase
[0x004048c5]>
```

ALIAS PERSISTENTES PARA TUS SCRIPTS

~/.config/radare2/radare2rc

```
e hex.flagsz=999
e scr.wheel=false
e cfg.fortunes=false

$decompile=#!pipe node /home/jaime/security-CVS/r2-scripts/decompi
$syscall-resolver=#!pipe node /home/jaime/security-CVS/r2-scripts/
```

SOLO VAMOS A VER R2PIPE PARA NODE.JS



R2PIPE DEFINE LA API Y EL PROTOCOLO PARA COMUNICARSE CON R_CORE_CMD_STR()

- cmd/cmdj
- syscmd/syscmdj
- NULL terminated

HELLO WORLD!

```
import r2pipe

if __name__ == "__main__":
    # Test r2pipe with local process
    print("[+] Testing python r2pipe local")
    rlocal = r2pipe.open("/bin/ls")
    print(rlocal.cmd("pi 5"))
    #print rlocal.cmd("pn")
    info = rlocal.cmdj("ij")
    print ("Architecture: " + info['bin']['machine'])

    print("[+] Testing python r2pipe tcp://")
    rremote = r2pipe.open("tcp://127.0.0.1:9080")
    disas = rremote.cmd("pi 5")
    if not disas:
        print("Error with remote tcp conection")
```

INSTALACIÓN NODE.JS

```
$ npm install r2pipe
-
> syspipe@0.1.4 install /tmp/nodetmp/node_modules/r2pipe/node_modu
> node-gyp rebuild

make: Entering directory '/tmp/nodetmp/node_modules/r2pipe/node_mo
  CXX(target) Release/obj.target/syspipe/pipe.o
  SOLINK_MODULE(target) Release/obj.target/syspipe.node
  SOLINK_MODULE(target) Release/obj.target/syspipe.node: Finished
  COPY Release/syspipe.node
make: Leaving directory '/tmp/nodetmp/node_modules/r2pipe/node_mod
r2pipe@0.4.2 node_modules/r2pipe
└── syspipe@0.1.4 (nan@1.8.4)
```

Funciona con node 4.X, 0.12.X, 0.10.X e iojs

INSTALACIÓN PYTHON

```
$ sudo pip install r2pipe
Downloading/unpacking r2pipe
  Downloading r2pipe-0.6.5.tar.gz
  Running setup.py (path:/tmp/pip-build-0HggIZ/r2pipe/setup.py) eg

Installing collected packages: r2pipe
  Running setup.py install for r2pipe

Successfully installed r2pipe
Cleaning up...
```

R2PIPE ASYNC API

- connect
- launch
- pipe
- lpipe / rlangpipe

CONNECT (URL, CALLBACK)

```
var r2pipe = require('r2pipe');

r2pipe.connect('http://cloud.radare.org/cmd/', function (r2) {
  r2.cmdj('pdj 1', function (res) {
     console.log(res);
     r2.quit();
  });
});
```

Output

```
[ { offset: 4197583,
fcn_addr: 4197583,
fcn_last: 4197598,
size: 2,
opcode: 'byte [eax] += al',
bytes: '0000',
type: 'add',
type_num: 17,
type2_num: 0,
flags: [ 'fcn.00400ccf' ],
comment: 'c3Ryb2th' } ]
```

LAUNCH (FILE, CALLBACK)

```
var r2pipe = require('r2pipe');

r2pipe.launch('/bin/ls', function (r2) {
  r2.cmdj('pdj 1', function (res) {
     console.log(res);
     r2.quit();
  });
});
```

Output

```
[ { offset: 4212933,
fcn_addr: 0,
fcn_last: 0,
size: 2,
opcode: 'xor ebp, ebp',
bytes: '31ed',
type: 'xor',
type_num: 28,
type2_num: 0,
flags: [ 'entry0' ] } ]
```

PIPE (FILE, CALLBACK)

```
var r2pipe = require('r2pipe');

r2pipe.pipe('/bin/ls', function (r2) {
  r2.cmdj('pdj 1', function (res) {
     console.log(res);
     r2.quit();
  });
});
```

Output

