GO FORTH AND REVERSE

TIM "DIFF" STRAZZERE
07.25.2017
BSides Las Vegas

REDNAGA

WHO ARE WE

RED NAGA

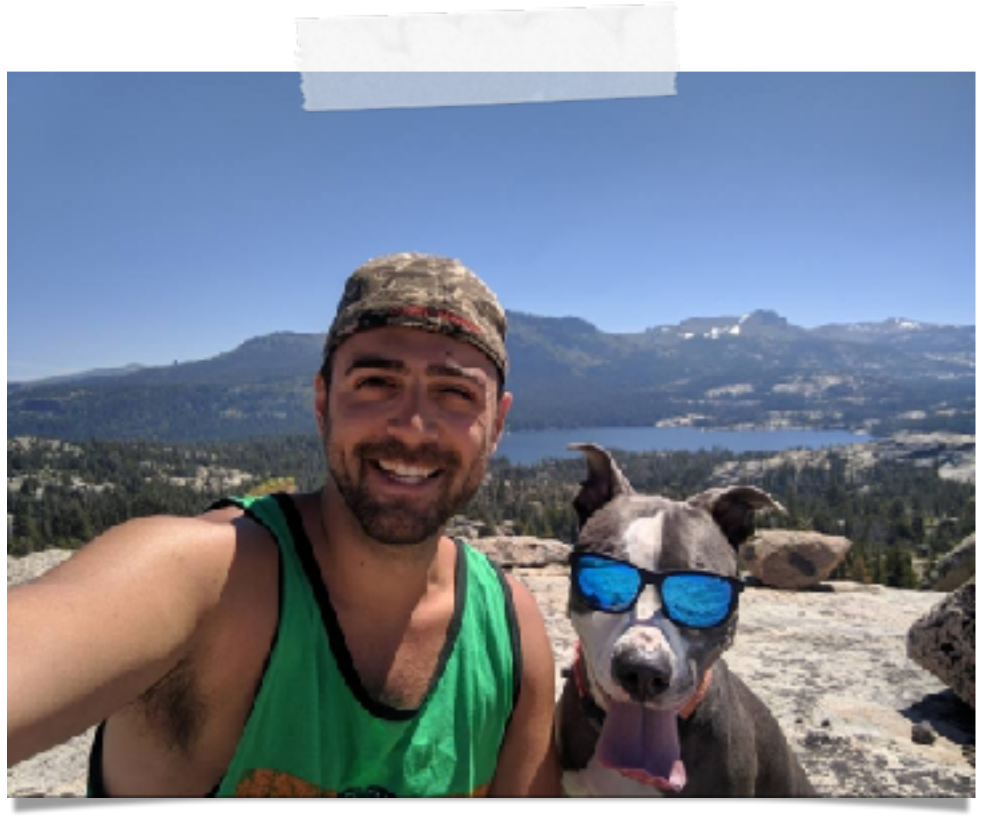
- Banded together by the love of 0days, fuzzing, making oem/bad guys lives harder, hot sauces
- Random out of work collaboration and pursuit of up-leveling the community
  - Disclosures / Code / Lessons available on GitHub
- rednaga.io
- github.com/RedNaga

WHOAMI

DIFF

- Security Engineer @ Cloudflare
- Previously
  Directory, Mobile Research @ SentinelOne
  Research & Response Engineer @ Lookout
- Obfuscation, Fuzzing and Packer Junkie
- Makes own hot sauce - cause why not?
- @timstrazz
- github.com/strazzere

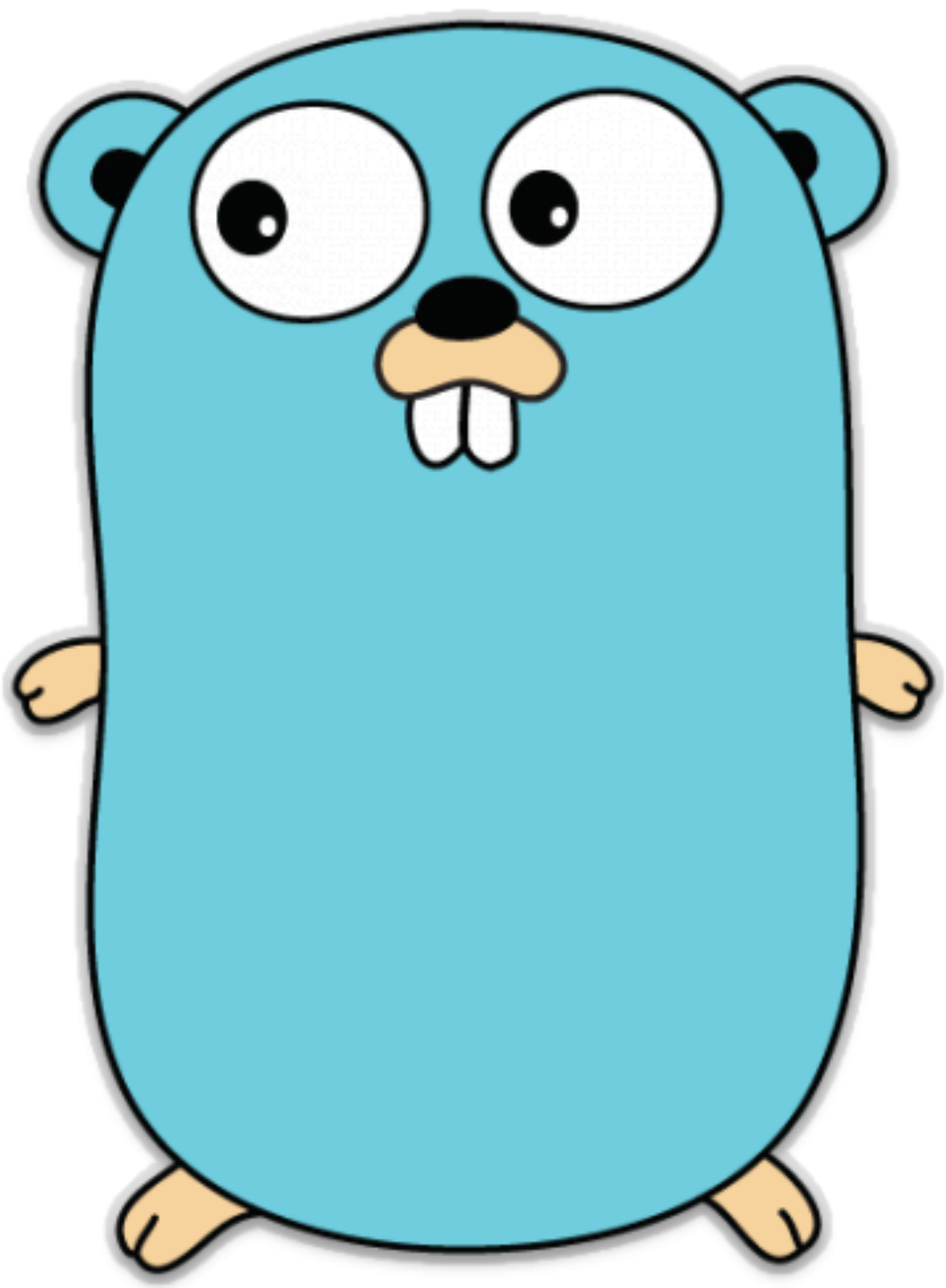
CLOUDFLARE



WHY AM I HERE

More importantly - why should you care?

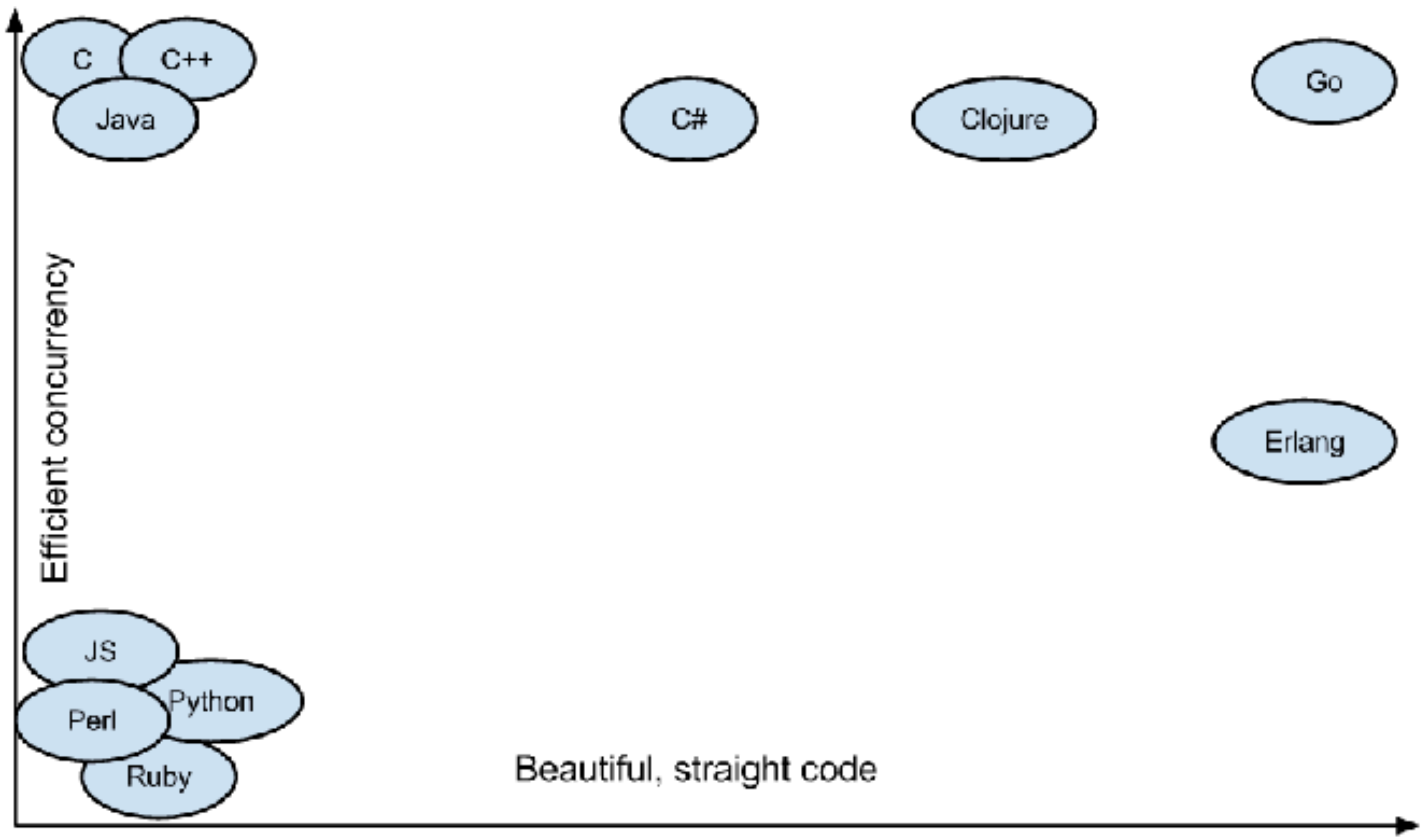
- How to approach a “new” language when reversing
- As malware and other binaries evolve, we must too
- If we break our tools now, we’re more prepared when we come across real life examples
- Building & expanding your reversing toolsets



WHY IS GO DIFFERENT?

It's an interesting beast…

- Gaining popularity… sort of similar concepts as Java with speed of C "Write once, run on any platform and is memory safe!"



- Easy, powerful coding setup which is often memory efficient and fast!
- Easy byte to byte reproducible builds (this helps us!)

Source: https://medium.com/@kevalpatel2106/why-should-you-learn-go-f607681fad65

WHY IS GO DIFFERENT?

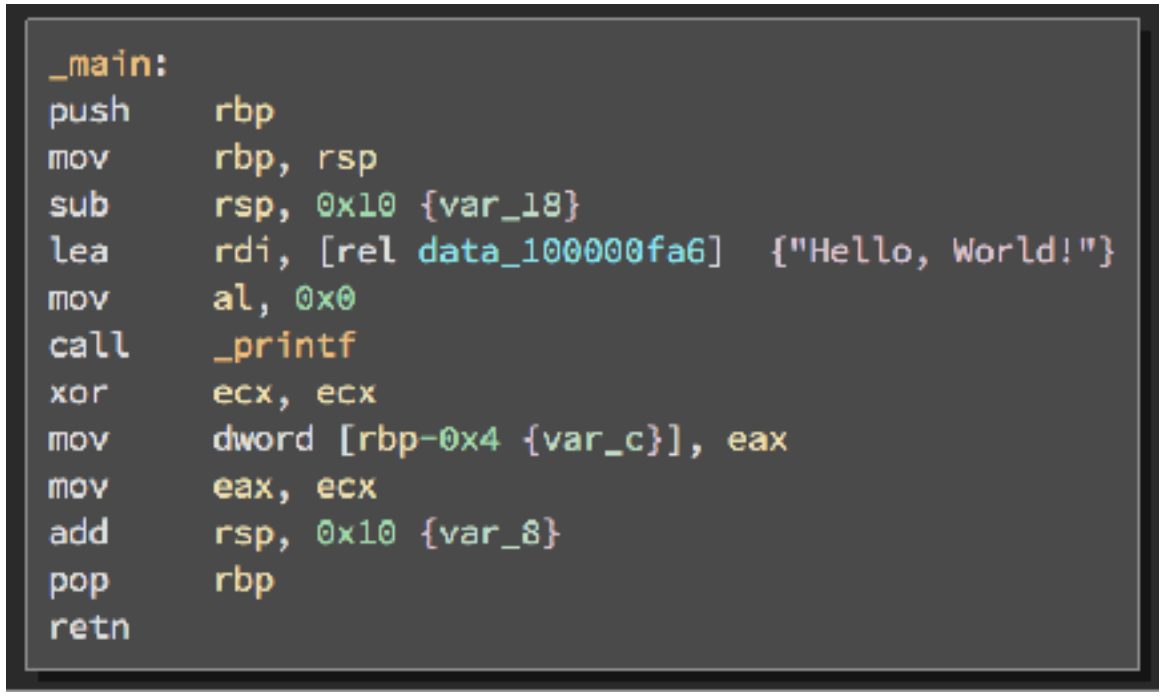
Yea but, I don't care for the sales pitch… I'm reversing…

```c
#include  <stdio.h>

int main() {
  printf("Hello, World!");
}
```

WHY IS GO DIFFERENT?

Yea but, I don't care for the sales pitch… I'm reversing…

```c
#include  <stdio.h>

int main() {
  printf("Hello, World!");
}
```



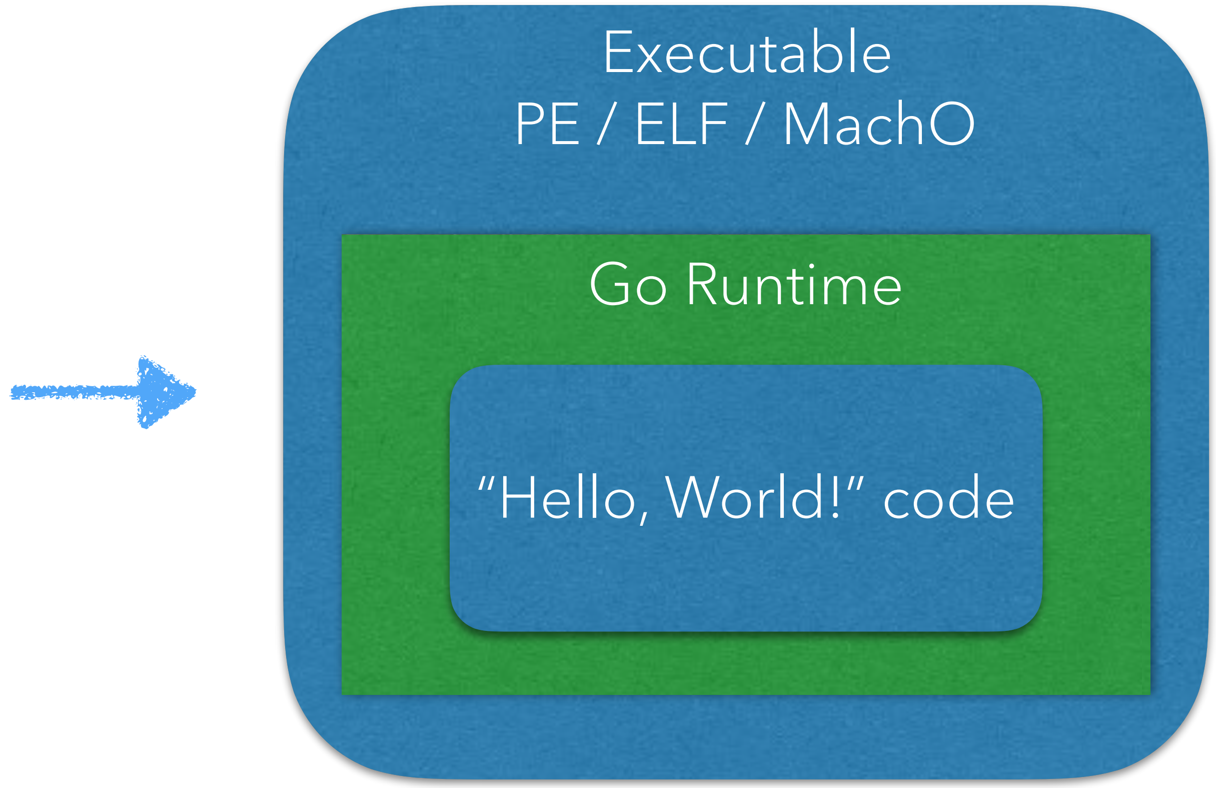
```
_main:
push     rbp
mov      rbp, rsp
sub      rsp, 0x10 {var_18}
lea      rdi, [rel data_100000fa6]  {"Hello, World!"}
mov      al, 0x0
call     _printf
xor      ecx, ecx
mov      dword [rbp-0x4 {var_c}], eax
mov      eax, ecx
add      rsp, 0x10 {var_8}
pop      rbp
retn
```

Very "low level" and minimal,
one function and an import (essentially)

WHY IS GO DIFFERENT?

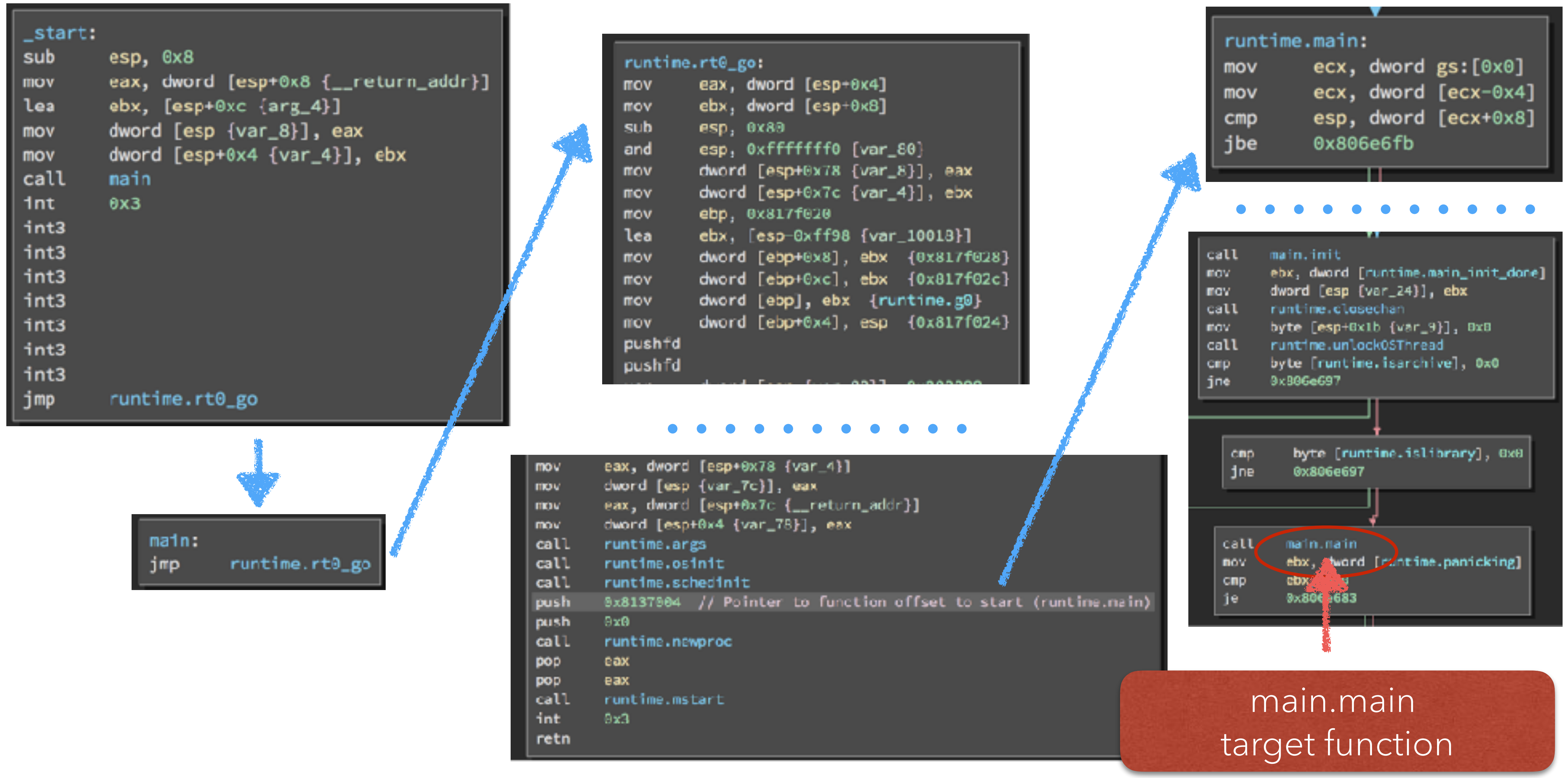
Yea but, I don't care for the sales pitch… I'm reversing…

```
package main

func main() {
  fmt.Println(“Hello, World!");
}
```



WHY IS GO DIFFERENT?

Yea but, I don't care for the sales pitch… I'm reversing…



```
_start:
sub     esp, 0x8
mov     eax, dword [esp+0x8 {__return_addr}]
lea     ebx, [esp+0xc {arg_4}]
mov     dword [esp {var_8}], eax
mov     dword [esp+0x4 {var_4}], ebx
call    main
int     0x3
int3
int3
int3
int3
int3
int3
int3
jmp     runtime.rt0_go
```

```
main:
jmp     runtime.rt0_go
```

```
runtime.rt0_go:
mov     eax, dword [esp+0x4]
mov     ebx, dword [esp+0x8]
sub     esp, 0x80
and     esp, 0xfffffff0 [var_80]
mov     dword [esp+0x78 {var_8}], eax
mov     dword [esp+0x7c {var_4}], ebx
mov     ebp, 0x817f020
lea     ebx, [esp-0xff98 {var_10018}]
mov     dword [ebp+0x8], ebx   {0x817f028}
mov     dword [ebp+0xc], ebx   {0x817f02c}
mov     dword [ebp], ebx   {runtime.g0}
mov     dword [ebp+0x4], esp   {0x817f024}
pushfd
pushfd
```

```
mov     eax, dword [esp+0x78 {var_4}]
mov     dword [esp {var_7c}], eax
mov     eax, dword [esp+0x7c {__return_addr}]
mov     dword [esp+0x4 {var_78}], eax
call    runtime.args
call    runtime.osinit
call    runtime.schedinit
push    0x8137904  // Pointer to function offset to start (runtime.main)
push    0x0
call    runtime.newproc
pop     eax
pop     eax
call    runtime.mstart
int     0x3
retn
```

```
runtime.main:
mov     ecx, dword gs:[0x0]
mov     ecx, dword [ecx-0x4]
cmp     esp, dword [ecx+0x8]
jbe     0x806e6fb
```

```
call    main.init
mov     ebx, dword [runtime.main_init_done]
mov     dword [esp {var_24}], ebx
call    runtime.closechan
mov     byte [esp+0x1b {var_9}], 0x0
call    runtime.unlockOSThread
cmp     byte [runtime.isarchive], 0x0
jne     0x806e697
```

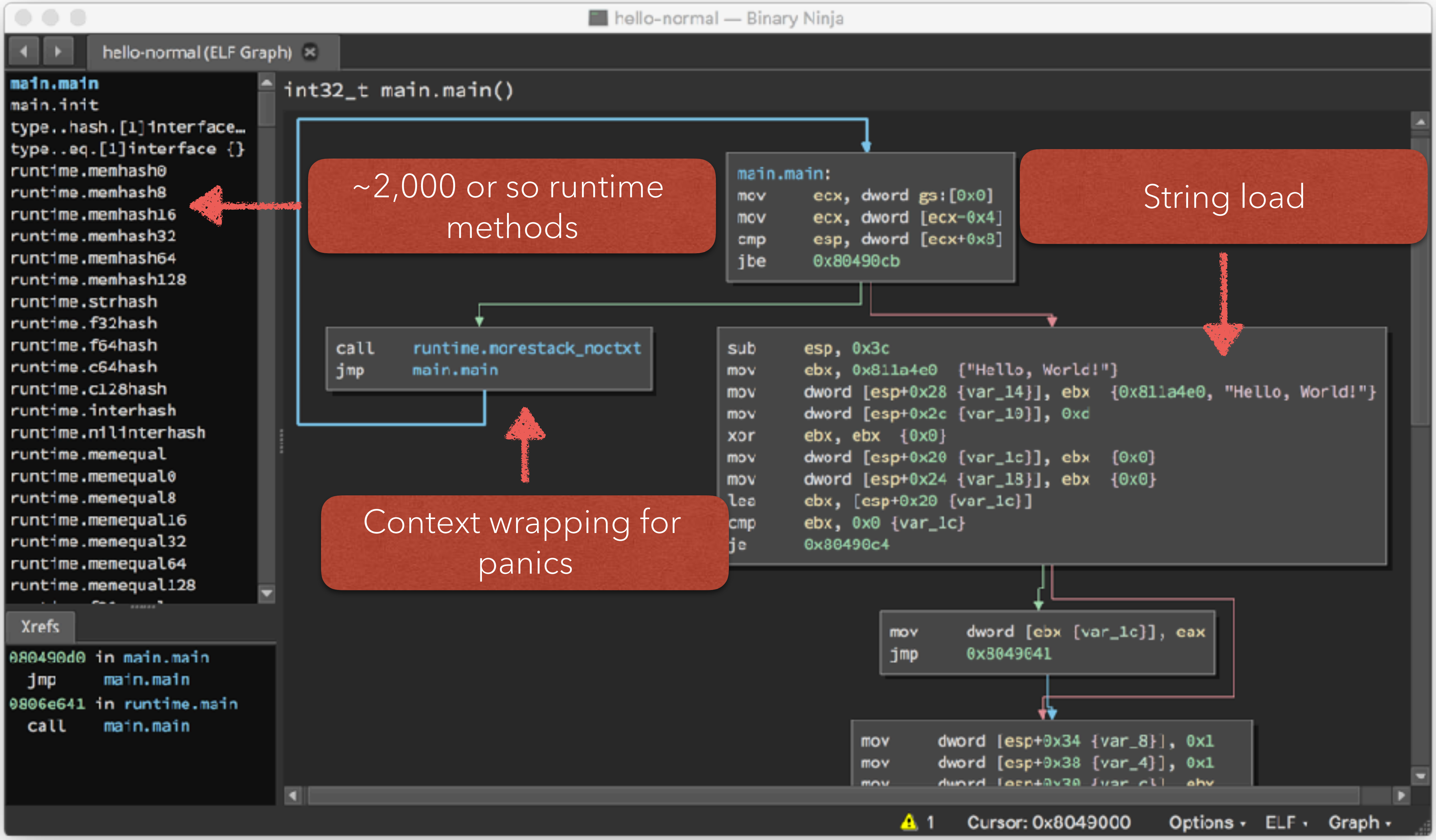
```
cmp     byte [runtime.islibrary], 0x0
jne     0x806e697
```

```
call    main.main
mov     ebx, dword [runtime.panicking]
cmp     ebx, [illegible]
je      0x806[illegible]683
```

_start -> main -> runtime.rt0_go -> runtime.newproc(*runtime.main) -> main.main

WHY IS GO DIFFERENT?

Yea but, I don't care for the sales pitch… I'm reversing…


hello-normal — Binary Ninja
hello-normal (ELF Graph)
~2,000 or so runtime methods
String load
Context wrapping for panics
Xrefs
Cursor: 0x8049000
Options
ELF
Graph

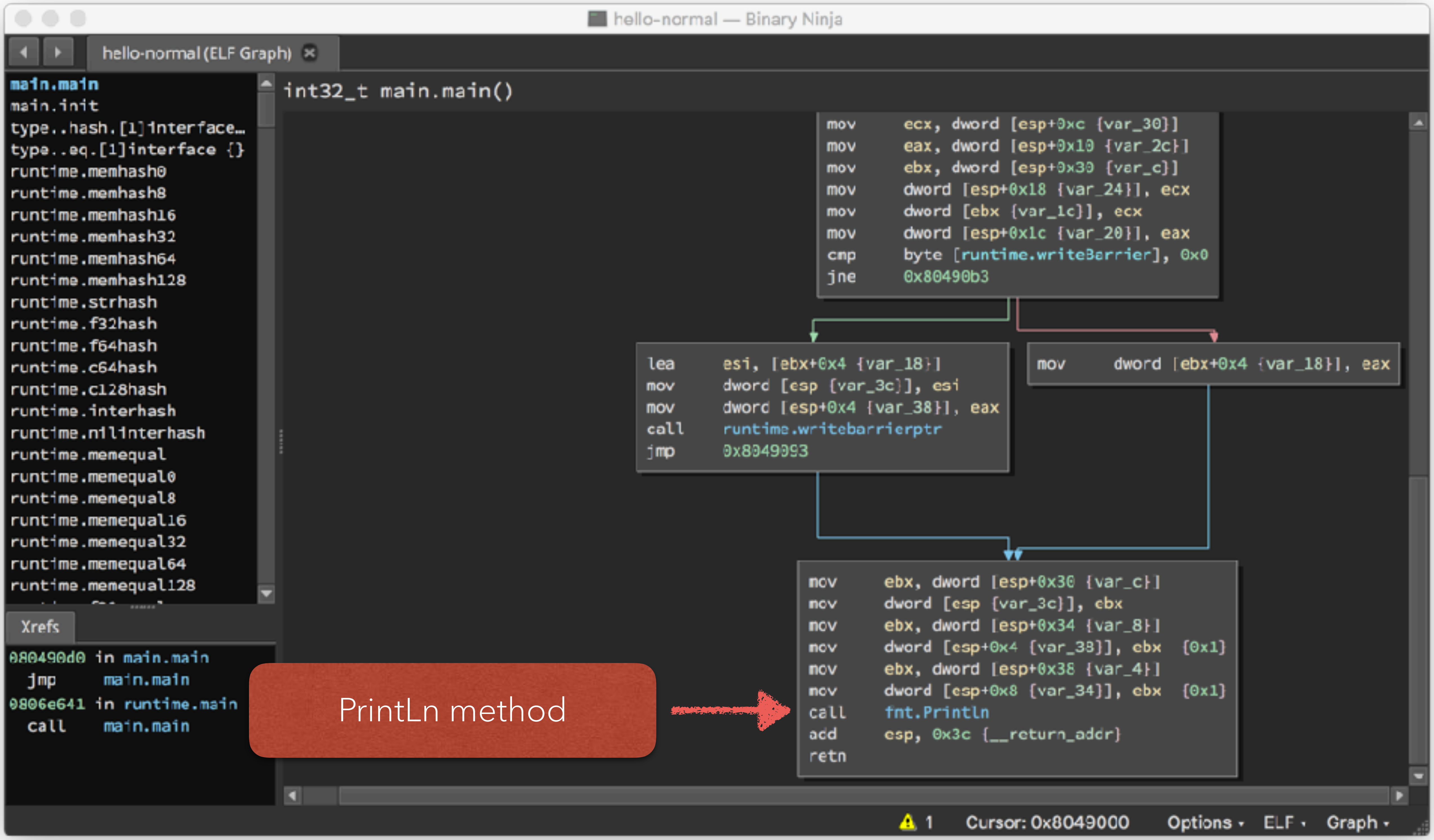
```
main.main
main.init
type..hash.[1]interface…
type..eq.[1]interface {}
runtime.memhash0
runtime.memhash8
runtime.memhash16
runtime.memhash32
runtime.memhash64
runtime.memhash128
runtime.strhash
runtime.f32hash
runtime.f64hash
runtime.c64hash
runtime.c128hash
runtime.interhash
runtime.nilinterhash
runtime.memequal
runtime.memequal0
runtime.memequal8
runtime.memequal16
runtime.memequal32
runtime.memequal64
runtime.memequal128
```

```
080490d0 in main.main
  jmp       main.main
0806e641 in runtime.main
  call      main.main
```

```
int32_t main.main()

main.main:
mov     ecx, dword gs:[0x0]
mov     ecx, dword [ecx-0x4]
cmp     esp, dword [ecx+0x8]
jbe     0x80490cb

call    runtime.morestack_noctxt
jmp     main.main

sub     esp, 0x3c
mov     ebx, 0x811a4e0  {"Hello, World!"}
mov     dword [esp+0x28 {var_14}], ebx  {0x811a4e0, "Hello, World!"}
mov     dword [esp+0x2c {var_10}], 0xd
xor     ebx, ebx  {0x0}
mov     dword [esp+0x20 {var_1c}], ebx  {0x0}
mov     dword [esp+0x24 {var_18}], ebx  {0x0}
lea     ebx, [esp+0x20 {var_1c}]
cmp     ebx, 0x0 {var_1c}
je      0x80490c4

mov     dword [ebx {var_1c}], eax
jmp     0x8049041

mov     dword [esp+0x34 {var_8}], 0x1
mov     dword [esp+0x38 {var_4}], 0x1
```

WHY IS GO DIFFERENT?

Yea but, I don't care for the sales pitch… I'm reversing…



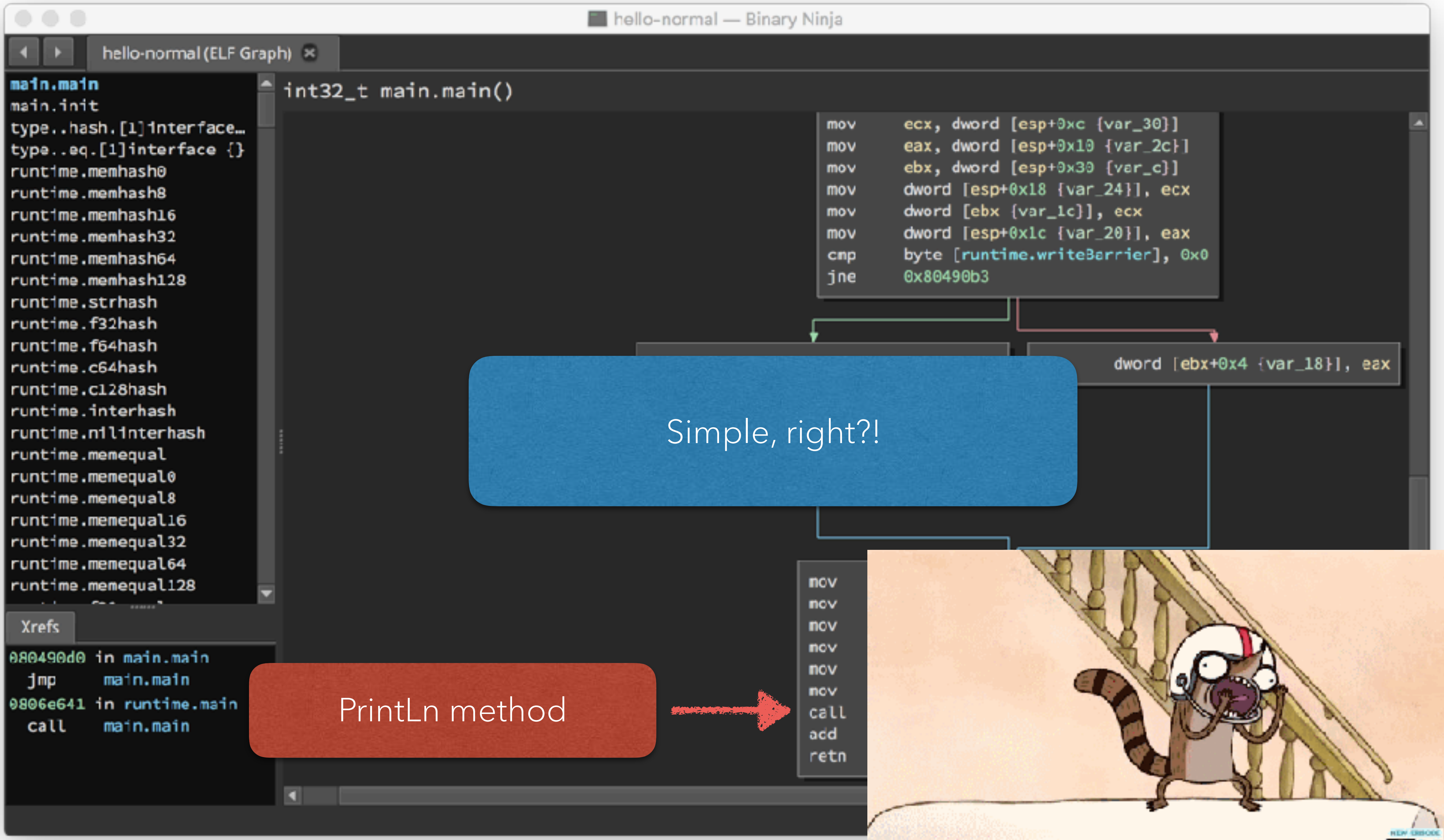
```
main.main
main.init
type..hash.[1]interface…
type..eq.[1]interface {}
runtime.memhash0
runtime.memhash8
runtime.memhash16
runtime.memhash32
runtime.memhash64
runtime.memhash128
runtime.strhash
runtime.f32hash
runtime.f64hash
runtime.c64hash
runtime.c128hash
runtime.interhash
runtime.nilinterhash
runtime.memequal
runtime.memequal0
runtime.memequal8
runtime.memequal16
runtime.memequal32
runtime.memequal64
runtime.memequal128
```

```
080490d0 in main.main
  jmp      main.main
0806e641 in runtime.main
  call     main.main
```

```
int32_t main.main()

mov     ecx, dword [esp+0xc {var_30}]
mov     eax, dword [esp+0x10 {var_2c}]
mov     ebx, dword [esp+0x30 {var_c}]
mov     dword [esp+0x18 {var_24}], ecx
mov     dword [ebx {var_1c}], ecx
mov     dword [esp+0x1c {var_20}], eax
cmp     byte [runtime.writeBarrier], 0x0
jne     0x80490b3

lea     esi, [ebx+0x4 {var_18}]
mov     dword [esp {var_3c}], esi
mov     dword [esp+0x4 {var_38}], eax
call    runtime.writebarrierptr
jmp     0x8049093

mov     dword [ebx+0x4 {var_18}], eax

mov     ebx, dword [esp+0x30 {var_c}]
mov     dword [esp {var_3c}], ebx
mov     ebx, dword [esp+0x34 {var_8}]
mov     dword [esp+0x4 {var_38}], ebx   {0x1}
mov     ebx, dword [esp+0x38 {var_4}]
mov     dword [esp+0x8 {var_34}], ebx   {0x1}
call    fmt.Println
add     esp, 0x3c {__return_addr}
retn
```

WHY IS GO DIFFERENT?

Yea but, I don't care for the sales pitch… I'm reversing…



```
main.main
main.init
type..hash.[1]interface…
type..eq.[1]interface {}
runtime.memhash0
runtime.memhash8
runtime.memhash16
runtime.memhash32
runtime.memhash64
runtime.memhash128
runtime.strhash
runtime.f32hash
runtime.f64hash
runtime.c64hash
runtime.c128hash
runtime.interhash
runtime.nilinterhash
runtime.memequal
runtime.memequal0
runtime.memequal8
runtime.memequal16
runtime.memequal32
runtime.memequal64
runtime.memequal128
```

```
080490d0 in main.main
  jmp      main.main
0806e641 in runtime.main
  call     main.main
```

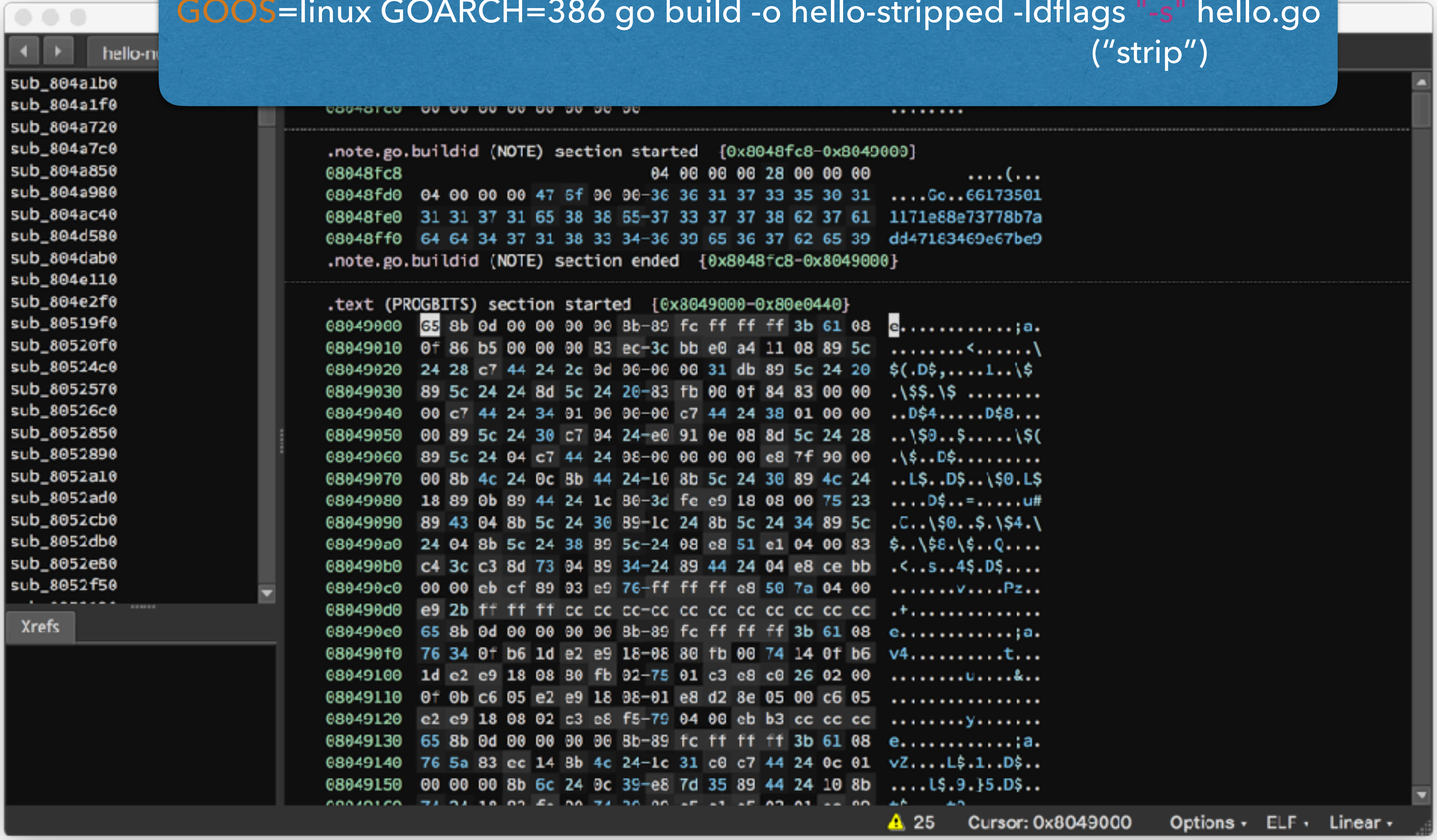
```
int32_t main.main()

mov     ecx, dword [esp+0xc {var_30}]
mov     eax, dword [esp+0x10 {var_2c}]
mov     ebx, dword [esp+0x30 {var_c}]
mov     dword [esp+0x18 {var_24}], ecx
mov     dword [ebx {var_1c}], ecx
mov     dword [esp+0x1c {var_20}], eax
cmp     byte [runtime.writeBarrier], 0x0
jne     0x80490b3

        dword [ebx+0x4 {var_18}], eax

mov
mov
mov
mov
mov
mov
call
add
retn
```

WHY IS GO DIFFERENT?

