MWR INFOSECURITY

How To Be An RSol: Effective Bug Hunting in Solaris

ShmooCon 2010

Matt Hillman

6th February 2010

So I've been working on 0wning Solaris

But then I got a bit distracted...

Who Am I?

Matt Hillman

Security Consultant at *MWR InfoSecurity*

http://labs.mwrinfosecurity.com/

Outline

1. How I Got Here
2. What Is RSol?
3. Demos!
4. Future Work

How I Got Here

- Fuzzing -> Sulley -> Process Monitor
- Core files, Procfs, DTrace
- Wait... DTrace?

DTrace Goodness

- Tracing at Wide Range of Levels
- Hit Probes are Queued
- Highly Specific Probe Conditions

DTrace Issues

- Designed to be Production Safe
- Probe Skips
- Effective Generic Solutions Less Easy
- We Can Do This Stuff Anyway...

What is RSol?

- A Library
- Programmatically Debug Solaris in Ruby
- Not an Interactive Debugger

RSol Architecture Overview

Core Classes

- RSol
- Procfs
- ElfUtils
- SparcInstructionType
- Signal
- Constant Classes

Native Extensions

- NativeBuffer
- DisasmSolaris
- LargeFile

RSol API Summary

attach(*to*)
run(*program*)
run_stopped(*program*)
detach
stop
continue
step
step_over
trace_entry(*set*)
trace_exit(*set*)
trace_fault(*set*)
trace_signal(*set*)
wait_for_event
read_memory(*address, length*)
write_memory(*address, buffer*)
set_register(*register, value*)
set_breakpoint(*address*)
unset_breakpoint(*address*)
disassemble(*instructions, start_address*)
malloc(*size*)
hook(*address, trampoline, handler*)
find_instruction_types(*data, flags*)