```
[ { offset: 4212933,
    fcn_addr: 0,
    fcn_last: 0,
    size: 2,
    opcode: 'xor ebp, ebp',
    bytes: '31ed',
    type: 'xor',
    type_num: 28,
    type2_num: 0,
    flags: [ 'entry0' ] } ]
```

LPIPE (CALLBACK)

```
var r2pipe = require('r2pipe');

r2pipe.lpipe(function (r2) {
  r2.cmdj('pdj 1', function (res) {
     console.log(res);
     r2.quit();
  });
});
```

Output

```
[0x004048c5]> #!pipe node /tmp/nodetmp/test-connect.js
[ { offset: 4212933,
    fcn_addr: 0,
    fcn_last: 0,
    size: 2,
    opcode: 'xor ebp, ebp',
    bytes: '31ed',
    type: 'xor',
    type_num: 28,
    type2_num: 0,
    flags: [ 'entry0' ] } ]
```

R2 OBJECT

Los diferentes métodos de conexión devuelven un objeto r2 a traves del callback

```
{ cmd: [Function],
  cmdj: [Function],
  syscmd: [Function],
  syscmdj: [Function],
  quit: [Function],
  promise: [Function] }
```

CMD (CMD, CALLBACK)

Ejecuta un comando normal de r2

```
var r2pipe = require('r2pipe');

r2pipe.pipe(function (r2) {
  r2.cmd('pd 1', function (res) {
     console.log(res);
     r2.quit();
  });
});
```

CMDJ (CMD, CALLBACK)

Ejecuta un comando normal de r2 e intenta convertir el resultado a JSON

```
var r2pipe = require('r2pipe');

r2pipe.pipe(function (r2) {
  r2.cmdj('pdj 1', function (res) {
     console.log(res);
     r2.quit();
  });
});
```

SYSCMD (CMD, CALLBACK)

Ejecuta un comando del sistema

```
var r2pipe = require('r2pipe');

r2pipe.pipe(function (r2) {
  r2.syscmd('rabin2 -zz /bin/ls', function (res) {
     console.log(res);
     r2.quit();
  });
});
```

SYSCMDJ (CMD, CALLBACK)

Ejecuta un comando del sistema e intenta convertir el resultado a JSON

```
var r2pipe = require('r2pipe');

r2pipe.pipe(function (r2) {
  r2.syscmdj('rabin2 -j -zz /bin/ls', function (res) {
     console.log(res);
     r2.quit();
  });
});
```

QUIT ()

Termina la sesion de radare. Dependiendo del metodo de conexion, termina la conexion, cierra el proceso hijo o cierra los descriptores.

```
var r2pipe = require('r2pipe');

r2pipe.pipe(function (r2) {
    r2.quit();
});
```


THIS API HAS NO END
PELASE KILL ME
memecenter.com
MemeCenter

PROMISES (DEPRECATED)

- Implementación propia de promises
- Nos permite ejecutar multiples comandos de forma secuencial
- Ayuda a evitar generar código "christmas tree"
- No tiene control de errores por el momento
- Proporciona los siguientes metodos
  - promise (r2_function, cmd, [callback])
  - then (r2_function, cmd, [callback])
  - done (callback)

CÓDIGO SIN PROMISES

```
var r2pipe = require ("r2pipe");

function doSomeStuff(r2) {
    r2.cmd('aemi', function (res) {
        r2.cmd('aer esp=0x001f0000', function (res) {
            r2.cmd('aer eip=sym.decrypt_remotestr', function (res)
                r2.cmd('aecu 0x08049164', function (res) {
                    r2.cmd('ps @ ebx', function (res) {
                        r2.quit();
                    });
                });
            });
        });
    });
}
```

PROMISES

```
var r2pipe = require ("r2pipe");

function doSomeStuff(r2) {
  r2.promise(r2.cmd, 'aeim', null)
    .then(r2.cmd, 'aer esp=0x001f0000', null)
    .then(r2.cmd, 'aer eip=sym.decrypt_remotestr', null)
    .then(r2.cmd, 'aecu 0x08049164', null)
    .then(r2.cmd, 'ps @ ebx', function (res) {
      console.log("The decrypted result is: " + res);
    })
    .done(function () {
      r2.quit();
    });
}

r2pipe.pipe ("/tmp/m1wre/sample", doSomeStuff);
```


3
APM?
0:02

R2PIPE SYNC API

- pipeSync
- lpipeSync

PIPESYNC(CMD)