That was easy!

```
GOOS=linux GOARCH=386 go build -o hello-stripped -ldflags "-s" hello.go
```

("strip")



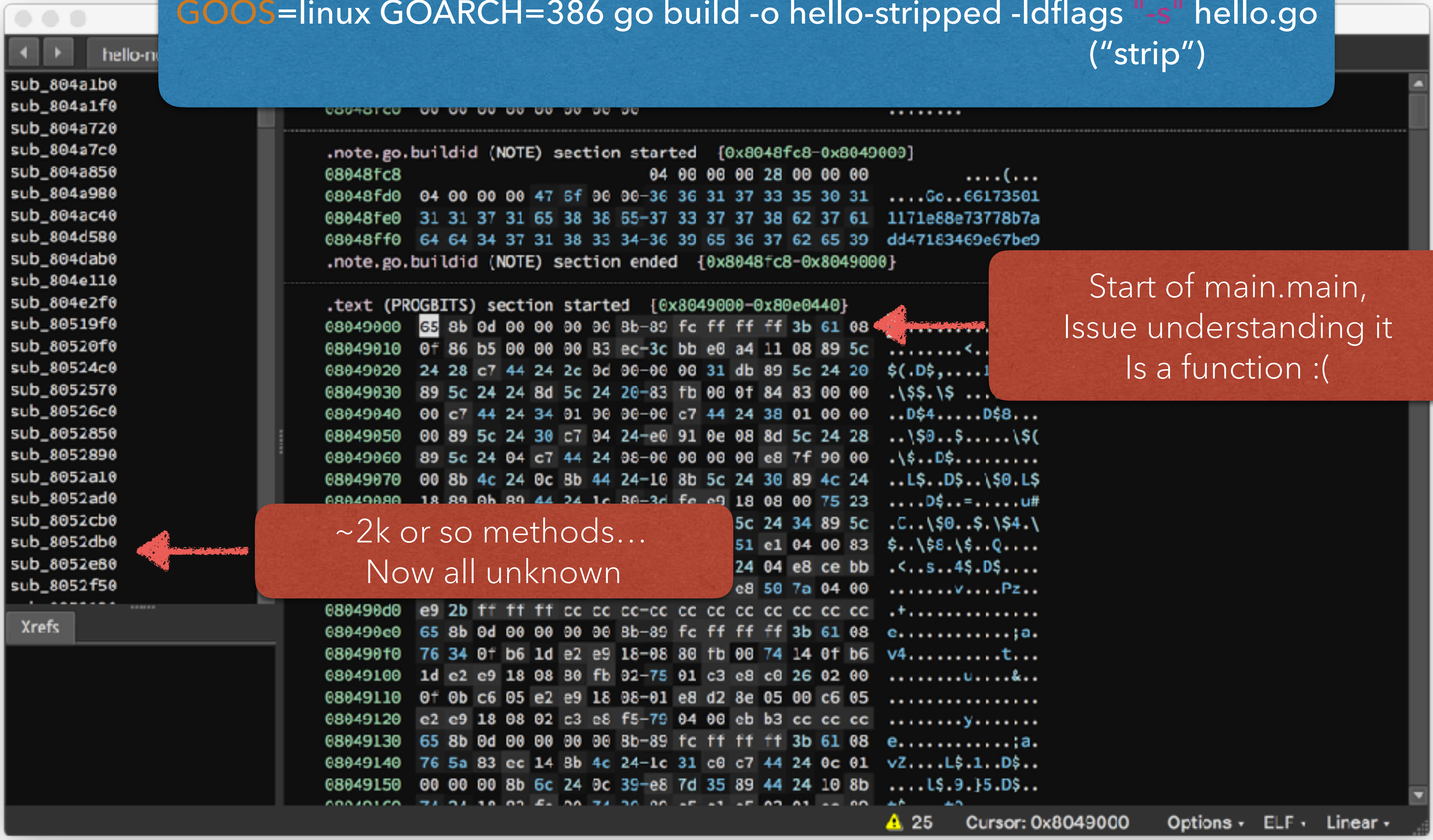
```
sub_804a1b0
sub_804a1f0
sub_804a720
sub_804a7c0
sub_804a850
sub_804a980
sub_804ac40
sub_804d580
sub_804dab0
sub_804e110
sub_804e2f0
sub_80519f0
sub_80520f0
sub_80524c0
sub_8052570
sub_80526c0
sub_8052850
sub_8052890
sub_8052a10
sub_8052ad0
sub_8052cb0
sub_8052db0
sub_8052e80
sub_8052f50
```

```
.note.go.buildid (NOTE) section started  [0x8048fc8-0x8049000]
08048fc8                         04 00 00 00 28 00 00 00          ....(...
08048fd0  04 00 00 00 47 6f 00 00-36 36 31 37 33 35 30 31  ....Go..66173501
08048fe0  31 31 37 31 65 38 38 65-37 33 37 37 38 62 37 61  1171e88e73778b7a
08048ff0  64 64 34 37 31 38 33 34-36 39 65 36 37 62 65 39  dd47183469e67be9
.note.go.buildid (NOTE) section ended  {0x8048fc8-0x8049000}

.text (PROGBITS) section started  [0x8049000-0x80e0440}
08049000  65 8b 0d 00 00 00 00 8b-89 fc ff ff ff 3b 61 08  e............;a.
08049010  0f 86 b5 00 00 00 83 ec-3c bb e0 a4 11 08 89 5c  ........<......\
08049020  24 28 c7 44 24 2c 0d 00-00 00 31 db 89 5c 24 20  $(.D$,....1..\$
08049030  89 5c 24 24 8d 5c 24 20-83 fb 00 0f 84 83 00 00  .\$$.\$ ........
08049040  00 c7 44 24 34 01 00 00-00 c7 44 24 38 01 00 00  ..D$4.....D$8...
08049050  00 89 5c 24 30 c7 04 24-e0 91 0e 08 8d 5c 24 28  ..\$0..$.....\$(
08049060  89 5c 24 04 c7 44 24 08-00 00 00 00 e8 7f 90 00  .\$..D$.........
08049070  00 8b 4c 24 0c 8b 44 24-10 8b 5c 24 30 89 4c 24  ..L$..D$..\$0.L$
08049080  18 89 0b 89 44 24 1c 80-3d fe e9 18 08 00 75 23  ....D$..=.....u#
08049090  89 43 04 8b 5c 24 30 89-1c 24 8b 5c 24 34 89 5c  .C..\$0..$.\$4.\
080490a0  24 04 8b 5c 24 38 89 5c-24 08 e8 51 e1 04 00 83  $..\$8.\$..Q....
080490b0  c4 3c c3 8d 73 04 89 34-24 89 44 24 04 e8 ce bb  .<..s..4$.D$....
080490c0  00 00 eb cf 89 03 e9 76-ff ff ff e8 50 7a 04 00  .......v....Pz..
080490d0  e9 2b ff ff ff cc cc cc-cc cc cc cc cc cc cc cc  .+..............
080490e0  65 8b 0d 00 00 00 00 8b-89 fc ff ff ff 3b 61 08  e............;a.
080490f0  76 34 0f b6 1d e2 e9 18-08 80 fb 00 74 14 0f b6  v4..........t...
08049100  1d e2 e9 18 08 80 fb 02-75 01 c3 e8 c0 26 02 00  ........u....&..
08049110  0f 0b c6 05 e2 e9 18 08-01 e8 d2 8e 05 00 c6 05  ................
08049120  e2 e9 18 08 02 c3 e8 f5-79 04 00 eb b3 cc cc cc  ........y.......
08049130  65 8b 0d 00 00 00 00 8b-89 fc ff ff ff 3b 61 08  e............;a.
08049140  76 5a 83 ec 14 8b 4c 24-1c 31 c0 c7 44 24 0c 01  vZ....L$.1..D$..
08049150  00 00 00 8b 6c 24 0c 39-e8 7d 35 89 44 24 10 8b  ....l$.9.}5.D$..
```

```
25   Cursor: 0x8049000   Options   ELF   Linear
```

WHY IS GO DIFFERENT?

Well… Crap



```
GOOS=linux GOARCH=386 go build -o hello-stripped -ldflags "-s" hello.go
```

("strip")

Start of main.main,
Issue understanding it
Is a function :(

~2k or so methods…
Now all unknown

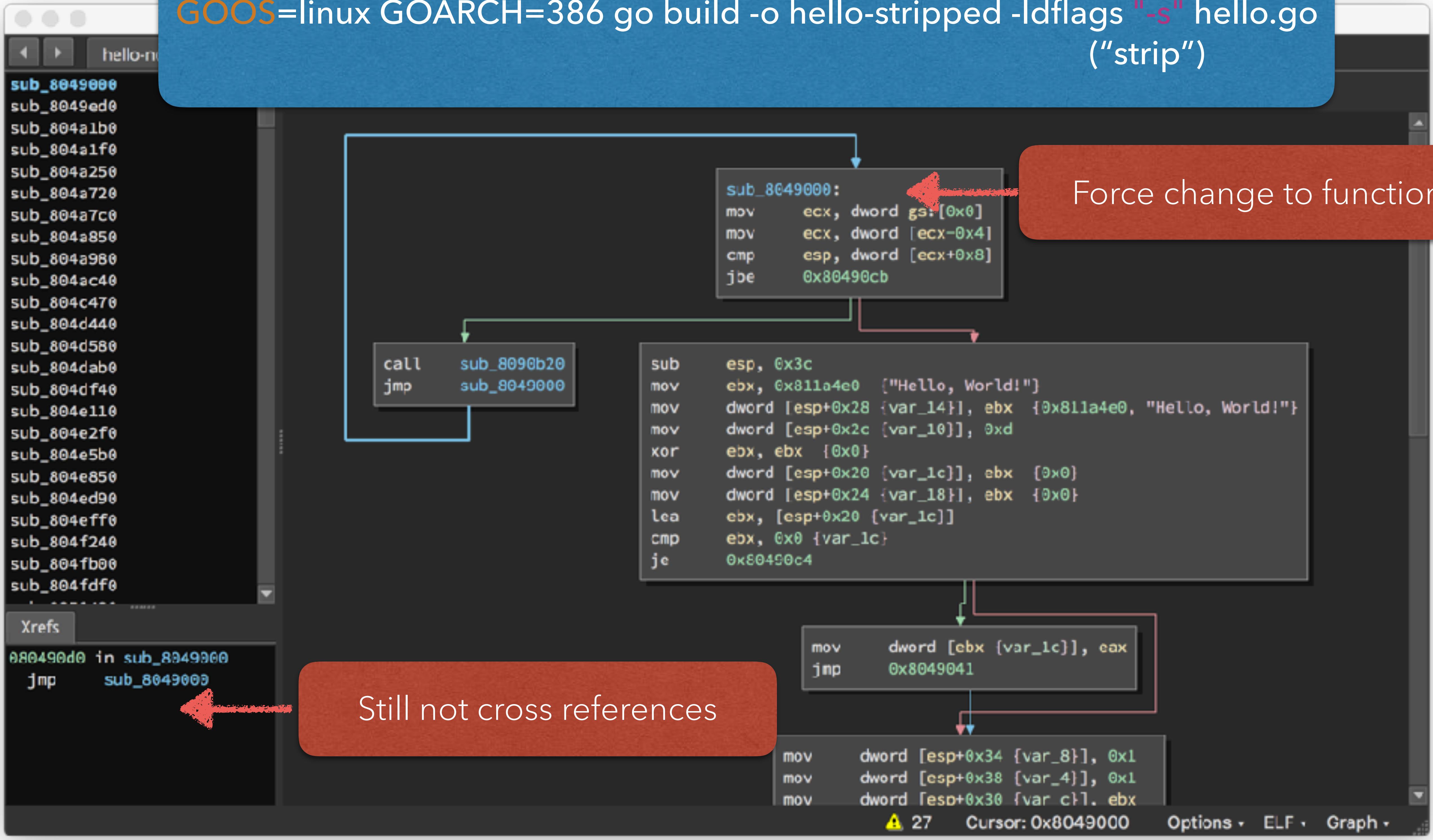
```
sub_804a1b0
sub_804a1f0
sub_804a720
sub_804a7c0
sub_804a850
sub_804a980
sub_804ac40
sub_804d580
sub_804dab0
sub_804e110
sub_804e2f0
sub_80519f0
sub_80520f0
sub_80524c0
sub_8052570
sub_80526c0
sub_8052850
sub_8052890
sub_8052a10
sub_8052ad0
sub_8052cb0
sub_8052db0
sub_8052e80
sub_8052f50
```

```
.note.go.buildid (NOTE) section started  [0x8048fc8-0x8049000]
08048fc8                            04 00 00 00 28 00 00 00     ....(...
08048fd0  04 00 00 00 47 6f 00 00-36 36 31 37 33 35 30 31  ....Go..66173501
08048fe0  31 31 37 31 65 38 38 65-37 33 37 37 38 62 37 61  1171e88e73778b7a
08048ff0  64 64 34 37 31 38 33 34-36 39 65 36 37 62 65 39  dd47183469e67be9
.note.go.buildid (NOTE) section ended  {0x8048fc8-0x8049000}

.text (PROGBITS) section started  {0x8049000-0x80e0440}
08049000  65 8b 0d 00 00 00 00 8b-89 fc ff ff ff 3b 61 08
08049010  0f 86 b5 00 00 00 83 ec-3c bb e0 a4 11 08 89 5c  ........<..
08049020  24 28 c7 44 24 2c 0d 00-00 00 31 db 89 5c 24 20  $(.D$,....1
08049030  89 5c 24 24 8d 5c 24 20-83 fb 00 0f 84 83 00 00  .\$$.\$ ...
08049040  00 c7 44 24 34 01 00 00-00 c7 44 24 38 01 00 00  ..D$4.....D$8...
08049050  00 89 5c 24 30 c7 04 24-e0 91 0e 08 8d 5c 24 28  ..\$0..$.....\$(
08049060  89 5c 24 04 c7 44 24 08-00 00 00 00 e8 7f 90 00  .\$..D$.........
08049070  00 8b 4c 24 0c 8b 44 24-10 8b 5c 24 30 89 4c 24  ..L$..D$..\$0.L$
08049080  18 89 0b 89 44 24 1c 80-3d fc e9 18 08 00 75 23  ....D$..=.....u#
                                          5c 24 34 89 5c  .C..\$0..$.\$4.\
                                          51 e1 04 00 83  $..\$8.\$..Q....
                                          24 04 e8 ce bb  .<..s..4$.D$....
                                          e8 50 7a 04 00  ........v....Pz..
080490d0  e9 2b ff ff ff cc cc cc-cc cc cc cc cc cc cc cc  .+..............
080490e0  65 8b 0d 00 00 00 00 8b-89 fc ff ff ff 3b 61 08  e............;a.
080490f0  76 34 0f b6 1d e2 e9 18-08 80 fb 00 74 14 0f b6  v4..........t...
08049100  1d e2 e9 18 08 80 fb 02-75 01 c3 e8 c0 26 02 00  .........u....&..
08049110  0f 0b c6 05 e2 e9 18 08-01 e8 d2 8e 05 00 c6 05  ................
08049120  e2 e9 18 08 02 c3 e8 f5-79 04 00 eb b3 cc cc cc  ........y.......
08049130  65 8b 0d 00 00 00 00 8b-89 fc ff ff ff 3b 61 08  e............;a.
08049140  76 5a 83 ec 14 8b 4c 24-1c 31 c0 c7 44 24 0c 01  vZ....L$.1..D$..
08049150  00 00 00 8b 6c 24 0c 39-e8 7d 35 89 44 24 10 8b  ....l$.9.}5.D$..
```

WHY IS GO DIFFERENT?

Well… Crap

GOOS=linux GOARCH=386 go build -o hello-stripped -ldflags "-s" hello.go ("strip")



Force change to function

Still not cross references

```
sub_8049000
sub_8049ed0
sub_804a1b0
sub_804a1f0
sub_804a250
sub_804a720
sub_804a7c0
sub_804a850
sub_804a980
sub_804ac40
sub_804c470
sub_804d440
sub_804d580
sub_804dab0
sub_804df40
sub_804e110
sub_804e2f0
sub_804e5b0
sub_804e850
sub_804ed90
sub_804eff0
sub_804f240
sub_804fb00
sub_804fdf0
```

Xrefs

```
080490d0 in sub_8049000
  jmp       sub_8049000
```

```
sub_8049000:
mov     ecx, dword gs:[0x0]
mov     ecx, dword [ecx-0x4]
cmp     esp, dword [ecx+0x8]
jbe     0x80490cb
```

```
call    sub_8090b20
jmp     sub_8049000
```

```
sub     esp, 0x3c
mov     ebx, 0x811a4e0  {"Hello, World!"}
mov     dword [esp+0x28 {var_14}], ebx  {0x811a4e0, "Hello, World!"}
mov     dword [esp+0x2c {var_10}], 0xd
xor     ebx, ebx  {0x0}
mov     dword [esp+0x20 {var_1c}], ebx  {0x0}
mov     dword [esp+0x24 {var_18}], ebx  {0x0}
lea     ebx, [esp+0x20 {var_1c}]
cmp     ebx, 0x0 {var_1c}
je      0x80490c4
```

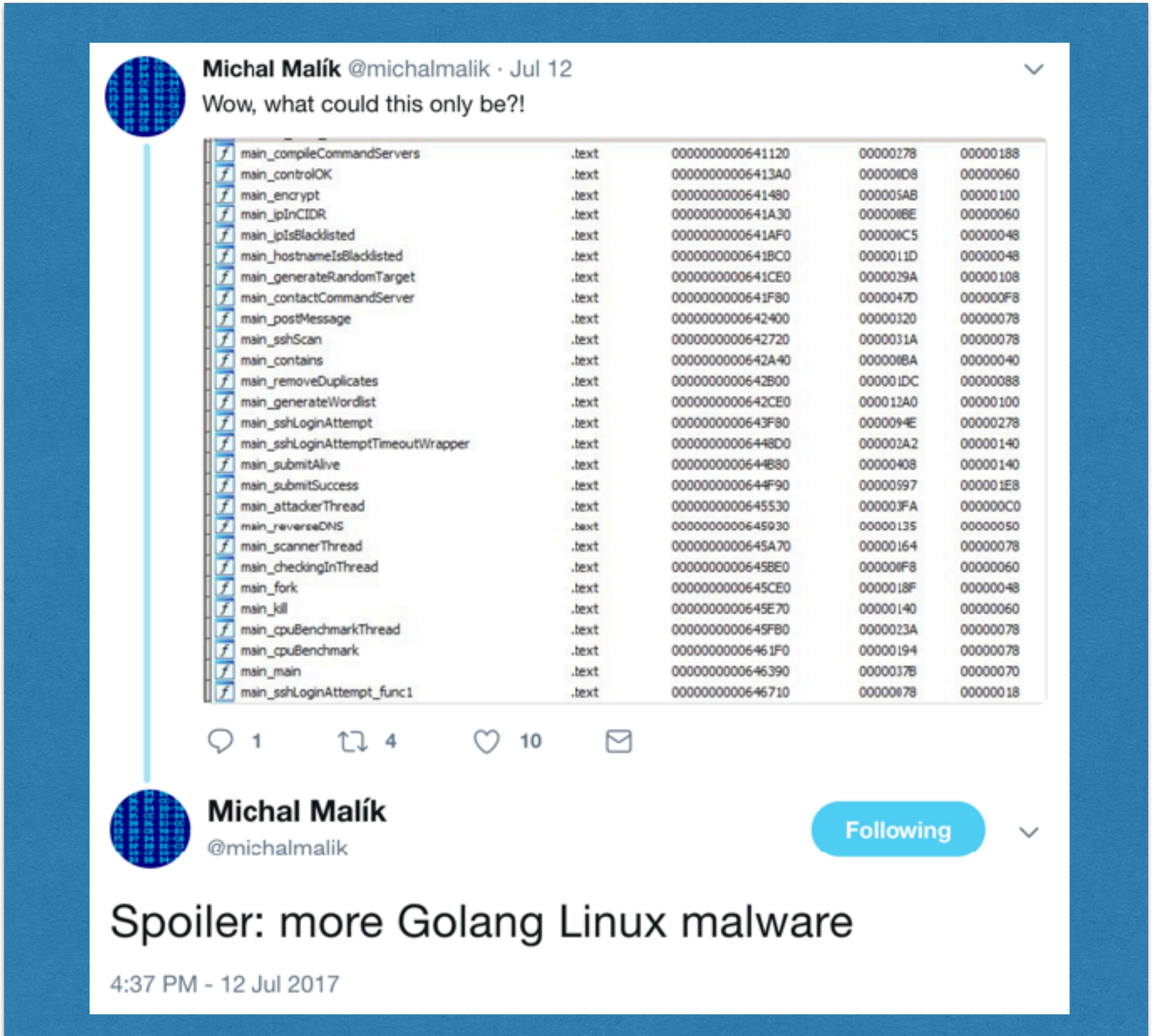
```
mov     dword [ebx {var_1c}], eax
jmp     0x8049041
```

```
mov     dword [esp+0x34 {var_8}], 0x1
mov     dword [esp+0x38 {var_4}], 0x1
mov     dword [esp+0x30 {var_c}], ebx
```

27 Cursor: 0x8049000 Options ELF Graph

WHY SHOULD I CARE?

Malware, Offense and defense!



Michal Malík @michalmalik · Jul 12

Wow, what could this only be?!

```
main_compileCommandServers              .text   0000000000641120   00000278   00000188
main_controlOK                          .text   00000000006413A0   000000D8   00000060
main_encrypt                            .text   0000000000641480   000005AB   00000100
main_ipInCIDR                           .text   0000000000641A30   000000BE   00000060
main_ipIsBlacklisted                    .text   0000000000641AF0   000000C5   00000048
main_hostnameIsBlacklisted              .text   0000000000641BC0   0000011D   00000048
main_generateRandomTarget               .text   0000000000641CE0   0000029A   00000108
main_contactCommandServer               .text   0000000000641F80   0000047D   000000F8
main_postMessage                        .text   0000000000642400   00000320   00000078
main_sshScan                            .text   0000000000642720   0000031A   00000078
main_contains                           .text   0000000000642A40   000000BA   00000040
main_removeDuplicates                   .text   0000000000642B00   000001DC   00000088
main_generateWordlist                   .text   0000000000642CE0   000012A0   00000100
main_sshLoginAttempt                    .text   0000000000643F80   0000094E   00000278
main_sshLoginAttemptTimeoutWrapper      .text   00000000006448D0   000002A2   00000140
main_submitAlive                        .text   0000000000644B80   00000408   00000140
main_submitSuccess                      .text   0000000000644F90   00000597   000001E8
main_attackerThread                     .text   0000000000645530   000003FA   000000C0
main_reverseDNS                         .text   0000000000645930   00000135   00000050
main_scannerThread                      .text   0000000000645A70   00000164   00000078
main_checkingInThread                   .text   0000000000645BE0   000000F8   00000060
main_fork                               .text   0000000000645CE0   0000018F   00000048
main_kill                               .text   0000000000645E70   00000140   00000060
main_cpuBenchmarkThread                 .text   0000000000645FB0   0000023A   00000078
main_cpuBenchmark                       .text   00000000006461F0   00000194   00000078
main_main                               .text   0000000000646390   0000037B   00000070
main_sshLoginAttempt_func1              .text   0000000000646710   00000078   00000018
```

1 4 10

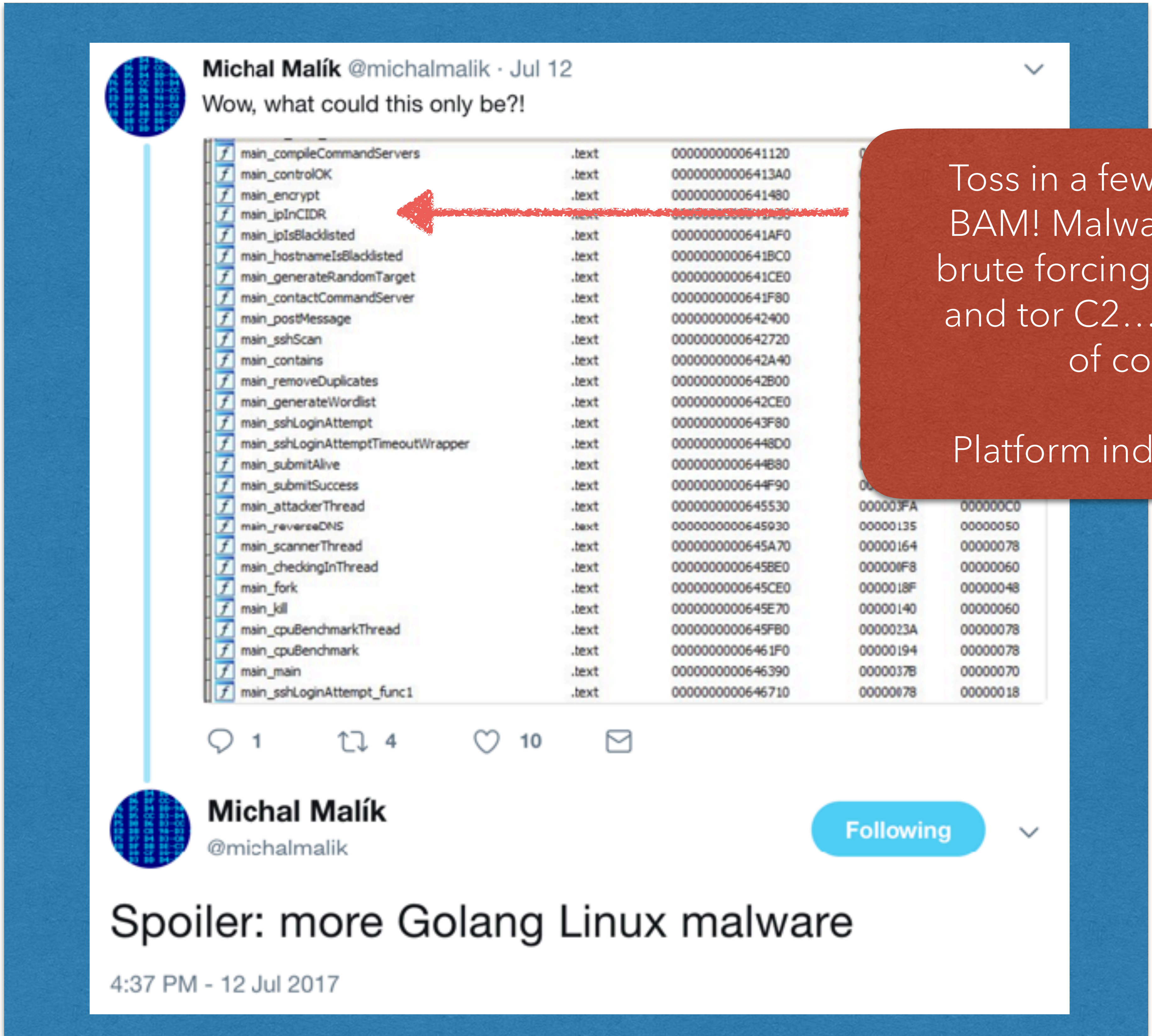
Michal Malík
@michalmalik

Spoiler: more Golang Linux malware

4:37 PM - 12 Jul 2017

WHY SHOULD I CARE?

Malware, Offense and defense!



Michal Malík @michalmalik · Jul 12

Wow, what could this only be?!

```
main_compileCommandServers              .text   0000000000641120
main_controlOK                          .text   00000000006413A0
main_encrypt                            .text   0000000000641480
main_ipInCIDR                           [illegible]
main_ipIsBlacklisted                    .text   0000000000641AF0
main_hostnameIsBlacklisted              .text   0000000000641BC0
main_generateRandomTarget               .text   0000000000641CE0
main_contactCommandServer               .text   0000000000641F80
main_postMessage                        .text   0000000000642400
main_sshScan                            .text   0000000000642720
main_contains                           .text   0000000000642A40
main_removeDuplicates                   .text   0000000000642B00
main_generateWordlist                   .text   0000000000642CE0
main_sshLoginAttempt                    .text   0000000000643F80
main_sshLoginAttemptTimeoutWrapper      .text   00000000006448D0
main_submitAlive                        .text   0000000000644B80
main_submitSuccess                      .text   0000000000644F90
main_attackerThread                     .text   0000000000645530   000003FA   000000C0
main_reverseDNS                         .text   0000000000645930   00000135   00000050
main_scannerThread                      .text   0000000000645A70   00000164   00000078
main_checkingInThread                   .text   0000000000645BE0   000000F8   00000060
main_fork                               .text   0000000000645CE0   0000018F   00000048
main_kill                               .text   0000000000645E70   00000140   00000060
main_cpuBenchmarkThread                 .text   0000000000645FB0   0000023A   00000078
main_cpuBenchmark                       .text   00000000006461F0   00000194   00000078
main_main                               .text   0000000000646390   0000037B   00000070
main_sshLoginAttempt_func1              .text   0000000000646710   00000078   00000018
```

Michal Malík
@michalmalik

Spoiler: more Golang Linux malware

4:37 PM - 12 Jul 2017

Toss in a few libraries… BAM! Malware with ssh brute forcing, ransoming and tor C2… ~200 lines of code?

Platform independent!

WHY SHOULD I CARE?

Malware, Offense and defense!

GO is straight forward, easy to use, easy to replicate functionality when reversing it :D

Mac malware, Windows malware, Linux malware - oh my! So portable

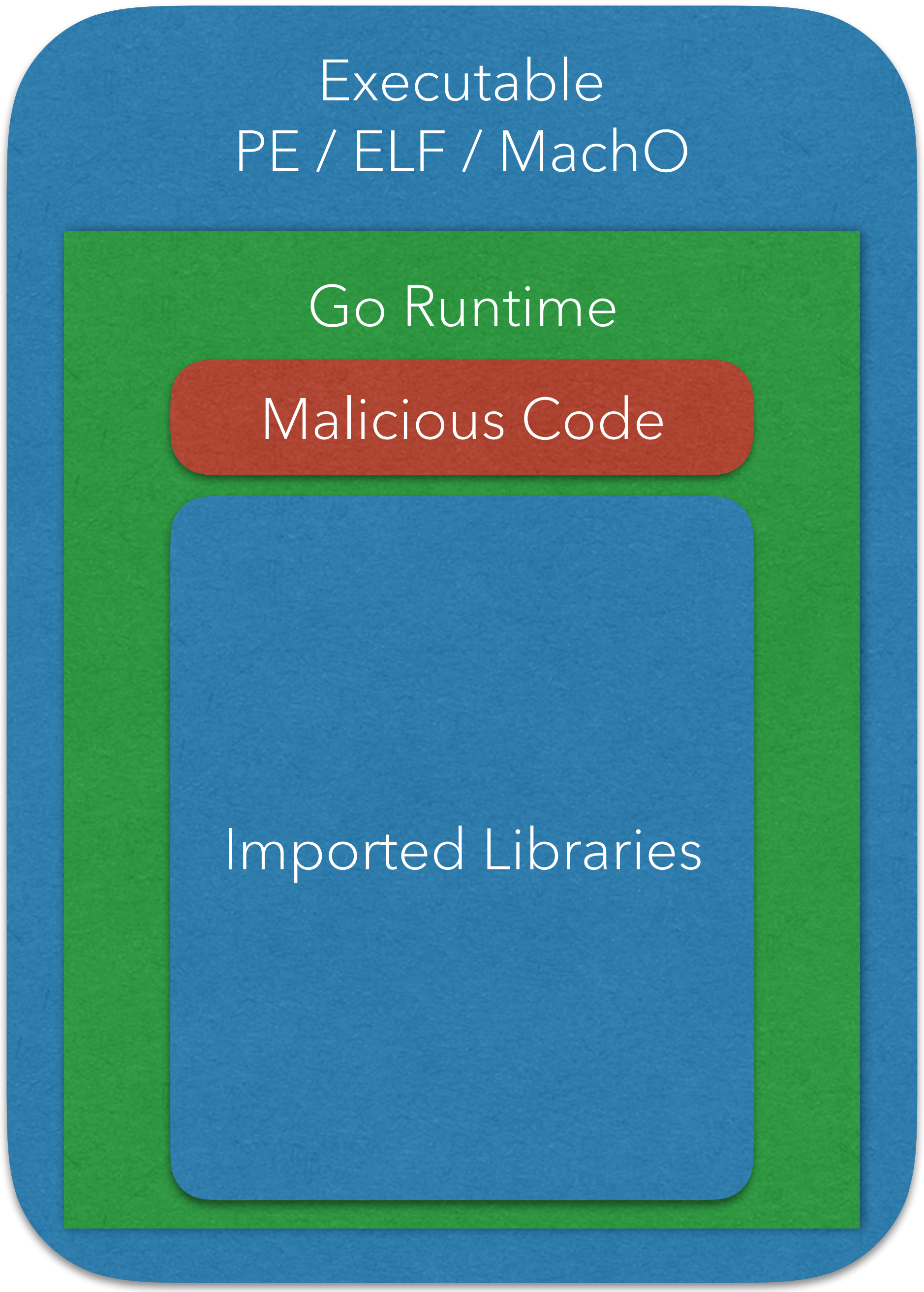
A few malware over the past year…

- Linux.Lady
- Linux.Rex
- Linux/Agent.DT
- YourRansom "educational" w/ wanna cry exploit
- GoBot(2) (POC)
- GoAT (POC)
- EGESPLOIT (POC)
- Ebowla (POC)
- Go-mimikatz
- Plenty more no one has classified

Pentesting:
"what makes Go awesome for AV-Avoidance, is use 'net/http' and parse it to a "import 'c'". Then in C you parse the buffer you get from Go and make a 'mistake' in order to trigger a vulnerability. That allows you to load meterpeter in memory." - Rapt0r- (reddit)

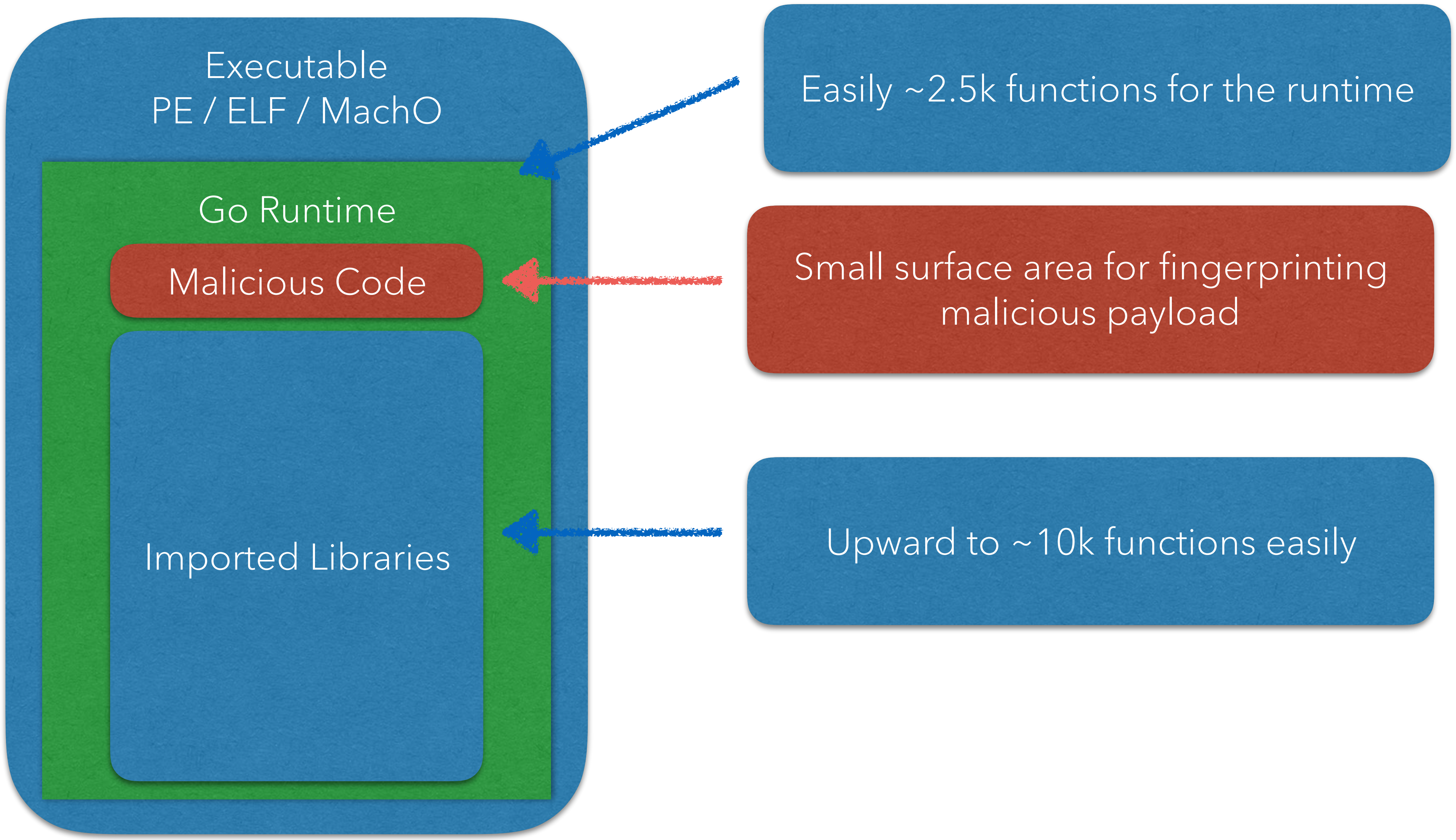
WHY SHOULD I CARE?

AV's Can have a hard time



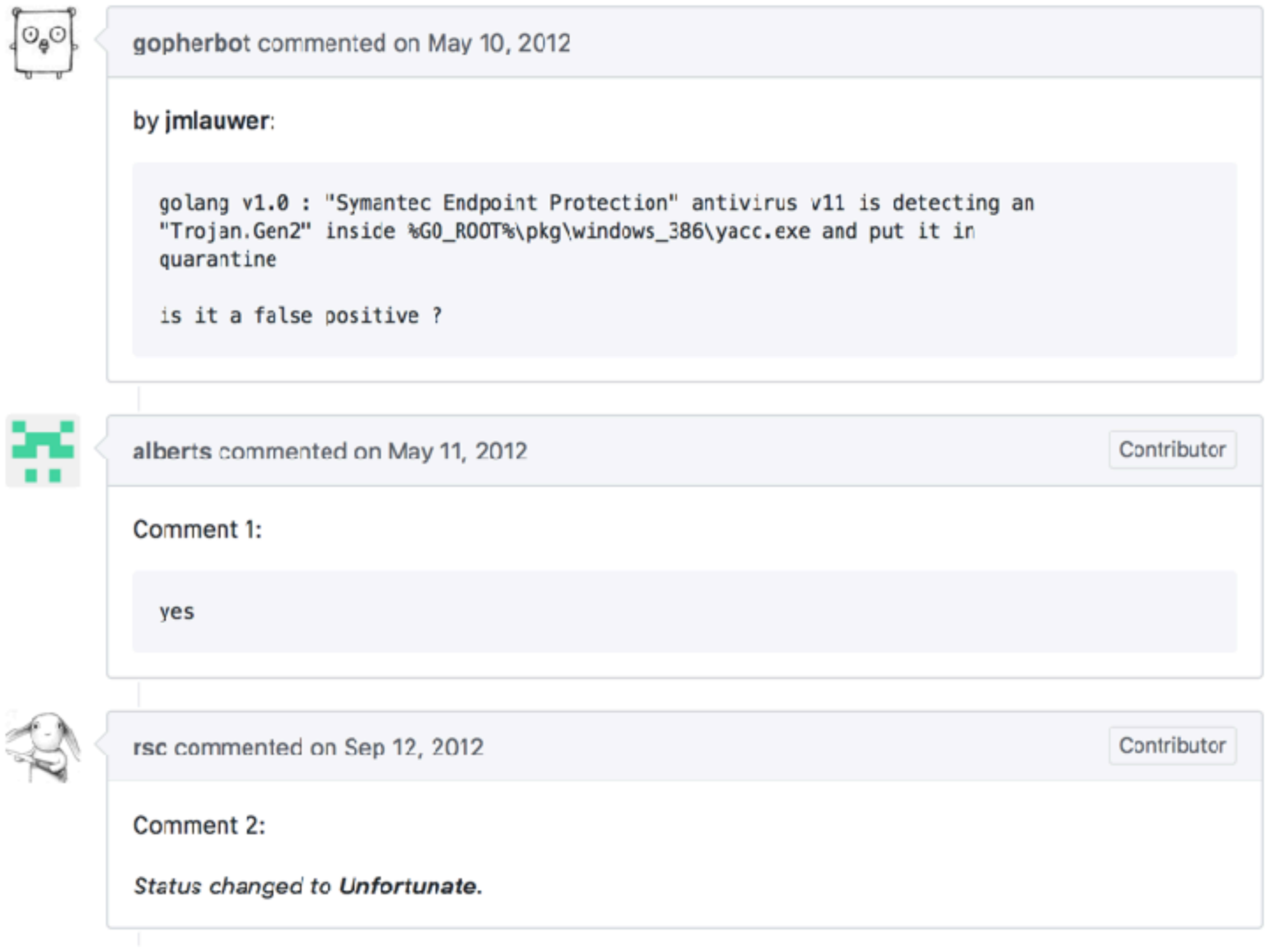
WHY SHOULD I CARE?