MWR LABS

RSol Fields

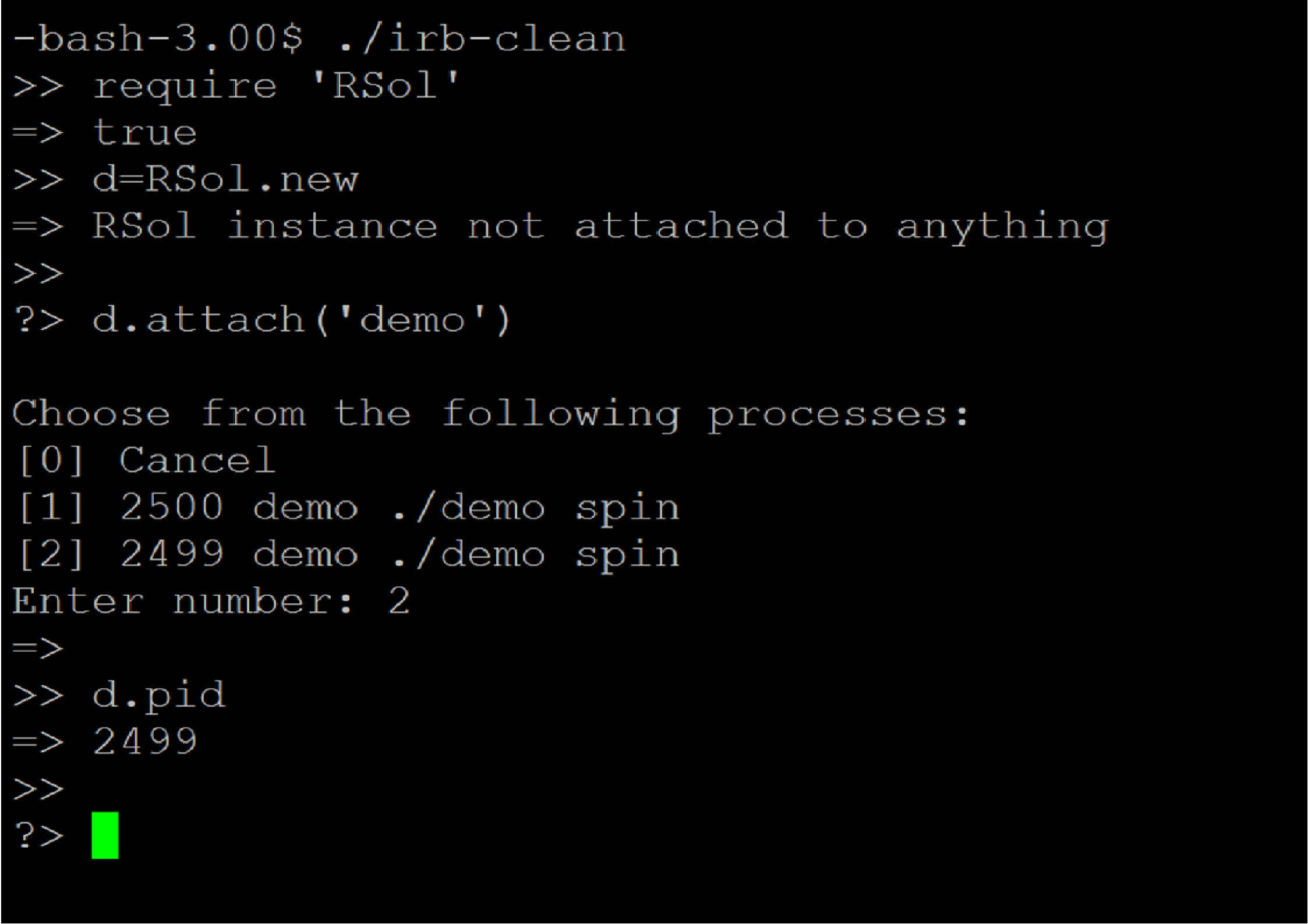
- .status.pr_lwp.pr_reg.pc

Quick Example

```
require 'RSol'
d=RSol.new
d.attach('my_program')
d.stop
print d
d.detach_and_continue
```

A shell is worth a thousand slides...

```
-bash-3.00$ ./irb-clean
>> require 'RSol'
=> true
>> d=RSol.new
=> RSol instance not attached to anything
>>
?>
```