```
var r2pipe = require('r2pipe');

var r2 = r2pipe.pipeSync('/bin/ls');
var res = r2.cmdj('pdj 4');
r2.quit();
console.log(res);
```

LPIPESYNC()

```
var r2pipe = require('r2pipe');

var r2 = r2pipe.lpipeSync();
var res = r2.cmdj('pdj 4');
r2.quit();
console.log(res);
```

R2 OBJECT

Los diferentes métodos de conexión devuelven un objeto r2

```
var r2pipe = require('r2pipe');
var r2 = r2pipe.lpipeSync();
```

```
{ cmd: [Function],
  cmdj: [Function],
  syscmd: [Function],
  syscmdj: [Function],
  quit: [Function] }
```

FUNCIONES DE R2 OBJECT

Las funciones a son iguales que en API Async a excepcion de promises que no son necesarias. Los resultados son devueltos con return en vez de a traves de callback.

- var res = cmd(cmd)
- var res = cmdj(cmd)
- var res = syscmd(cmd)
- var res = syscmdj(cmd)
- quit()

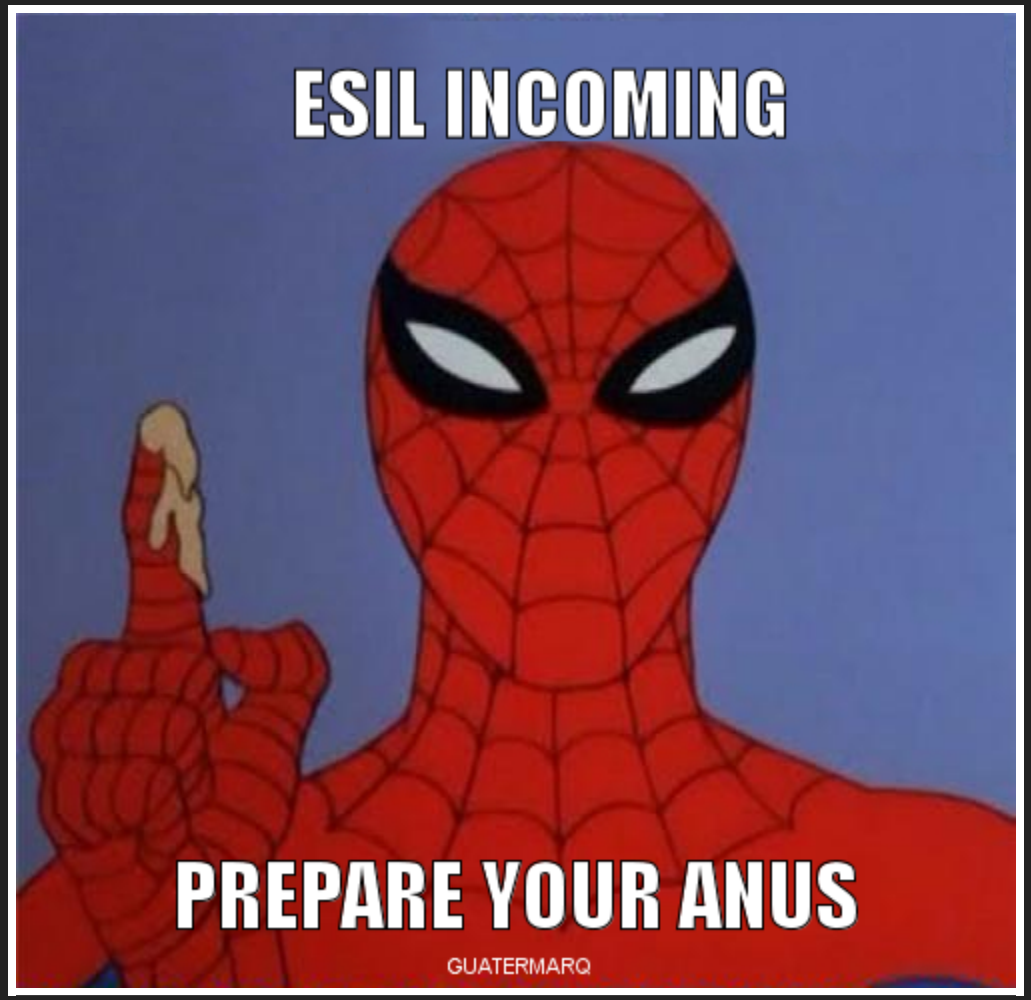
RLANG-IOPLUGIN

Ejecuta scripts de node, python, o cualquier lenguaje soportado por r2pipe para comunicarse con la uri pipe:// para ofrecer todas las funcionalidades de un plugin de io.

- open/read/write/close
- system (usando =!)

IOPLUGIN