AV's Can have a hard time


Executable
PE / ELF / MachO
Go Runtime
Malicious Code
Imported Libraries
Easily ~2.5k functions for the runtime
Small surface area for fingerprinting malicious payload
Upward to ~10k functions easily

WHY SHOULD I CARE?

AV's Can have a hard time



gopherbot commented on May 10, 2012

by **jmlauwer**:

```
golang v1.0 : "Symantec Endpoint Protection" antivirus v11 is detecting an
"Trojan.Gen2" inside %GO_ROOT%\pkg\windows_386\yacc.exe and put it in
quarantine

is it a false positive ?
```

alberts commented on May 11, 2012 Contributor

Comment 1:

```
yes
```

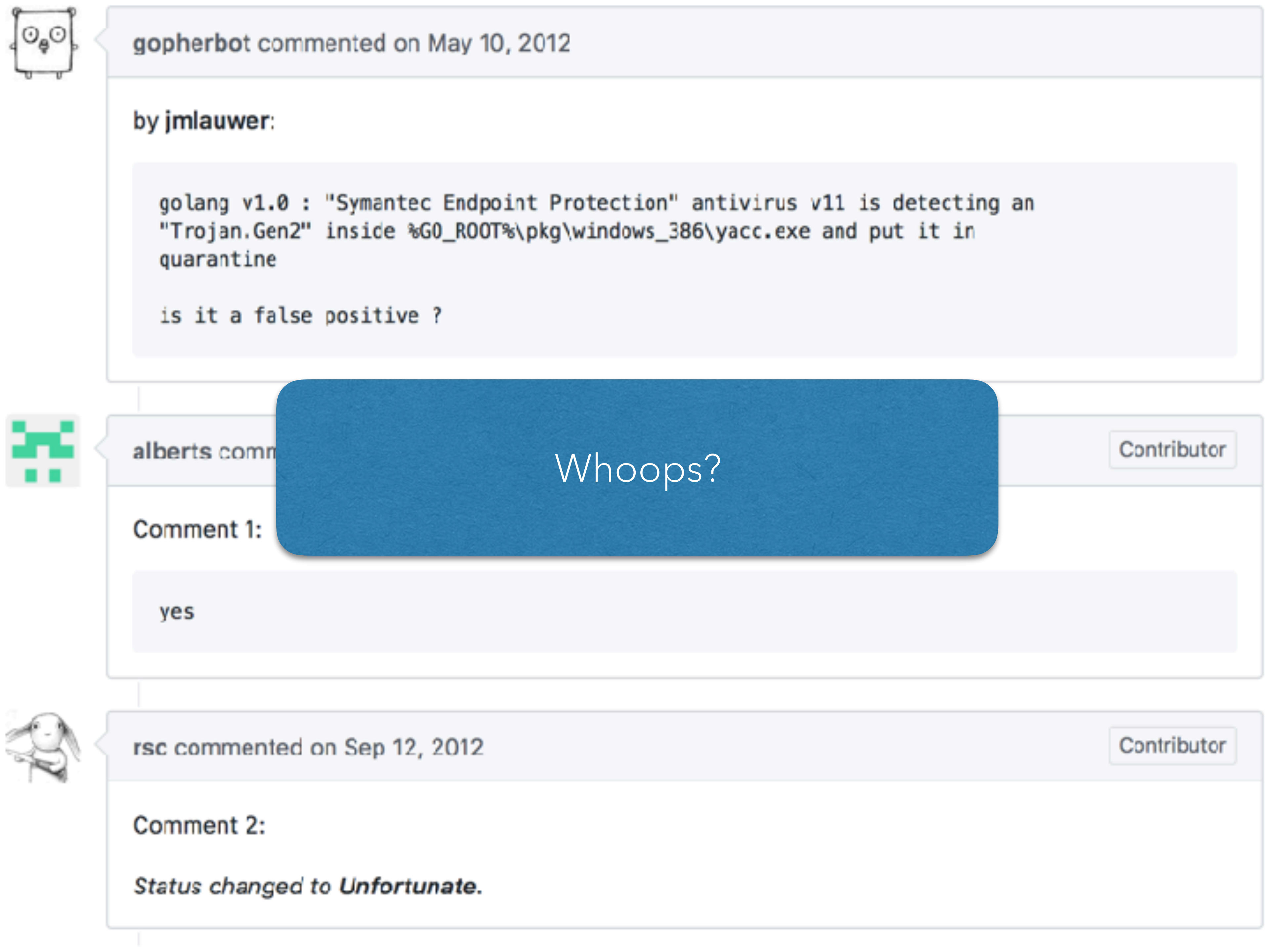
rsc commented on Sep 12, 2012 Contributor

Comment 2:

*Status changed to **Unfortunate**.*

WHY SHOULD I CARE?

AV's Can have a hard time



gopherbot commented on May 10, 2012

by **jmlauwer**:

```
golang v1.0 : "Symantec Endpoint Protection" antivirus v11 is detecting an
"Trojan.Gen2" inside %GO_ROOT%\pkg\windows_386\yacc.exe and put it in
quarantine

is it a false positive ?
```

alberts comm

Contributor

Comment 1:

```
yes
```

rsc commented on Sep 12, 2012

Contributor

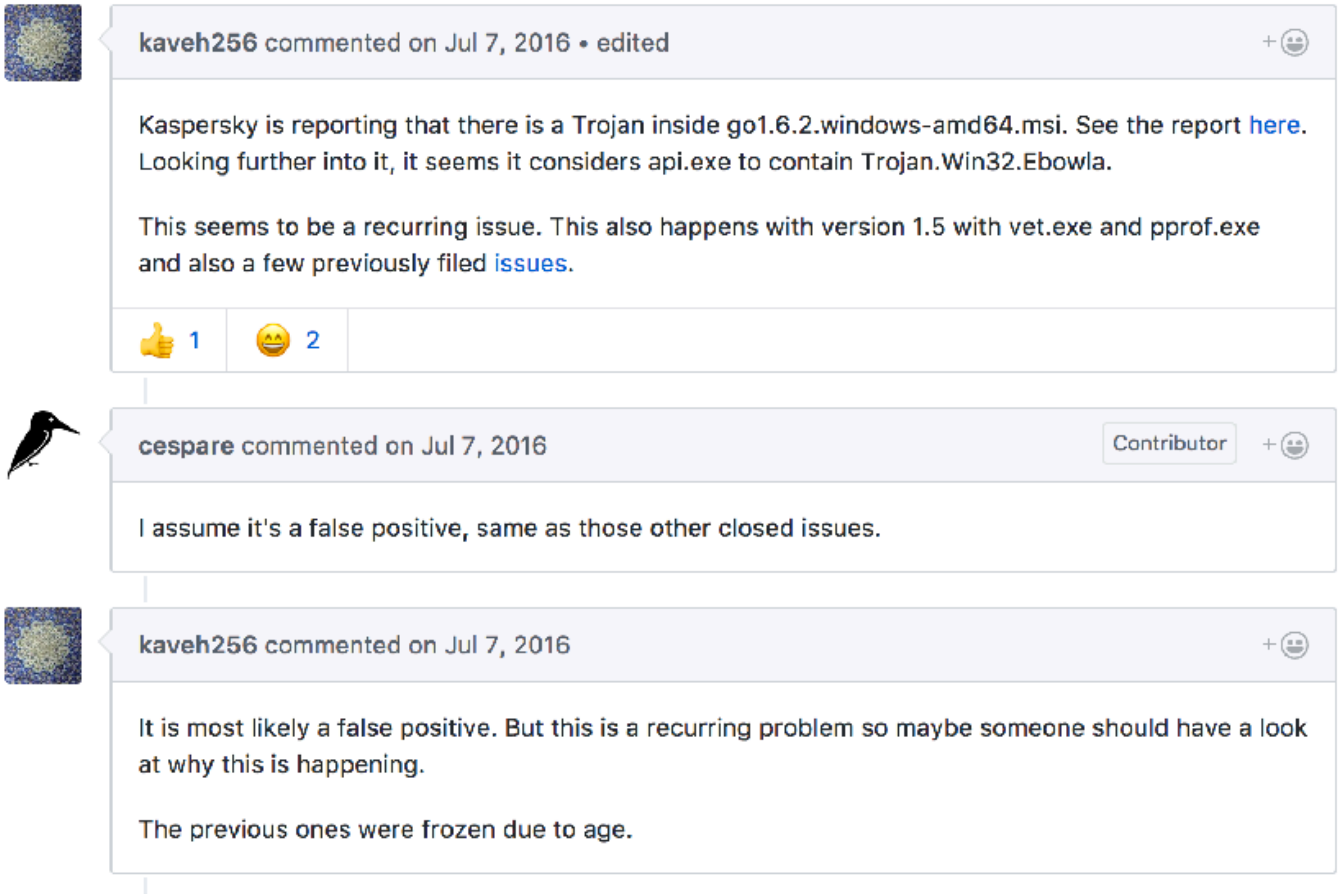
Comment 2:

*Status changed to **Unfortunate**.*

Whoops?

WHY SHOULD I CARE?

AV's Can have a hard time



kaveh256 commented on Jul 7, 2016 • edited

Kaspersky is reporting that there is a Trojan inside go1.6.2.windows-amd64.msi. See the report here. Looking further into it, it seems it considers api.exe to contain Trojan.Win32.Ebowla.

This seems to be a recurring issue. This also happens with version 1.5 with vet.exe and pprof.exe and also a few previously filed issues.

👍 1 😄 2

cespare commented on Jul 7, 2016 Contributor

I assume it's a false positive, same as those other closed issues.

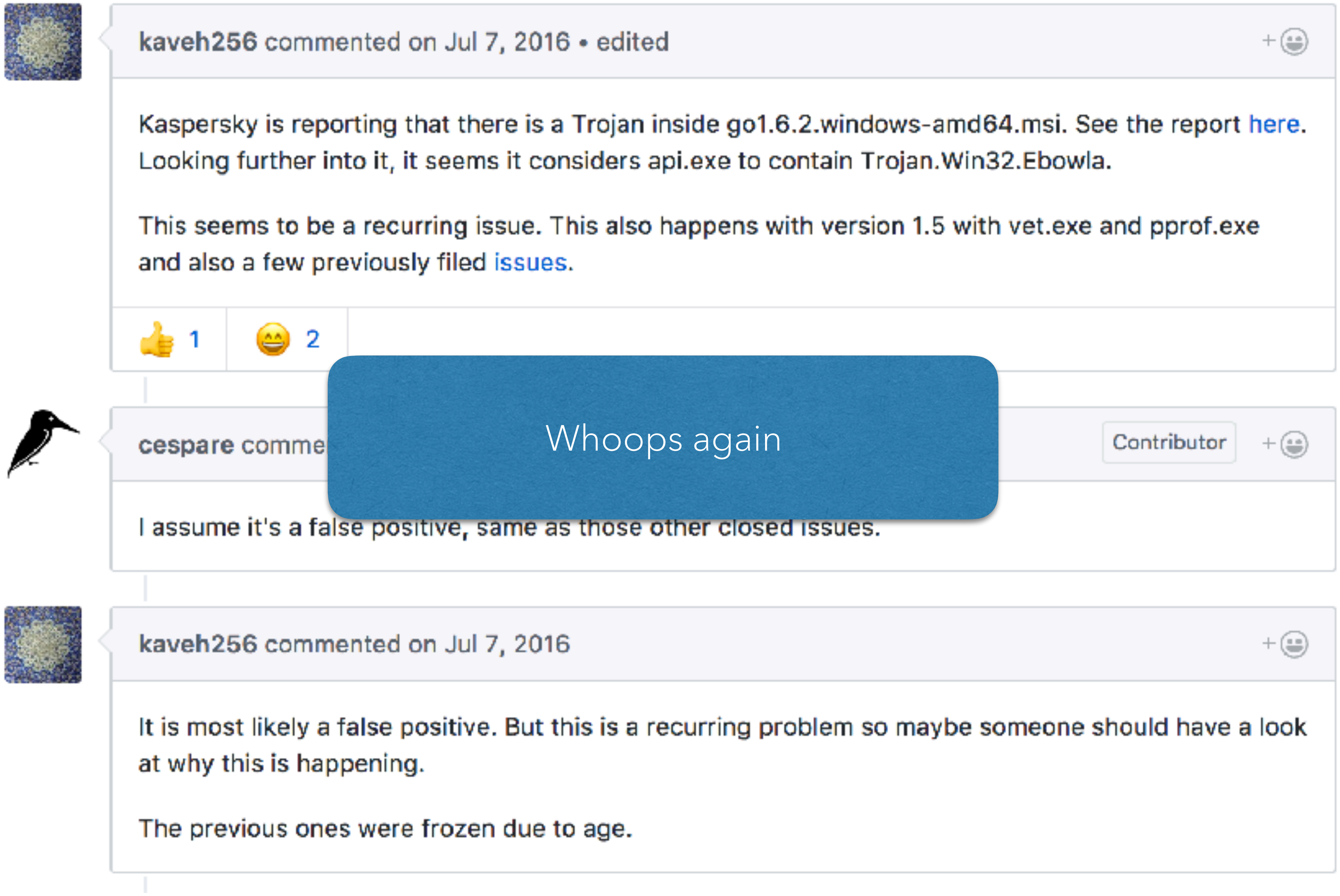
kaveh256 commented on Jul 7, 2016

It is most likely a false positive. But this is a recurring problem so maybe someone should have a look at why this is happening.

The previous ones were frozen due to age.

WHY SHOULD I CARE?

AV's Can have a hard time



kaveh256 commented on Jul 7, 2016 • edited

Kaspersky is reporting that there is a Trojan inside go1.6.2.windows-amd64.msi. See the report here. Looking further into it, it seems it considers api.exe to contain Trojan.Win32.Ebowla.

This seems to be a recurring issue. This also happens with version 1.5 with vet.exe and pprof.exe and also a few previously filed issues.

👍 1 😄 2

cespare comme

Contributor

I assume it's a false positive, same as those other closed issues.

kaveh256 commented on Jul 7, 2016

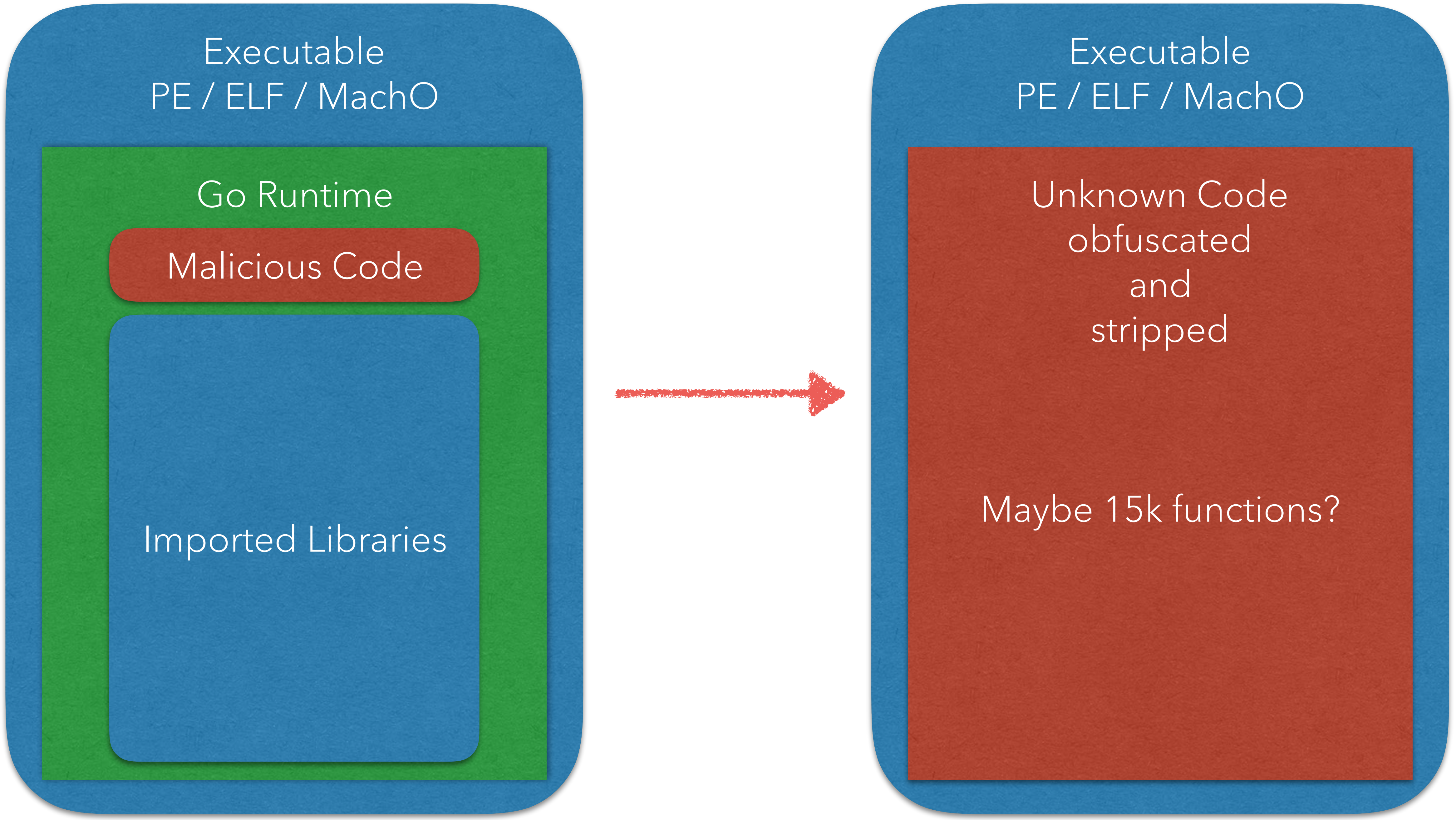
It is most likely a false positive. But this is a recurring problem so maybe someone should have a look at why this is happening.

The previous ones were frozen due to age.

Whoops again

WHY SHOULD I CARE?

AV's Can have a hard time



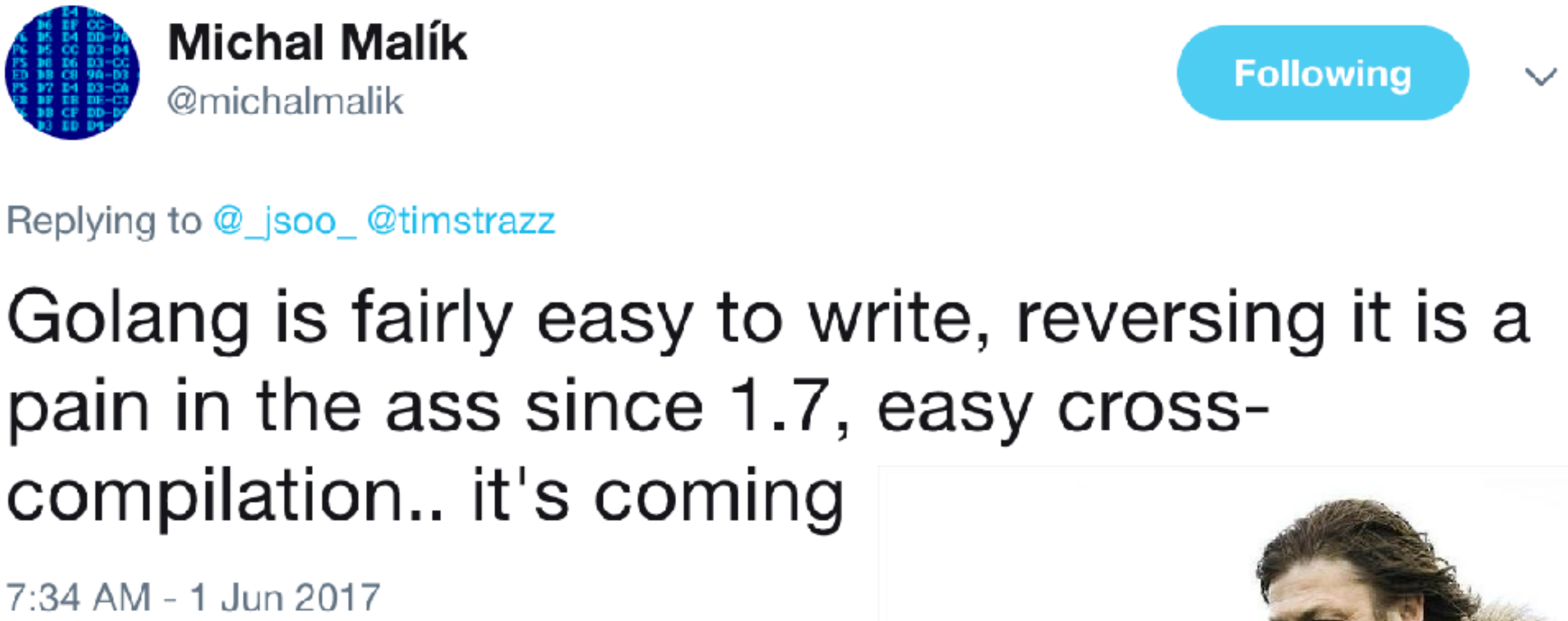
WHY SHOULD I CARE?

Other things than defense…

- Many more server side apps moving towards GO
- While “memory safe” programmers still make many issues
- Lots of licensing protection is starting to make use of GO
- Bug hunting and bounties are ripe due to lack of reverses with skill set
- Some “expensive" bugs are better detected in disassembled code than auditing the code, especially since most audit tend to be blackboxes
- Developers feel invulnerable! (Show them otherwise…)

WHY SHOULD I CARE?

TLDR



Michal Malík
@michalmalik

Following

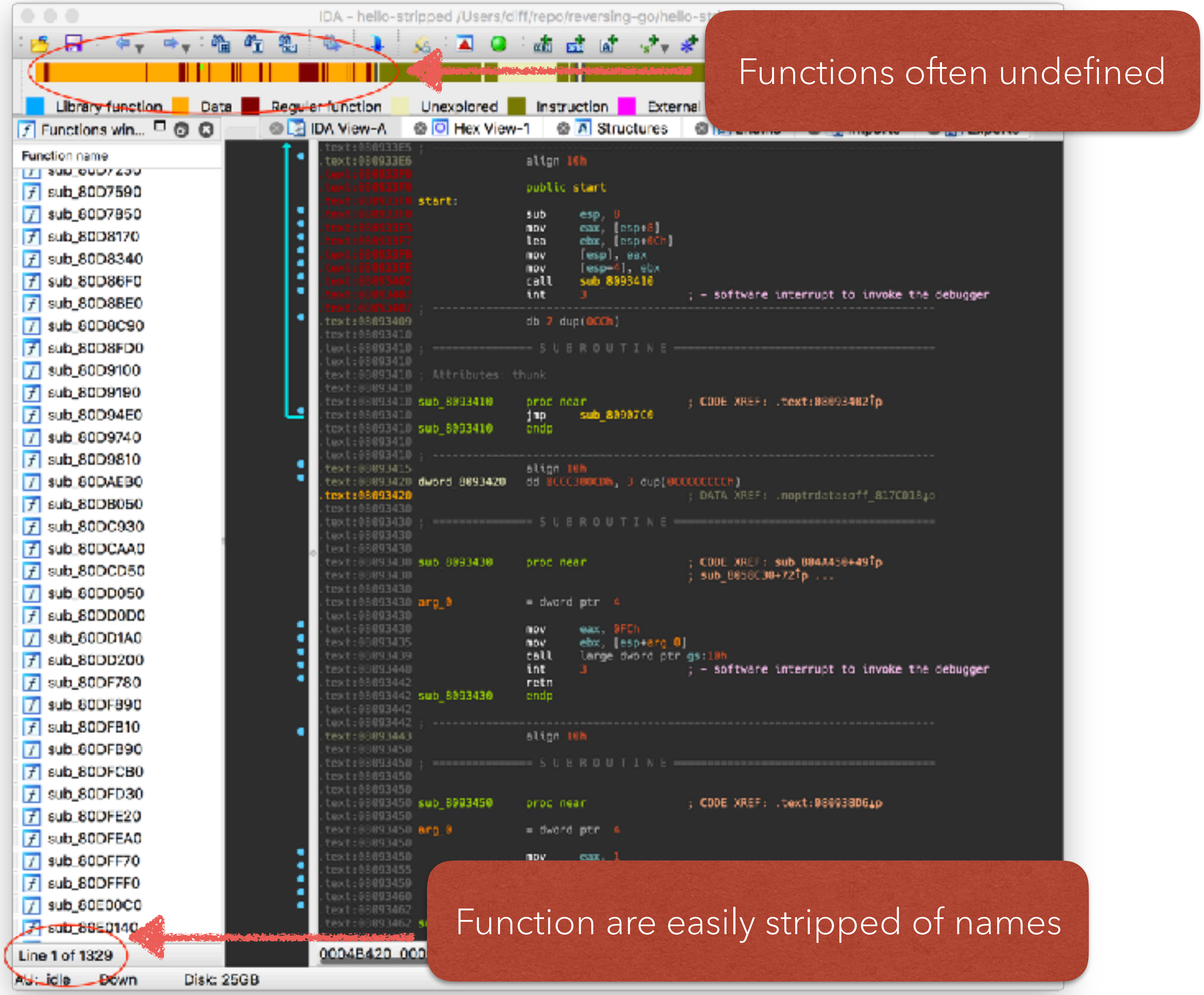
Replying to @_jsoo_ @timstrazz

Golang is fairly easy to write, reversing it is a pain in the ass since 1.7, easy cross-compilation.. it's coming

7:34 AM - 1 Jun 2017

FIXING OUR TOOLS…

IDA Pro and Binary Ninja don't have great Go support :\



Functions often undefined

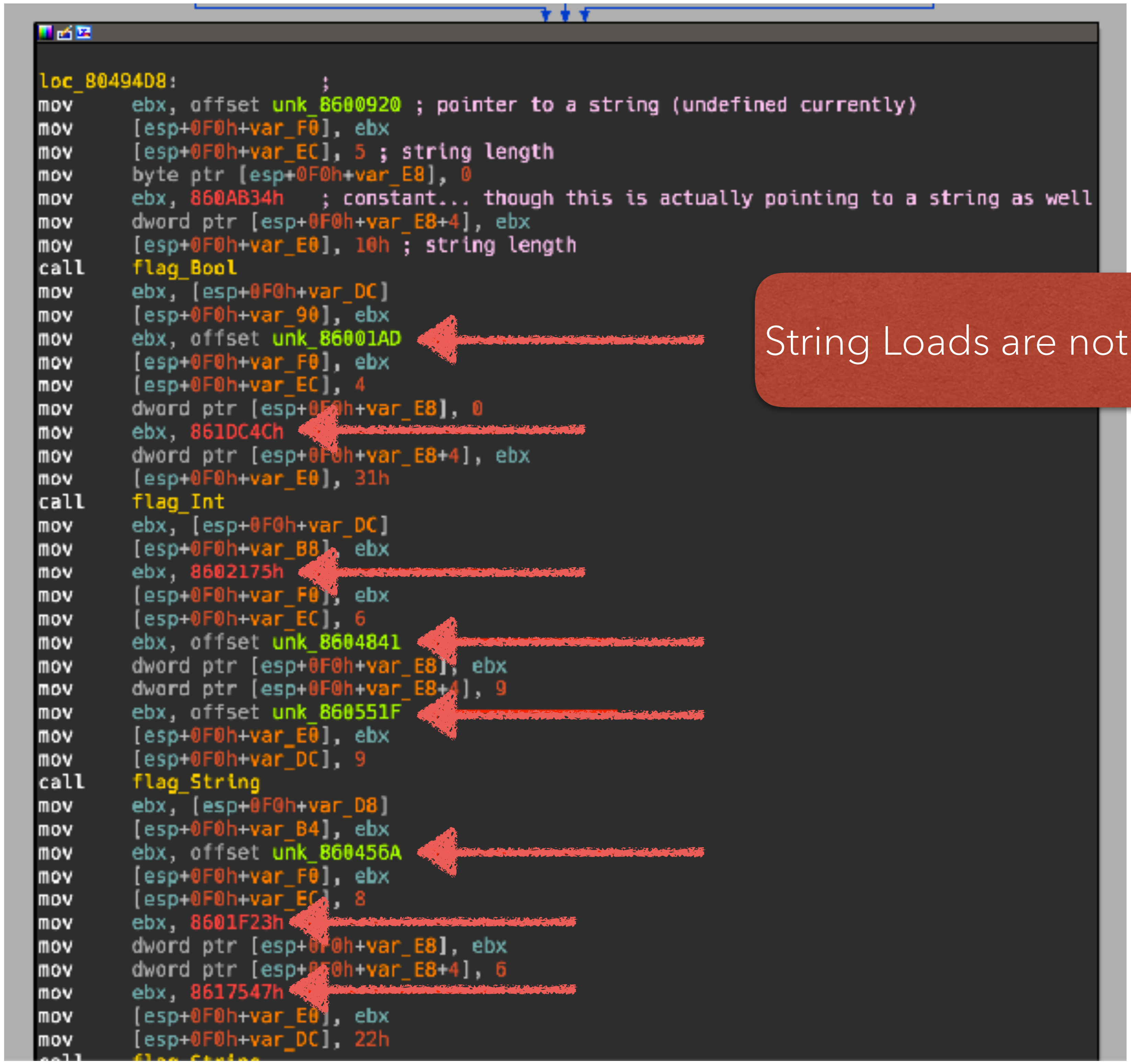
```
sub_80D7590
sub_80D7850
sub_80D8170
sub_80D8340
sub_80D86F0
sub_80D8BE0
sub_80D8C90
sub_80D8FD0
sub_80D9100
sub_80D9190
sub_80D94E0
sub_80D9740
sub_80D9810
sub_80DAE30
sub_80DB050
sub_80DC030
sub_80DCAA0
sub_80DCD50
sub_80DD050
sub_80DD0D0
sub_80DD1A0
sub_80DD200
sub_80DF780
sub_80DF890
sub_80DFB10
sub_80DFB90
sub_80DFCB0
sub_80DFD30
sub_80DFE20
sub_80DFEA0
sub_80DFF70
sub_80DFFF0
sub_80E00C0
sub_80E0140
```

```
.text:080933E5 ; ---------------------------------------------------------------
.text:080933E6                 align 10h
.text:080933F0
.text:080933F0                 public start
.text:080933F0 start:
.text:080933F0                 sub     esp, 8
.text:080933F3                 mov     eax, [esp+8]
.text:080933F7                 lea     ebx, [esp+0Ch]
.text:080933FB                 mov     [esp], eax
.text:080933FE                 mov     [esp+4], ebx
.text:08093402                 call    sub_8093410
.text:08093407                 int     3               ; - software interrupt to invoke the debugger
.text:08093407 ; ---------------------------------------------------------------
.text:08093409                 db 7 dup(0CCh)
.text:08093410
.text:08093410 ; =============== S U B R O U T I N E ===========================
.text:08093410
.text:08093410 ; Attributes: thunk
.text:08093410
.text:08093410 sub_8093410     proc near               ; CODE XREF: .text:08093402↑p
.text:08093410                 jmp     sub_80907C0
.text:08093410 sub_8093410     endp
.text:08093410
.text:08093410 ; ---------------------------------------------------------------
.text:08093415                 align 10h
.text:08093420 dword_8093420   dd 0CCC300CDh, 3 dup(0CCCCCCCCh)
.text:08093420                                         ; DATA XREF: .noptrdata:off_817C018↓o
.text:08093430
.text:08093430 ; =============== S U B R O U T I N E ===========================
.text:08093430
.text:08093430
.text:08093430 sub_8093430     proc near               ; CODE XREF: sub_804A450+49↑p
.text:08093430                                         ; sub_8050C30+72↑p ...
.text:08093430
.text:08093430 arg_0           = dword ptr  4
.text:08093430
.text:08093430                 mov     eax, 0FCh
.text:08093435                 mov     ebx, [esp+arg_0]
.text:08093439                 call    large dword ptr gs:10h
.text:08093440                 int     3               ; - software interrupt to invoke the debugger
.text:08093442                 retn
.text:08093442 sub_8093430     endp
.text:08093442
.text:08093442 ; ---------------------------------------------------------------
.text:08093443                 align 10h
.text:08093450
.text:08093450 ; =============== S U B R O U T I N E ===========================
.text:08093450
.text:08093450
.text:08093450 sub_8093450     proc near               ; CODE XREF: .text:080933D6↓p
.text:08093450
.text:08093450 arg_0           = dword ptr  4
.text:08093450
.text:08093450                 mov     eax, 1
.text:08093455
.text:08093459
.text:08093460
.text:08093462
.text:08093462
0004B420 000
```

Function are easily stripped of names

FIXING OUR TOOLS…

IDA Pro and Binary Ninja don't have great Go support :\



```
loc_80494D8:                ;
mov     ebx, offset unk_8600920 ; pointer to a string (undefined currently)
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 5 ; string length
mov     byte ptr [esp+0F0h+var_E8], 0
mov     ebx, 860AB34h   ; constant... though this is actually pointing to a string as well
mov     dword ptr [esp+0F0h+var_E8+4], ebx
mov     [esp+0F0h+var_E0], 10h ; string length
call    flag_Bool
mov     ebx, [esp+0F0h+var_DC]
mov     [esp+0F0h+var_90], ebx
mov     ebx, offset unk_86001AD
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 4
mov     dword ptr [esp+0F0h+var_E8], 0
mov     ebx, 861DC4Ch
mov     dword ptr [esp+0F0h+var_E8+4], ebx
mov     [esp+0F0h+var_E0], 31h
call    flag_Int
mov     ebx, [esp+0F0h+var_DC]
mov     [esp+0F0h+var_B8], ebx
mov     ebx, 8602175h
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 6
mov     ebx, offset unk_8604841
mov     dword ptr [esp+0F0h+var_E8], ebx
mov     dword ptr [esp+0F0h+var_E8+4], 9
mov     ebx, offset unk_860551F
mov     [esp+0F0h+var_E0], ebx
mov     [esp+0F0h+var_DC], 9
call    flag_String
mov     ebx, [esp+0F0h+var_D8]
mov     [esp+0F0h+var_B4], ebx
mov     ebx, offset unk_860456A
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 8
mov     ebx, 8601F23h
mov     dword ptr [esp+0F0h+var_E8], ebx
mov     dword ptr [esp+0F0h+var_E8+4], 6
mov     ebx, 8617547h
mov     [esp+0F0h+var_E0], ebx
mov     [esp+0F0h+var_DC], 22h
```

String Loads are not "normal"

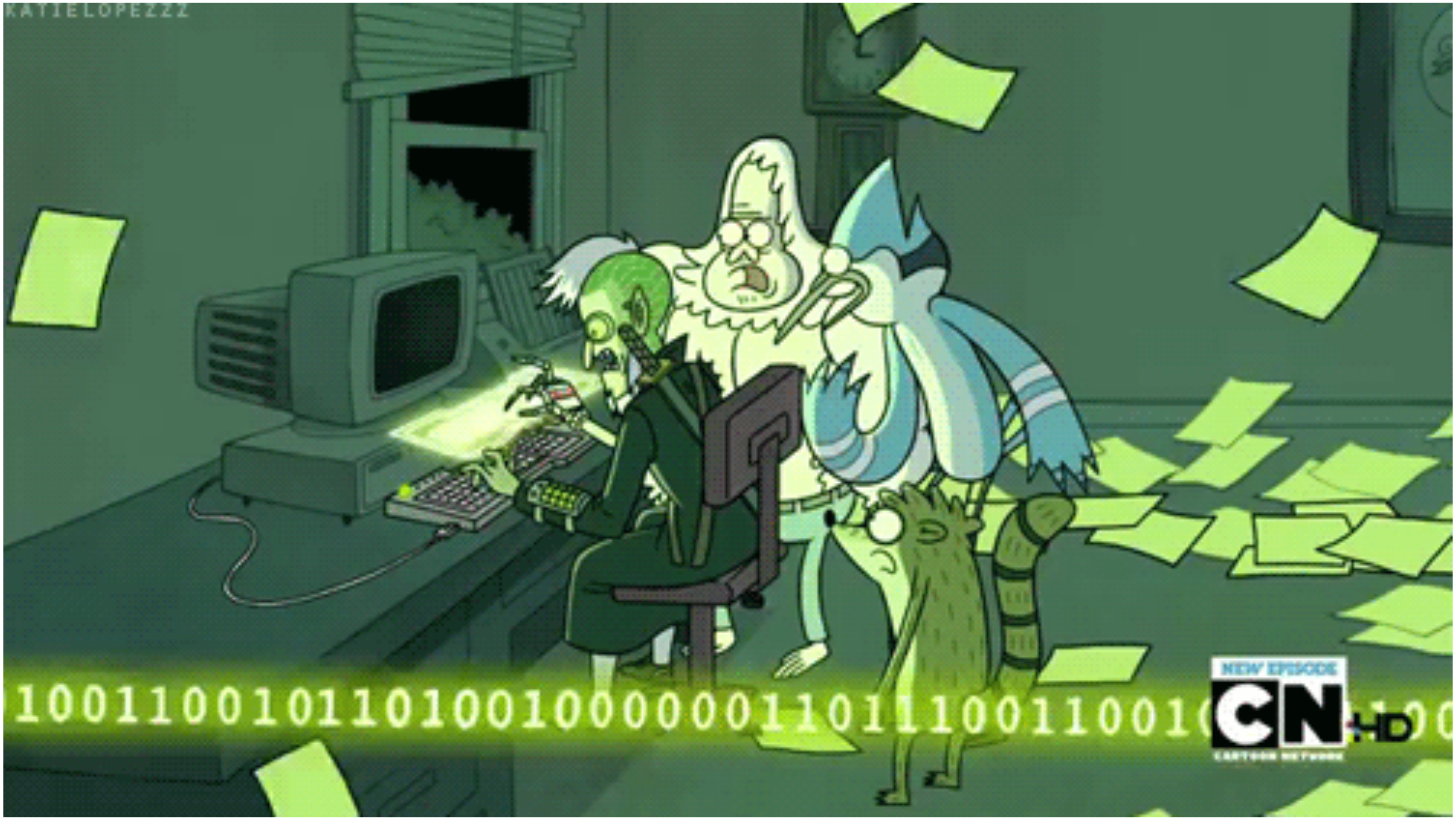
FIXING OUR TOOLS…

Issues Identified

- Functions are not all easily defined
- Functions do not retain their name when stripped
- String loads can be funky - dependent on architecture and Go version
- For above, we need to easily identify Go version!