-bash-3.00$ ./irb-clean
>> require 'RSol'
=> true
>> d=RSol.new
=> RSol instance not attached to anything
>>
?> d.attach('demo')
Choose from the following processes:
[0] Cancel
[1] 2500 demo ./demo spin
[2] 2499 demo ./demo spin
Enter number: 2
=>
>> d.pid
=> 2499
>>
?>

```
>> d.pid
=> 2499
>>
?> print d
Process Information
--------------------
Flags: PR_ASLEEP|PR_PCINVAL|PR_KLC|PR_MSACCT|PR_
MSFORK
Lwp count: 1
Pid: 2499
Parent pid: 2495
Process gid: 2499
Session ID: 2495
Agent ID: 0
Pending Signals: NONE
Heap address: 0x0002191C
Heap size: 0
Stack address: 0xFFBFE000
Stack size: 8192
```

```
============================
Representative lwpid 1 LWP Information
----------------
Flags: PR_ASLEEP|PR_PCINVAL|PR_KLC|PR_MSACCT|PR_
MSFORK
Lwp ID: 1
Why stopped: NOFLAGS
Current signal: NONE
Signals pending: NONE
Signals blocked: NONE
Action for current signal: [Handler: 0x00000000,
 SA Flags: NOFLAGS, SA Mask: NONE]
Alt stack: [Address: 0x00000000, Size: 0, Flags:
 2]
Prev ucontext: 0x00000000
Syscall number: SYS_nanosleep
Syscall arg count: 2
Syscall args: [0xFFBFFC00, 0xFFBFFBF8]
Syscall return value1: 0x00000000
```

```
============================
lwpid 1 No signal information
============================
lwpid 1 Registers
---------
g0: 0x00000000  g1: 0x00000000
g2: 0x00000000  g3: 0x00000000
g4: 0x00000000  g5: 0x00000001
g6: 0x00000000  g7: 0xFF373684
o0: 0xFFBFFC00  o1: 0xFFBFFBF8
o2: 0x00000040  o3: 0x0000002F
o4: 0xFF3773AA  o5: 0x00000000
o6: 0xFFBFFB98  o7: 0xFF337BE8
l0: 0xFF392A00  l1: 0x00000000
l2: 0x00000000  l3: 0x00000000
l4: 0x00000001  l5: 0x00000001
l6: 0x00000000  l7: 0xFF373684
i0: 0x00000001  i1: 0x000217D0
```

```
============================
lwpid 1 Stack Trace
------------
Register window:
l0: 0x00000002  l1: 0xFFBFFD3C
l2: 0xFFBFFD48  l3: 0x00021400
l4: 0xFF3900C0  l5: 0xFF390100
l6: 0xFFBFFCD8  l7: 0x00010AC8
i0: 0x00000000  i1: 0x000115C4
i2: 0x00000000  i3: 0x00000000
i4: 0xFF392A00  i5: 0xFF36E308
i6: 0xFF3746C0  i7: 0xFF3F4910
sp: 0xFFBFFCD8  Ret address: 0x00010AC8
Structure pointer: 0x00000000
Other data:
0x00000000  0x00000003
0xFFBFFD3C  0x00000004
0xFFBFFD48  0x00000005
0xFFBFFDA0  0x00000000
```

```
?>
?>
?>
?> d.set_breakpoint('main')
=> #<File:0x0000001007a8810>
>>
?> d.stop
=>
>> puts d.disassemble(4)
clr         %o0
cmp         %o0, 0x5b
move        %icc, 0x4, %o0
retl
=>
>> d.continue
=>
>>
?>
?>
```

```
?>
?> d.disassemble_function('main', '$address: ')
=> 0x00011278: ta          %icc, %g0 + 0x1
0x0001127C: cmp         %i0, 0x2
0x00011280: bge         +0x18           <main+0x20>
0x00011284: nop
0x00011288: call        -0x328          <overflow_me
>
0x0001128C: nop
0x00011290: ret
0x00011294: restore
0x00011298: sethi       %hi(0x11400), %l1
0x0001129C: or          %l1, 0x1c4, %l1
0x000112A0: ld          [%i1 + 0x4], %o1
0x000112A4: call        +0x10410        <strcmp>
0x000112A8: or          %l1, %g0, %o0
0x000112AC: cmp         %o0, 0x0
0x000112B0: bne         +0x18           <main+0x50>
0x000112B4: nop
```

```
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?> d.run_stopped('./demo')
=>
>>
```

```
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?> d.run_stopped('./demo')
=>
>> flt = Fltset.new
=> NONE
>>
?> flt.praddset!(ProcfsConst::FLTBOUNDS)
=> 32
>>
?>
```

```
?>
?>
?>
?>
?>
?>
?>
?>
?>
?> d.run_stopped('./demo')
=>
>> flt = Fltset.new
=> NONE
>>
?> flt.praddset!(ProcfsConst::FLTBOUNDS)
=> 32
>>
?> d.trace_fault(flt)
=>
>>
```

```
?>
?>
?>
?>
?> d.run_stopped('./demo')
=>
>> flt = Fltset.new
=> NONE
>>
?> flt.praddset!(ProcfsConst::FLTBOUNDS)
=> 32
>>
?> d.trace_fault(flt)
=>
>> d.continue
=>
>>

Enter data, but not too much, I am only little:
```

```
?> d.run_stopped('./demo')
=>
>> flt = Fltset.new
=> NONE
>>
?> flt.praddset!(ProcfsConst::FLTBOUNDS)
=> 32
>>
?> d.trace_fault(flt)
=>
>> d.continue
=>
>>

Enter data, but not too much, I am only little:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaa
```

```
>> flt = Fltset.new
=> NONE
>>
?> flt.praddset!(ProcfsConst::FLTBOUNDS)
=> 32
>>
?> d.trace_fault(flt)
=>
>> d.continue
=>
>>

Enter data, but not too much, I am only little:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa
```

```
>>
?> d.trace_fault(flt)
=>
>> d.continue
=>
>>

Enter data, but not too much, I am only little:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa

?> d.wait_for_event
=> true
>>
?>
```

```
?> d.wait_for_event
=> true
>>
?> print d
Process Information
--------------------
Flags: PR_STOPPED|PR_ISTOP|PR_KLC|PR_MSACCT|PR_M
SFORK
Lwp count: 1
Pid: 2517
Parent pid: 2516
Process gid: 2515
Session ID: 2477
Agent ID: 0
Pending Signals: NONE
Heap address: 0x0002191C
Heap size: 0
Stack address: 0xFFBFE000
```

```
Zombie count: 0
Zone ID: 0

============================
Representative lwpid 1 LWP Information
----------------
Flags: PR_STOPPED|PR_ISTOP|PR_KLC|PR_MSACCT|PR_M
SFORK
Lwp ID: 1
Why stopped: PR_FAULTED (FLTBOUNDS)
Current signal: NONE
Signals pending: NONE
Signals blocked: NONE
Action for current signal: [Handler: 0x00000000,
 SA Flags: NOFLAGS, SA Mask: NONE]
Alt stack: [Address: 0x00000000, Size: 0, Flags:
 2]
Prev ucontext: 0x00000000
Syscall number: SYS_syscall
```