```
var r2p = require ("r2pipe");
r2p.ioplugin(function(io, msg) {
  switch (msg.op) {
    case 'read':
      var obj = {
        result: msg.count,
        data: [1, 2, 3]
      };
      io.send(obj);
      break;
    case 'write':
      /* not implemented */
      console.error("not implemented")
      break;
    case 'system':
      io.send({
```


ESIL INCOMING
PREPARE YOUR ANUS
GUATERMARQ

ESIL

Esil is the language for the vm in r2

- Easy-to-parse-syntax
- You can hook almost everything debug hardware that you usually cannot debug
- Plugins may create customop for special instructions (des on avr for example)

@condret
http://runas-racer.com/foo/r2_solving_talk.pdf

LO MALO

- No soporta syscalls por el momento
- No soporta ejecutables enlazados dinamicamente
- Dificil de leer
- Shit, ermm... Bugs happens
- ESIL ruined my life

LO BUENO

- Facil de parsear
- Independiente de arquitectura
- No ejecuta el código (no hay riesgo de infección)
- Es opensource, así que puedes ayudar a mejorarlo

BUGS... YOU FIND IT, YOU FIX IT

ESIL BASIC OPS

op	esil	op	esil
mov	=	add	+
mul	*	sub	-
div	/	xor	^
and	&	or	|
neg	!	cmp	==
read	[]	write	=[]
if	?{		

@condret

http://runas-racer.com/foo/r2_solving_talk.pdf

ESIL INTERNAL VARS

- prefix is '%'
- calculated on the go
- ro-accessible
- provide information for flag-registers
- only affected by ops that end with '='
- only affected if the src is not an internal var

ESIL INTERNAL VARS

- carry from bit x → %cx
- borrow from bit x → %bx
- zero-flag → %z
- parity of dst → %p
- sign-flag → %s
- overflow-flag → %o

ESIL CONTROL FLOW

- BREAK = stop parsing and emulate next instruction
- LOOP = restart emulation of instruction
- GOTO n = jump to op N
- TODO = stop emulation and eprintf (“TODO %s“, ins)

@condret
http://runas-racer.com/foo/r2_solving_talk.pdf