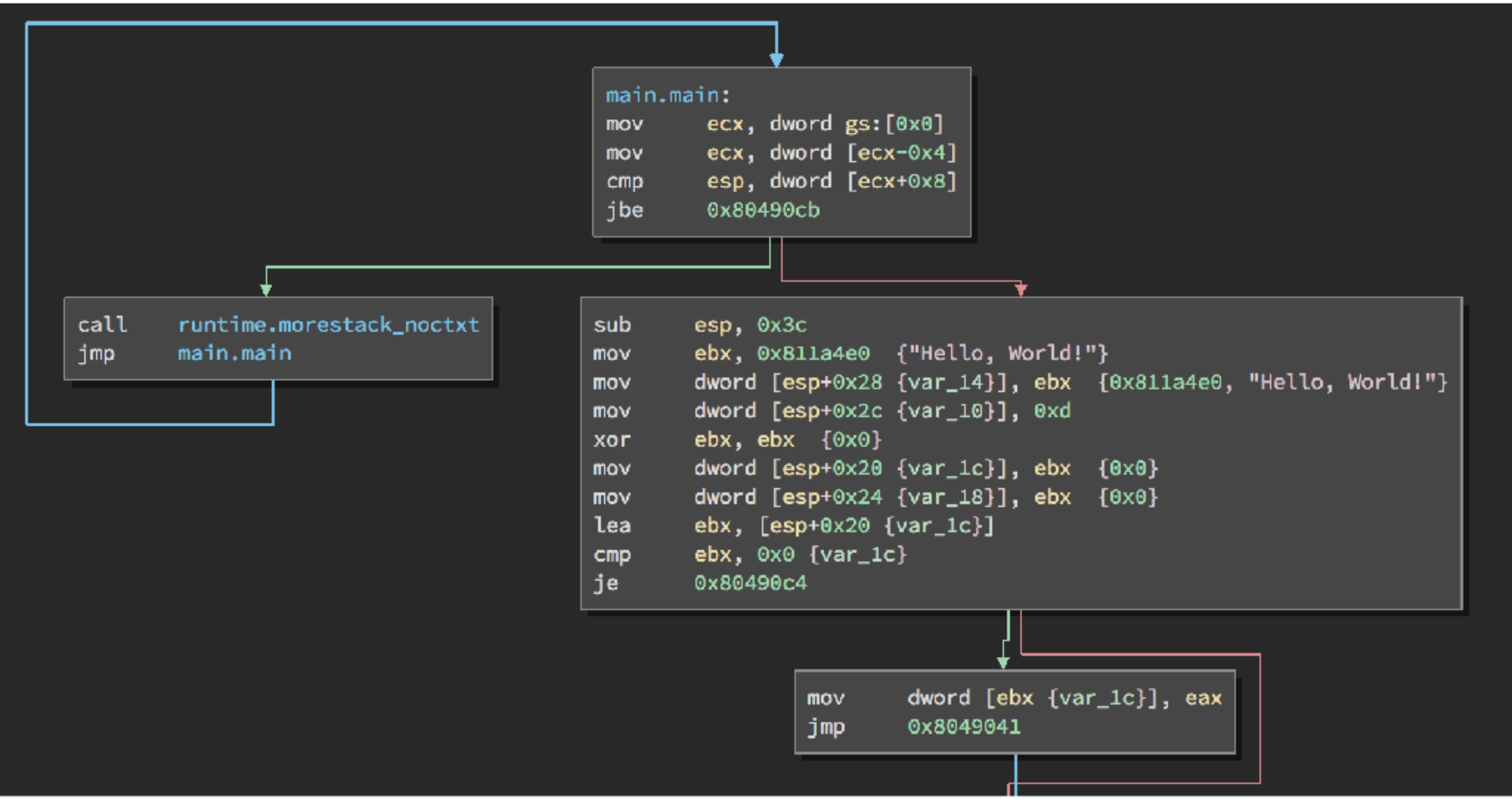
FIXING OUR TOOLS…

Issues Identified



TACKLING THE FUNCTIONS

How do functions normally look?



```
main.main:
mov     ecx, dword gs:[0x0]
mov     ecx, dword [ecx-0x4]
cmp     esp, dword [ecx+0x8]
jbe     0x80490cb
```

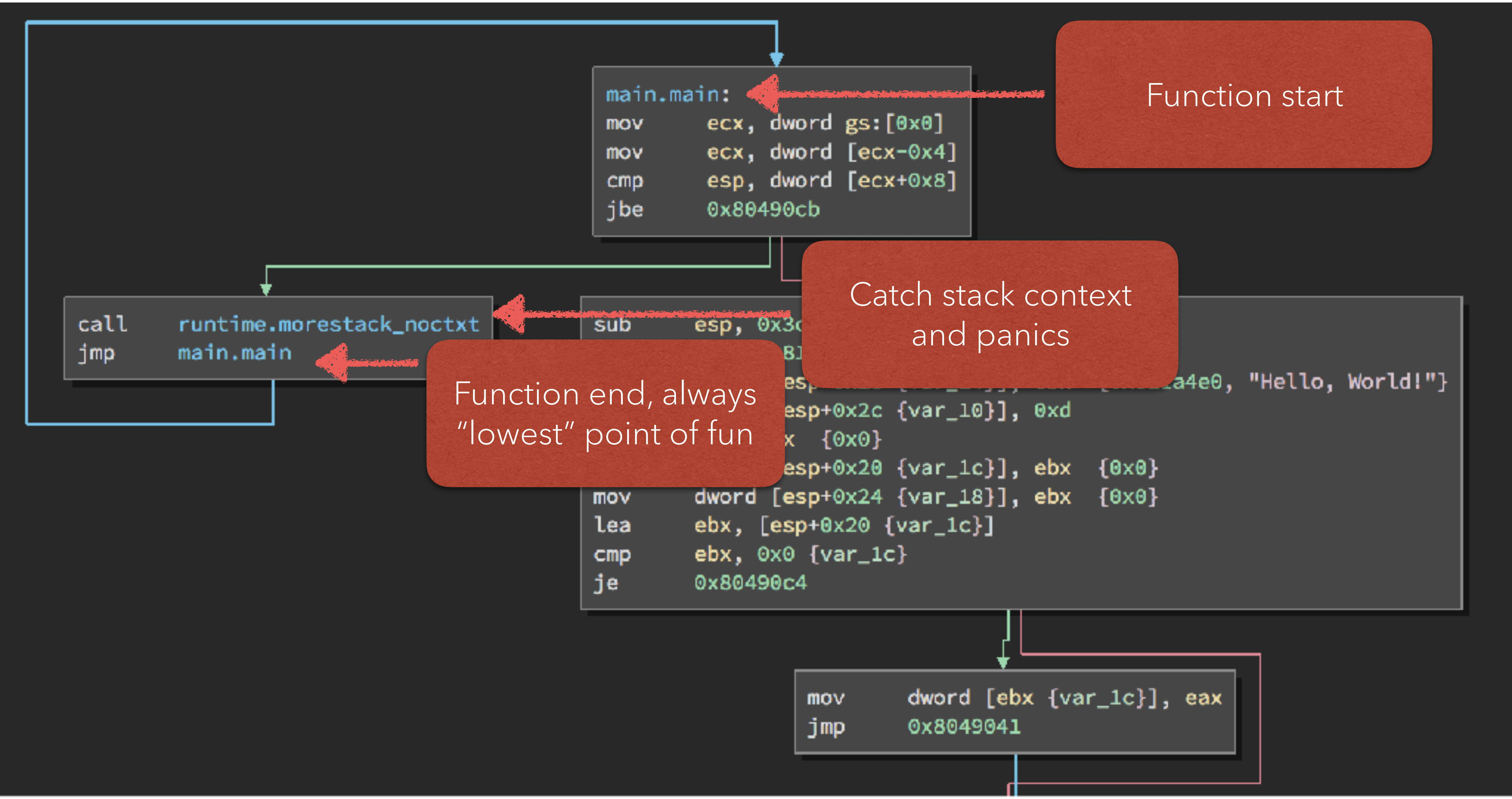
```
call    runtime.morestack_noctxt
jmp     main.main
```

```
sub     esp, 0x3c
mov     ebx, 0x811a4e0  {"Hello, World!"}
mov     dword [esp+0x28 {var_14}], ebx  {0x811a4e0, "Hello, World!"}
mov     dword [esp+0x2c {var_10}], 0xd
xor     ebx, ebx  {0x0}
mov     dword [esp+0x20 {var_1c}], ebx  {0x0}
mov     dword [esp+0x24 {var_18}], ebx  {0x0}
lea     ebx, [esp+0x20 {var_1c}]
cmp     ebx, 0x0 {var_1c}
je      0x80490c4
```

```
mov     dword [ebx {var_1c}], eax
jmp     0x8049041
```

TACKLING THE FUNCTIONS

How do functions normally look?


Function start
Catch stack context and panics
Function end, always "lowest" point of fun

```
main.main:
mov      ecx, dword gs:[0x0]
mov      ecx, dword [ecx-0x4]
cmp      esp, dword [ecx+0x8]
jbe      0x80490cb
```

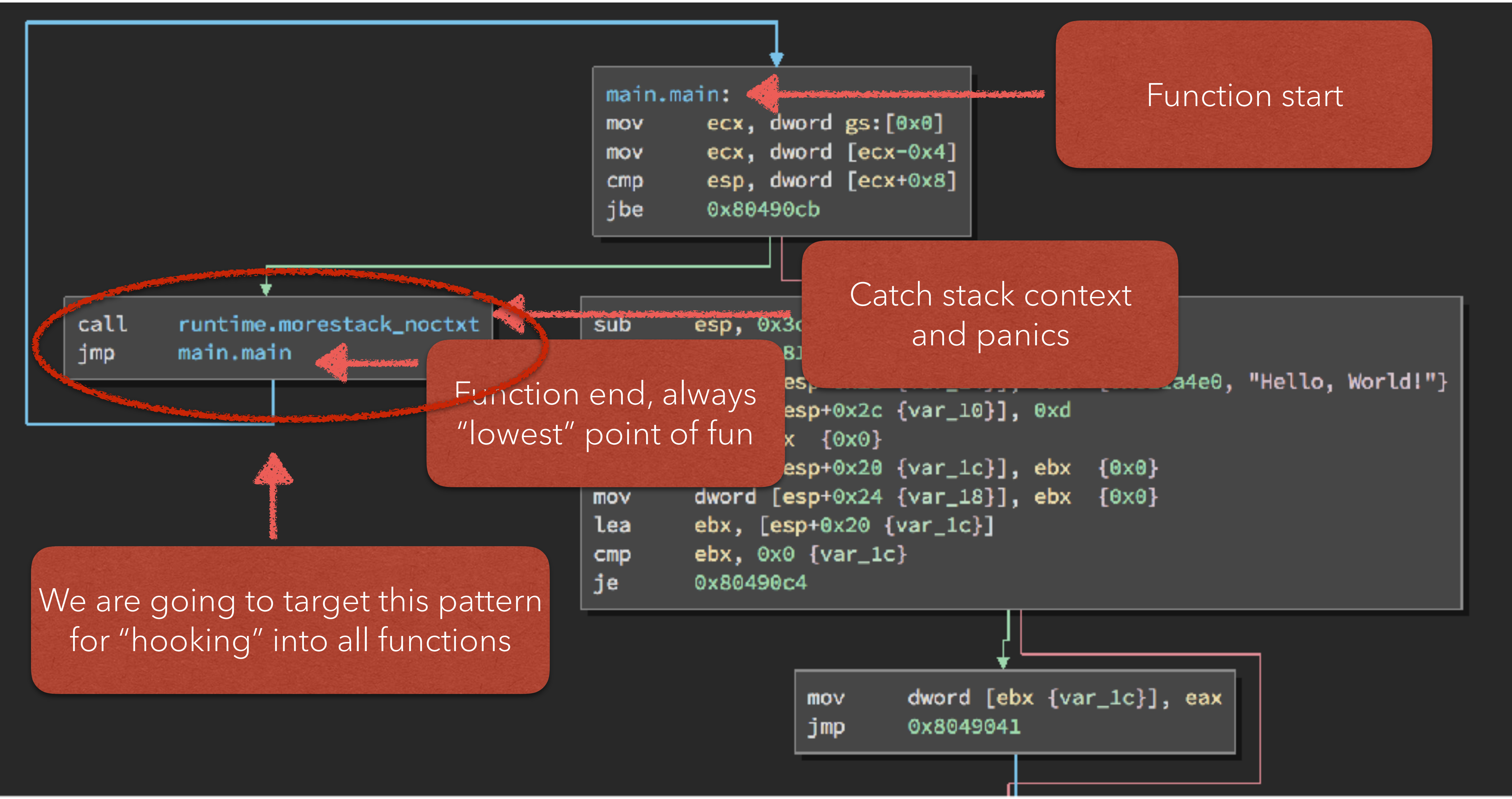
```
call     runtime.morestack_noctxt
jmp      main.main
```

```
sub      esp, 0x3c
                8]
         esp                                 a4e0, "Hello, World!"}
         esp+0x2c {var_10}], 0xd
         x   {0x0}
         esp+0x20 {var_1c}], ebx   {0x0}
mov      dword [esp+0x24 {var_18}], ebx   {0x0}
lea      ebx, [esp+0x20 {var_1c}]
cmp      ebx, 0x0 {var_1c}
je       0x80490c4
```

```
mov      dword [ebx {var_1c}], eax
jmp      0x8049041
```

TACKLING THE FUNCTIONS

How do functions normally look?


Function start
Catch stack context and panics
Function end, always "lowest" point of fun
We are going to target this pattern for "hooking" into all functions

```
main.main:
mov     ecx, dword gs:[0x0]
mov     ecx, dword [ecx-0x4]
cmp     esp, dword [ecx+0x8]
jbe     0x80490cb
```

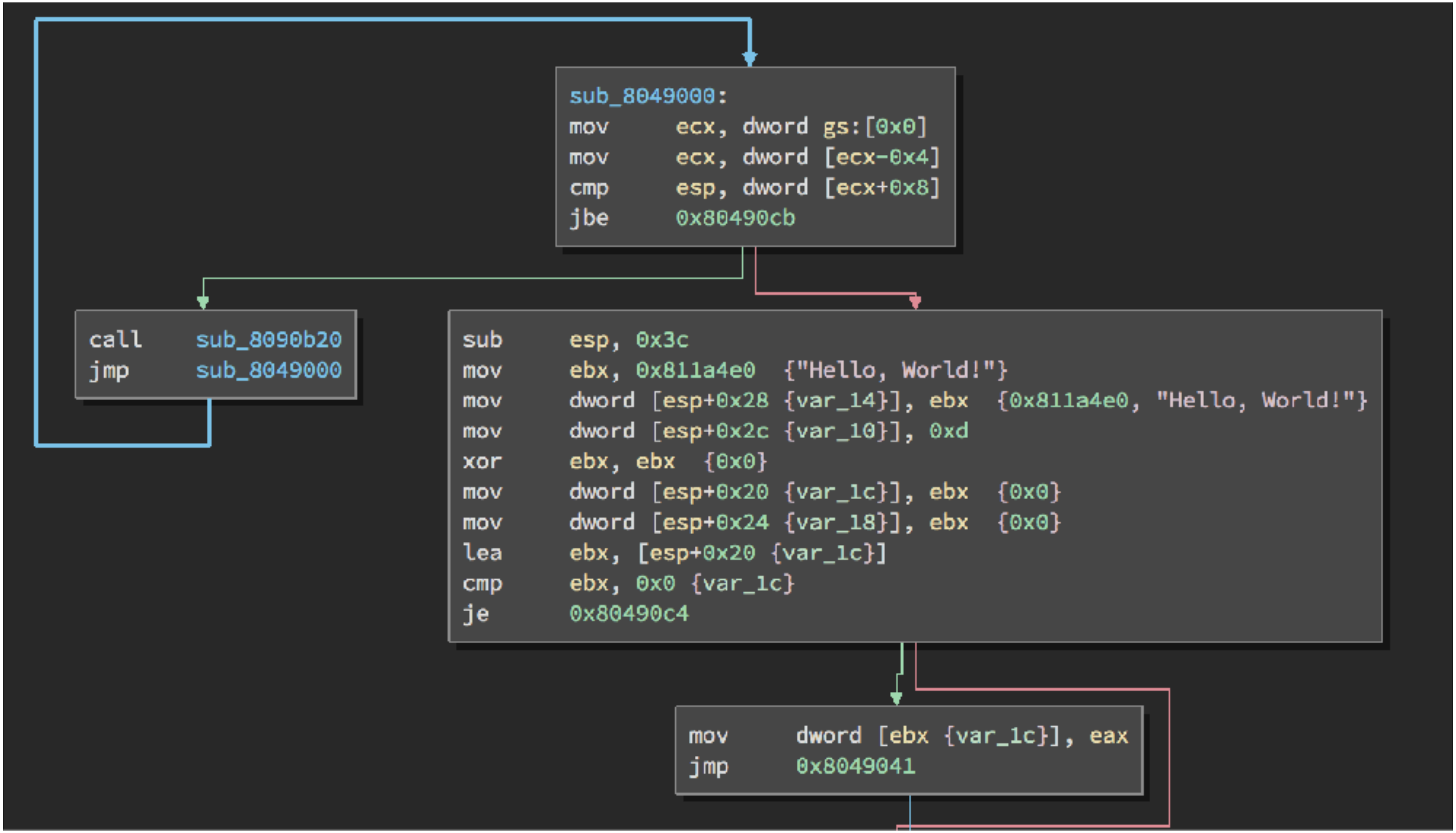
```
call    runtime.morestack_noctxt
jmp     main.main
```

```
sub     esp, 0x3c
                8]
                es                                      a4e0, "Hello, World!"}
                esp+0x2c {var_10}], 0xd
                x  {0x0}
                esp+0x20 {var_1c}], ebx  {0x0}
mov     dword [esp+0x24 {var_18}], ebx  {0x0}
lea     ebx, [esp+0x20 {var_1c}]
cmp     ebx, 0x0 {var_1c}
je      0x80490c4
```

```
mov     dword [ebx {var_1c}], eax
jmp     0x8049041
```

TACKLING THE FUNCTIONS

Same function, but binary stripped - hooray reproducible builds!



```
sub_8049000:
mov      ecx, dword gs:[0x0]
mov      ecx, dword [ecx-0x4]
cmp      esp, dword [ecx+0x8]
jbe      0x80490cb
```

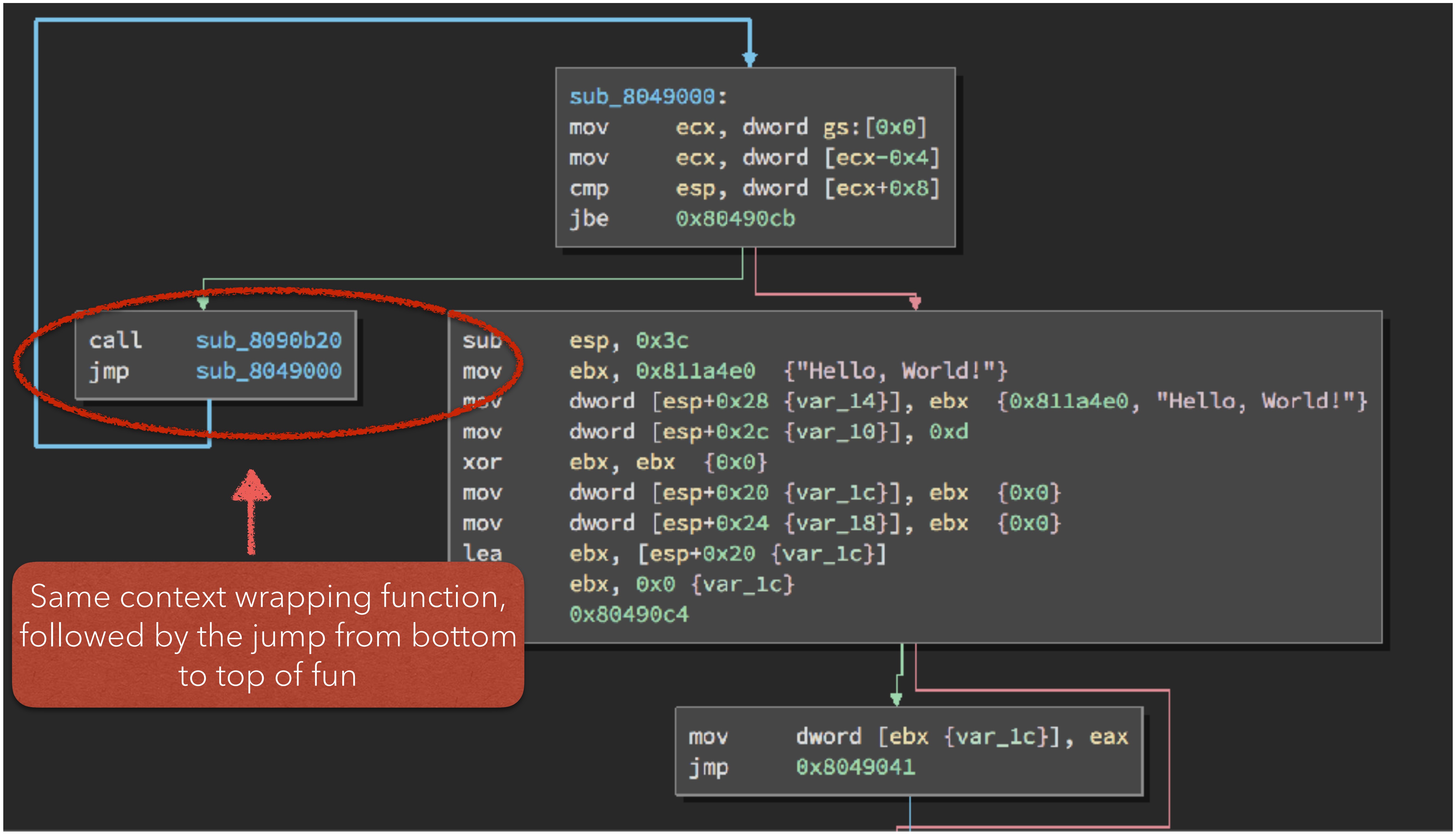
```
call     sub_8090b20
jmp      sub_8049000
```

```
sub      esp, 0x3c
mov      ebx, 0x811a4e0  {"Hello, World!"}
mov      dword [esp+0x28 {var_14}], ebx  {0x811a4e0, "Hello, World!"}
mov      dword [esp+0x2c {var_10}], 0xd
xor      ebx, ebx  {0x0}
mov      dword [esp+0x20 {var_1c}], ebx  {0x0}
mov      dword [esp+0x24 {var_18}], ebx  {0x0}
lea      ebx, [esp+0x20 {var_1c}]
cmp      ebx, 0x0 {var_1c}
je       0x80490c4
```

```
mov      dword [ebx {var_1c}], eax
jmp      0x8049041
```

TACKLING THE FUNCTIONS

Same function, but binary stripped - hooray reproducible builds!



```
sub_8049000:
mov     ecx, dword gs:[0x0]
mov     ecx, dword [ecx-0x4]
cmp     esp, dword [ecx+0x8]
jbe     0x80490cb
```

```
call    sub_8090b20
jmp     sub_8049000
```

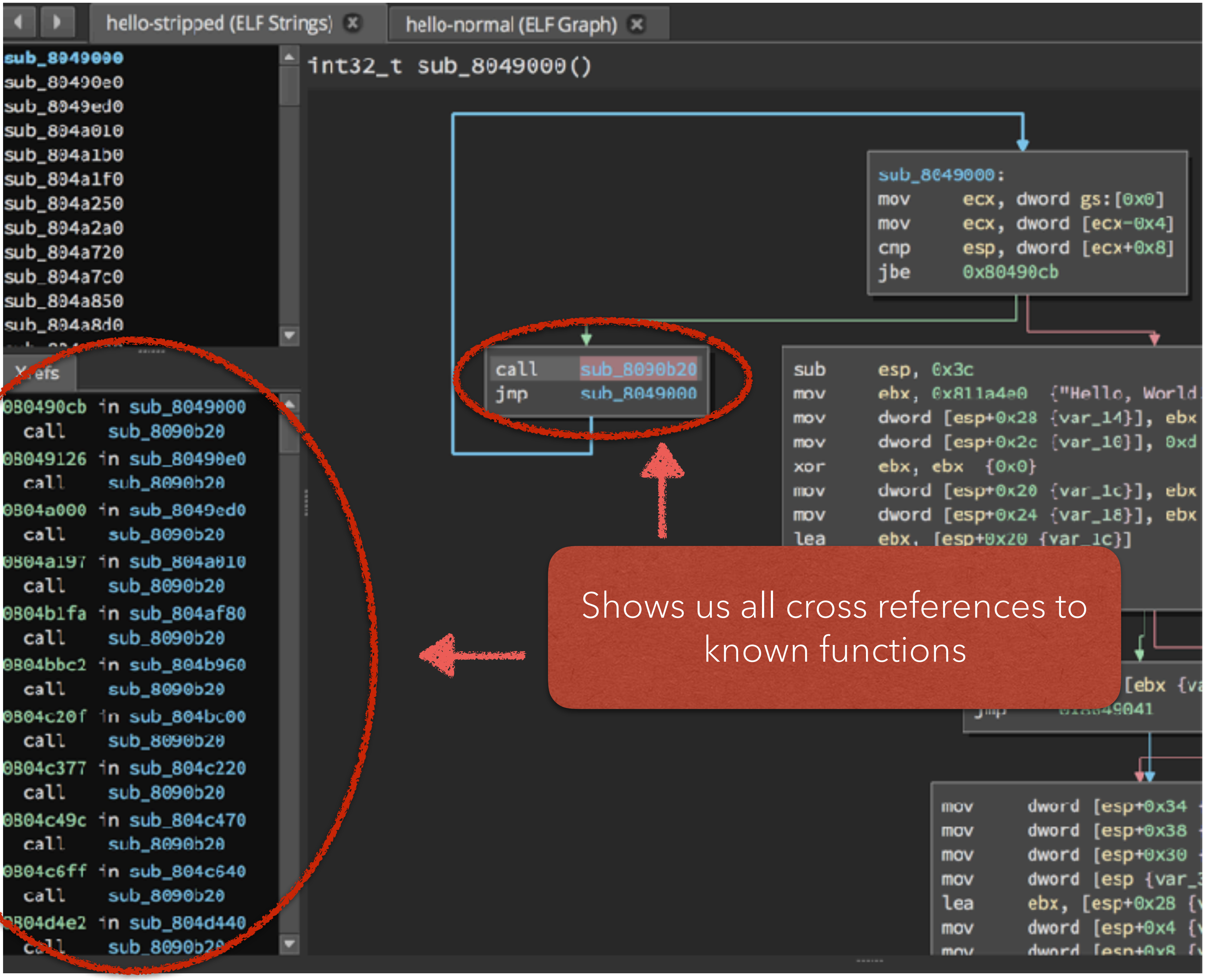
```
sub     esp, 0x3c
mov     ebx, 0x811a4e0  {"Hello, World!"}
mov     dword [esp+0x28 {var_14}], ebx  {0x811a4e0, "Hello, World!"}
mov     dword [esp+0x2c {var_10}], 0xd
xor     ebx, ebx  {0x0}
mov     dword [esp+0x20 {var_1c}], ebx  {0x0}
mov     dword [esp+0x24 {var_18}], ebx  {0x0}
lea     ebx, [esp+0x20 {var_1c}]
        ebx, 0x0 {var_1c}
        0x80490c4
```

```
mov     dword [ebx {var_1c}], eax
jmp     0x8049041
```

Same context wrapping function, followed by the jump from bottom to top of fun

TACKLING THE FUNCTIONS

Look for cross references



```
sub_8049000
sub_80490e0
sub_8049ed0
sub_804a010
sub_804a1b0
sub_804a1f0
sub_804a250
sub_804a2a0
sub_804a720
sub_804a7c0
sub_804a850
sub_804a8d0
```

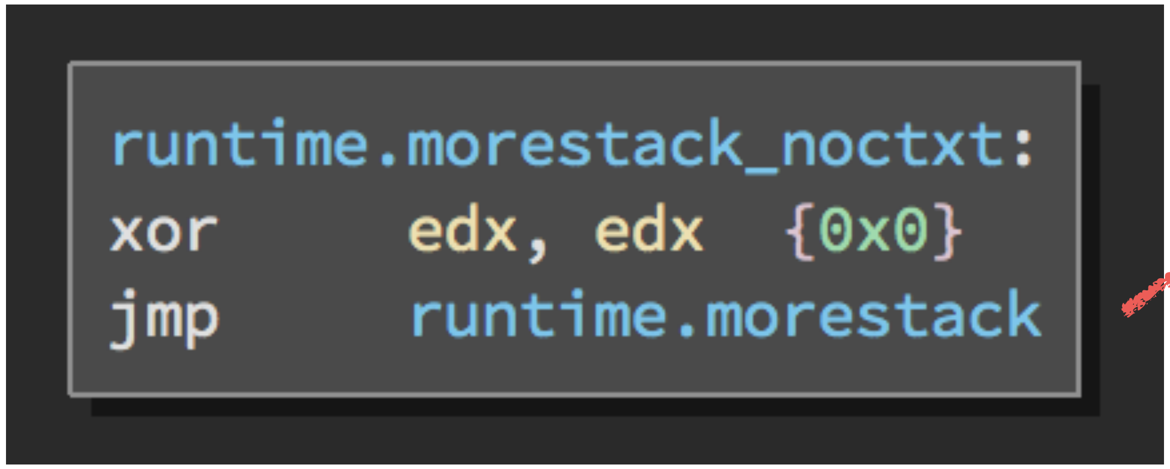
```
080490cb  in sub_8049000
  call    sub_8090b20
08049126  in sub_80490e0
  call    sub_8090b20
0804a000  in sub_8049ed0
  call    sub_8090b20
0804a197  in sub_804a010
  call    sub_8090b20
0804b1fa  in sub_804af80
  call    sub_8090b20
0804bbc2  in sub_804b960
  call    sub_8090b20
0804c20f  in sub_804bc00
  call    sub_8090b20
0804c377  in sub_804c220
  call    sub_8090b20
0804c49c  in sub_804c470
  call    sub_8090b20
0804c6ff  in sub_804c640
  call    sub_8090b20
0804d4e2  in sub_804d440
  call    sub_8090b20
```

```
int32_t sub_8049000()

sub_8049000:
mov     ecx, dword gs:[0x0]
mov     ecx, dword [ecx-0x4]
cmp     esp, dword [ecx+0x8]
jbe     0x80490cb

call    sub_8090b20
jmp     sub_8049000

sub     esp, 0x3c
mov     ebx, 0x811a4e0  {"Hello, World
mov     dword [esp+0x28 {var_14}], ebx
mov     dword [esp+0x2c {var_10}], 0xd
xor     ebx, ebx  {0x0}
mov     dword [esp+0x20 {var_1c}], ebx
mov     dword [esp+0x24 {var_18}], ebx
lea     ebx, [esp+0x20 {var_1c}]

        [ebx {va
jmp     0x8049041

mov     dword [esp+0x34
mov     dword [esp+0x38
mov     dword [esp+0x30
mov     dword [esp {var_3
lea     ebx, [esp+0x28 {v
mov     dword [esp+0x4 {v
mov     dword [esp+0x8 {v
```

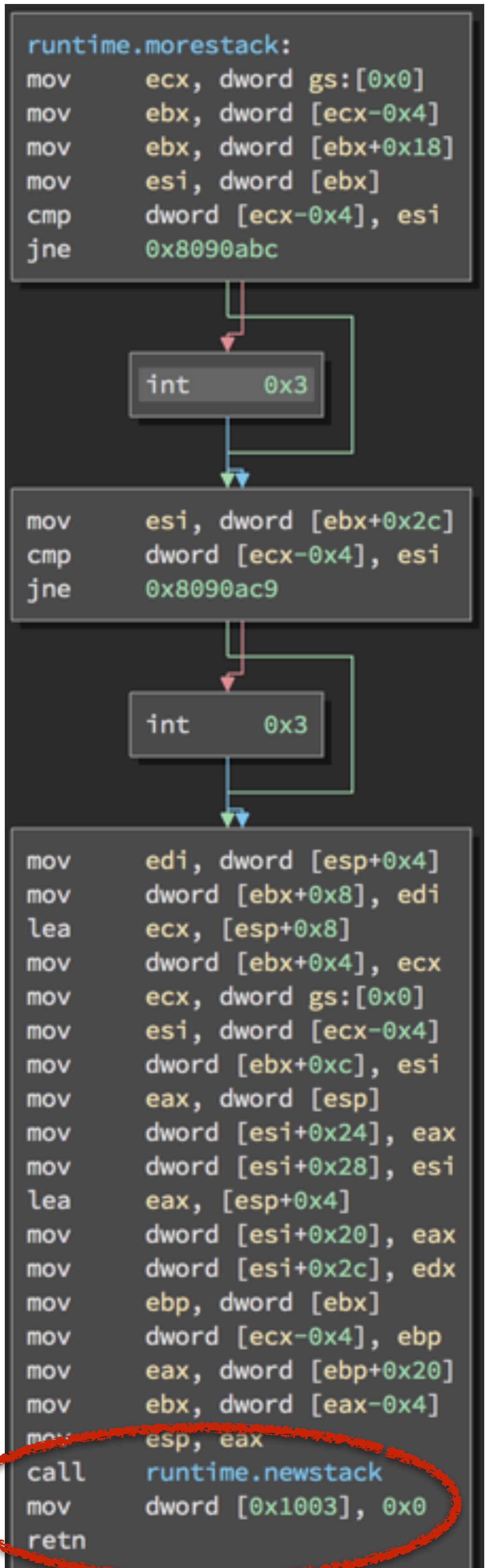
Shows us all cross references to known functions

TACKLING THE FUNCTIONS

Look for cross references



```
runtime.morestack_noctxt:
xor        edx, edx   {0x0}
jmp        runtime.morestack
```

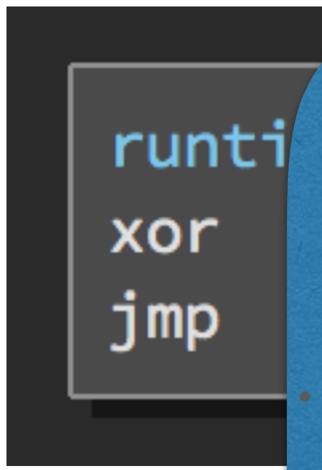


```
runtime.morestack:
mov        ecx, dword gs:[0x0]
mov        ebx, dword [ecx-0x4]
mov        ebx, dword [ebx+0x18]
mov        esi, dword [ebx]
cmp        dword [ecx-0x4], esi
jne        0x8090abc

int        0x3

mov        esi, dword [ebx+0x2c]
cmp        dword [ecx-0x4], esi
jne        0x8090ac9

int        0x3

mov        edi, dword [esp+0x4]
mov        dword [ebx+0x8], edi
lea        ecx, [esp+0x8]
mov        dword [ebx+0x4], ecx
mov        ecx, dword gs:[0x0]
mov        esi, dword [ecx-0x4]
mov        dword [ebx+0xc], esi
mov        eax, dword [esp]
mov        dword [esi+0x24], eax
mov        dword [esi+0x28], esi
lea        eax, [esp+0x4]
mov        dword [esi+0x20], eax
mov        dword [esi+0x2c], edx
mov        ebp, dword [ebx]
mov        dword [ecx-0x4], ebp
mov        eax, dword [ebp+0x20]
mov        ebx, dword [eax-0x4]
mov        esp, eax
call       runtime.newstack
mov        dword [0x1003], 0x0
retn
```

Unique feature of this relatively unique function

TACKLING THE FUNCTIONS

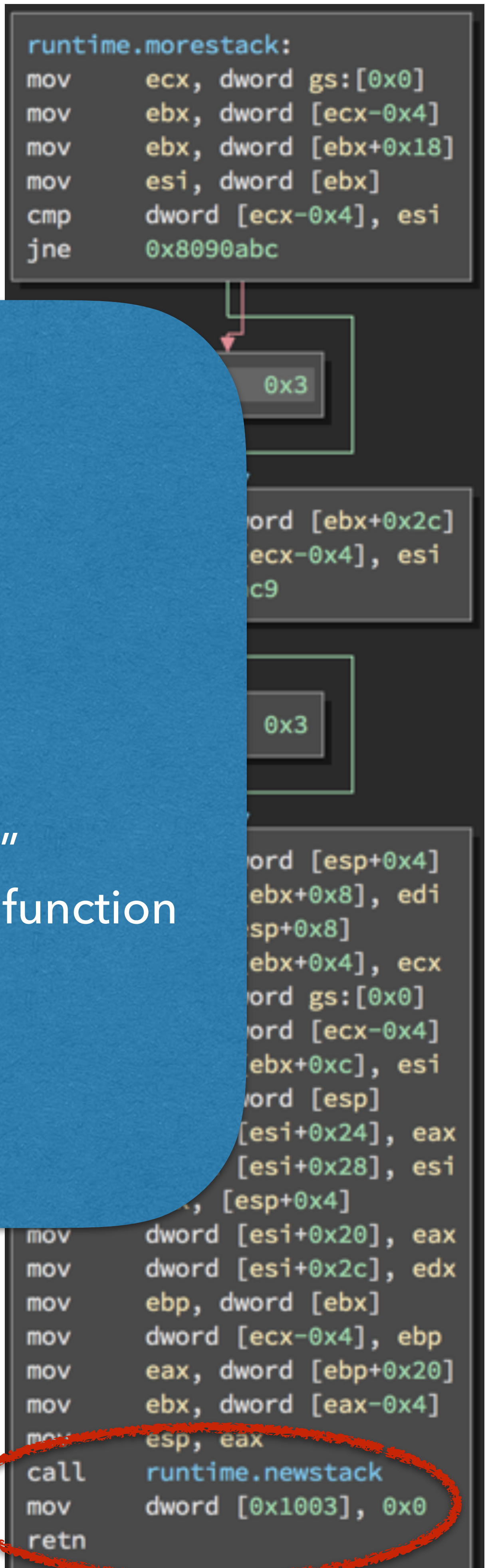
Look for cross references



```
runti
xor
jmp
```

Awesome - we've got a pseudo solution!

1. Find the runtime.newstack function
2. Recurse backwards into each function
3. The reference is always 1 line from the "bottom"
4. The bottom of the function is a jump to the "top of the function

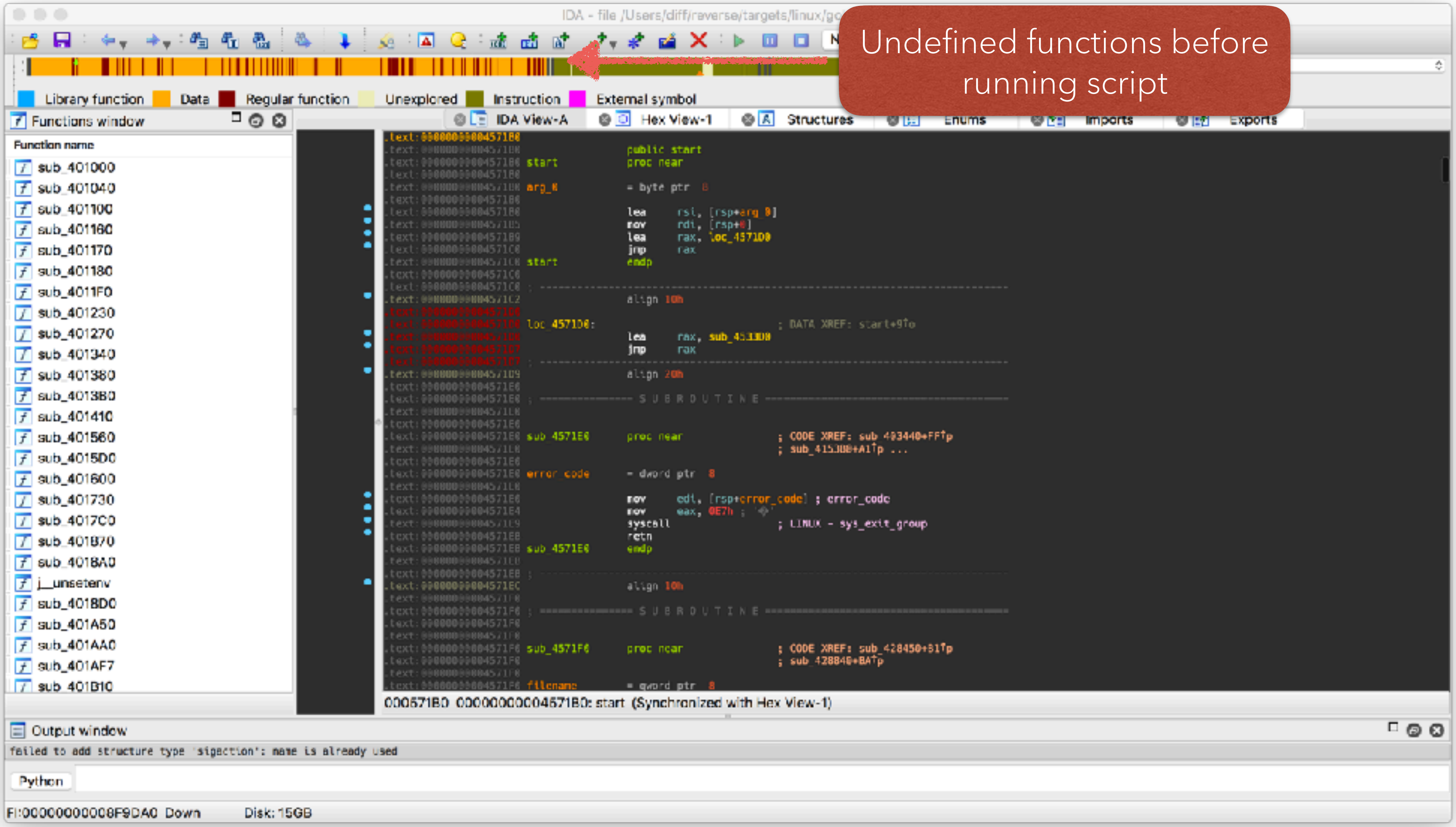


```
runtime.morestack:
mov     ecx, dword gs:[0x0]
mov     ebx, dword [ecx-0x4]
mov     ebx, dword [ebx+0x18]
mov     esi, dword [ebx]
cmp     dword [ecx-0x4], esi
jne     0x8090abc

        0x3

        ord [ebx+0x2c]
        ecx-0x4], esi
        c9

        0x3

        ord [esp+0x4]
        ebx+0x8], edi
        sp+0x8]
        ebx+0x4], ecx
        ord gs:[0x0]
        ord [ecx-0x4]
        ebx+0xc], esi
        ord [esp]
        [esi+0x24], eax
        [esi+0x28], esi
        , [esp+0x4]
mov     dword [esi+0x20], eax
mov     dword [esi+0x2c], edx
mov     ebp, dword [ebx]
mov     dword [ecx-0x4], ebp
mov     eax, dword [ebp+0x20]
mov     ebx, dword [eax-0x4]
mov     esp, eax
call    runtime.newstack
mov     dword [0x1003], 0x0
retn
```

Unique feature of this relatively unique function

TACKLING THE FUNCTIONS

Now we got things defined!



```
sub_401000
sub_401040
sub_401100
sub_401160
sub_401170
sub_401180
sub_4011F0
sub_401230
sub_401270
sub_401340
sub_401380
sub_4013B0
sub_401410
sub_401560
sub_4015D0
sub_401600
sub_401730
sub_4017C0
sub_401B70
sub_401BA0
j__unsetenv
sub_401BD0
sub_401A50
sub_401AA0
sub_401AF7
sub_401B10
```

```
.text:00000000004571B0
.text:00000000004571B0                  public start
.text:00000000004571B0 start            proc near
.text:00000000004571B0
.text:00000000004571B0 arg_0            = byte ptr  8
.text:00000000004571B0
.text:00000000004571B0                  lea     rsi, [rsp+arg_0]
.text:00000000004571B5                  mov     rdi, [rsp+0]
.text:00000000004571B9                  lea     rax, loc_4571D0
.text:00000000004571C0                  jmp     rax
.text:00000000004571C0 start            endp
.text:00000000004571C0
.text:00000000004571C0 ; ---------------------------------------------------------------
.text:00000000004571C2                  align 10h
.text:00000000004571D0
.text:00000000004571D0 loc_4571D0:                             ; DATA XREF: start+9↑o
.text:00000000004571D0                  lea     rax, sub_4533D0
.text:00000000004571D7                  jmp     rax
.text:00000000004571D7 ; ---------------------------------------------------------------
.text:00000000004571D9                  align 20h
.text:00000000004571E0
.text:00000000004571E0 ; =============== S U B R O U T I N E =======================================
.text:00000000004571E0
.text:00000000004571E0
.text:00000000004571E0 sub_4571E0       proc near               ; CODE XREF: sub_403440+FF↑p
.text:00000000004571E0                                          ; sub_4153D0+A1↑p ...
.text:00000000004571E0
.text:00000000004571E0 error_code       = dword ptr  8
.text:00000000004571E0
.text:00000000004571E0                  mov     edi, [rsp+error_code] ; error_code
.text:00000000004571E4                  mov     eax, 0E7h ; 'ç'
.text:00000000004571E9                  syscall                 ; LINUX - sys_exit_group
.text:00000000004571EB                  retn
.text:00000000004571EB sub_4571E0       endp
.text:00000000004571EB
.text:00000000004571EB ; ---------------------------------------------------------------
.text:00000000004571EC                  align 10h
.text:00000000004571F0
.text:00000000004571F0 ; =============== S U B R O U T I N E =======================================
.text:00000000004571F0
.text:00000000004571F0
.text:00000000004571F0 sub_4571F0       proc near               ; CODE XREF: sub_428450+31↑p
.text:00000000004571F0                                          ; sub_428840+BA↑p
.text:00000000004571F0
.text:00000000004571F0 filename         = qword ptr  8
```