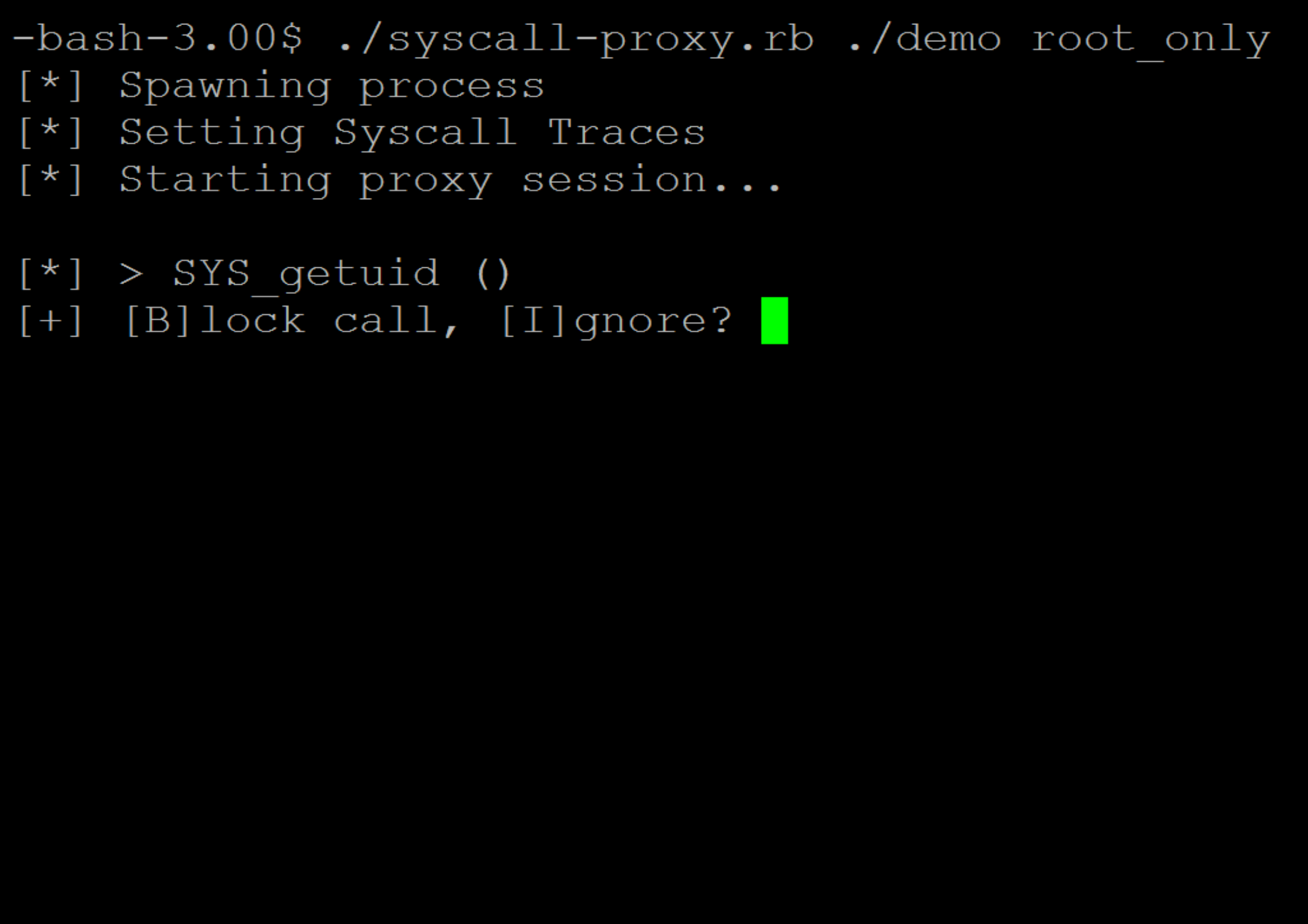
```
Lwp CPU time (system): [0, 81502249]
Missing Privilege: 0
Stack boundary: 0xFF392A88
Current instruction: ret

==============================
lwpid 1 Signal/Fault Information
--------------------
Signal: SIGSEGV
Signal code: SEGV_MAPERR
Signal errno: 0
Extra:
  addr: 0x00000000
  trapno: 0
  pc: 0x00000000

==============================
lwpid 1 Registers
---------
```

```
==========================
lwpid 1 Registers
---------
g0: 0x00000000  g1: 0x61616161
g2: 0x00000000  g3: 0x61616161
g4: 0x00000000  g5: 0x61616161
g6: 0x00000000  g7: 0x61616161
o0: 0x00000000  o1: 0x00000000
o2: 0x00000000  o3: 0xFF392A00
o4: 0x00000004  o5: 0x00000004
o6: 0xFFBFFC90  o7: 0x00011288
l0: 0x61616161  l1: 0x61616161
l2: 0x61616161  l3: 0x61616161
l4: 0x61616161  l5: 0x61616161
l6: 0x61616161  l7: 0x61616161
i0: 0x61616161  i1: 0x61616161
i2: 0x61616161  i3: 0x61616161
i4: 0x61616161  i5: 0x61616161
```

```
l6: 0x61616161  l7: 0x61616161
i0: 0x61616161  i1: 0x61616161
i2: 0x61616161  i3: 0x61616161
i4: 0x61616161  i5: 0x61616161
i6: 0x61616161  i7: 0x61616161
ccr:  0x00000044
pc:   0x00011290
npc:  0x00011294
y:    0x00000000
asi:  0x00000082
fprs: 0x00000000
Register Aliases:
ps: 0x00000044
sp: 0xFFBFFC90
fp: 0x61616161
r0: 0x00000000
r1: 0x00000000
return_address: 0x61616161
```

```
-bash-3.00$ python solaris_crashbin_explorer.py
 audits/srv1
[1] Crash at 0x000110E8 - SIGSEGV (SEGV_MAPERR)
addr: 0x2A100B3F2E1 trapno: 0 pc: 0x00000000 [re
t: 0x41414141, fp: 0x41414141, sp:0xFFBFFAF8]
        Test Case: 18

-bash-3.00$
```

```
-bash-3.00$ ./demo root_only


Your UID is: 100

Go away lamer!

-bash-3.00$
```