ESIL BASIC OPS

X86	ESIL
mov eax,ebx	ebx,eax,=
jz 0xaabbccdd	zf,?{,0xaabbccdd,eip,=,}
cmp ecx,edx	edx,ecx,==,%z,zf,=,%b32,cf,=,%p,pf,=,%s,sf,=
push ebp	4,esp,-=,ebp,esp,=[4]

@condret

http://runas-racer.com/foo/r2_solving_talk.pdf

USING ESIL

You can access esil via analysis (a):

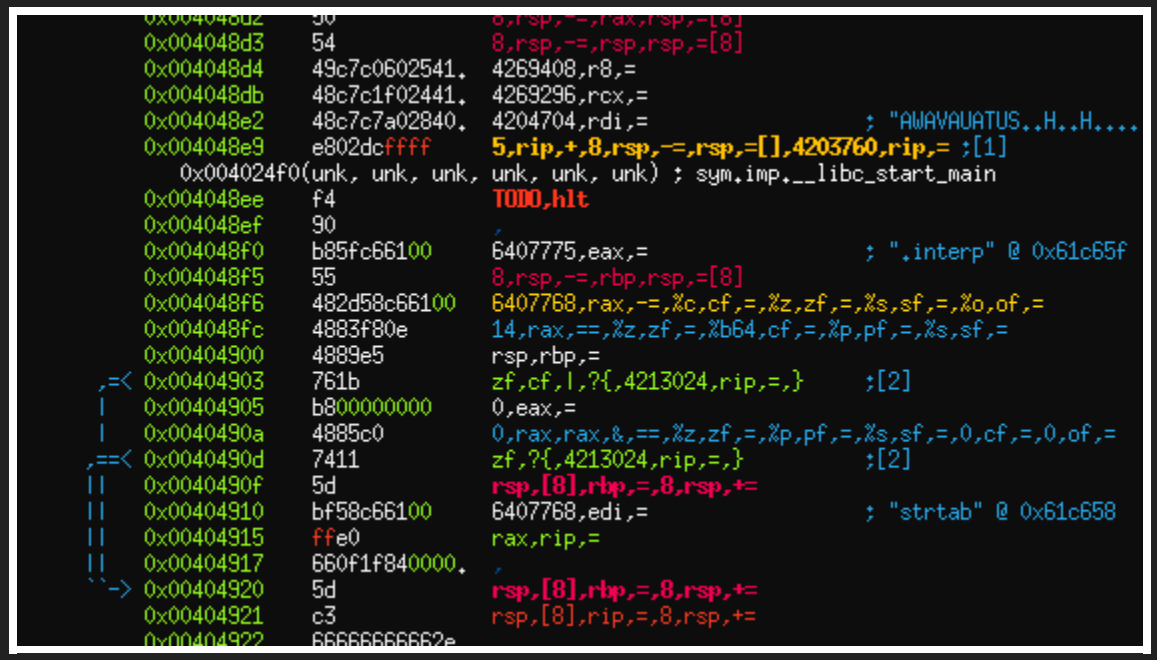
- aei = analysis esil initialize
- aeim = analysis esil initialize memory
- aer = analysis esil registers
- aes = analysis esil step
- aesu = analysis esil step until

For visual mode use O (captital "o")

ESIL DOC

https://github.com/radare/radare2book/blob/master/esil.md

ESIL DEMO


0x004048d3 54 8,rsp,-=,rsp,rsp,=[8]
0x004048d4 49c7c0602541. 4269408,r8,=
0x004048db 48c7c1f02441. 4269296,rcx,=
0x004048e2 48c7c7a02840. 4204704,rdi,= ; "AWAVAUATUS..H..H....
0x004048e9 e802dcffff 5,rip,+,8,rsp,-=,rsp,=[],4203760,rip,= ;[1]
0x004024f0(unk, unk, unk, unk, unk, unk) ; sym.imp.__libc_start_main
0x004048ee f4 TODO,hlt
0x004048ef 90
0x004048f0 b85fc66100 6407775,eax,= ; ".interp" @ 0x61c65f
0x004048f5 55 8,rsp,-=,rbp,rsp,=[8]
0x004048f6 482d58c66100 6407768,rax,-=,%c,cf,=,%z,zf,=,%s,sf,=,%o,of,=
0x004048fc 4883f80e 14,rax,==,%z,zf,=,%b64,cf,=,%p,pf,=,%s,sf,=
0x00404900 4889e5 rsp,rbp,=
0x00404903 761b zf,cf,|,?{,4213024,rip,=,} ;[2]
0x00404905 b800000000 0,eax,=
0x0040490a 4885c0 0,rax,rax,&,==,%z,zf,=,%p,pf,=,%s,sf,=,0,cf,=,0,of,=
0x0040490d 7411 zf,?{,4213024,rip,=,} ;[2]
0x0040490f 5d rsp,[8],rbp,=,8,rsp,+=
0x00404910 bf58c66100 6407768,edi,= ; "strtab" @ 0x61c658
0x00404915 ffe0 rax,rip,=
0x00404917 660f1f840000.
0x00404920 5d rsp,[8],rbp,=,8,rsp,+=
0x00404921 c3 rsp,[8],rip,=,8,rsp,+=

INTERRUPCIONES

&

TRAPS

OJO. NO ME REFIERO A ESTOS TRAPS

Para definir un script para manejar las interrupciones utilizamos la evar "cmd.esil.intr"