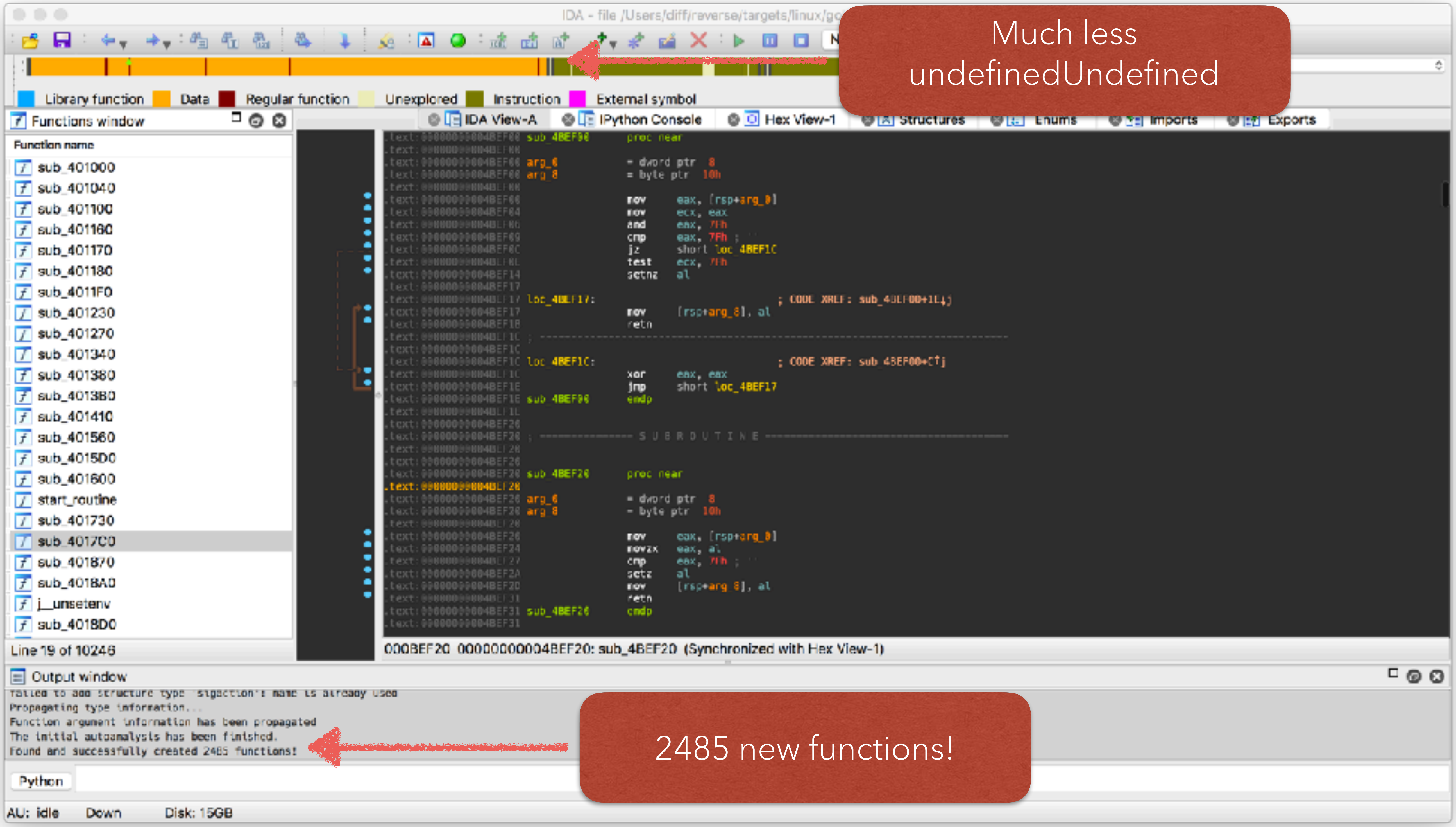
```
000571B0 00000000004571B0: start (Synchronized with Hex View-1)
```

```
failed to add structure type 'sigaction': name is already used
```

```
FI:00000000008F9DA0  Down      Disk: 15GB
```

TACKLING THE FUNCTIONS

Now we got things defined!



```
Function name
sub_401000
sub_401040
sub_401100
sub_401160
sub_401170
sub_401180
sub_4011F0
sub_401230
sub_401270
sub_401340
sub_401380
sub_4013B0
sub_401410
sub_401560
sub_4015D0
sub_401600
start_routine
sub_401730
sub_4017C0
sub_401870
sub_401BA0
j__unsetenv
sub_401BD0
Line 19 of 10246
```

```
.text:00000000004BEF00 sub_4BEF00      proc near
.text:00000000004BEF00
.text:00000000004BEF00 arg_0           = dword ptr  8
.text:00000000004BEF00 arg_8           = byte ptr  10h
.text:00000000004BEF00
.text:00000000004BEF00                 mov     eax, [rsp+arg_0]
.text:00000000004BEF04                 mov     ecx, eax
.text:00000000004BEF06                 and     eax, 7Fh
.text:00000000004BEF09                 cmp     eax, 7Fh ; ''
.text:00000000004BEF0C                 jz      short loc_4BEF1C
.text:00000000004BEF0E                 test    ecx, 7Fh
.text:00000000004BEF14                 setnz   al
.text:00000000004BEF17
.text:00000000004BEF17 loc_4BEF17:                             ; CODE XREF: sub_4BEF00+1E↓j
.text:00000000004BEF17                 mov     [rsp+arg_8], al
.text:00000000004BEF1B                 retn
.text:00000000004BEF1C ; ---------------------------------------------------------------------------
.text:00000000004BEF1C
.text:00000000004BEF1C loc_4BEF1C:                             ; CODE XREF: sub_4BEF00+C↑j
.text:00000000004BEF1C                 xor     eax, eax
.text:00000000004BEF1E                 jmp     short loc_4BEF17
.text:00000000004BEF1E sub_4BEF00      endp
.text:00000000004BEF1E
.text:00000000004BEF20
.text:00000000004BEF20 ; --------------- S U B R O U T I N E ---------------------------------------
.text:00000000004BEF20
.text:00000000004BEF20
.text:00000000004BEF20 sub_4BEF20      proc near
.text:00000000004BEF20
.text:00000000004BEF20 arg_0           = dword ptr  8
.text:00000000004BEF20 arg_8           = byte ptr  10h
.text:00000000004BEF20
.text:00000000004BEF20                 mov     eax, [rsp+arg_0]
.text:00000000004BEF24                 movzx   eax, al
.text:00000000004BEF27                 cmp     eax, 7Fh ; ''
.text:00000000004BEF2A                 setz    al
.text:00000000004BEF2D                 mov     [rsp+arg_8], al
.text:00000000004BEF31                 retn
.text:00000000004BEF31 sub_4BEF20      endp
.text:00000000004BEF31
000BEF20 00000000004BEF20: sub_4BEF20 (Synchronized with Hex View-1)
```

Output window

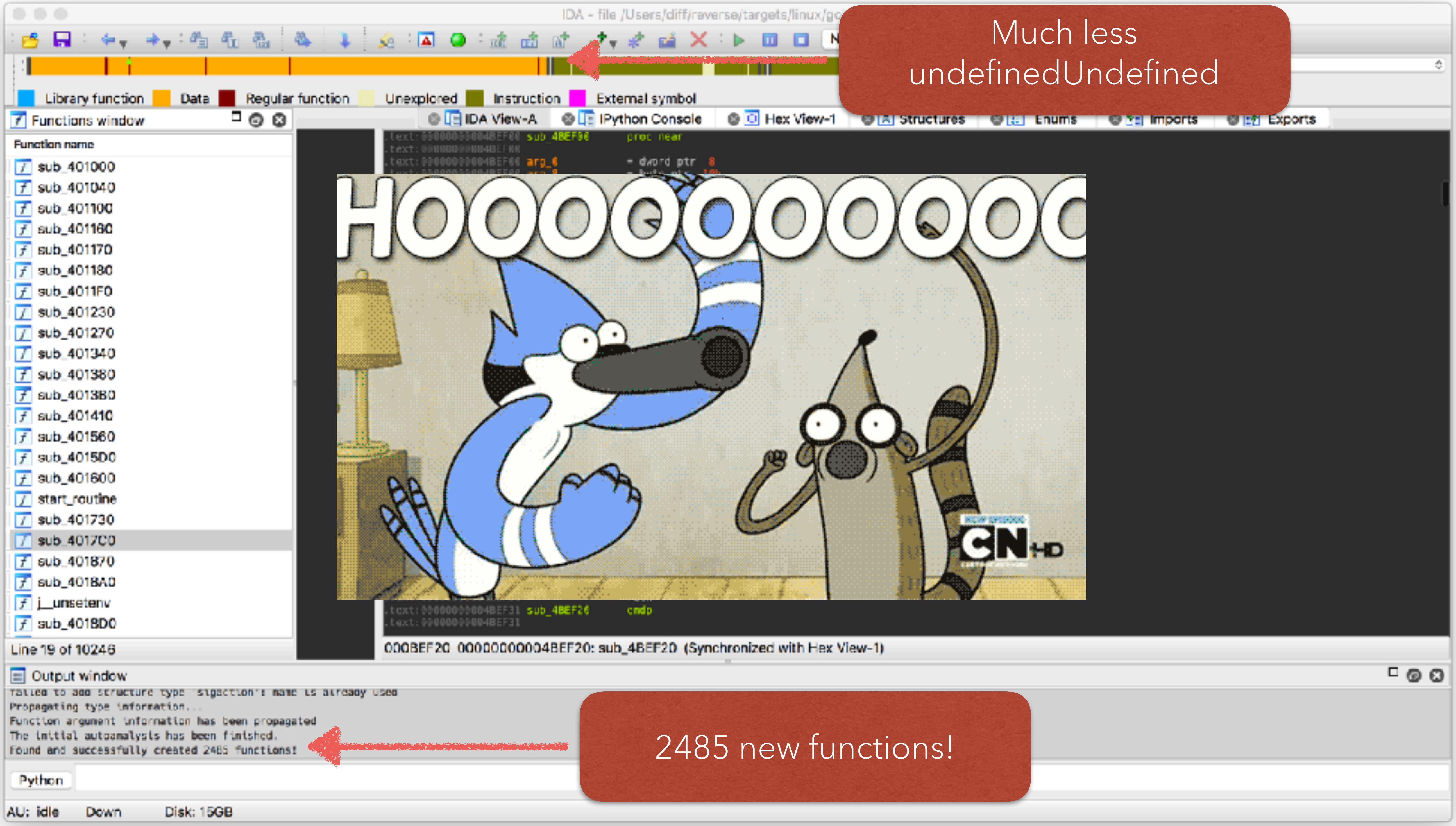
```
Failed to add structure type 'sigaction': name is already used
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
Found and successfully created 2485 functions!
```

Much less undefinedUndefined

2485 new functions!

TACKLING THE FUNCTIONS

Now we got things defined!



```
sub_401000
sub_401040
sub_401100
sub_401160
sub_401170
sub_401180
sub_4011F0
sub_401230
sub_401270
sub_401340
sub_401380
sub_4013B0
sub_401410
sub_401560
sub_4015D0
sub_401600
start_routine
sub_401730
sub_4017C0
sub_401870
sub_401BA0
j__unsetenv
sub_401BD0
Line 19 of 10246
```

```
.text:00000000004BEF00 sub_4BEF00      proc near
.text:00000000004BEF00
.text:00000000004BEF00 arg_0           = dword ptr  8
.text:00000000004BEF31 sub_4BEF26      cmdp
.text:00000000004BEF31
000BEF20 00000000004BEF20: sub_4BEF20 (Synchronized with Hex View-1)
```

```
Failed to add structure type 'sigaction': name is already used
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
Found and successfully created 2485 functions!
```

```
AU: idle     Down     Disk: 15GB
```

Much less undefinedUndefined

2485 new functions!

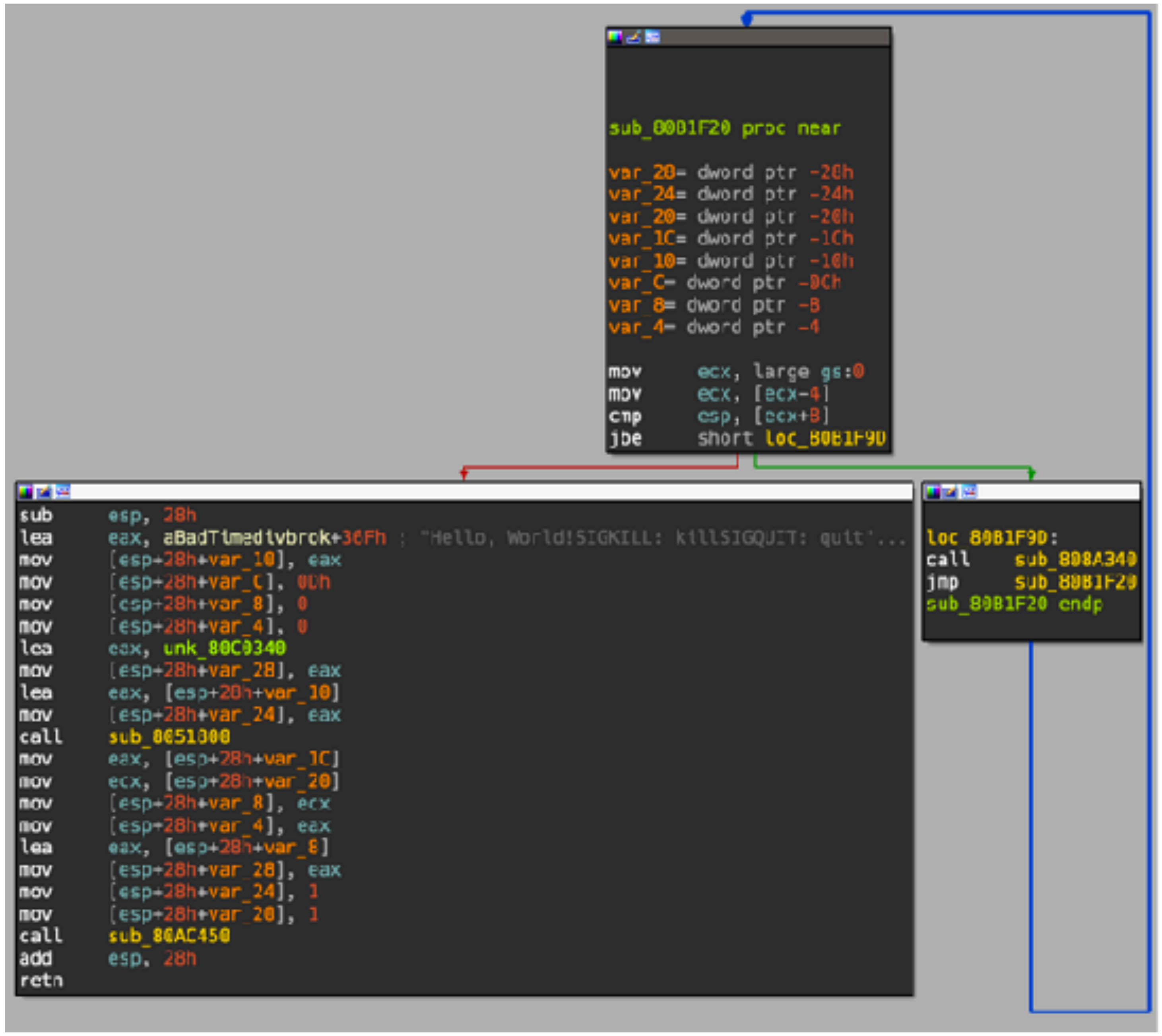
FIXING OUR TOOLS…

Issues Identified

- ~~Functions are not all easily defined~~
- Functions do not retain their name when stripped
- String loads can be funky - dependent on architecture and Go version
- For above, we need to easily identify Go version!

FUNCTION NAMES RECOVERABLE?

How do functions normally look?

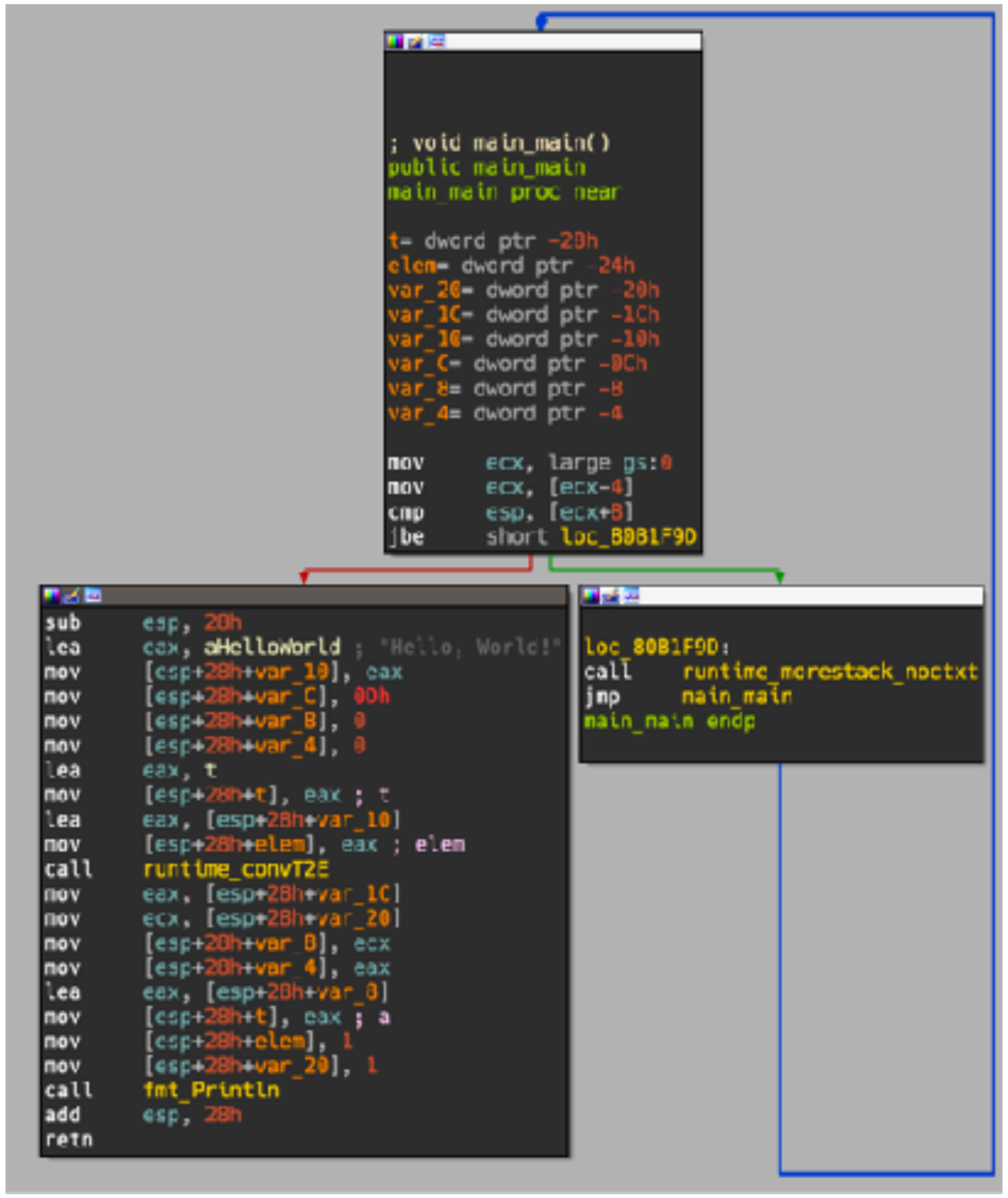


```
sub_8081F20 proc near

var_28= dword ptr -28h
var_24= dword ptr -24h
var_20= dword ptr -20h
var_1C= dword ptr -1Ch
var_10= dword ptr -10h
var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= dword ptr -4

mov     ecx, large gs:0
mov     ecx, [ecx-4]
cmp     esp, [ecx+8]
jbe     short loc_8081F9D
```

```
sub     esp, 28h
lea     eax, aBadTimedivbrck+36Fh ; "Hello, World!SIGKILL: killSIGQUIT: quit'...
mov     [esp+28h+var_10], eax
mov     [esp+28h+var_C], 0Dh
mov     [esp+28h+var_8], 0
mov     [esp+28h+var_4], 0
lea     eax, unk_80C0340
mov     [esp+28h+var_28], eax
lea     eax, [esp+28h+var_10]
mov     [esp+28h+var_24], eax
call    sub_8051000
mov     eax, [esp+28h+var_1C]
mov     ecx, [esp+28h+var_20]
mov     [esp+28h+var_8], ecx
mov     [esp+28h+var_4], eax
lea     eax, [esp+28h+var_8]
mov     [esp+28h+var_28], eax
mov     [esp+28h+var_24], 1
mov     [esp+28h+var_20], 1
call    sub_80AC450
add     esp, 28h
retn
```

```
loc_8081F9D:
call    sub_808A340
jmp     sub_8081F20
sub_8081F20 endp
```

Stripped binary



```
; void main_main()
public main_main
main_main proc near

t= dword ptr -28h
elem= dword ptr -24h
var_20= dword ptr -20h
var_1C= dword ptr -1Ch
var_10= dword ptr -10h
var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= dword ptr -4

mov     ecx, large gs:0
mov     ecx, [ecx-4]
cmp     esp, [ecx+8]
jbe     short loc_8081F9D
```

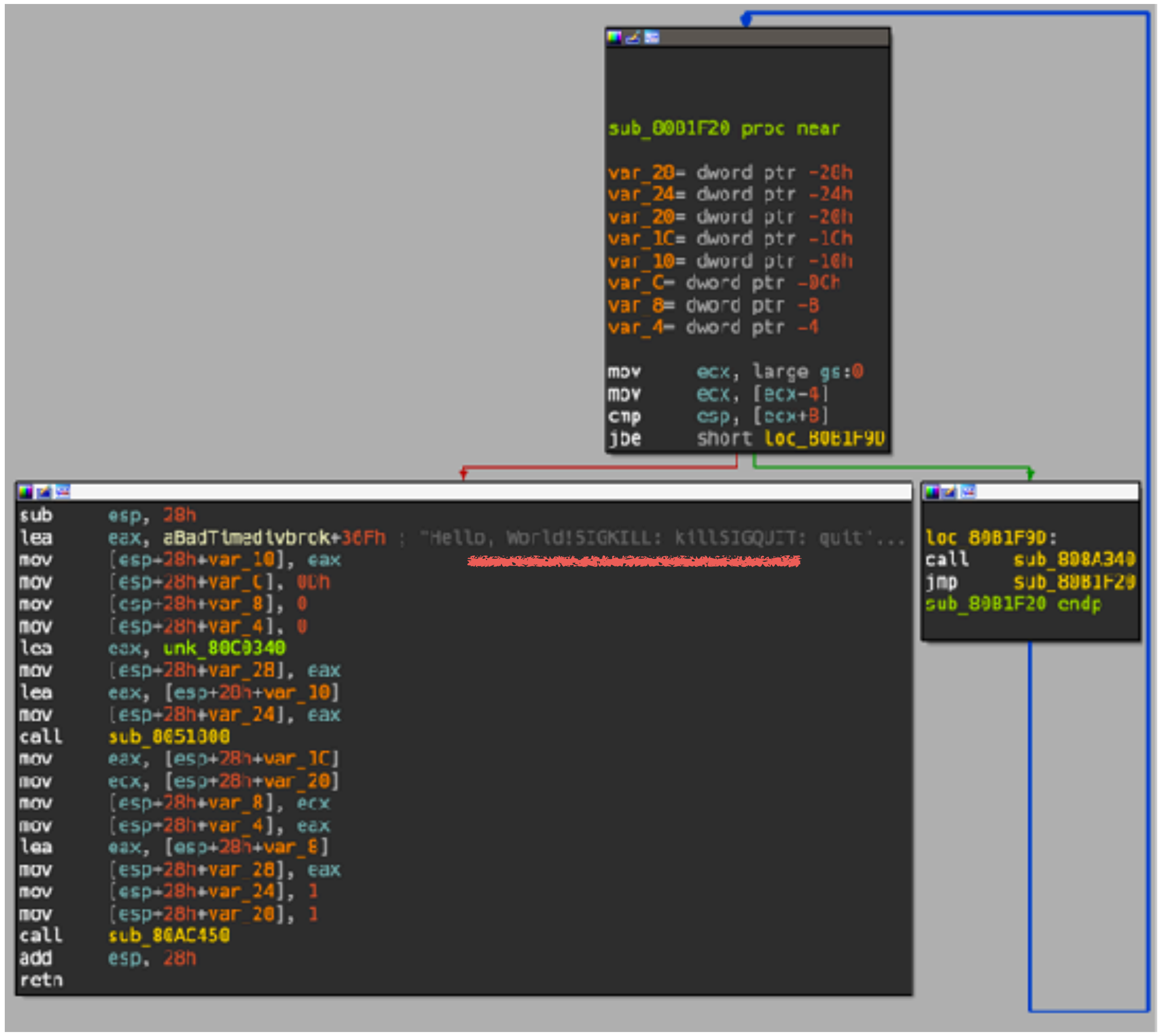
```
sub     esp, 28h
lea     eax, aHelloWorld ; "Hello, World!"
mov     [esp+28h+var_10], eax
mov     [esp+28h+var_C], 0Dh
mov     [esp+28h+var_8], 0
mov     [esp+28h+var_4], 0
lea     eax, t
mov     [esp+28h+t], eax ; t
lea     eax, [esp+28h+var_10]
mov     [esp+28h+elem], eax ; elem
call    runtime_convT2E
mov     eax, [esp+28h+var_1C]
mov     ecx, [esp+28h+var_20]
mov     [esp+28h+var_8], ecx
mov     [esp+28h+var_4], eax
lea     eax, [esp+28h+var_8]
mov     [esp+28h+t], eax ; a
mov     [esp+28h+elem], 1
mov     [esp+28h+var_20], 1
call    fmt_Println
add     esp, 28h
retn
```

```
loc_8081F9D:
call    runtime_morestack_noctxt
jmp     main_main
main_main endp
```

Normal binary

FUNCTION NAMES RECOVERABLE?

How do functions normally look?

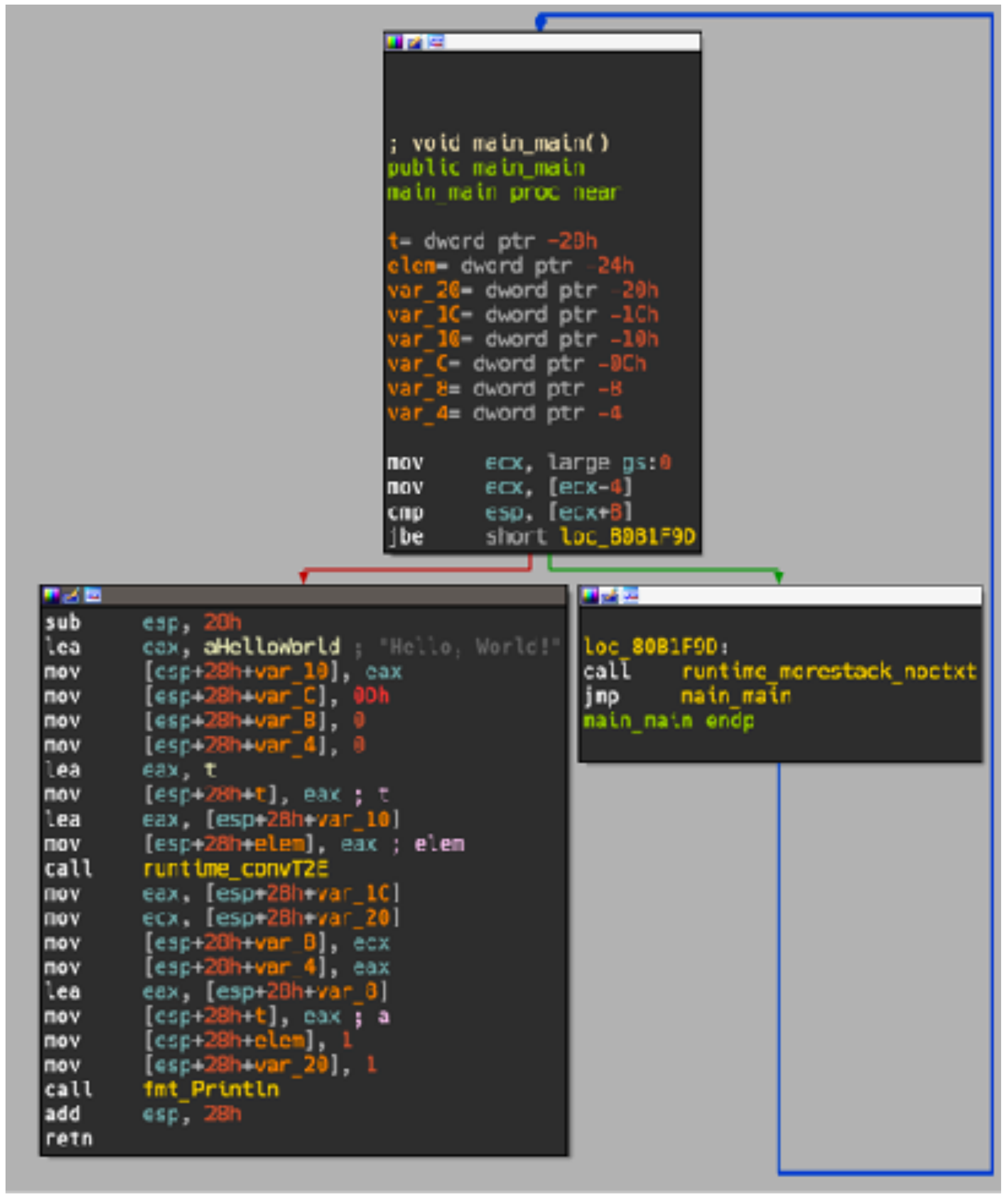


```
sub_80B1F20 proc near

var_28= dword ptr -28h
var_24= dword ptr -24h
var_20= dword ptr -20h
var_1C= dword ptr -1Ch
var_10= dword ptr -10h
var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= dword ptr -4

mov     ecx, large gs:0
mov     ecx, [ecx-4]
cmp     esp, [ecx+8]
jbe     short loc_80B1F9D
```

```
sub     esp, 28h
lea     eax, aBadTimedivbrck+36Fh ; "Hello, World!SIGKILL: killSIGQUIT: quit"...
mov     [esp+28h+var_10], eax
mov     [esp+28h+var_C], 0Dh
mov     [esp+28h+var_8], 0
mov     [esp+28h+var_4], 0
lea     eax, unk_80C0340
mov     [esp+28h+var_28], eax
lea     eax, [esp+28h+var_10]
mov     [esp+28h+var_24], eax
call    sub_8051000
mov     eax, [esp+28h+var_1C]
mov     ecx, [esp+28h+var_20]
mov     [esp+28h+var_8], ecx
mov     [esp+28h+var_4], eax
lea     eax, [esp+28h+var_8]
mov     [esp+28h+var_28], eax
mov     [esp+28h+var_24], 1
mov     [esp+28h+var_20], 1
call    sub_80AC450
add     esp, 28h
retn
```

```
loc_80B1F9D:
call    sub_808A340
jmp     sub_80B1F20
sub_80B1F20 endp
```

Stripped binary



```
; void main_main()
public main_main
main_main proc near

t= dword ptr -28h
elem= dword ptr -24h
var_20= dword ptr -20h
var_1C= dword ptr -1Ch
var_10= dword ptr -10h
var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= dword ptr -4

mov     ecx, large gs:0
mov     ecx, [ecx-4]
cmp     esp, [ecx+8]
jbe     short loc_80B1F9D
```

```
sub     esp, 28h
lea     eax, aHelloWorld ; "Hello, World!"
mov     [esp+28h+var_10], eax
mov     [esp+28h+var_C], 0Dh
mov     [esp+28h+var_8], 0
mov     [esp+28h+var_4], 0
lea     eax, t
mov     [esp+28h+t], eax ; t
lea     eax, [esp+28h+var_10]
mov     [esp+28h+elem], eax ; elem
call    runtime_convT2E
mov     eax, [esp+28h+var_1C]
mov     ecx, [esp+28h+var_20]
mov     [esp+28h+var_8], ecx
mov     [esp+28h+var_4], eax
lea     eax, [esp+28h+var_8]
mov     [esp+28h+t], eax ; a
mov     [esp+28h+elem], 1
mov     [esp+28h+var_20], 1
call    fmt_Println
add     esp, 28h
retn
```

```
loc_80B1F9D:
call    runtime_morestack_noctxt
jmp     main_main
main_main endp
```

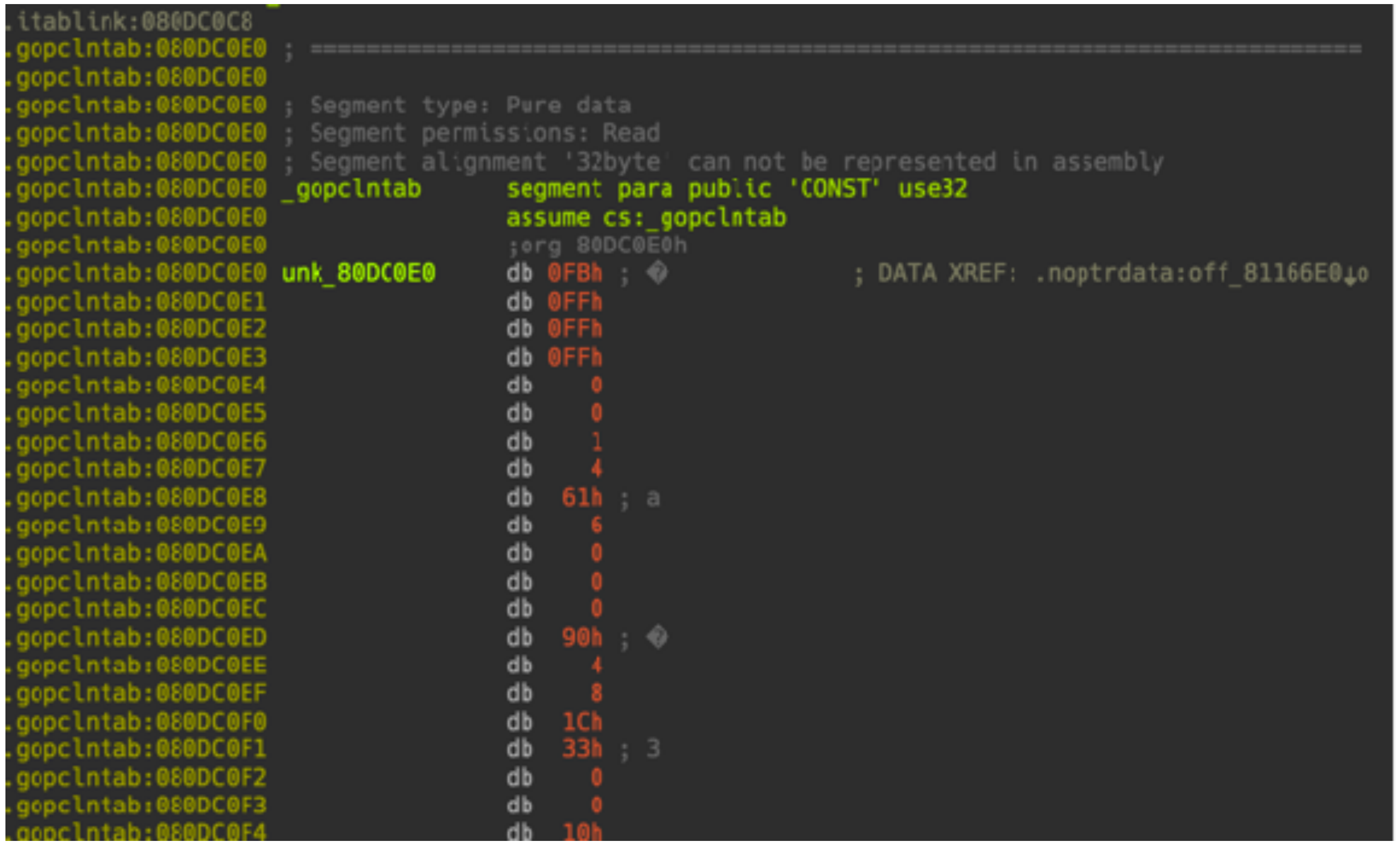
Normal binary

FUNCTION NAMES RECOVERABLE?

Enter .gopclntab!

- Oh how nice!
  A go specific segment!

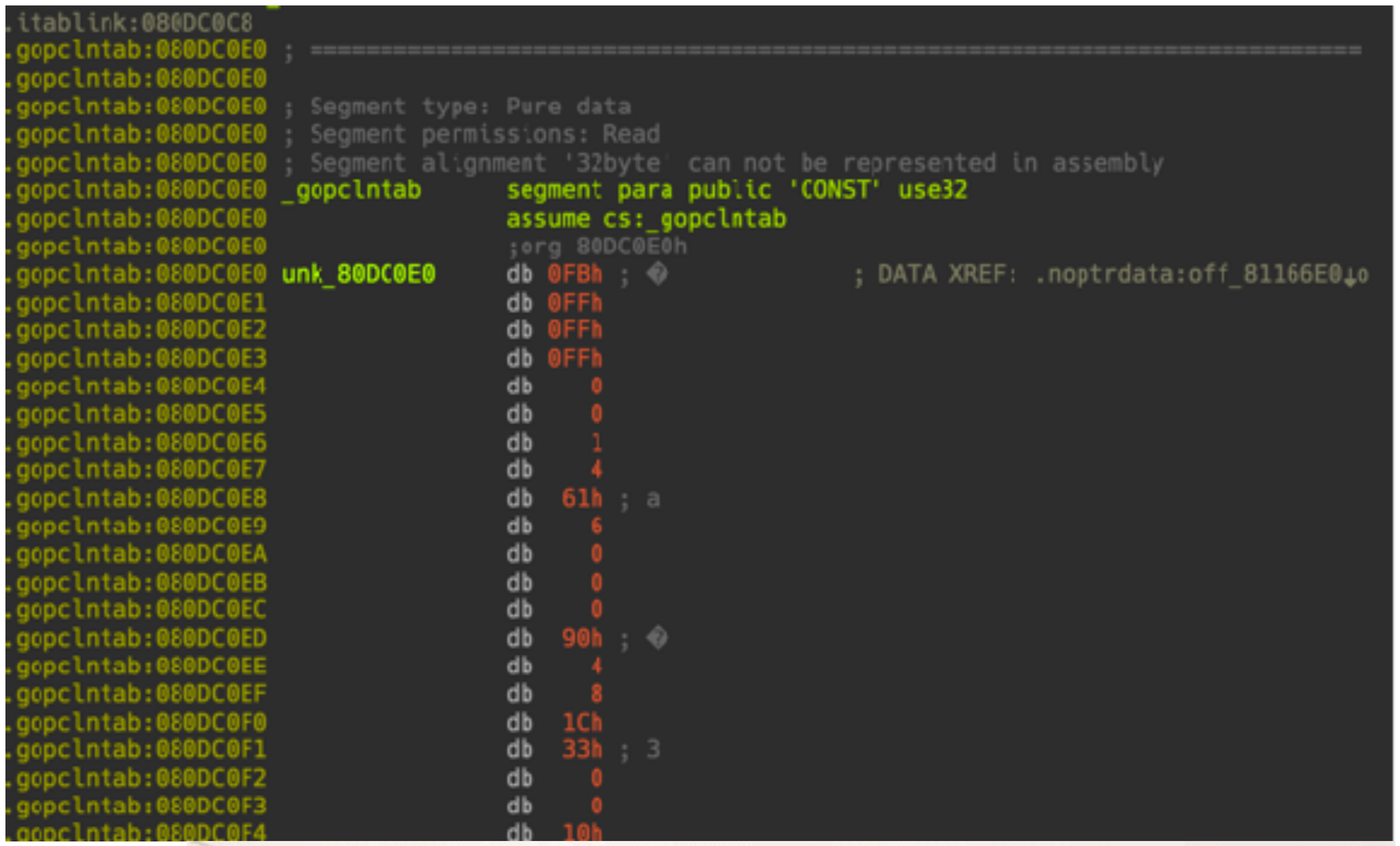
- Dig into the source and figure out the structure



```
.itablink:080DC0C8
.gopclntab:080DC0E0 ; ===========================================================================
.gopclntab:080DC0E0
.gopclntab:080DC0E0 ; Segment type: Pure data
.gopclntab:080DC0E0 ; Segment permissions: Read
.gopclntab:080DC0E0 ; Segment alignment '32byte' can not be represented in assembly
.gopclntab:080DC0E0 _gopclntab      segment para public 'CONST' use32
.gopclntab:080DC0E0                 assume cs:_gopclntab
.gopclntab:080DC0E0                 ;org 80DC0E0h
.gopclntab:080DC0E0 unk_80DC0E0     db 0FBh ; �               ; DATA XREF: .noptrdata:off_81166E0↓o
.gopclntab:080DC0E1                 db 0FFh
.gopclntab:080DC0E2                 db 0FFh
.gopclntab:080DC0E3                 db 0FFh
.gopclntab:080DC0E4                 db    0
.gopclntab:080DC0E5                 db    0
.gopclntab:080DC0E6                 db    1
.gopclntab:080DC0E7                 db    4
.gopclntab:080DC0E8                 db  61h ; a
.gopclntab:080DC0E9                 db    6
.gopclntab:080DC0EA                 db    0
.gopclntab:080DC0EB                 db    0
.gopclntab:080DC0EC                 db    0
.gopclntab:080DC0ED                 db  90h ; �
.gopclntab:080DC0EE                 db    4
.gopclntab:080DC0EF                 db    8
.gopclntab:080DC0F0                 db  1Ch
.gopclntab:080DC0F1                 db  33h ; 3
.gopclntab:080DC0F2                 db    0
.gopclntab:080DC0F3                 db    0
.gopclntab:080DC0F4                 db  10h
```

FUNCTION NAMES RECOVERABLE?

Enter .gopclntab!