-bash-3.00$ ./syscall-proxy.rb ./demo root_only
[*] Spawning process
[*] Setting Syscall Traces
[*] Starting proxy session...
[*] > SYS_getuid ()
[+] [B]lock call, [I]gnore?

```
-bash-3.00$ ./syscall-proxy.rb ./demo root_only
[*] Spawning process
[*] Setting Syscall Traces
[*] Starting proxy session...

[*] > SYS_getuid ()
[+] [B]lock call, [I]gnore? i

[*] < SYS_getuid return1: 0x00000064 return2: 0x
00000064
[+] Modify return value(s) (y/N)?
```

```
-bash-3.00$ ./syscall-proxy.rb ./demo root_only
[*] Spawning process
[*] Setting Syscall Traces
[*] Starting proxy session...

[*] > SYS_getuid ()
[+] [B]lock call, [I]gnore? i

[*] < SYS_getuid return1: 0x00000064 return2: 0x
00000064
[+] Modify return value(s) (y/N)? y
[+] New return1: 0x0
[+] New return2: 0x0
```

```
-bash-3.00$ ./syscall-proxy.rb ./demo root_only
[*] Spawning process
[*] Setting Syscall Traces
[*] Starting proxy session...

[*] > SYS_getuid ()
[+] [B]lock call, [I]gnore? i

[*] < SYS_getuid return1: 0x00000064 return2: 0x
00000064
[+] Modify return value(s) (y/N)? y
[+] New return1: 0x0
[+] New return2: 0x0

[*] > SYS_open (0xFF3F7ED8, 0x00000000, 0x000000
00)
[+] [M]odify arguments, [B]lock call, [I]gnore?
```