```
[0x004048c5]> e cmd.esil.intr=#!pipe node /tmp/myscript.js
```

Para definir un script para manejar los traps utilizamos la evar "cmd.esil.trap"

```
[0x004048c5]> e cmd.esil.trap=#!pipe node /tmp/myscript.js
```

INTERRUPCIONES

Nuestro script sera llamado con el numero de interrupcion como primer argumento

```
#!pipe node /tmp/myscript.js 0x80
```

De esta forma podemos por ejemplo manejar:

- 0x80: SYSCALLS
- 0x03: INT3 (breakpoint traps)

TRAPS

Nuestro script sera llamado con trap_type como primer argumento y trap_code como segundo argumento

```
#!pipe node /tmp/myscript.js 6 0x004048c5
```

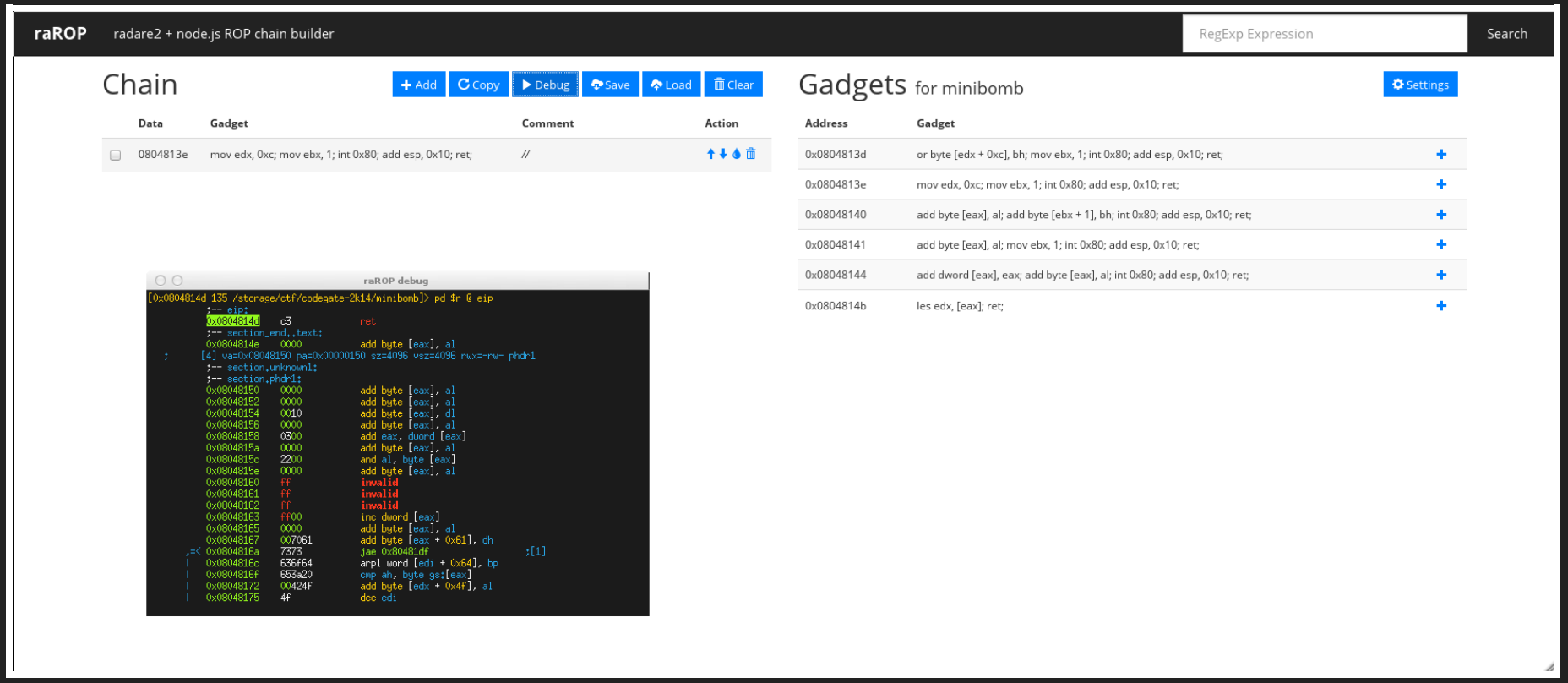
Actualmente estan definidos los siguientes traps