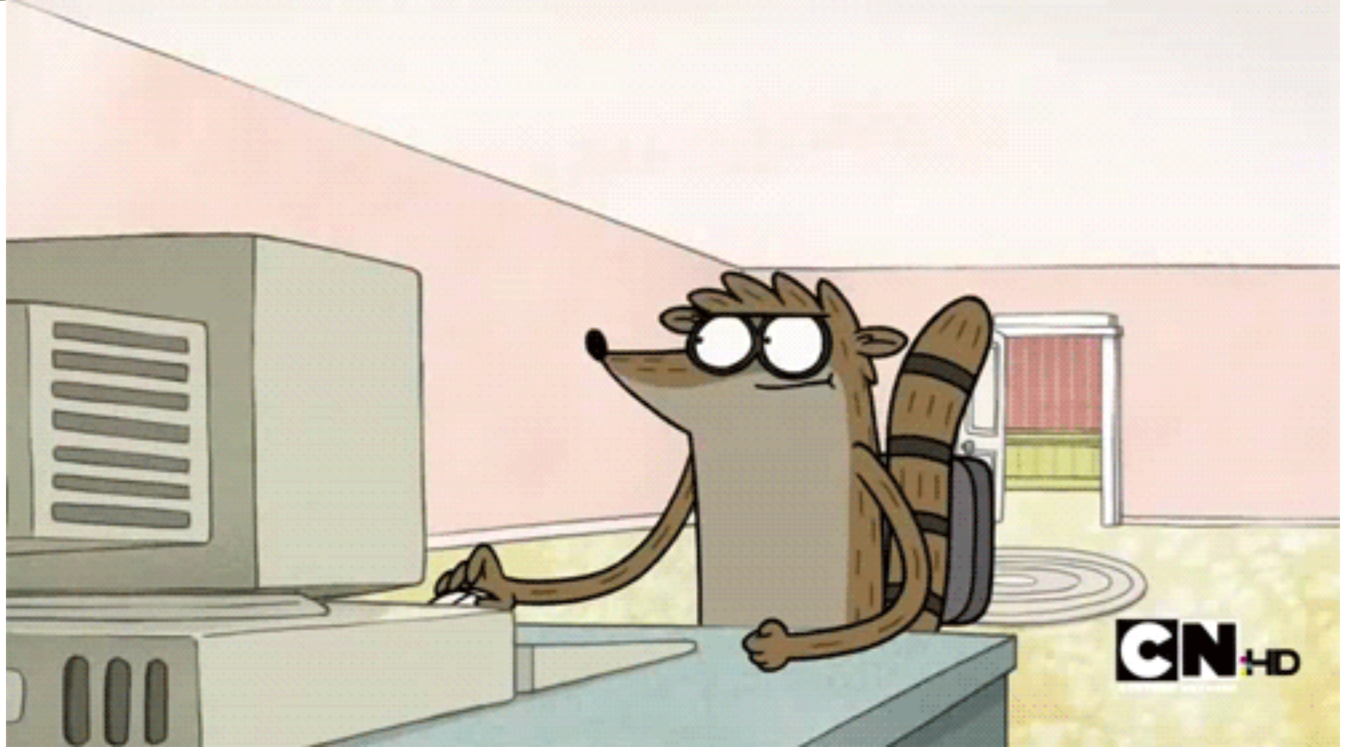


```
.itablink:080DC0C8
.gopclntab:080DC0E0 ; ===========================================================================
.gopclntab:080DC0E0
.gopclntab:080DC0E0 ; Segment type: Pure data
.gopclntab:080DC0E0 ; Segment permissions: Read
.gopclntab:080DC0E0 ; Segment alignment '32byte' can not be represented in assembly
.gopclntab:080DC0E0 _gopclntab      segment para public 'CONST' use32
.gopclntab:080DC0E0                 assume cs:_gopclntab
.gopclntab:080DC0E0                 ;org 80DC0E0h
.gopclntab:080DC0E0 unk_80DC0E0     db 0FBh ; �             ; DATA XREF: .noptrdata:off_81166E0↓o
.gopclntab:080DC0E1                 db 0FFh
.gopclntab:080DC0E2                 db 0FFh
.gopclntab:080DC0E3                 db 0FFh
.gopclntab:080DC0E4                 db    0
.gopclntab:080DC0E5                 db    0
.gopclntab:080DC0E6                 db    1
.gopclntab:080DC0E7                 db    4
.gopclntab:080DC0E8                 db  61h ; a
.gopclntab:080DC0E9                 db    6
.gopclntab:080DC0EA                 db    0
.gopclntab:080DC0EB                 db    0
.gopclntab:080DC0EC                 db    0
.gopclntab:080DC0ED                 db  90h ; �
.gopclntab:080DC0EE                 db    4
.gopclntab:080DC0EF                 db    8
.gopclntab:080DC0F0                 db  1Ch
.gopclntab:080DC0F1                 db  33h ; 3
.gopclntab:080DC0F2                 db    0
.gopclntab:080DC0F3                 db    0
.gopclntab:080DC0F4                 db  10h
```

- Oh how nice!
  A go specific segment!

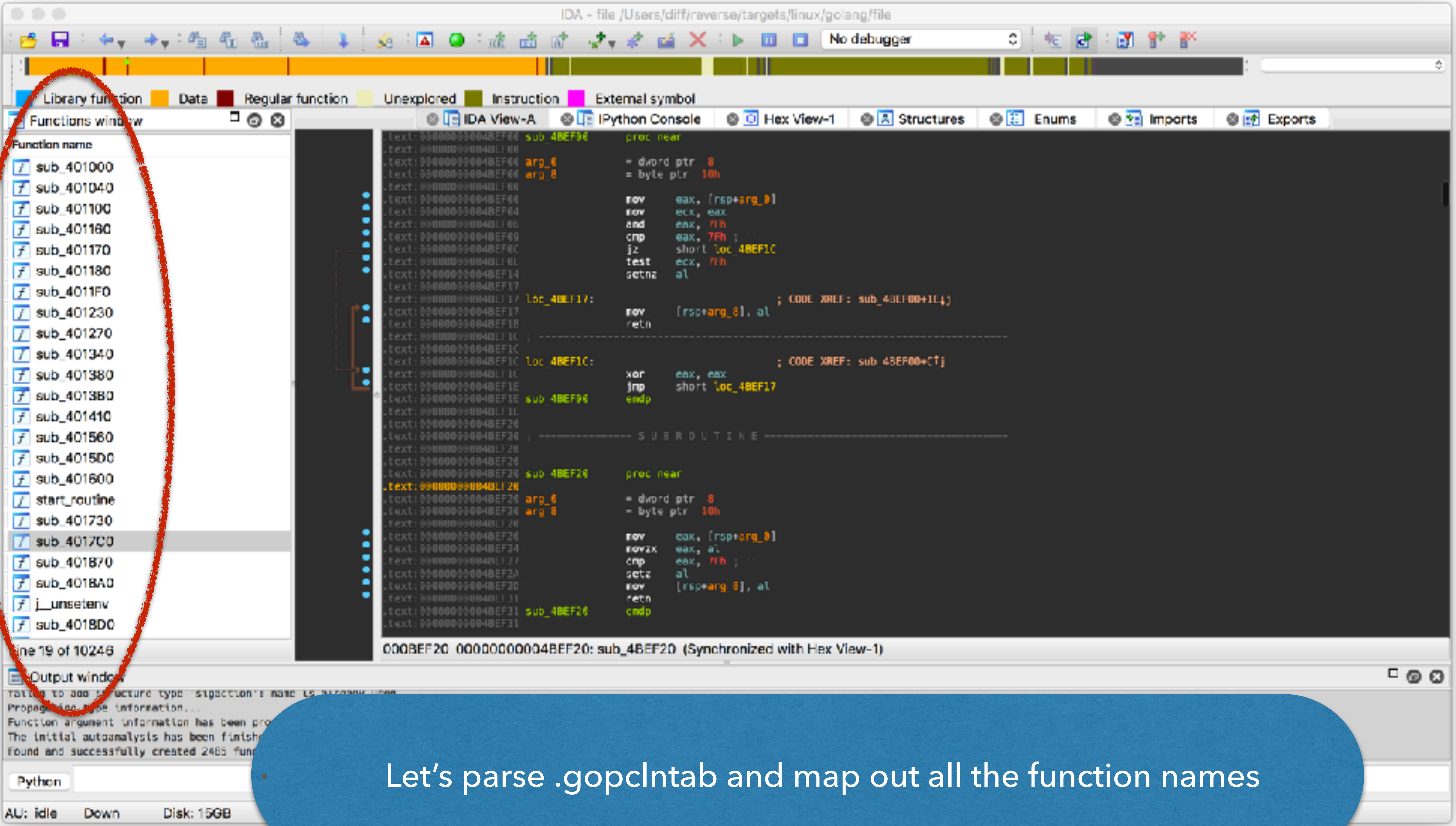
- Dig into the source and figure out the structure

```
struct gopclntab {
        char header[8];
        uint function_count;
        struct function {
            char address[4];
            char offsetToFuncName[4];
         }[function_count]
}
```



FUNCTION NAMES RECOVERABLE?

Enter .gopclntab!



IDA - file /Users/diff/reverse/targets/linux/golang/file

No debugger

Library function | Data | Regular function | Unexplored | Instruction | External symbol

Functions window

Function name

```
sub_401000
sub_401040
sub_401100
sub_401160
sub_401170
sub_401180
sub_4011F0
sub_401230
sub_401270
sub_401340
sub_401380
sub_4013B0
sub_401410
sub_401560
sub_4015D0
sub_401600
start_routine
sub_401730
sub_4017C0
sub_401870
sub_4018A0
j__unsetenv
sub_4018D0
```

Line 19 of 10246

IDA View-A | IPython Console | Hex View-1 | Structures | Enums | Imports | Exports

```
.text:00000000004BEF00 sub_4BEF00      proc near
.text:00000000004BEF00
.text:00000000004BEF00 arg_0           = dword ptr  8
.text:00000000004BEF00 arg_8           = byte ptr  10h
.text:00000000004BEF00
.text:00000000004BEF00                 mov     eax, [rsp+arg_0]
.text:00000000004BEF04                 mov     ecx, eax
.text:00000000004BEF06                 and     eax, 7Fh
.text:00000000004BEF09                 cmp     eax, 7Fh ; ''
.text:00000000004BEF0C                 jz      short loc_4BEF1C
.text:00000000004BEF0E                 test    ecx, 7Fh
.text:00000000004BEF14                 setnz   al
.text:00000000004BEF17
.text:00000000004BEF17 loc_4BEF17:                             ; CODE XREF: sub_4BEF00+1E↓j
.text:00000000004BEF17                 mov     [rsp+arg_8], al
.text:00000000004BEF1B                 retn
.text:00000000004BEF1C ; ---------------------------------------------------------------------------
.text:00000000004BEF1C
.text:00000000004BEF1C loc_4BEF1C:                             ; CODE XREF: sub_4BEF00+C↑j
.text:00000000004BEF1C                 xor     eax, eax
.text:00000000004BEF1E                 jmp     short loc_4BEF17
.text:00000000004BEF1E sub_4BEF00      endp
.text:00000000004BEF1E
.text:00000000004BEF20
.text:00000000004BEF20 ; =============== S U B R O U T I N E =======================================
.text:00000000004BEF20
.text:00000000004BEF20
.text:00000000004BEF20 sub_4BEF20      proc near
.text:00000000004BEF20
.text:00000000004BEF20 arg_0           = dword ptr  8
.text:00000000004BEF20 arg_8           = byte ptr  10h
.text:00000000004BEF20
.text:00000000004BEF20                 mov     eax, [rsp+arg_0]
.text:00000000004BEF24                 movzx   eax, al
.text:00000000004BEF27                 cmp     eax, 7Fh ; ''
.text:00000000004BEF2A                 setz    al
.text:00000000004BEF2D                 mov     [rsp+arg_8], al
.text:00000000004BEF31                 retn
.text:00000000004BEF31 sub_4BEF20      endp
.text:00000000004BEF31
```

000BEF20 00000000004BEF20: sub_4BEF20 (Synchronized with Hex View-1)

Output window

```
Failed to add structure type 'sigaction': name is already used
Propagating type information...
Function argument information has been pro
The initial autoanalysis has been finish
Found and successfully created 2485 fun
```

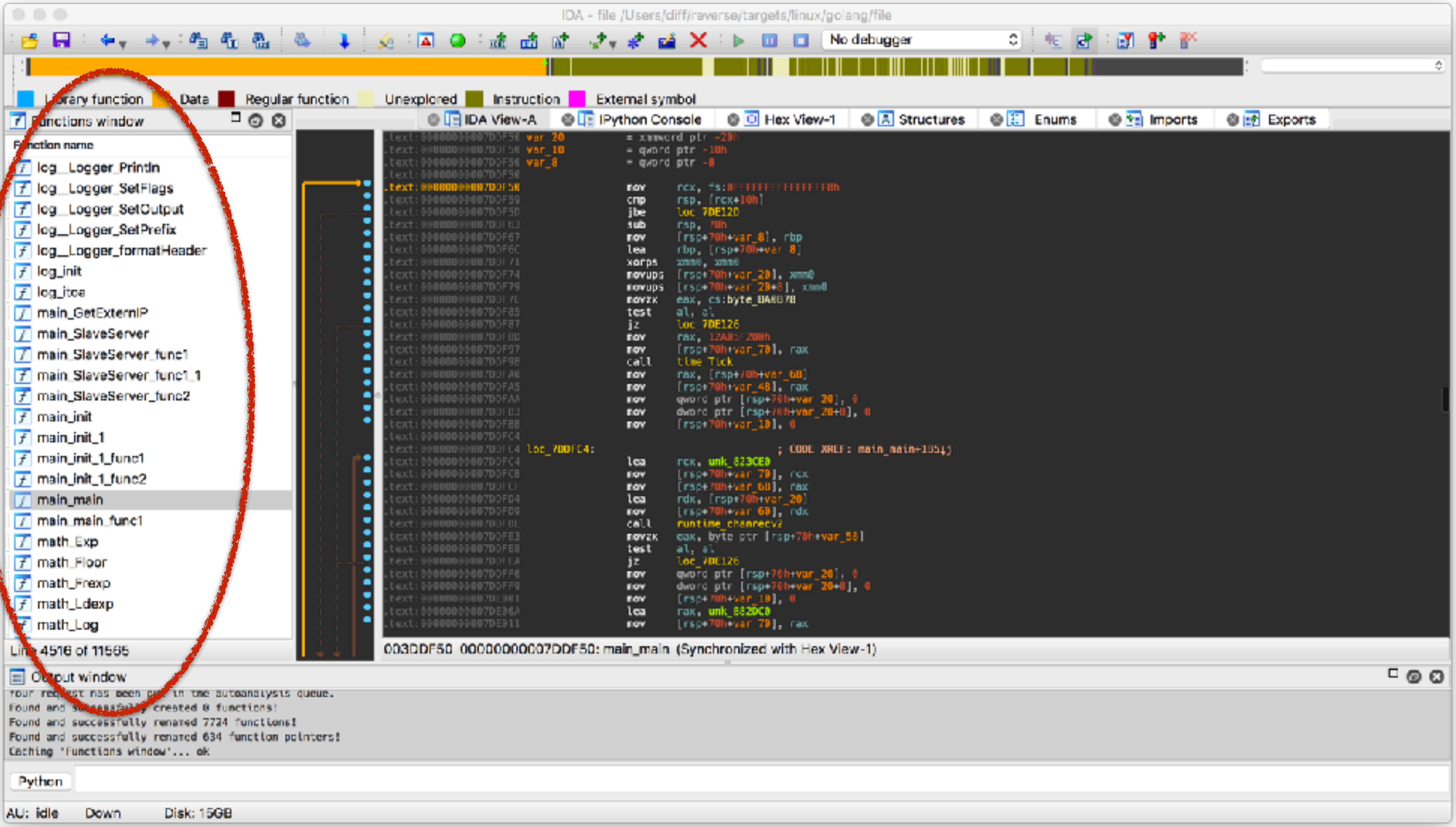
Python

AU: idle   Down   Disk: 15GB

Let's parse .gopclntab and map out all the function names

FUNCTION NAMES RECOVERABLE?

Enter .gopclntab!



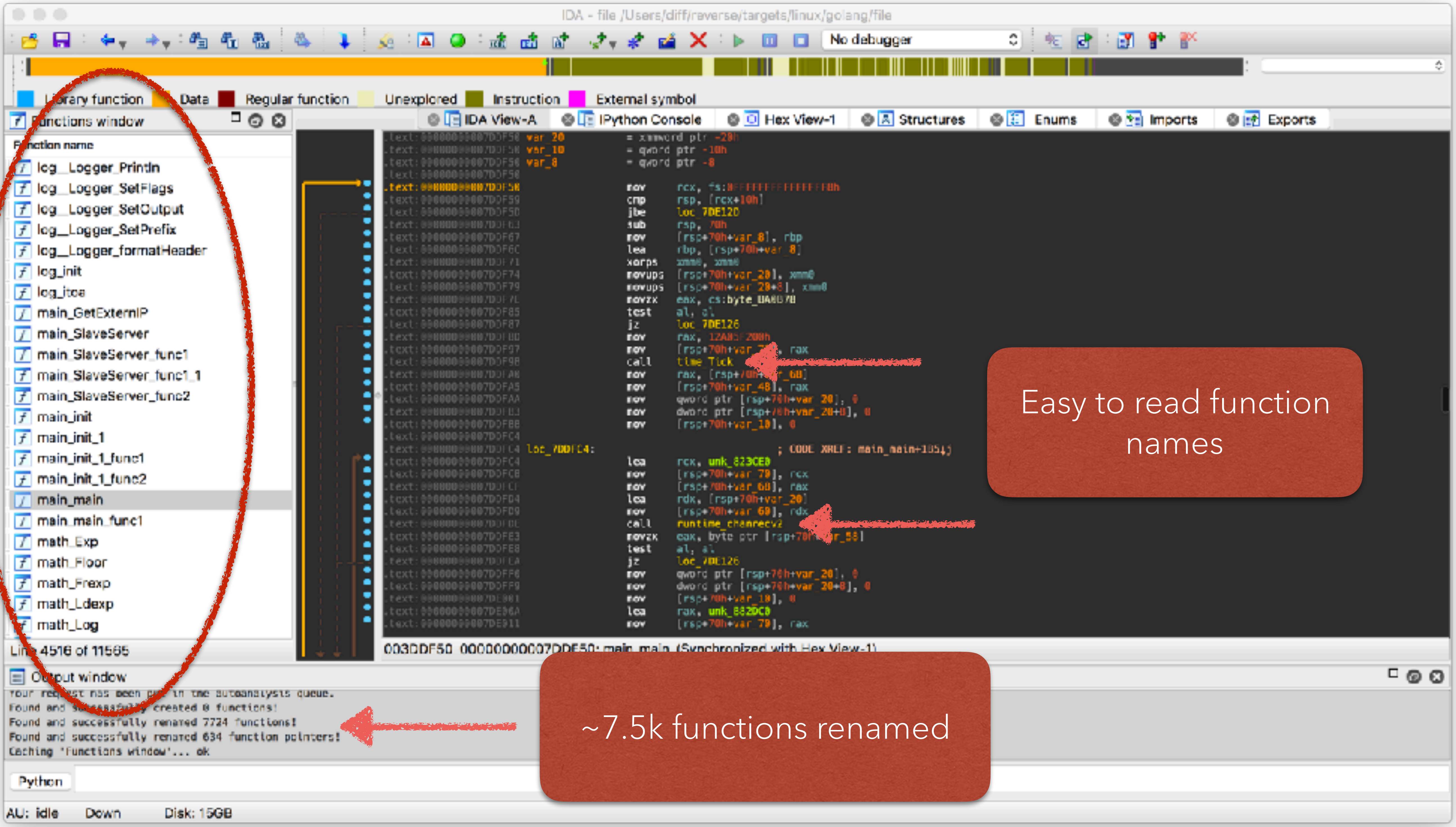
```
log__Logger_Println
log__Logger_SetFlags
log__Logger_SetOutput
log__Logger_SetPrefix
log__Logger_formatHeader
log_init
log_itoa
main_GetExternIP
main_SlaveServer
main_SlaveServer_func1
main_SlaveServer_func1_1
main_SlaveServer_func2
main_init
main_init_1
main_init_1_func1
main_init_1_func2
main_main
main_main_func1
math_Exp
math_Floor
math_Frexp
math_Ldexp
math_Log
Line 4516 of 11565
```

```
.text:00000000007DDF50 var_20          = xmmword ptr -20h
.text:00000000007DDF50 var_10          = qword ptr -10h
.text:00000000007DDF50 var_8           = qword ptr -8
.text:00000000007DDF50
.text:00000000007DDF50                 mov     rcx, fs:0FFFFFFFFFFFFFFF8h
.text:00000000007DDF59                 cmp     rsp, [rcx+10h]
.text:00000000007DDF5D                 jbe     loc_7DE120
.text:00000000007DDF63                 sub     rsp, 70h
.text:00000000007DDF67                 mov     [rsp+70h+var_8], rbp
.text:00000000007DDF6C                 lea     rbp, [rsp+70h+var_8]
.text:00000000007DDF71                 xorps   xmm0, xmm0
.text:00000000007DDF74                 movups  [rsp+70h+var_20], xmm0
.text:00000000007DDF79                 movups  [rsp+70h+var_20+8], xmm0
.text:00000000007DDF7E                 movzx   eax, cs:byte_UA8B78
.text:00000000007DDF85                 test    al, al
.text:00000000007DDF87                 jz      loc_7DE126
.text:00000000007DDF8D                 mov     rax, 12A05F200h
.text:00000000007DDF97                 mov     [rsp+70h+var_70], rax
.text:00000000007DDF9B                 call    time_Tick
.text:00000000007DDFA0                 mov     rax, [rsp+70h+var_68]
.text:00000000007DDFA5                 mov     [rsp+70h+var_48], rax
.text:00000000007DDFAA                 mov     qword ptr [rsp+70h+var_20], 0
.text:00000000007DDFB3                 mov     dword ptr [rsp+70h+var_20+8], 0
.text:00000000007DDFBB                 mov     [rsp+70h+var_10], 0
.text:00000000007DDFC4
.text:00000000007DDFC4 loc_7DDFC4:                             ; CODE XREF: main_main+1D5↓j
.text:00000000007DDFC4                 lea     rcx, unk_823CE0
.text:00000000007DDFCB                 mov     [rsp+70h+var_70], rcx
.text:00000000007DDFCF                 mov     [rsp+70h+var_68], rax
.text:00000000007DDFD4                 lea     rdx, [rsp+70h+var_20]
.text:00000000007DDFD9                 mov     [rsp+70h+var_60], rdx
.text:00000000007DDFDE                 call    runtime_chanrecv2
.text:00000000007DDFE3                 movzx   eax, byte ptr [rsp+70h+var_58]
.text:00000000007DDFE8                 test    al, al
.text:00000000007DDFEA                 jz      loc_7DE126
.text:00000000007DDFF0                 mov     qword ptr [rsp+70h+var_20], 0
.text:00000000007DDFF9                 mov     dword ptr [rsp+70h+var_20+8], 0
.text:00000000007DE001                 mov     [rsp+70h+var_10], 0
.text:00000000007DE00A                 lea     rax, unk_882DC0
.text:00000000007DE011                 mov     [rsp+70h+var_70], rax
003DDF50 00000000007DDF50: main_main (Synchronized with Hex View-1)
```

```
Your request has been put in the autoanalysis queue.
Found and successfully created 0 functions!
Found and successfully renamed 7724 functions!
Found and successfully renamed 634 function pointers!
Caching 'Functions window'... ok
```

FUNCTION NAMES RECOVERABLE?

Enter .gopclntab!


IDA - file /Users/diff/reverse/targets/linux/golang/file

Functions window

```
Function name
log__Logger_Println
log__Logger_SetFlags
log__Logger_SetOutput
log__Logger_SetPrefix
log__Logger_formatHeader
log_init
log_itoa
main_GetExternIP
main_SlaveServer
main_SlaveServer_func1
main_SlaveServer_func1_1
main_SlaveServer_func2
main_init
main_init_1
main_init_1_func1
main_init_1_func2
main_main
main_main_func1
math_Exp
math_Floor
math_Frexp
math_Ldexp
math_Log
Line 4516 of 11565
```

```
.text:00000000007DDF50 var_20          = xmmword ptr -20h
.text:00000000007DDF50 var_10          = qword ptr -10h
.text:00000000007DDF50 var_8           = qword ptr -8
.text:00000000007DDF50
.text:00000000007DDF50                 mov     rcx, fs:0FFFFFFFFFFFFFFF8h
.text:00000000007DDF59                 cmp     rsp, [rcx+10h]
.text:00000000007DDF5D                 jbe     loc_7DE120
.text:00000000007DDF63                 sub     rsp, 70h
.text:00000000007DDF67                 mov     [rsp+70h+var_8], rbp
.text:00000000007DDF6C                 lea     rbp, [rsp+70h+var_8]
.text:00000000007DDF71                 xorps   xmm0, xmm0
.text:00000000007DDF74                 movups  [rsp+70h+var_20], xmm0
.text:00000000007DDF79                 movups  [rsp+70h+var_20+8], xmm0
.text:00000000007DDF7E                 movzx   eax, cs:byte_DA8B70
.text:00000000007DDF85                 test    al, al
.text:00000000007DDF87                 jz      loc_7DE126
.text:00000000007DDF8D                 mov     rax, 12A05F200h
.text:00000000007DDF97                 mov     [rsp+70h+var_70], rax
.text:00000000007DDF9B                 call    time_Tick
.text:00000000007DDFA0                 mov     rax, [rsp+70h+var_68]
.text:00000000007DDFA5                 mov     [rsp+70h+var_48], rax
.text:00000000007DDFAA                 mov     qword ptr [rsp+70h+var_20], 0
.text:00000000007DDFB3                 mov     dword ptr [rsp+70h+var_20+8], 0
.text:00000000007DDFBB                 mov     [rsp+70h+var_10], 0
.text:00000000007DDFC4
.text:00000000007DDFC4 loc_7DDFC4:                             ; CODE XREF: main_main+1D5↓j
.text:00000000007DDFC4                 lea     rcx, unk_823CE0
.text:00000000007DDFCB                 mov     [rsp+70h+var_70], rcx
.text:00000000007DDFCF                 mov     [rsp+70h+var_68], rax
.text:00000000007DDFD4                 lea     rdx, [rsp+70h+var_20]
.text:00000000007DDFD9                 mov     [rsp+70h+var_60], rdx
.text:00000000007DDFDE                 call    runtime_chanrecv2
.text:00000000007DDFE3                 movzx   eax, byte ptr [rsp+70h+var_58]
.text:00000000007DDFE8                 test    al, al
.text:00000000007DDFEA                 jz      loc_7DE126
.text:00000000007DDFF0                 mov     qword ptr [rsp+70h+var_20], 0
.text:00000000007DDFF9                 mov     dword ptr [rsp+70h+var_20+8], 0
.text:00000000007DE001                 mov     [rsp+70h+var_10], 0
.text:00000000007DE00A                 lea     rax, unk_8820C0
.text:00000000007DE011                 mov     [rsp+70h+var_70], rax
003DDF50 00000000007DDF50: main_main (Synchronized with Hex View-1)
```

Output window

```
Your request has been put in the autoanalysis queue.
Found and successfully created 0 functions!
Found and successfully renamed 7724 functions!
Found and successfully renamed 634 function pointers!
Caching 'Functions window'... ok
```

AU: idle   Down   Disk: 15GB

Easy to read function names

~7.5k functions renamed

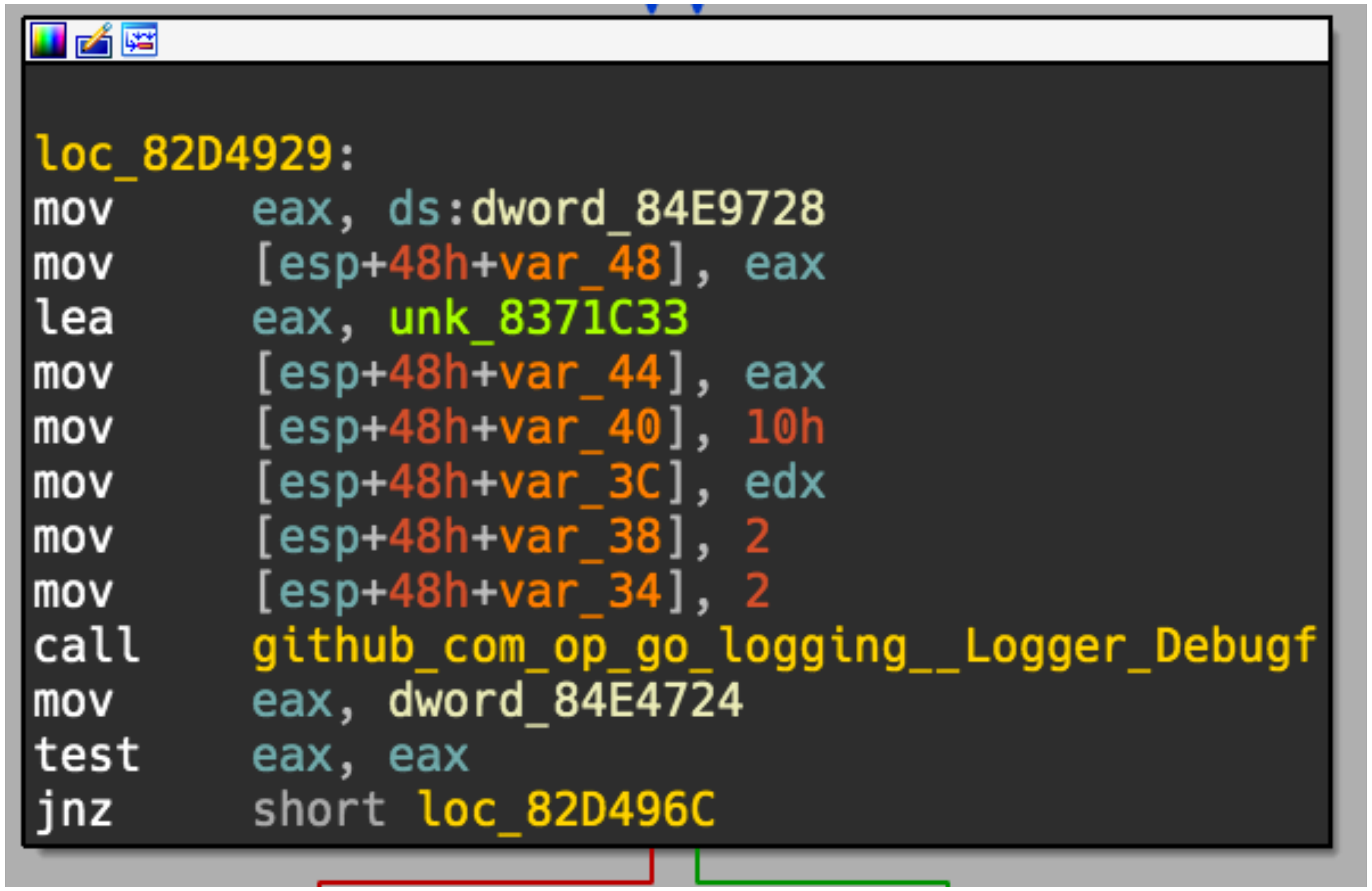
FIXING OUR TOOLS…

Issues Identified

- ~~Functions are not all easily defined~~
- ~~Functions do not retain their name when stripped~~
- String loads can be funky - dependent on architecture and Go version
- For above, we need to easily identify Go version!

STRING LOADS? WHERE ARE YOU?

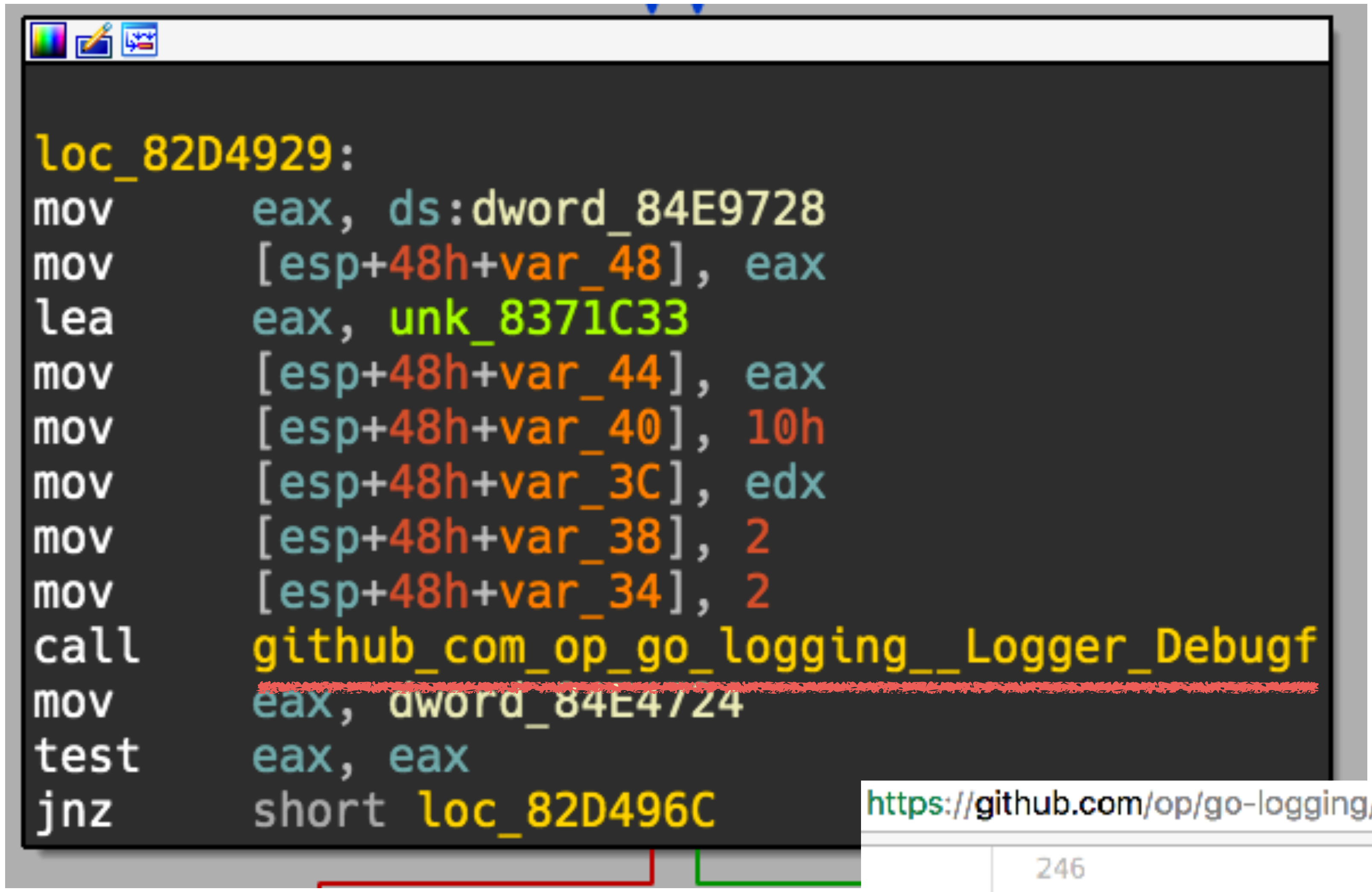
This doesn't seem very nice…



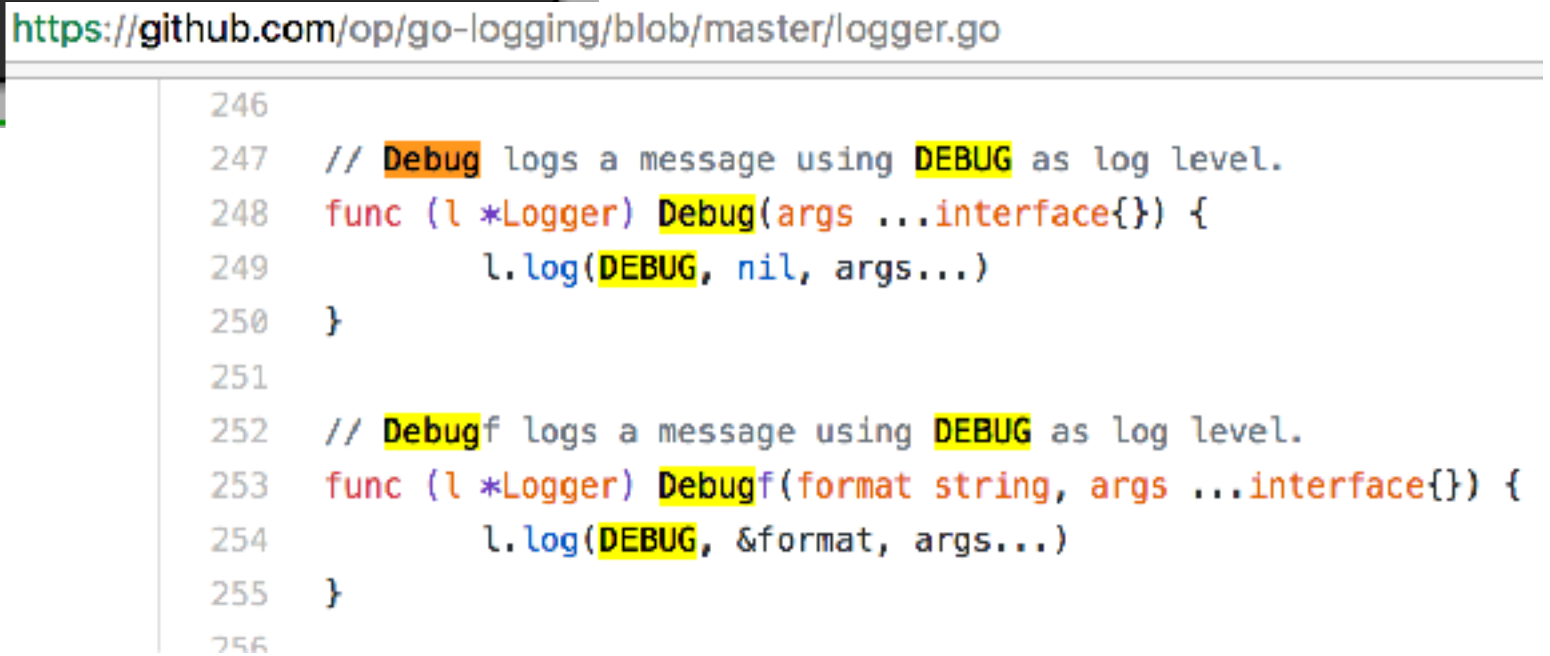
```
loc_82D4929:
mov     eax, ds:dword_84E9728
mov     [esp+48h+var_48], eax
lea     eax, unk_8371C33
mov     [esp+48h+var_44], eax
mov     [esp+48h+var_40], 10h
mov     [esp+48h+var_3C], edx
mov     [esp+48h+var_38], 2
mov     [esp+48h+var_34], 2
call    github_com_op_go_logging__Logger_Debugf
mov     eax, dword_84E4724
test    eax, eax
jnz     short loc_82D496C
```

STRING LOADS? WHERE ARE YOU?