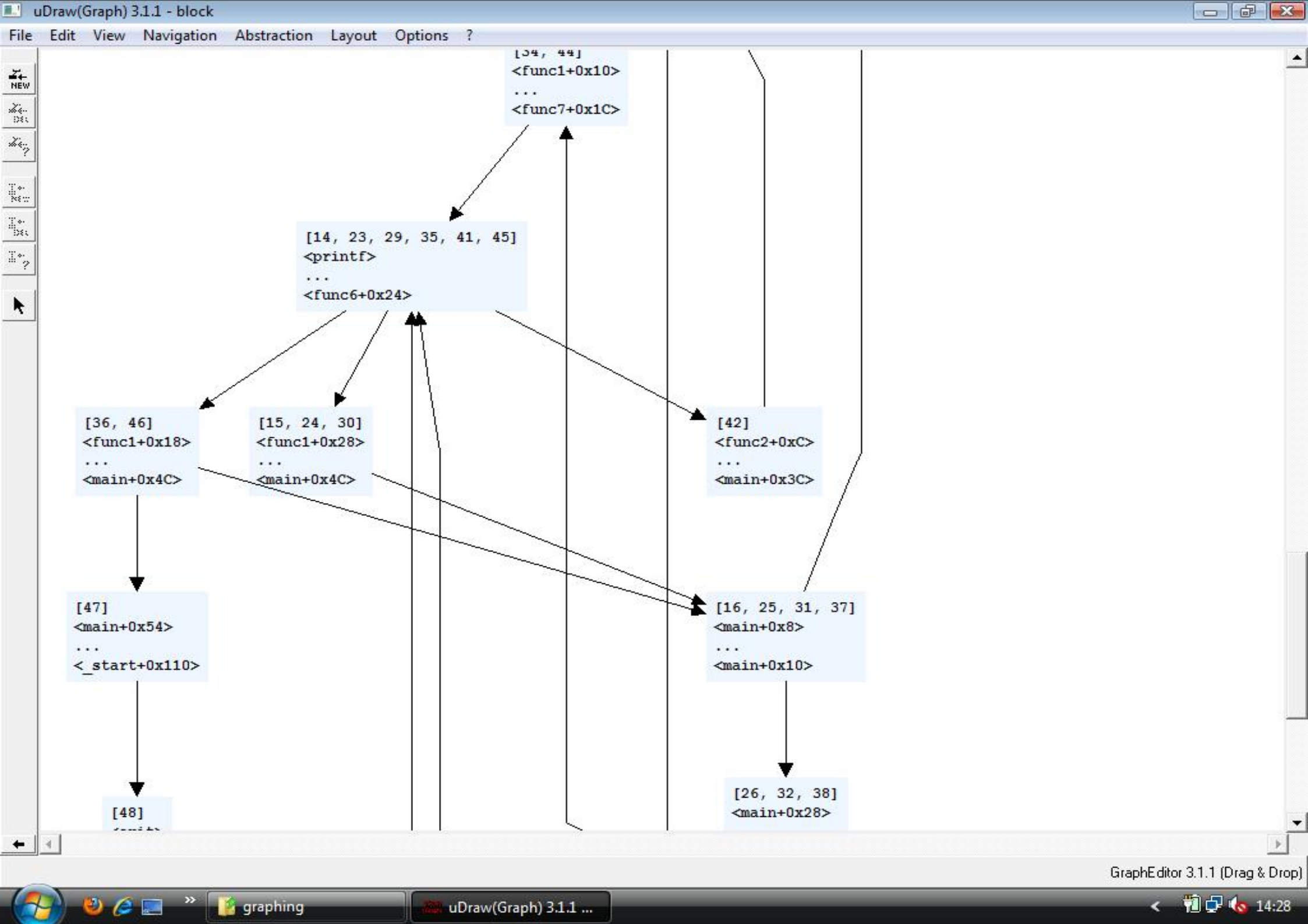
```
[+] New return1: 0x0
[+] New return2: 0x0

[*] > SYS_open (0xFF3F7ED8, 0x00000000, 0x00000000)
[+] [M]odify arguments, [B]lock call, [I]gnore?


[*] < SYS_open return1: 0x00000003 return2: 0x00000000
[+] Modify return value(s) (y/N)?


Your UID is: 0

If you were Chuck Norris you wouldn't need RSol.

[*] Process terminated
-bash-3.00$
```