```
enum {
    R_ANAL_TRAP_NONE = 0,
    R_ANAL_TRAP_UNHANDLED = 1,
    R_ANAL_TRAP_BREAKPOINT = 2,
    R_ANAL_TRAP_DIVBYZERO = 3,
    R_ANAL_TRAP_WRITE_ERR = 4,
    R_ANAL_TRAP_READ_ERR = 5,
    R_ANAL_TRAP_EXEC_ERR = 6,
    R_ANAL_TRAP_TODO = 7,
    R_ANAL_TRAP_HALT = 8,
};
```

CASOS DE USO

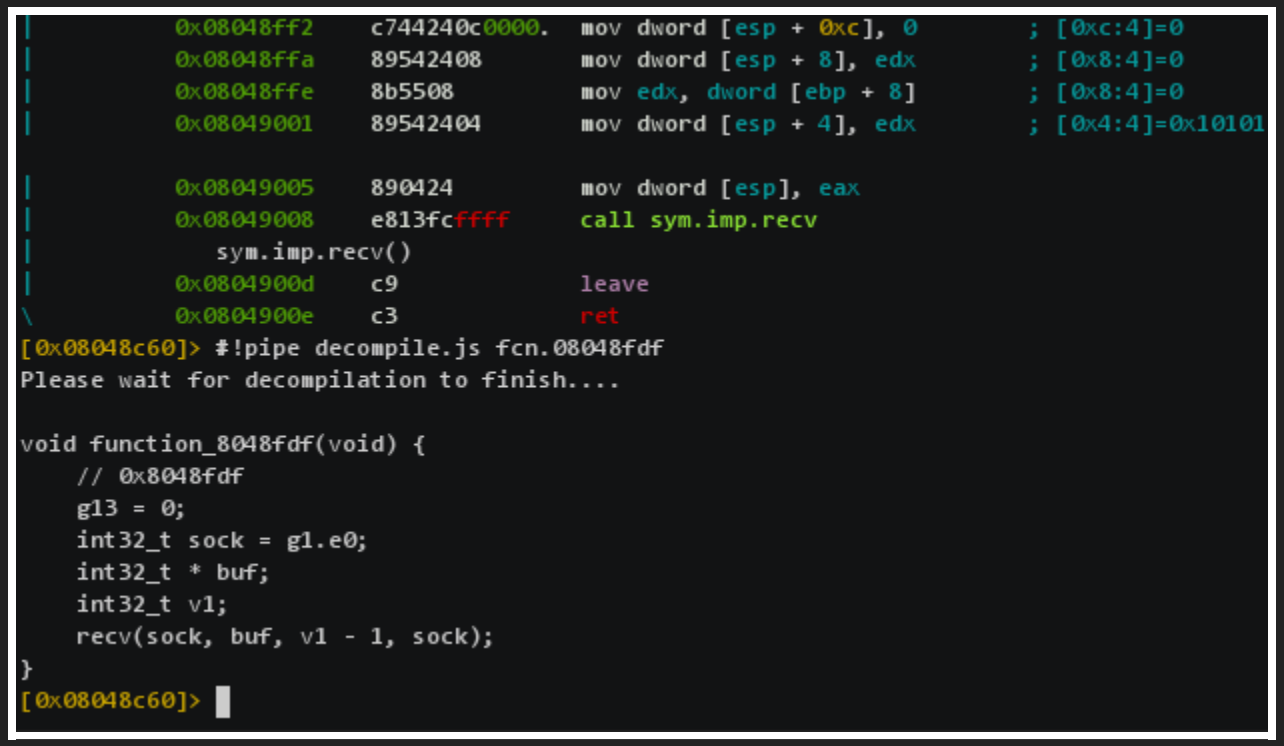
RAROP

Crea y debuggea ropchains de forma visual



RETDEC DECOMPILER

Decompiler usando el API REST de retdec.com

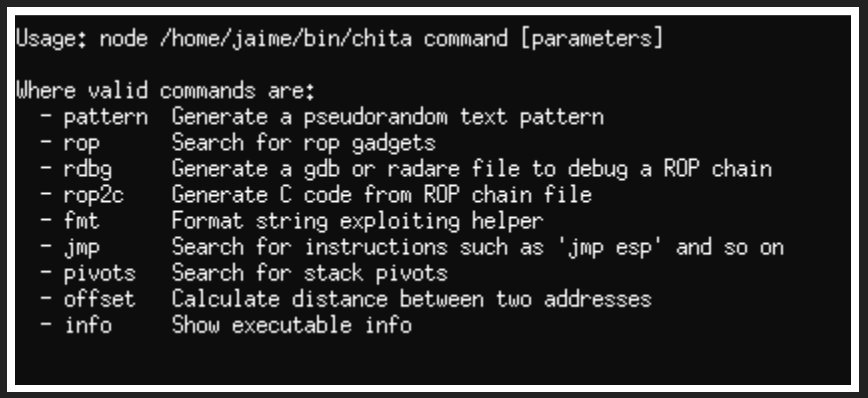