This doesn't seem very nice…



```
loc_82D4929:
mov     eax, ds:dword_84E9728
mov     [esp+48h+var_48], eax
lea     eax, unk_8371C33
mov     [esp+48h+var_44], eax
mov     [esp+48h+var_40], 10h
mov     [esp+48h+var_3C], edx
mov     [esp+48h+var_38], 2
mov     [esp+48h+var_34], 2
call    github_com_op_go_logging__Logger_Debugf
mov     eax, dword_84E4724
test    eax, eax
jnz     short loc_82D496C
```

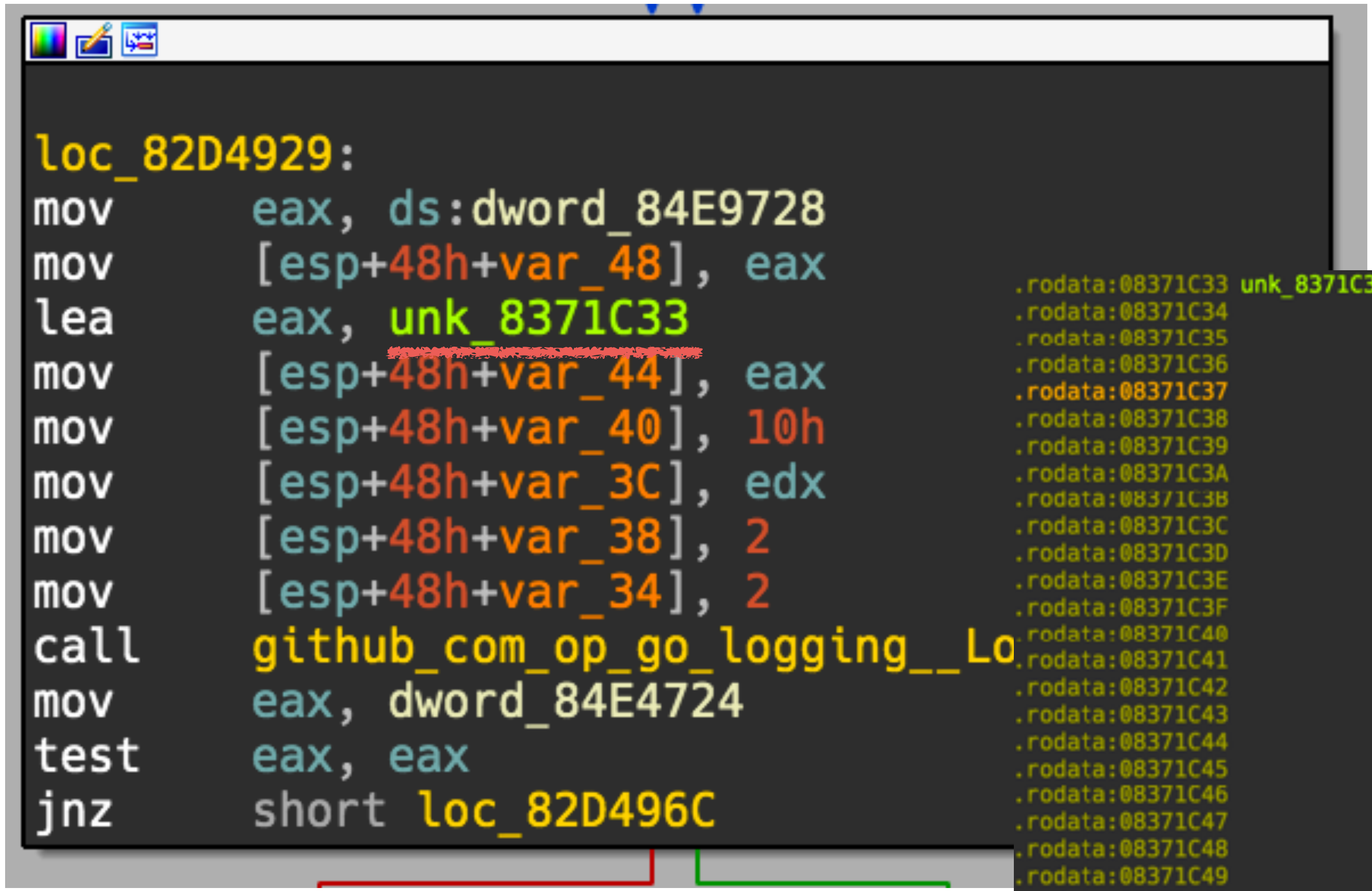


https://github.com/op/go-logging/blob/master/logger.go

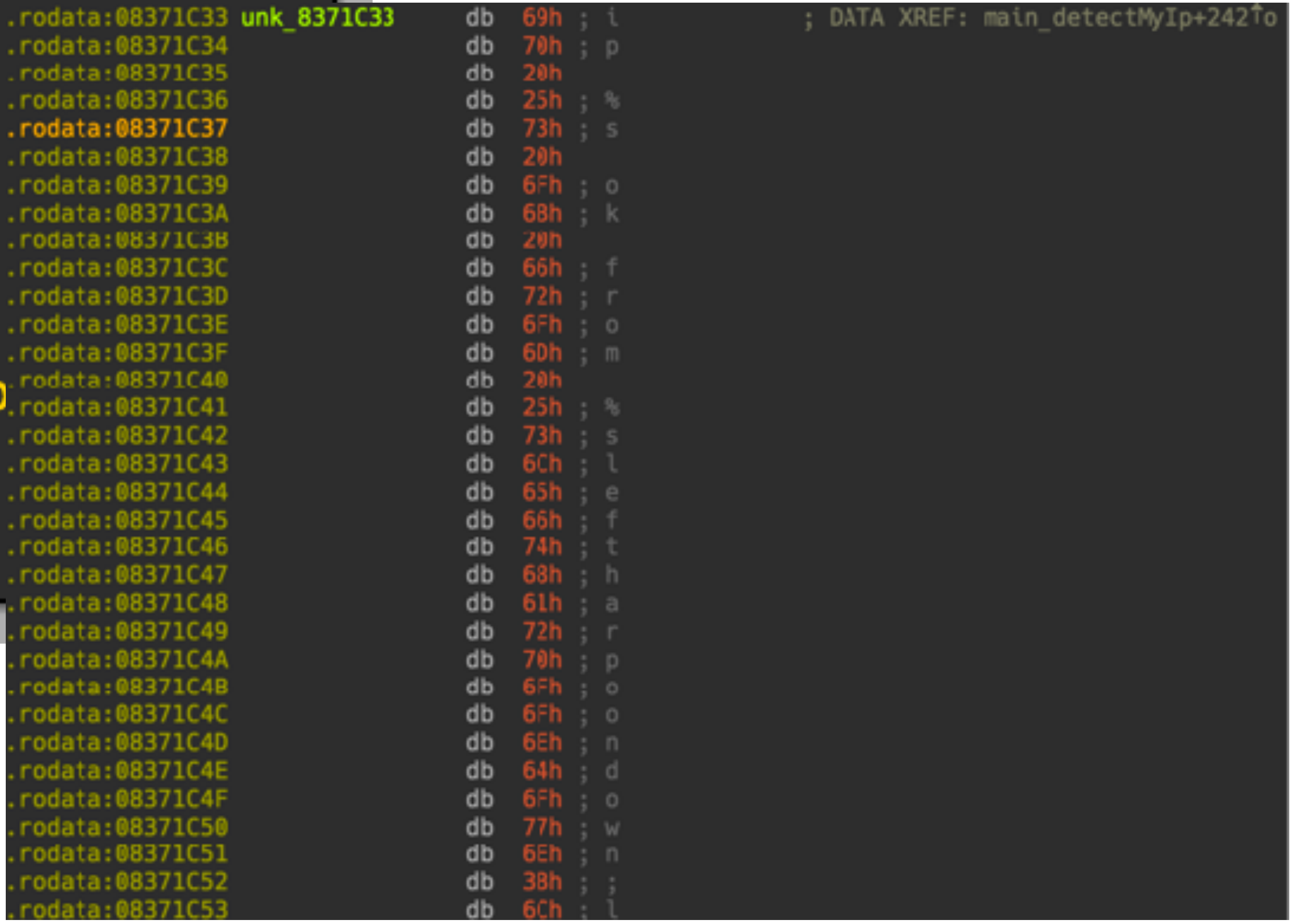
```go
// Debug logs a message using DEBUG as log level.
func (l *Logger) Debug(args ...interface{}) {
        l.log(DEBUG, nil, args...)
}

// Debugf logs a message using DEBUG as log level.
func (l *Logger) Debugf(format string, args ...interface{}) {
        l.log(DEBUG, &format, args...)
}
```

STRING LOADS? WHERE ARE YOU?

This doesn't seem very nice…



```
loc_82D4929:
mov     eax, ds:dword_84E9728
mov     [esp+48h+var_48], eax
lea     eax, unk_8371C33
mov     [esp+48h+var_44], eax
mov     [esp+48h+var_40], 10h
mov     [esp+48h+var_3C], edx
mov     [esp+48h+var_38], 2
mov     [esp+48h+var_34], 2
call    github_com_op_go_logging__Lo
mov     eax, dword_84E4724
test    eax, eax
jnz     short loc_82D496C
```

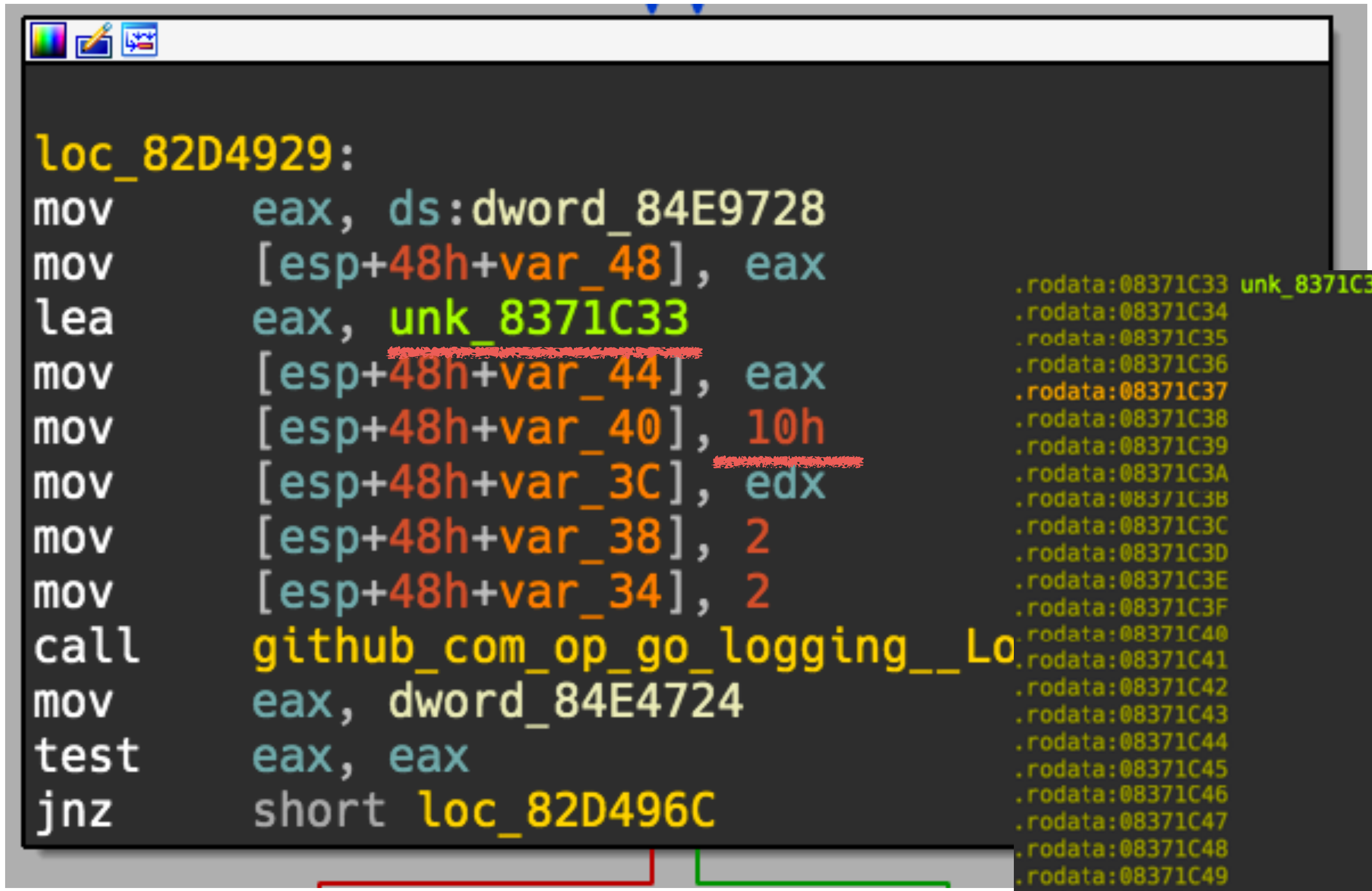


```
.rodata:08371C33 unk_8371C33     db  69h ; i            ; DATA XREF: main_detectMyIp+242↑o
.rodata:08371C34                 db  70h ; p
.rodata:08371C35                 db  20h
.rodata:08371C36                 db  25h ; %
.rodata:08371C37                 db  73h ; s
.rodata:08371C38                 db  20h
.rodata:08371C39                 db  6Fh ; o
.rodata:08371C3A                 db  6Bh ; k
.rodata:08371C3B                 db  20h
.rodata:08371C3C                 db  66h ; f
.rodata:08371C3D                 db  72h ; r
.rodata:08371C3E                 db  6Fh ; o
.rodata:08371C3F                 db  6Dh ; m
.rodata:08371C40                 db  20h
.rodata:08371C41                 db  25h ; %
.rodata:08371C42                 db  73h ; s
.rodata:08371C43                 db  6Ch ; l
.rodata:08371C44                 db  65h ; e
.rodata:08371C45                 db  66h ; f
.rodata:08371C46                 db  74h ; t
.rodata:08371C47                 db  68h ; h
.rodata:08371C48                 db  61h ; a
.rodata:08371C49                 db  72h ; r
.rodata:08371C4A                 db  70h ; p
.rodata:08371C4B                 db  6Fh ; o
.rodata:08371C4C                 db  6Fh ; o
.rodata:08371C4D                 db  6Eh ; n
.rodata:08371C4E                 db  64h ; d
.rodata:08371C4F                 db  6Fh ; o
.rodata:08371C50                 db  77h ; w
.rodata:08371C51                 db  6Eh ; n
.rodata:08371C52                 db  3Bh ; ;
.rodata:08371C53                 db  6Ch ; l
```

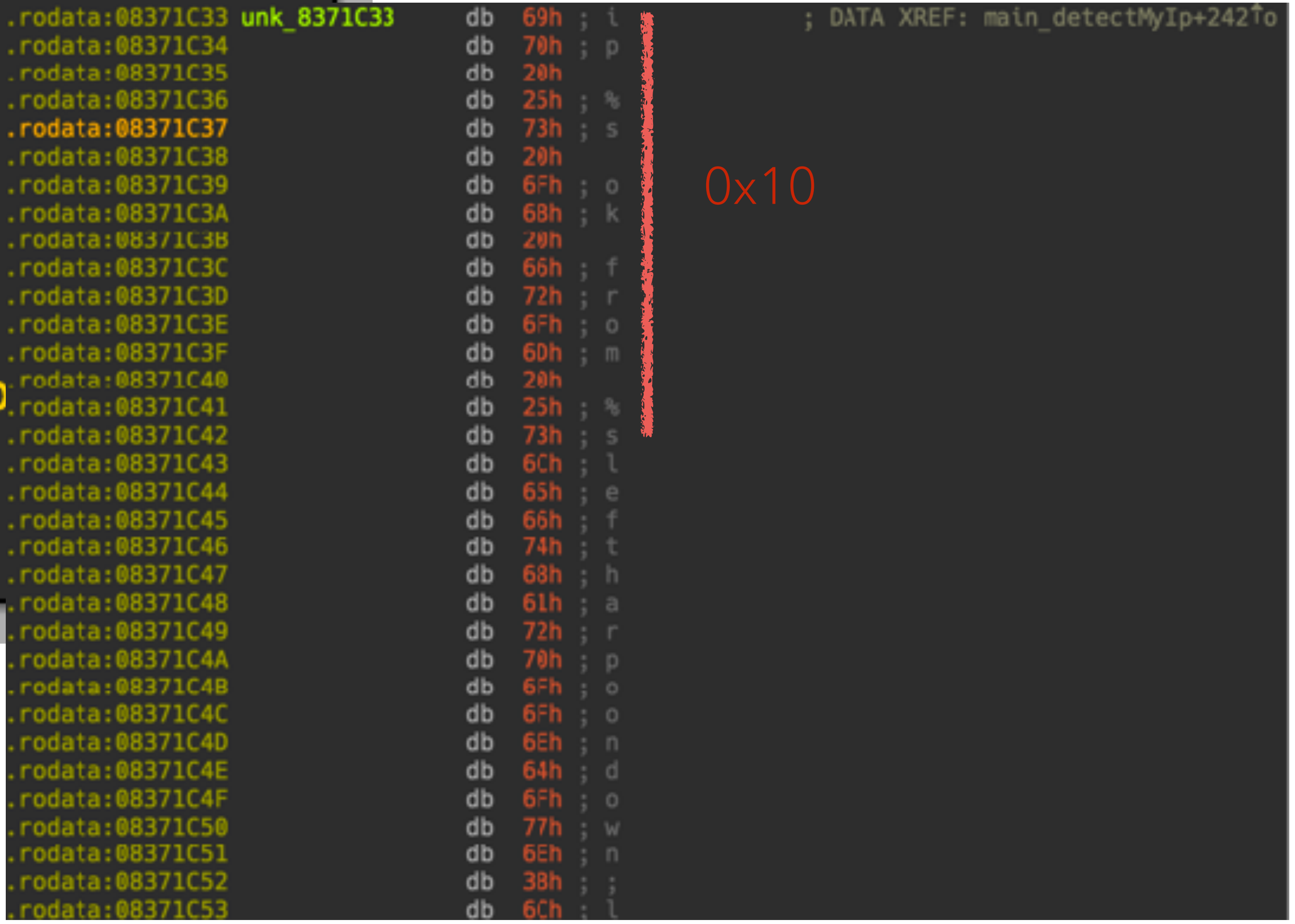
No null terminator?

STRING LOADS? WHERE ARE YOU?

This doesn't seem very nice…



```
loc_82D4929:
mov     eax, ds:dword_84E9728
mov     [esp+48h+var_48], eax
lea     eax, unk_8371C33
mov     [esp+48h+var_44], eax
mov     [esp+48h+var_40], 10h
mov     [esp+48h+var_3C], edx
mov     [esp+48h+var_38], 2
mov     [esp+48h+var_34], 2
call    github_com_op_go_logging__Lo
mov     eax, dword_84E4724
test    eax, eax
jnz     short loc_82D496C
```

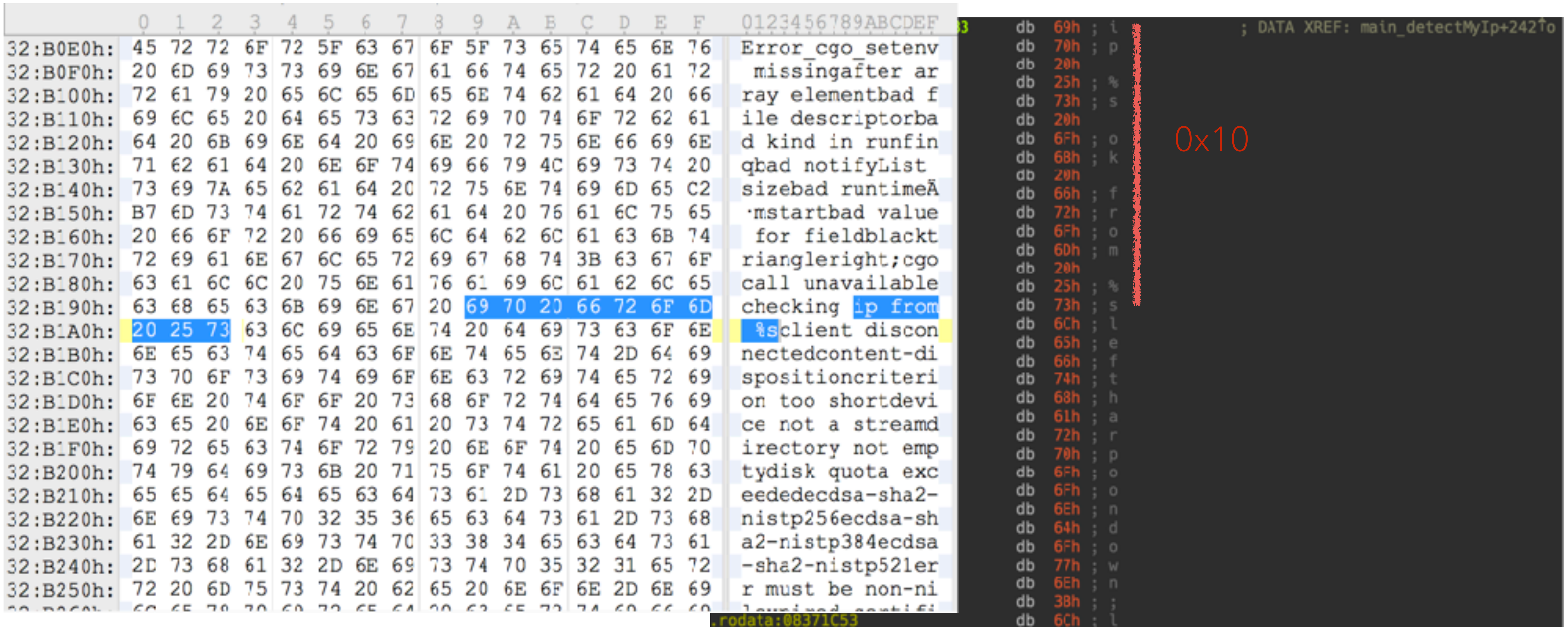


```
.rodata:08371C33 unk_8371C33     db  69h ; i              ; DATA XREF: main_detectMyIp+242↑o
.rodata:08371C34                 db  70h ; p
.rodata:08371C35                 db  20h
.rodata:08371C36                 db  25h ; %
.rodata:08371C37                 db  73h ; s
.rodata:08371C38                 db  20h
.rodata:08371C39                 db  6Fh ; o
.rodata:08371C3A                 db  6Bh ; k
.rodata:08371C3B                 db  20h
.rodata:08371C3C                 db  66h ; f
.rodata:08371C3D                 db  72h ; r
.rodata:08371C3E                 db  6Fh ; o
.rodata:08371C3F                 db  6Dh ; m
.rodata:08371C40                 db  20h
.rodata:08371C41                 db  25h ; %
.rodata:08371C42                 db  73h ; s
.rodata:08371C43                 db  6Ch ; l
.rodata:08371C44                 db  65h ; e
.rodata:08371C45                 db  66h ; f
.rodata:08371C46                 db  74h ; t
.rodata:08371C47                 db  68h ; h
.rodata:08371C48                 db  61h ; a
.rodata:08371C49                 db  72h ; r
.rodata:08371C4A                 db  70h ; p
.rodata:08371C4B                 db  6Fh ; o
.rodata:08371C4C                 db  6Fh ; o
.rodata:08371C4D                 db  6Eh ; n
.rodata:08371C4E                 db  64h ; d
.rodata:08371C4F                 db  6Fh ; o
.rodata:08371C50                 db  77h ; w
.rodata:08371C51                 db  6Eh ; n
.rodata:08371C52                 db  3Bh ; ;
.rodata:08371C53                 db  6Ch ; l
```

Length seems to be in code below loading of pointer!

STRING LOADS? WHERE ARE YOU?

Weird, oddly familiar… Sort of like Dalvik string tables?



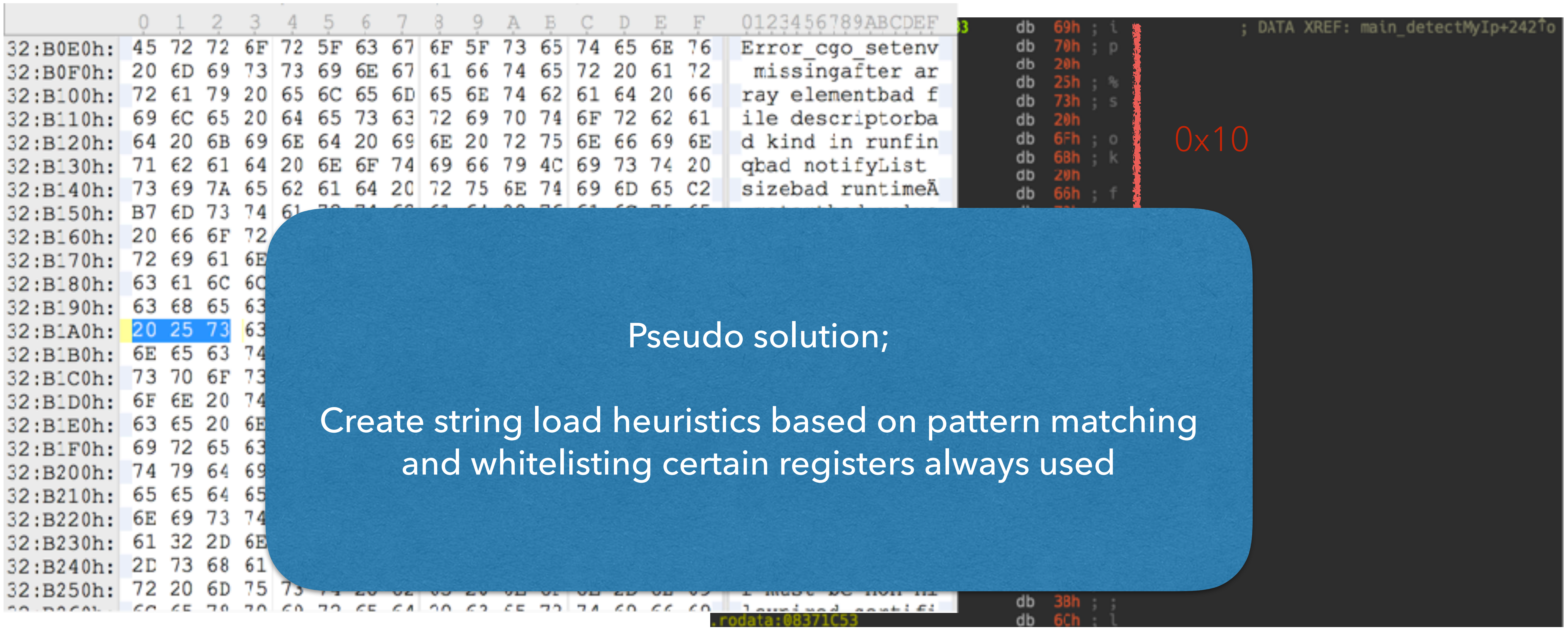
```
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
32:B0E0h: 45 72 72 6F 72 5F 63 67 6F 5F 73 65 74 65 6E 76  Error_cgo_setenv
32:B0F0h: 20 6D 69 73 73 69 6E 67 61 66 74 65 72 20 61 72   missingafter ar
32:B100h: 72 61 79 20 65 6C 65 6D 65 6E 74 62 61 64 20 66  ray elementbad f
32:B110h: 69 6C 65 20 64 65 73 63 72 69 70 74 6F 72 62 61  ile descriptorba
32:B120h: 64 20 6B 69 6E 64 20 69 6E 20 72 75 6E 66 69 6E  d kind in runfin
32:B130h: 71 62 61 64 20 6E 6F 74 69 66 79 4C 69 73 74 20  qbad notifyList
32:B140h: 73 69 7A 65 62 61 64 20 72 75 6E 74 69 6D 65 C2  sizebad runtimeÂ
32:B150h: B7 6D 73 74 61 72 74 62 61 64 20 76 61 6C 75 65  ·mstartbad value
32:B160h: 20 66 6F 72 20 66 69 65 6C 64 62 6C 61 63 6B 74   for fieldblackt
32:B170h: 72 69 61 6E 67 6C 65 72 69 67 68 74 3B 63 67 6F  riangleright;cgo
32:B180h: 63 61 6C 6C 20 75 6E 61 76 61 69 6C 61 62 6C 65  call unavailable
32:B190h: 63 68 65 63 6B 69 6E 67 20 69 70 20 66 72 6F 6D  checking ip from
32:B1A0h: 20 25 73 63 6C 69 65 6E 74 20 64 69 73 63 6F 6E   %sclient discon
32:B1B0h: 6E 65 63 74 65 64 63 6F 6E 74 65 6E 74 2D 64 69  nectedcontent-di
32:B1C0h: 73 70 6F 73 69 74 69 6F 6E 63 72 69 74 65 72 69  spositioncriteri
32:B1D0h: 6F 6E 20 74 6F 6F 20 73 68 6F 72 74 64 65 76 69  on too shortdevi
32:B1E0h: 63 65 20 6E 6F 74 20 61 20 73 74 72 65 61 6D 64  ce not a streamd
32:B1F0h: 69 72 65 63 74 6F 72 79 20 6E 6F 74 20 65 6D 70  irectory not emp
32:B200h: 74 79 64 69 73 6B 20 71 75 6F 74 61 20 65 78 63  tydisk quota exc
32:B210h: 65 65 64 65 64 65 63 64 73 61 2D 73 68 61 32 2D  eededecdsa-sha2-
32:B220h: 6E 69 73 74 70 32 35 36 65 63 64 73 61 2D 73 68  nistp256ecdsa-sh
32:B230h: 61 32 2D 6E 69 73 74 70 33 38 34 65 63 64 73 61  a2-nistp384ecdsa
32:B240h: 2D 73 68 61 32 2D 6E 69 73 74 70 35 32 31 65 72  -sha2-nistp521er
32:B250h: 72 20 6D 75 73 74 20 62 65 20 6E 6F 6E 2D 6E 69  r must be non-ni
```

```
db 69h ; i                    ; DATA XREF: main_detectMyIp+242↑o
db 70h ; p
db 20h
db 25h ; %
db 73h ; s
db 20h
db 6Fh ; o
db 6Bh ; k
db 20h
db 66h ; f
db 72h ; r
db 6Fh ; o
db 6Dh ; m
db 20h
db 25h ; %
db 73h ; s
db 6Ch ; l
db 65h ; e
db 66h ; f
db 74h ; t
db 68h ; h
db 61h ; a
db 72h ; r
db 70h ; p
db 6Fh ; o
db 6Fh ; o
db 6Eh ; n
db 64h ; d
db 6Fh ; o
db 77h ; w
db 6Eh ; n
db 3Bh ; ;
db 6Ch ; l
.rodata:08371C53
```

- Strings grouped together with no null bytes
- Groups are collected together by length, then alpha numerical order

STRING LOADS? WHERE ARE YOU?

Weird, oddly familiar… Sort of like Dalvik string tables?



```
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
32:B0E0h: 45 72 72 6F 72 5F 63 67 6F 5F 73 65 74 65 6E 76  Error_cgo_setenv
32:B0F0h: 20 6D 69 73 73 69 6E 67 61 66 74 65 72 20 61 72   missingafter ar
32:B100h: 72 61 79 20 65 6C 65 6D 65 6E 74 62 61 64 20 66  ray elementbad f
32:B110h: 69 6C 65 20 64 65 73 63 72 69 70 74 6F 72 62 61  ile descriptorba
32:B120h: 64 20 6B 69 6E 64 20 69 6E 20 72 75 6E 66 69 6E  d kind in runfin
32:B130h: 71 62 61 64 20 6E 6F 74 69 66 79 4C 69 73 74 20  qbad notifyList 
32:B140h: 73 69 7A 65 62 61 64 20 72 75 6E 74 69 6D 65 C2  sizebad runtimeÃ
32:B150h: B7 6D 73 74 61
32:B160h: 20 66 6F 72
32:B170h: 72 69 61 6E
32:B180h: 63 61 6C 6C
32:B190h: 63 68 65 63
32:B1A0h: 20 25 73 63
32:B1B0h: 6E 65 63 74
32:B1C0h: 73 70 6F 73
32:B1D0h: 6F 6E 20 74
32:B1E0h: 63 65 20 6E
32:B1F0h: 69 72 65 63
32:B200h: 74 79 64 69
32:B210h: 65 65 64 65
32:B220h: 6E 69 73 74
32:B230h: 61 32 2D 6E
32:B240h: 2D 73 68 61
32:B250h: 72 20 6D 75 73
```

```
.rodata:08371C53          db  69h ; i        ; DATA XREF: main_detectMyIp+242↑o
                          db  70h ; p
                          db  20h
                          db  25h ; %
                          db  73h ; s
                          db  20h
                          db  6Fh ; o
                          db  6Bh ; k
                          db  20h
                          db  66h ; f
                          db  38h ; 8
                          db  6Ch ; l
```

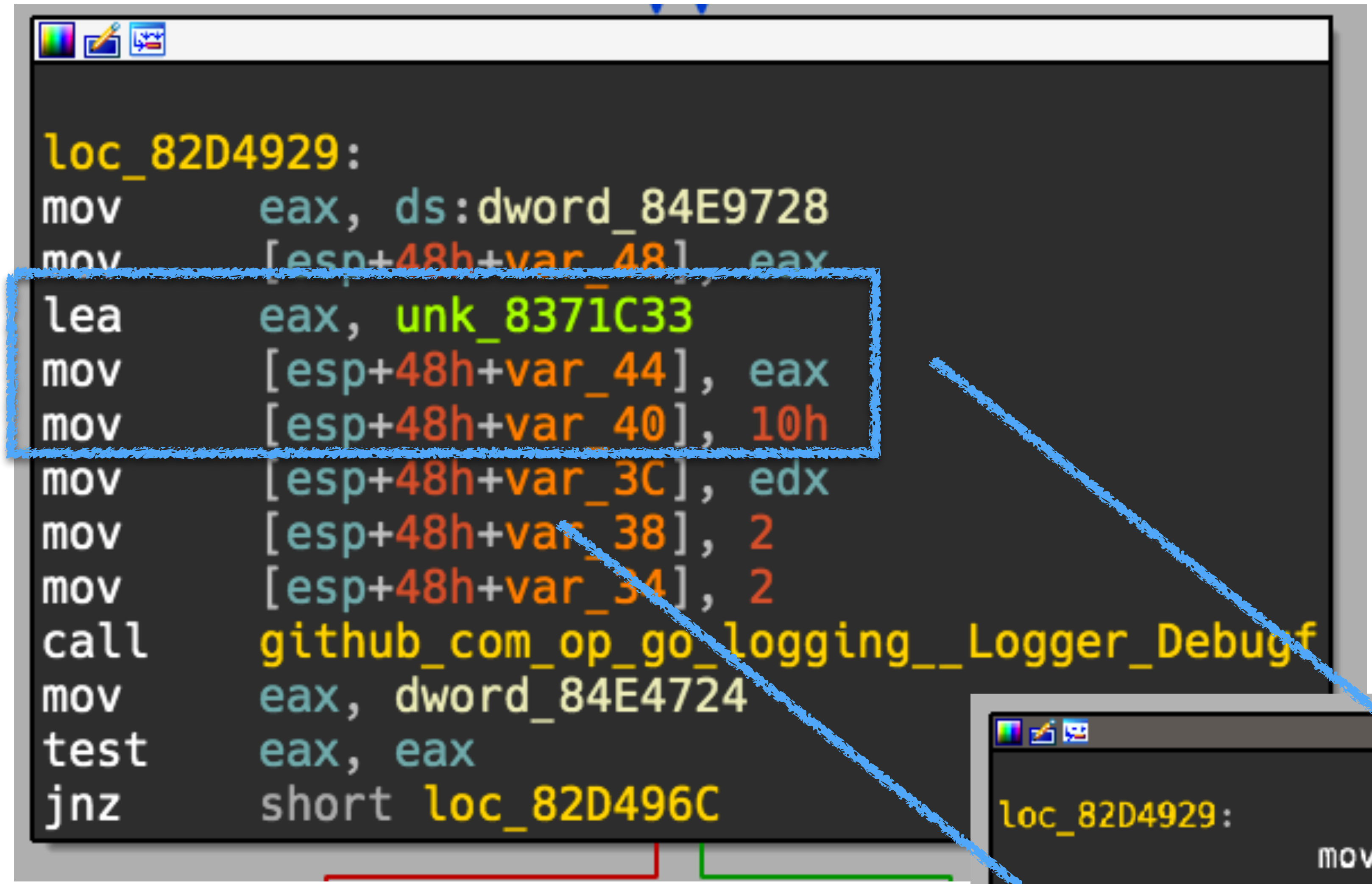
Pseudo solution;

Create string load heuristics based on pattern matching and whitelisting certain registers always used

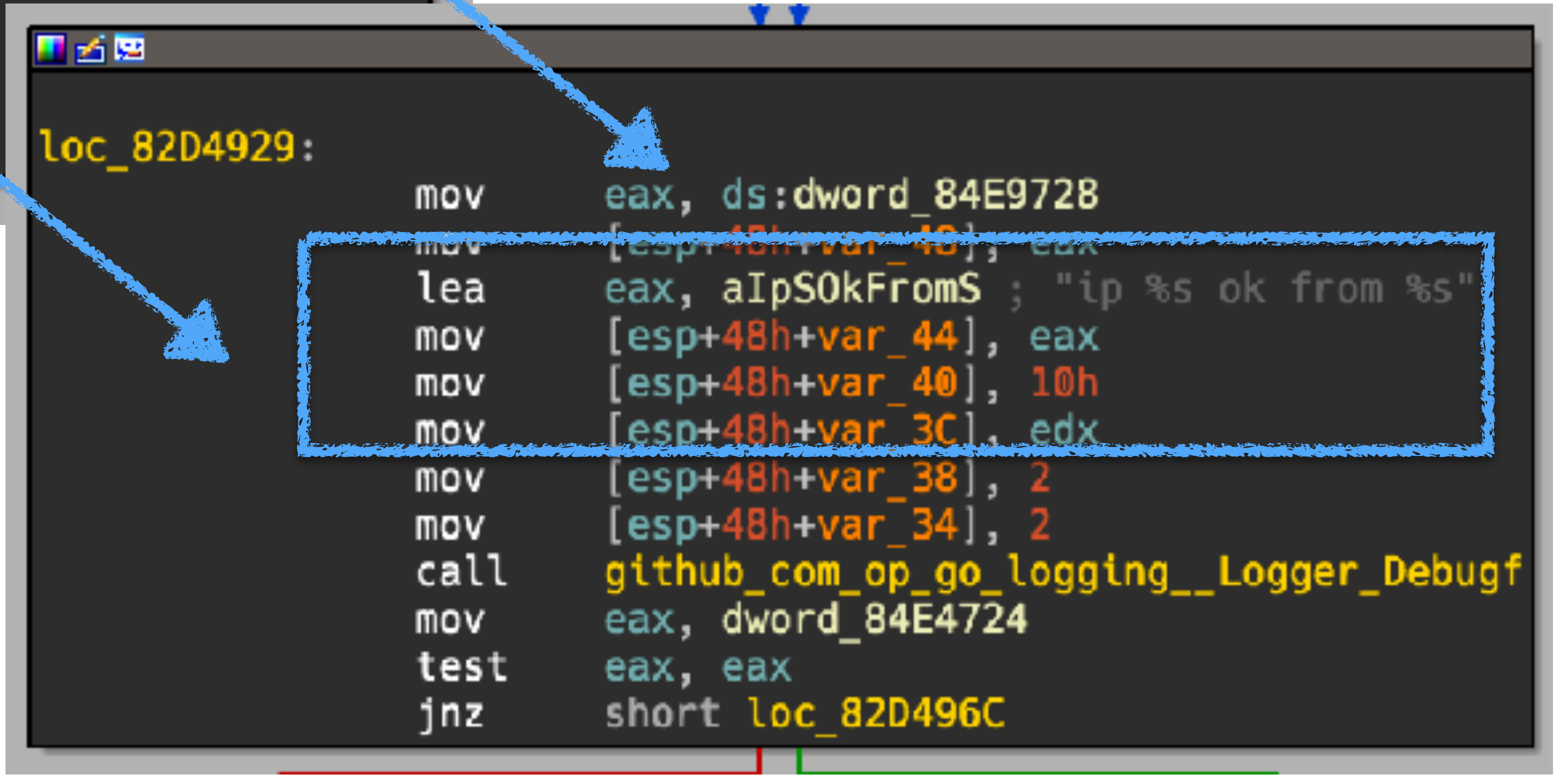
- Strings grouped together with no null bytes
- Groups are collected together by length, then alpha numerical order

STRING LOADS? WHERE ARE YOU?

Testing heuristics…



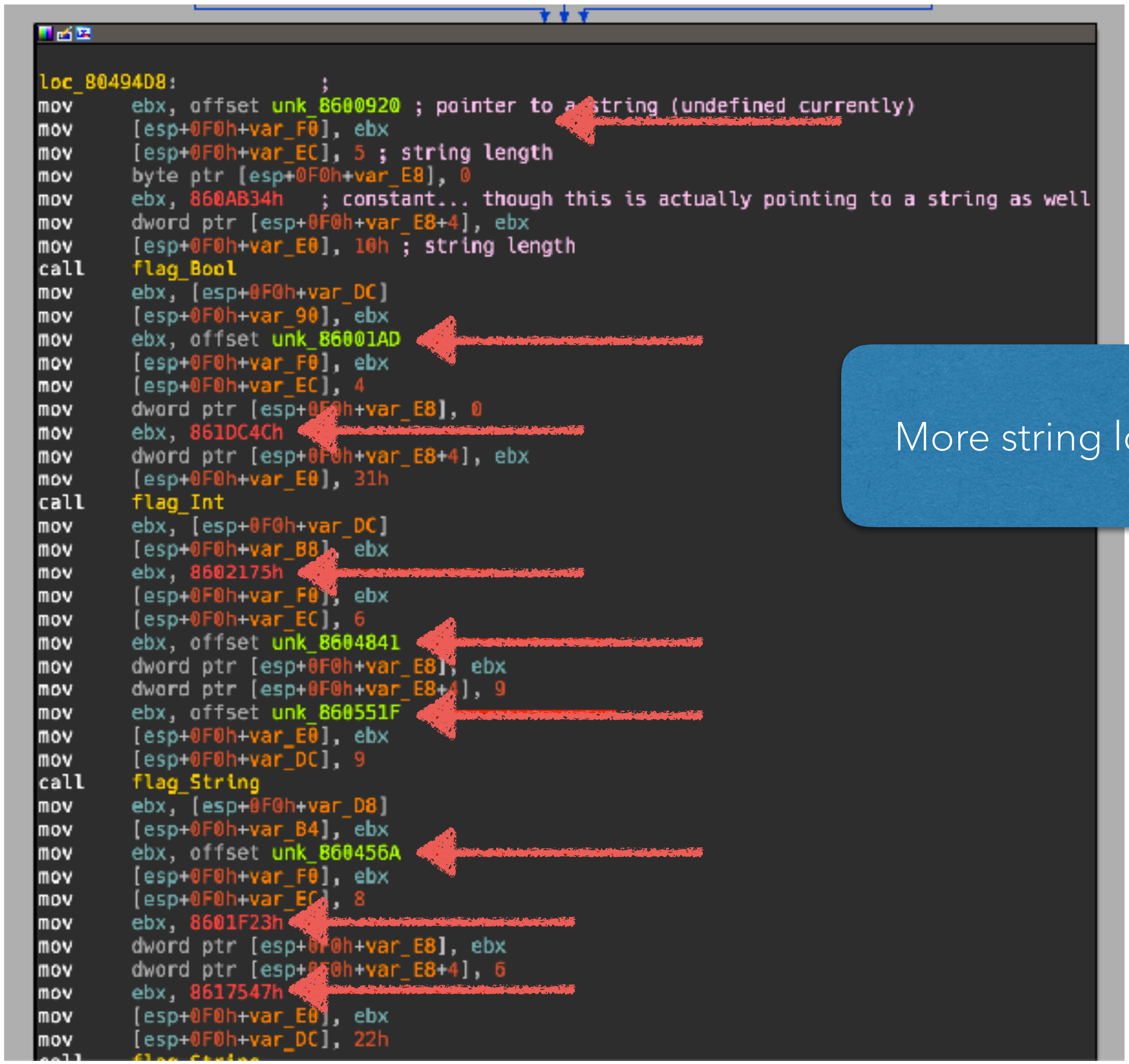
```
loc_82D4929:
mov     eax, ds:dword_84E9728
mov     [esp+48h+var_48], eax
lea     eax, unk_8371C33
mov     [esp+48h+var_44], eax
mov     [esp+48h+var_40], 10h
mov     [esp+48h+var_3C], edx
mov     [esp+48h+var_38], 2
mov     [esp+48h+var_34], 2
call    github_com_op_go_logging__Logger_Debugf
mov     eax, dword_84E4724
test    eax, eax
jnz     short loc_82D496C
```



```
loc_82D4929:
                mov     eax, ds:dword_84E9728
                mov     [esp+48h+var_48], eax
                lea     eax, aIpSOkFromS ; "ip %s ok from %s"
                mov     [esp+48h+var_44], eax
                mov     [esp+48h+var_40], 10h
                mov     [esp+48h+var_3C], edx
                mov     [esp+48h+var_38], 2
                mov     [esp+48h+var_34], 2
                call    github_com_op_go_logging__Logger_Debugf
                mov     eax, dword_84E4724
                test    eax, eax
                jnz     short loc_82D496C
```

STRING LOADS? WHERE ARE YOU?