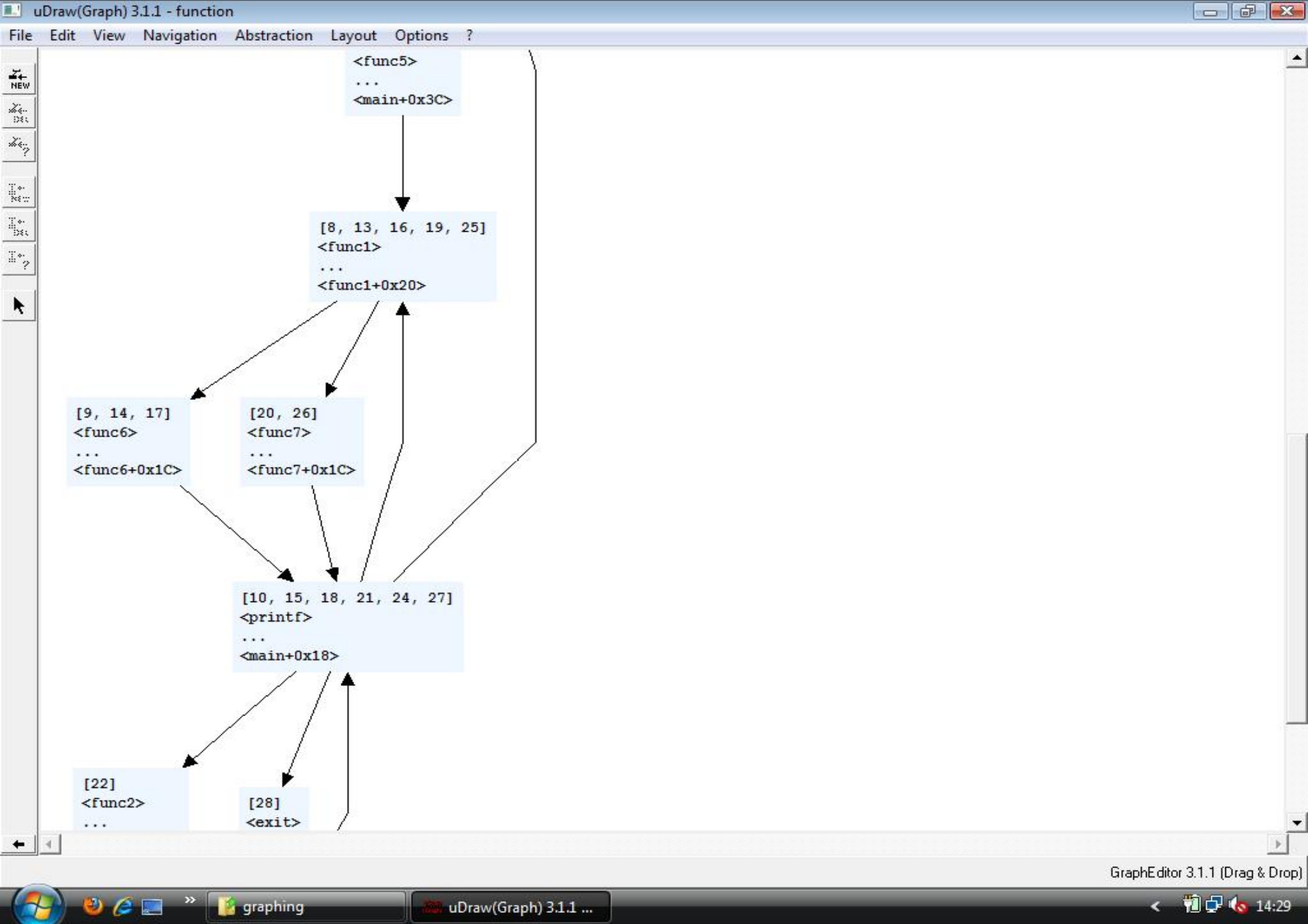
```
-bash-3.00$ ./trace.rb block b.txt ./demo_trace
bleep
bleep
bleep
9
blip
9
Program terminated
-bash-3.00$
```

```
-bash-3.00$ ./trace.rb function f.txt ./demo_tra
ce
bleep
bleep
bleep
9
blip
9
Program terminated
-bash-3.00$
```

```
0x00010740<_start> 0x0001077C<_start+0x3C>
0x00010788<_start+0x48> 0x00010794<_start+0x54>
0x000107DC<_start+0x9C> 0x000107E4<_start+0xA4>
0x000107EC<_start+0xAC> 0x000107F8<_start+0xB8>
0x00020EF8<atexit> 0x0001080C<_start+0xCC>
0x00010830<_start+0xF0> 0x00010848<_start+0x108>
0x00010E30<main> 0x00010E40<main+0x10>
0x00010E48<main+0x18> 0x00010D8C<func3+0x14>
0x00010D08<func5> 0x00010D10<func5+0x8>
0x00010D28<func5+0x20> 0x00010D94<func3+0x1C>
0x00010E50<main+0x20> 0x00010E6C<main+0x3C>
0x00010DE8<func1> 0x00010DF0<func1+0x8>
0x00010E08<func1+0x20> 0x00010CDC<func6+0x1C>
0x00020F1C<printf> 0x00010CE4<func6+0x24>
0x00010E10<func1+0x28> 0x00010E7C<main+0x4C>
0x00010E38<main+0x8> 0x00010E40<main+0x10>
0x00010E48<main+0x18> 0x00010D8C<func3+0x14>
0x00010D08<func5> 0x00010D10<func5+0x8>
b.txt
```

```
0x00010740<_start> 0x000107EC<_start+0xAC>
0x00020EF8<atexit> 0x000107F8<_start+0xB8>
0x00020EF8<atexit> 0x00010830<_start+0xF0>
0x00010E90<_init> 0x00010848<_start+0x108>
0x00010E30<main> 0x00010E48<main+0x18>
0x00010D78<func3> 0x00010D8C<func3+0x14>
0x00010D08<func5> 0x00010E6C<main+0x3C>
0x00010DE8<func1> 0x00010E08<func1+0x20>
0x00010CC0<func6> 0x00010CDC<func6+0x1C>
0x00020F1C<printf> 0x00010E48<main+0x18>
0x00010D78<func3> 0x00010D8C<func3+0x14>
0x00010D08<func5> 0x00010E6C<main+0x3C>
0x00010DE8<func1> 0x00010E08<func1+0x20>
0x00010CC0<func6> 0x00010CDC<func6+0x1C>
0x00020F1C<printf> 0x00010E6C<main+0x3C>
0x00010DE8<func1> 0x00010E08<func1+0x20>
0x00010CC0<func6> 0x00010CDC<func6+0x1C>
0x00020F1C<printf> 0x00010E6C<main+0x3C>
f.txt
```