```
|          0x08048ff2    c744240c0000.   mov dword [esp + 0xc], 0        ; [0xc:4]=0
|          0x08048ffa    89542408        mov dword [esp + 8], edx        ; [0x8:4]=0
|          0x08048ffe    8b5508          mov edx, dword [ebp + 8]        ; [0x8:4]=0
|          0x08049001    89542404        mov dword [esp + 4], edx        ; [0x4:4]=0x10101

|          0x08049005    890424          mov dword [esp], eax
|          0x08049008    e813fcffff      call sym.imp.recv
|             sym.imp.recv()
|          0x0804900d    c9              leave
\          0x0804900e    c3              ret
[0x08048c60]> #!pipe decompile.js fcn.08048fdf
Please wait for decompilation to finish....

void function_8048fdf(void) {
    // 0x8048fdf
    g13 = 0;
    int32_t sock = g1.e0;
    int32_t * buf;
    int32_t v1;
    recv(sock, buf, v1 - 1, sock);
}
[0x08048c60]>
```

https://github.com/jpenalbae/r2-scripts

CHITA.JS

x86 Exploiting helper



https://github.com/jpenalbae/chita

SYSCALL RESOLVER

Resolución de nombres de syscalls

https://github.com/jpenalbae/r2-scripts

SPIKE & XOR DDOS DEOBFUSCATOR

Desofuscacion de strings en malware spike

```
  - ls
  - ls -la
  - top
  - netstat -an
  - netstat -antop
  - grep "A"
  - sleep 1
  - cd /etc
  - echo "find"
  - ifconfig eth0
  - ifconfig
  - route -n
  - gnome-terminal
  - id
  - who
  - whoami
  - pwd
  - uptime

[+] remotestr
  - gh.dsaj2a1.org:2857
  - navert0p.com:2857
  - wangzongfacai.com:2857


(21:44:06) [jaime@Bhola]
~/docs/presentaciones/mlwre-2015/deofuscate$
```

https://github.com/jpenalbae/r2-scripts

INDIRECT CALLS RESOLVER

TODO

Usando frida & r2pipe

https://github.com/jpenalbae/r2-scripts

DYNAMICALLY LINKED ELF RESOLVER

TODO

- Abrir los ficheros de las librerias en radare
- resolver las llamadas .got/.plt

PRACTICAS

SIMPLE ANAL

realizar un analisis simple de un binario, obteniendo strings, secciones, simbolos e imports de este.

KALLSYMS LOADER

Definir flags en una imagen de kernel stripeada a partir de /proc/kallsyms o un fichero System.map

SYSCALL HANDLER

Crear un pequeño script para emular la ejecucion de las syscalls en ESIL

MSFDECODER

Crear un decoder de payloads encodeados con x86/shikata_ga_nai de msfencode utilizando ESIL

DDOS-XOR-DEOBFUSCATOR

Utilizando ESIL, deobfuscar automaticamente las cadenas que contienen los hosts de los paneles de control del malware ddos-xor

¡¡¡ ADIOS BEBES !!!