uDraw(Graph) 3.1.1 - block
File Edit View Navigation Abstraction Layout Options ?
[34, 44]
<func1+0x10>
...
<func7+0x1C>
[14, 23, 29, 35, 41, 45]
<printf>
...
<func6+0x24>
[36, 46]
<func1+0x18>
...
<main+0x4C>
[15, 24, 30]
<func1+0x28>
...
<main+0x4C>
[42]
<func2+0xC>
...
<main+0x3C>
[47]
<main+0x54>
...
<_start+0x110>
[16, 25, 31, 37]
<main+0x8>
...
<main+0x10>
[48]
[26, 32, 38]
<main+0x28>
GraphEditor 3.1.1 (Drag & Drop)
graphing
uDraw(Graph) 3.1.1 ...
14:28


uDraw(Graph) 3.1.1 - function
File Edit View Navigation Abstraction Layout Options ?
<func5>
...
<main+0x3C>
[8, 13, 16, 19, 25]
<func1>
...
<func1+0x20>
[9, 14, 17]
<func6>
...
<func6+0x1C>
[20, 26]
<func7>
...
<func7+0x1C>
[10, 15, 18, 21, 24, 27]
<printf>
...
<main+0x18>
[22]
<func2>
...
[28]
<exit>
GraphEditor 3.1.1 (Drag & Drop)
graphing
uDraw(Graph) 3.1.1 ...
14:29

```
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?>
?> d.run_stopped('./demo')
=>
>> mem = d.malloc(100)
=> 137512
>>
?> d.address_format(mem)
=> 0x00021928
>>
?>
```

```
>>
?>
?>
?> d.write_memory(mem, "\x01\x00\x00\x00")
=> #<File:0x00000100946190>
>>
?> puts d.disassemble(3, mem)
nop
illtrap   0x0
illtrap   0x0
=>
>>
?>
?>
?>
?>
?>
?>
?>
?>
```

Debugger vs DTrace

Debugger

- Easier Generic Solutions
- Can Be Slow
- No Info Loss

DTrace

- Best For Specific Solutions
- Fast
- Can Lose Info

Combine For Much Win

Future Work

- DTrace in RSol
- Metrics and Fuller Comparisons
- Polished Working Tools
- Actual 0wnage...

Greetz and Propz

- VulnDev – thanks for the servers :D
- Squidly1 – project Sparc USA!
- LEH – why I’m here at all
- Schutzey – for putting up with me

Questions, Comments, Shmoo balls?