Reorganize, collect more binaries and whitelist more heuristics

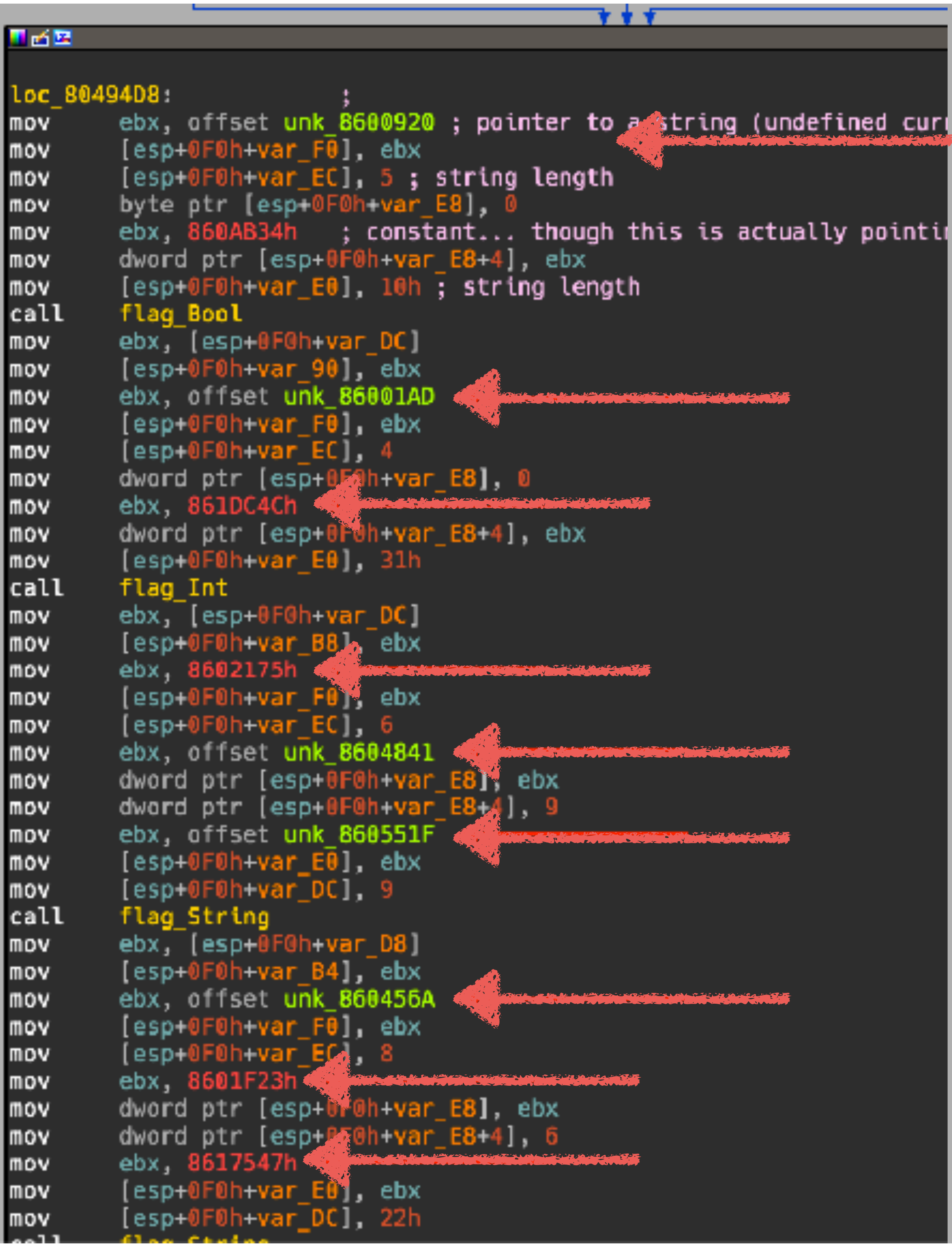


```
loc_80494D8:               ;
mov     ebx, offset unk_8600920 ; pointer to a string (undefined currently)
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 5 ; string length
mov     byte ptr [esp+0F0h+var_E8], 0
mov     ebx, 860AB34h    ; constant... though this is actually pointing to a string as well
mov     dword ptr [esp+0F0h+var_E8+4], ebx
mov     [esp+0F0h+var_E0], 10h ; string length
call    flag_Bool
mov     ebx, [esp+0F0h+var_DC]
mov     [esp+0F0h+var_90], ebx
mov     ebx, offset unk_86001AD
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 4
mov     dword ptr [esp+0F0h+var_E8], 0
mov     ebx, 861DC4Ch
mov     dword ptr [esp+0F0h+var_E8+4], ebx
mov     [esp+0F0h+var_E0], 31h
call    flag_Int
mov     ebx, [esp+0F0h+var_DC]
mov     [esp+0F0h+var_B8], ebx
mov     ebx, 8602175h
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 6
mov     ebx, offset unk_8604841
mov     dword ptr [esp+0F0h+var_E8], ebx
mov     dword ptr [esp+0F0h+var_E8+4], 9
mov     ebx, offset unk_860551F
mov     [esp+0F0h+var_E0], ebx
mov     [esp+0F0h+var_DC], 9
call    flag_String
mov     ebx, [esp+0F0h+var_D8]
mov     [esp+0F0h+var_B4], ebx
mov     ebx, offset unk_860456A
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 8
mov     ebx, 8601F23h
mov     dword ptr [esp+0F0h+var_E8], ebx
mov     dword ptr [esp+0F0h+var_E8+4], 6
mov     ebx, 8617547h
mov     [esp+0F0h+var_E0], ebx
mov     [esp+0F0h+var_DC], 22h
```

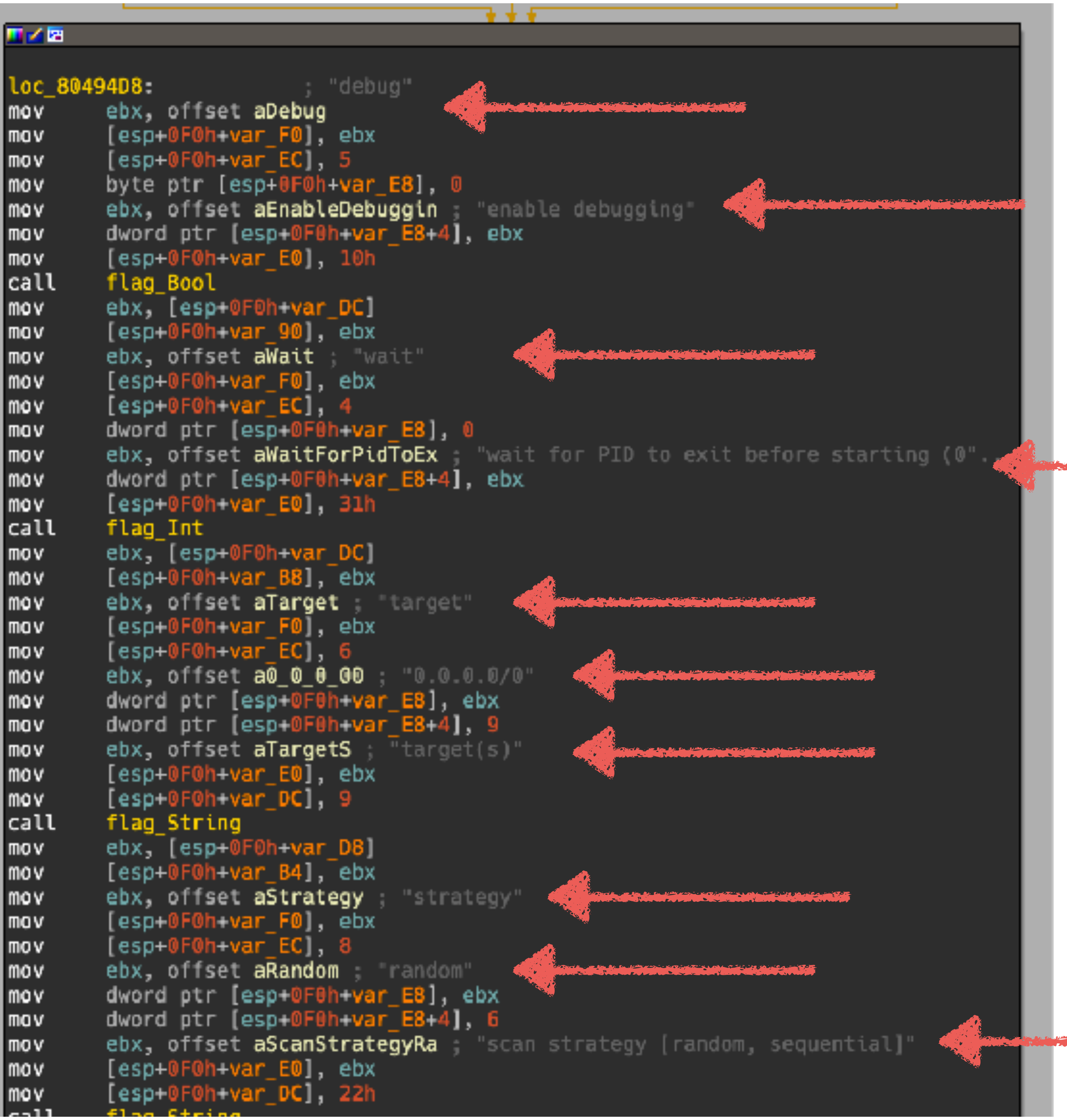
More string loads!

STRING LOADS? WHERE ARE YOU?

Reorganize, collect more binaries and whitelist more heuristics



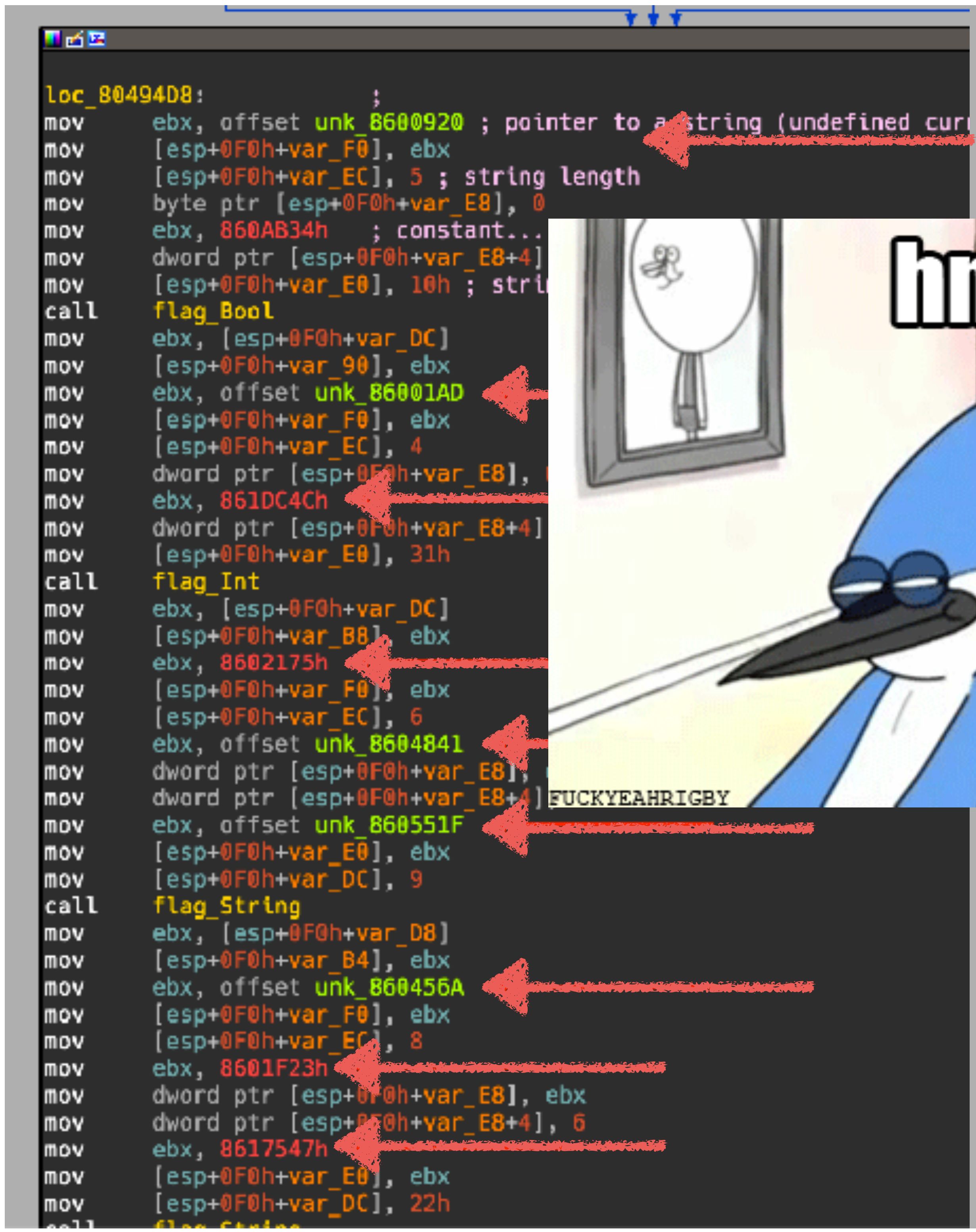
```
loc_80494D8:            ;
mov     ebx, offset unk_8600920 ; pointer to a string (undefined cur
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 5 ; string length
mov     byte ptr [esp+0F0h+var_E8], 0
mov     ebx, 860AB34h   ; constant... though this is actually pointi
mov     dword ptr [esp+0F0h+var_E8+4], ebx
mov     [esp+0F0h+var_E0], 10h ; string length
call    flag_Bool
mov     ebx, [esp+0F0h+var_DC]
mov     [esp+0F0h+var_90], ebx
mov     ebx, offset unk_86001AD
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 4
mov     dword ptr [esp+0F0h+var_E8], 0
mov     ebx, 861DC4Ch
mov     dword ptr [esp+0F0h+var_E8+4], ebx
mov     [esp+0F0h+var_E0], 31h
call    flag_Int
mov     ebx, [esp+0F0h+var_DC]
mov     [esp+0F0h+var_B8], ebx
mov     ebx, 8602175h
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 6
mov     ebx, offset unk_8604841
mov     dword ptr [esp+0F0h+var_E8], ebx
mov     dword ptr [esp+0F0h+var_E8+4], 9
mov     ebx, offset unk_860551F
mov     [esp+0F0h+var_E0], ebx
mov     [esp+0F0h+var_DC], 9
call    flag_String
mov     ebx, [esp+0F0h+var_D8]
mov     [esp+0F0h+var_B4], ebx
mov     ebx, offset unk_860456A
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 8
mov     ebx, 8601F23h
mov     dword ptr [esp+0F0h+var_E8], ebx
mov     dword ptr [esp+0F0h+var_E8+4], 6
mov     ebx, 8617547h
mov     [esp+0F0h+var_E0], ebx
mov     [esp+0F0h+var_DC], 22h
```



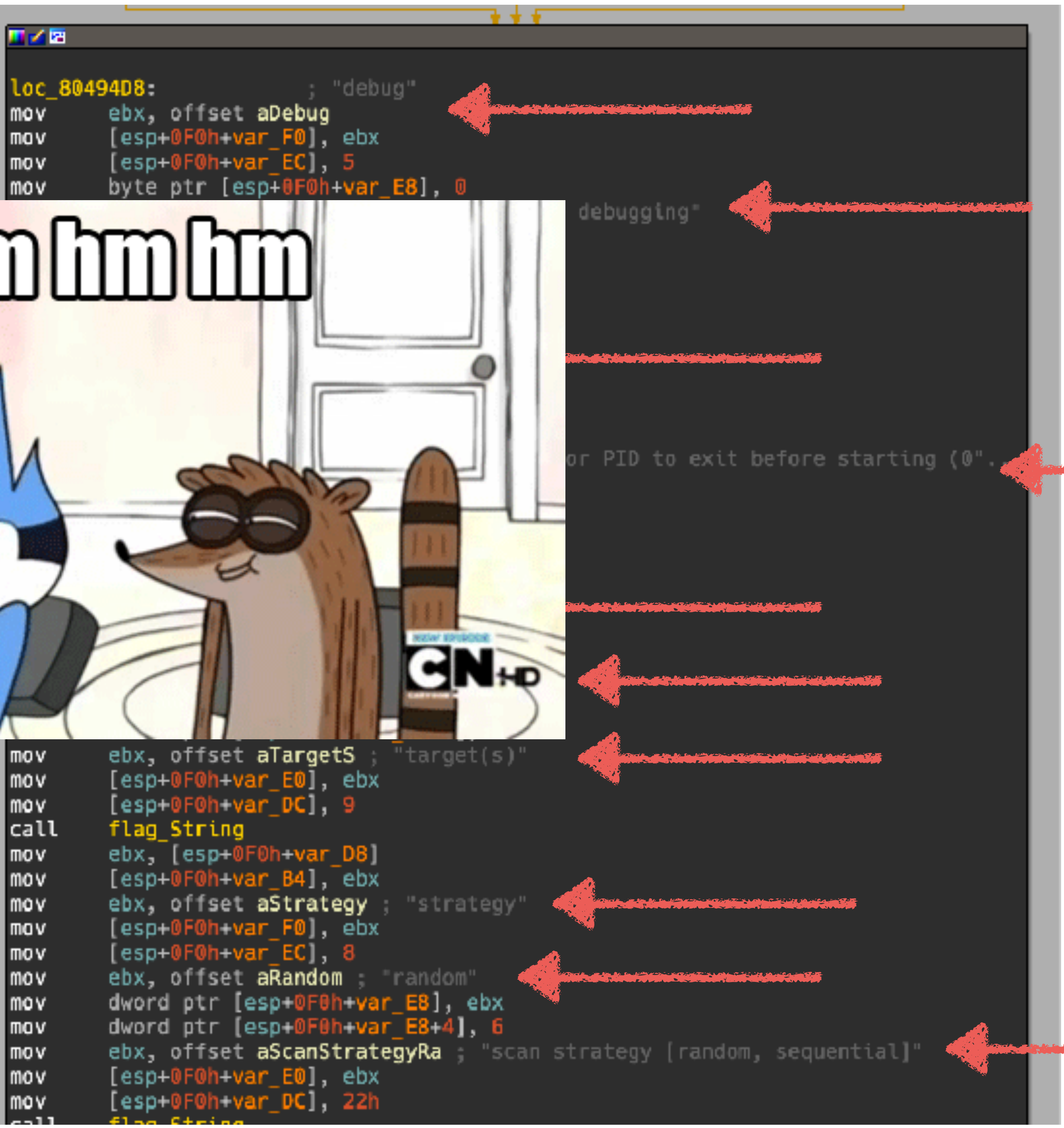
```
loc_80494D8:            ; "debug"
mov     ebx, offset aDebug
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 5
mov     byte ptr [esp+0F0h+var_E8], 0
mov     ebx, offset aEnableDebuggin ; "enable debugging"
mov     dword ptr [esp+0F0h+var_E8+4], ebx
mov     [esp+0F0h+var_E0], 10h
call    flag_Bool
mov     ebx, [esp+0F0h+var_DC]
mov     [esp+0F0h+var_90], ebx
mov     ebx, offset aWait ; "wait"
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 4
mov     dword ptr [esp+0F0h+var_E8], 0
mov     ebx, offset aWaitForPidToEx ; "wait for PID to exit before starting (0"...
mov     dword ptr [esp+0F0h+var_E8+4], ebx
mov     [esp+0F0h+var_E0], 31h
call    flag_Int
mov     ebx, [esp+0F0h+var_DC]
mov     [esp+0F0h+var_B8], ebx
mov     ebx, offset aTarget ; "target"
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 6
mov     ebx, offset a0_0_0_00 ; "0.0.0.0/0"
mov     dword ptr [esp+0F0h+var_E8], ebx
mov     dword ptr [esp+0F0h+var_E8+4], 9
mov     ebx, offset aTargetS ; "target(s)"
mov     [esp+0F0h+var_E0], ebx
mov     [esp+0F0h+var_DC], 9
call    flag_String
mov     ebx, [esp+0F0h+var_D8]
mov     [esp+0F0h+var_B4], ebx
mov     ebx, offset aStrategy ; "strategy"
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 8
mov     ebx, offset aRandom ; "random"
mov     dword ptr [esp+0F0h+var_E8], ebx
mov     dword ptr [esp+0F0h+var_E8+4], 6
mov     ebx, offset aScanStrategyRa ; "scan strategy [random, sequential]"
mov     [esp+0F0h+var_E0], ebx
mov     [esp+0F0h+var_DC], 22h
```

STRING LOADS? WHERE ARE YOU?

Reorganize, collect more binaries and whitelist more heuristics



```
loc_80494D8:                ;
mov     ebx, offset unk_8600920 ; pointer to a string (undefined cur
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 5 ; string length
mov     byte ptr [esp+0F0h+var_E8], 0
mov     ebx, 860AB34h    ; constant...
mov     dword ptr [esp+0F0h+var_E8+4]
mov     [esp+0F0h+var_E0], 10h ; stri
call    flag_Bool
mov     ebx, [esp+0F0h+var_DC]
mov     [esp+0F0h+var_90], ebx
mov     ebx, offset unk_86001AD
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 4
mov     dword ptr [esp+0F0h+var_E8],
mov     ebx, 861DC4Ch
mov     dword ptr [esp+0F0h+var_E8+4]
mov     [esp+0F0h+var_E0], 31h
call    flag_Int
mov     ebx, [esp+0F0h+var_DC]
mov     [esp+0F0h+var_B8], ebx
mov     ebx, 8602175h
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 6
mov     ebx, offset unk_8604841
mov     dword ptr [esp+0F0h+var_E8],
mov     dword ptr [esp+0F0h+var_E8+4]
mov     ebx, offset unk_860551F
mov     [esp+0F0h+var_E0], ebx
mov     [esp+0F0h+var_DC], 9
call    flag_String
mov     ebx, [esp+0F0h+var_D8]
mov     [esp+0F0h+var_B4], ebx
mov     ebx, offset unk_860456A
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 8
mov     ebx, 8601F23h
mov     dword ptr [esp+0F0h+var_E8], ebx
mov     dword ptr [esp+0F0h+var_E8+4], 6
mov     ebx, 8617547h
mov     [esp+0F0h+var_E0], ebx
mov     [esp+0F0h+var_DC], 22h
```



```
loc_80494D8:                ; "debug"
mov     ebx, offset aDebug
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 5
mov     byte ptr [esp+0F0h+var_E8], 0
                                  debugging"
                                or PID to exit before starting (0".
mov     ebx, offset aTargetS ; "target(s)"
mov     [esp+0F0h+var_E0], ebx
mov     [esp+0F0h+var_DC], 9
call    flag_String
mov     ebx, [esp+0F0h+var_D8]
mov     [esp+0F0h+var_B4], ebx
mov     ebx, offset aStrategy ; "strategy"
mov     [esp+0F0h+var_F0], ebx
mov     [esp+0F0h+var_EC], 8
mov     ebx, offset aRandom ; "random"
mov     dword ptr [esp+0F0h+var_E8], ebx
mov     dword ptr [esp+0F0h+var_E8+4], 6
mov     ebx, offset aScanStrategyRa ; "scan strategy [random, sequential]"
mov     [esp+0F0h+var_E0], ebx
mov     [esp+0F0h+var_DC], 22h
```

FIXING OUR TOOLS…

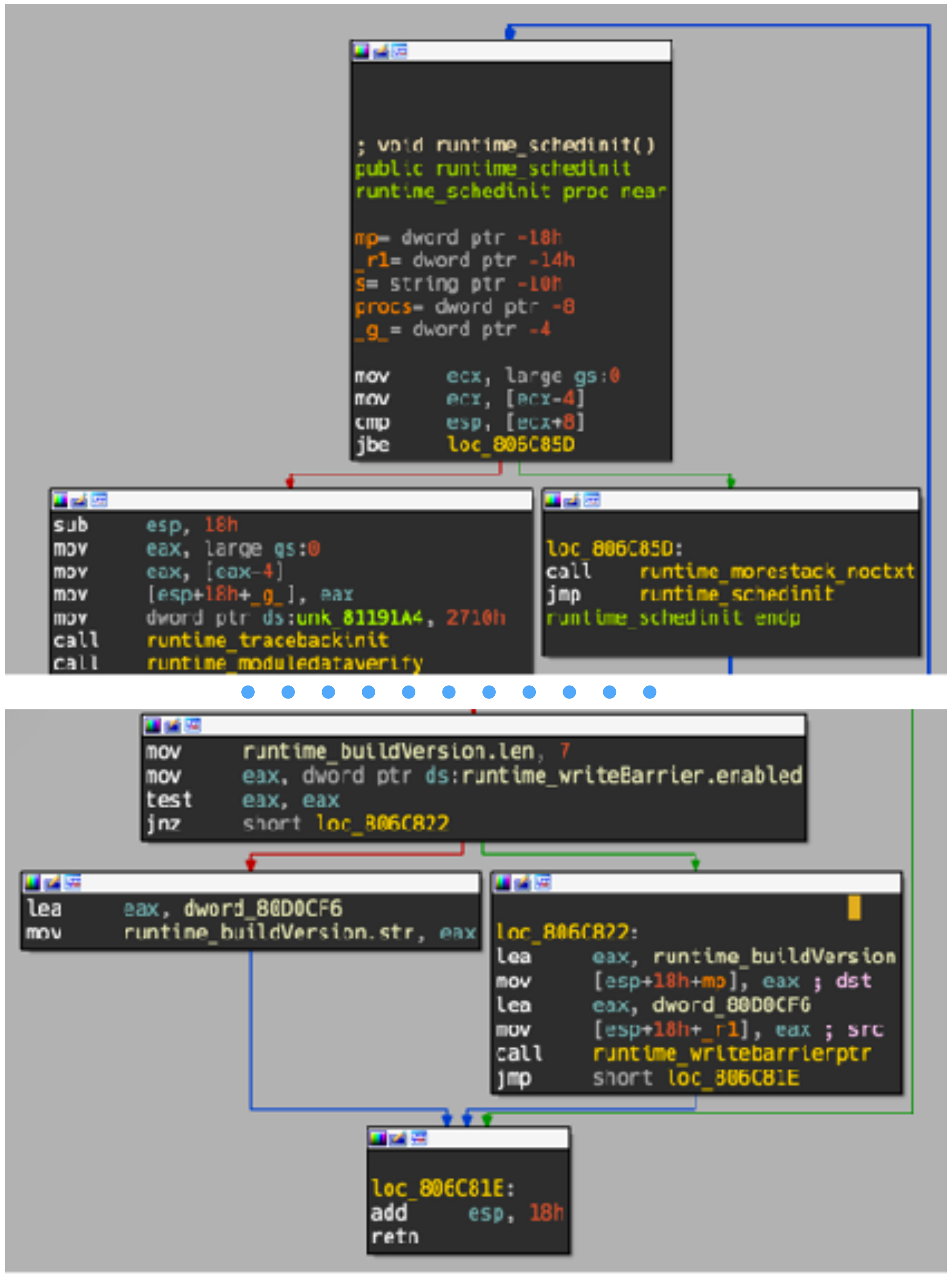
Issues Identified

- ~~Functions are not all easily defined~~
- ~~Functions do not retain their name when stripped~~
- ~~String loads can be funky - dependent on architecture and Go version~~
- For above, we need to easily identify Go version!

HEURISTICS BREAK ON NEW REVISIONS

Damn it, nothing is ever easy…

- Heuristics can be brittle as Go evolves
- Runtime has versioning!



```
; void runtime_schedinit()
public runtime_schedinit
runtime_schedinit proc near

mp= dword ptr -18h
_r1= dword ptr -14h
s= string ptr -10h
procs= dword ptr -8
_g_ = dword ptr -4

mov     ecx, large gs:0
mov     ecx, [ecx-4]
cmp     esp, [ecx+8]
jbe     loc_806C85D
```

```
sub     esp, 18h
mov     eax, large gs:0
mov     eax, [eax-4]
mov     [esp+18h+_g_], eax
mov     dword ptr ds:unk_81191A4, 2710h
call    runtime_tracebackinit
call    runtime_moduledataverify
```

```
loc_806C85D:
call    runtime_morestack_noctxt
jmp     runtime_schedinit
runtime_schedinit endp
```

```
mov     runtime_buildVersion.len, 7
mov     eax, dword ptr ds:runtime_writeBarrier.enabled
test    eax, eax
jnz     short loc_806C822
```

```
lea     eax, dword_80D0CF6
mov     runtime_buildVersion.str, eax
```

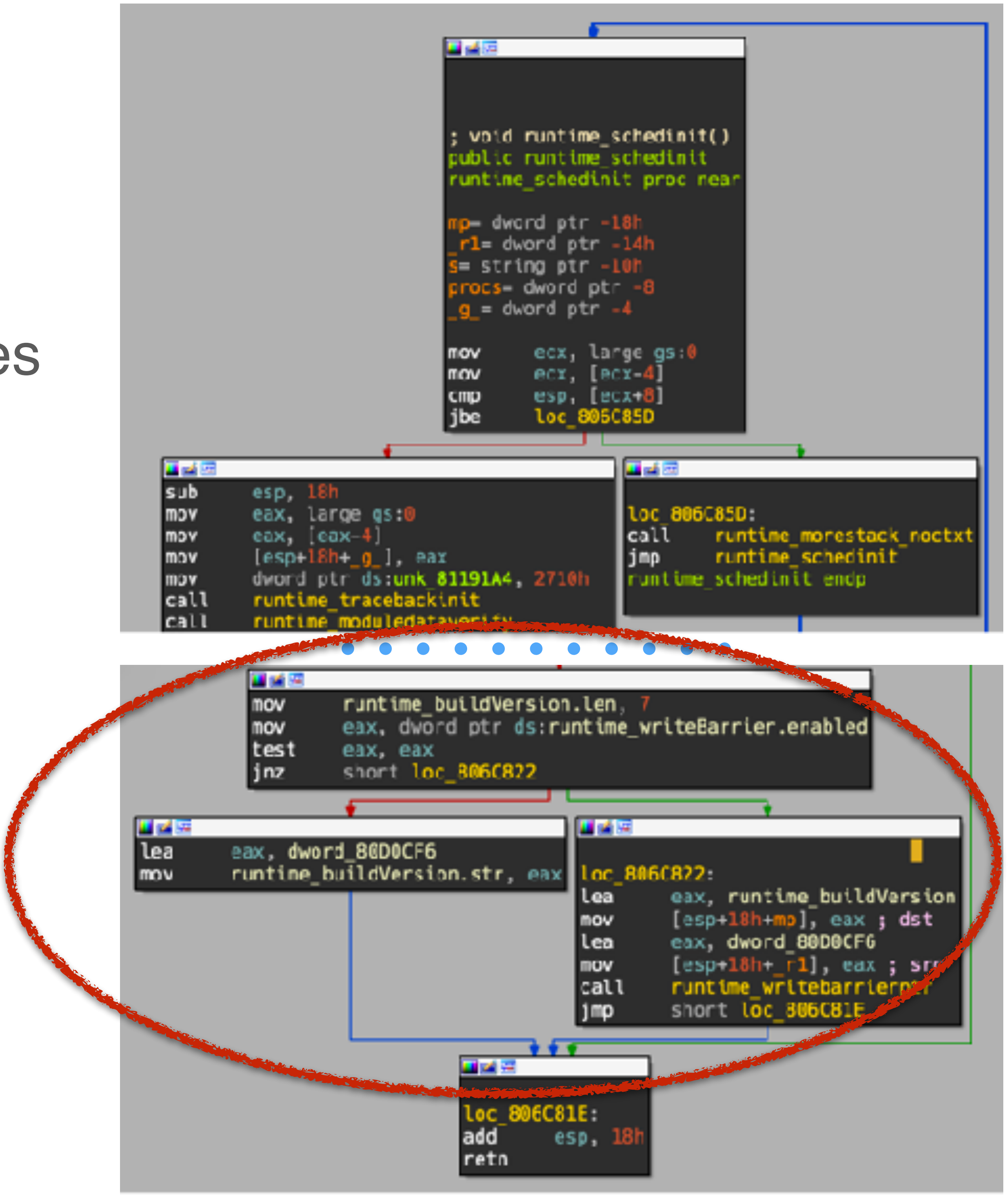
```
loc_806C822:
lea     eax, runtime_buildVersion
mov     [esp+18h+mp], eax ; dst
lea     eax, dword_80D0CF6
mov     [esp+18h+_r1], eax ; src
call    runtime_writebarrierptr
jmp     short loc_806C81E
```

```
loc_806C81E:
add     esp, 18h
retn
```

HEURISTICS BREAK ON NEW REVISIONS

Damn it, nothing is ever easy…

- Heuristics can be brittle as Go evolves
- Runtime has versioning!



```
; void runtime_schedinit()
public runtime_schedinit
runtime_schedinit proc near

mp= dword ptr -18h
_r1= dword ptr -14h
s= string ptr -10h
procs= dword ptr -8
_g_= dword ptr -4

mov     ecx, large gs:0
mov     ecx, [ecx-4]
cmp     esp, [ecx+8]
jbe     loc_806C85D
```

```
sub     esp, 18h
mov     eax, large gs:0
mov     eax, [eax-4]
mov     [esp+18h+_g_], eax
mov     dword ptr ds:unk_81191A4, 2710h
call    runtime_tracebackinit
call    runtime_moduledataverify
```

```
loc_806C85D:
call    runtime_morestack_noctxt
jmp     runtime_schedinit
runtime_schedinit endp
```

```
mov     runtime_buildVersion.len, 7
mov     eax, dword ptr ds:runtime_writeBarrier.enabled
test    eax, eax
jnz     short loc_806C822
```

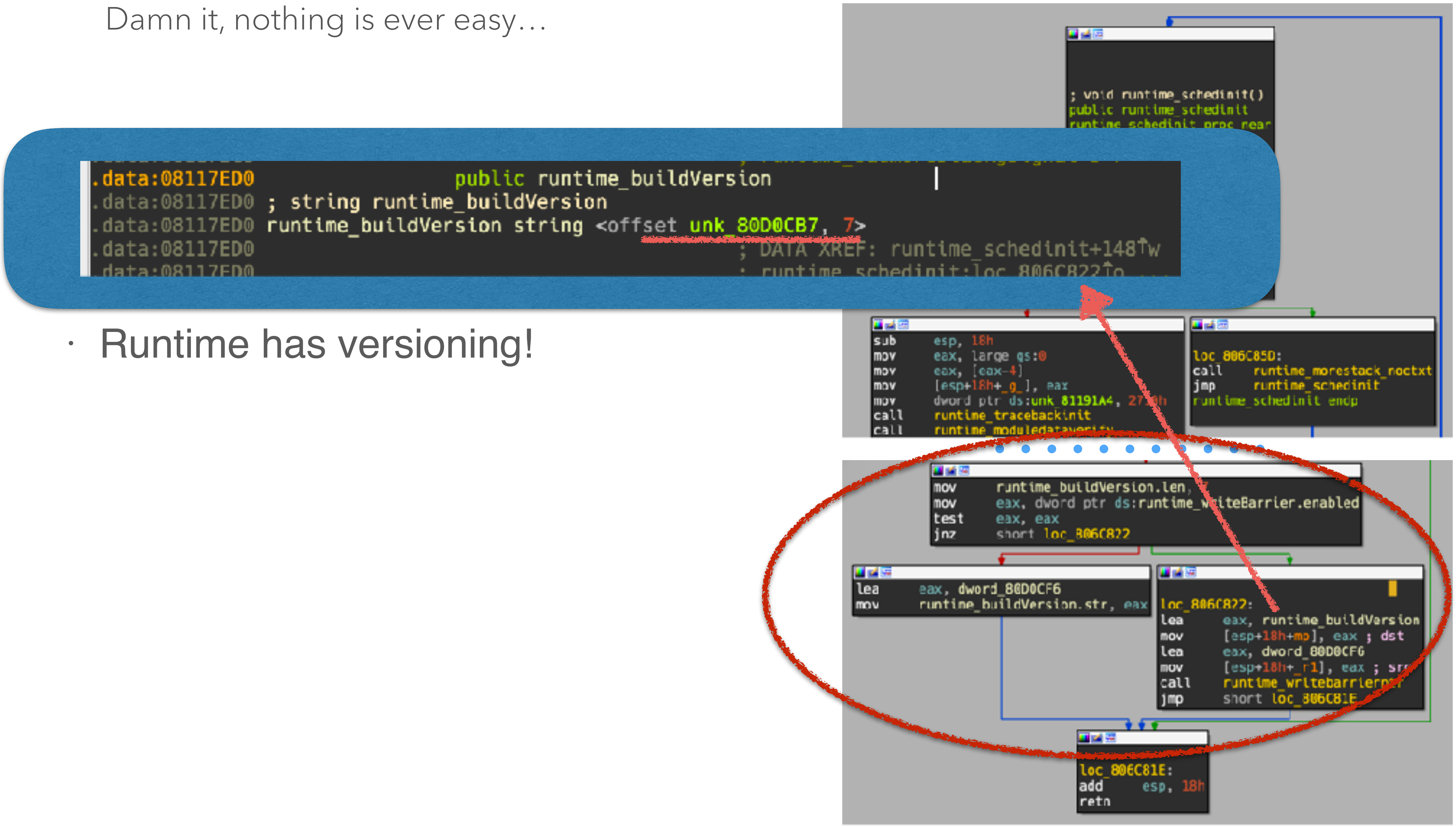
```
lea     eax, dword_80D0CF6
mov     runtime_buildVersion.str, eax
```

```
loc_806C822:
lea     eax, runtime_buildVersion
mov     [esp+18h+mp], eax ; dst
lea     eax, dword_80D0CF6
mov     [esp+18h+_r1], eax ; sr
call    runtime_writebarrierptr
jmp     short loc_806C81E
```

```
loc_806C81E:
add     esp, 18h
retn
```

HEURISTICS BREAK ON NEW REVISIONS

Damn it, nothing is ever easy…



```
.data:08117ED0                 public runtime_buildVersion
.data:08117ED0 ; string runtime_buildVersion
.data:08117ED0 runtime_buildVersion string <offset unk_80D0CB7, 7>
.data:08117ED0                                         ; DATA XREF: runtime_schedinit+148↑w
.data:08117ED0                                         ; runtime_schedinit:loc_806C822↑o ...
```

```
; void runtime_schedinit()
public runtime_schedinit
runtime_schedinit proc near
```

```
sub     esp, 18h
mov     eax, large gs:0
mov     eax, [eax-4]
mov     [esp+18h+_g_], eax
mov     dword ptr ds:unk_81191A4, 2710h
call    runtime_tracebackinit
call    runtime_moduledataverify
```

```
loc_806C85D:
call    runtime_morestack_noctxt
jmp     runtime_schedinit
runtime_schedinit endp
```

```
mov     runtime_buildVersion.len, 7
mov     eax, dword ptr ds:runtime_writeBarrier.enabled
test    eax, eax
jnz     short loc_806C822
```

```
lea     eax, dword_80D0CF6
mov     runtime_buildVersion.str, eax
```

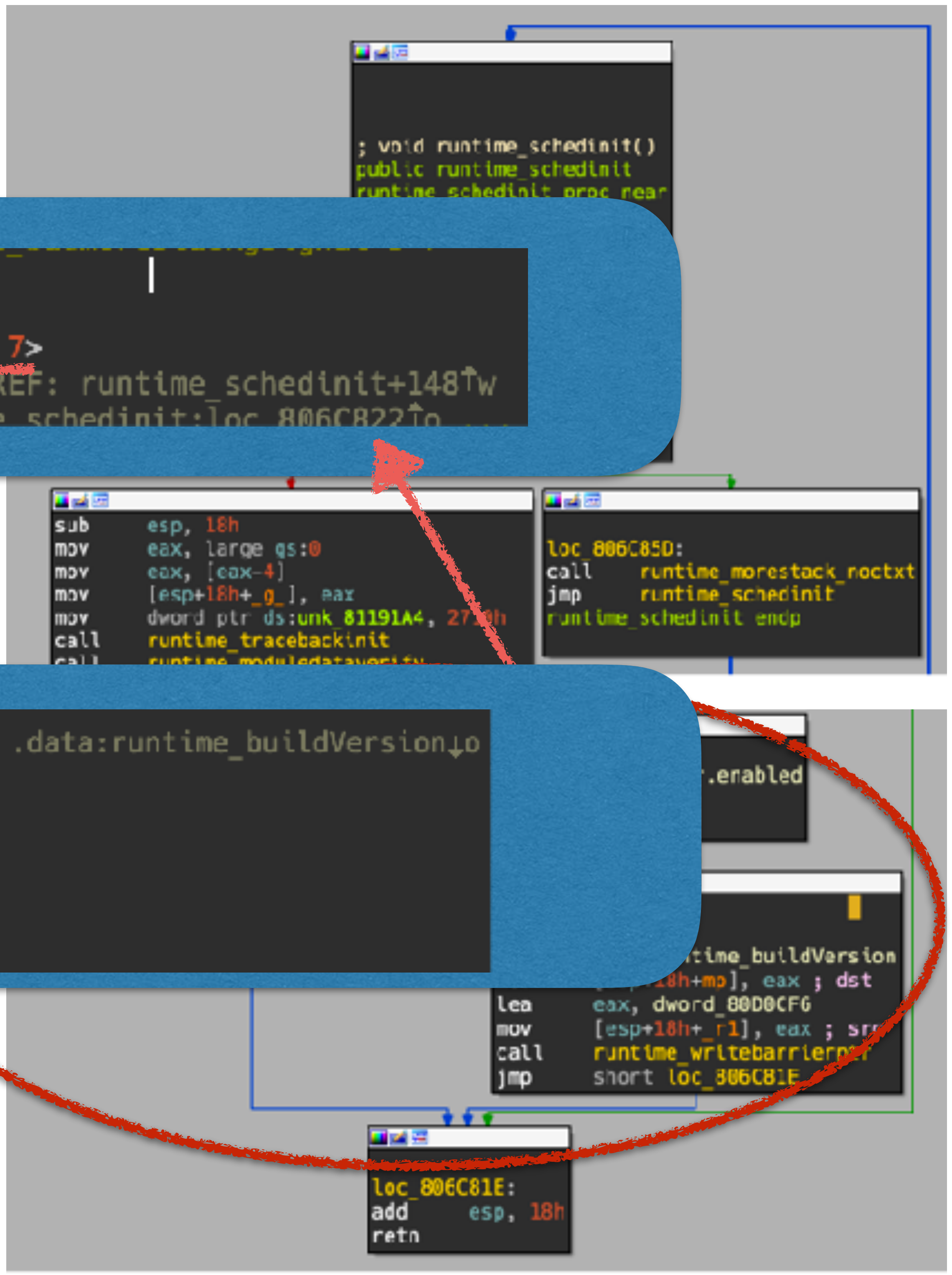
```
loc_806C822:
lea     eax, runtime_buildVersion
mov     [esp+18h+mp], eax ; dst
lea     eax, dword_80D0CF6
mov     [esp+18h+_r1], eax ; src
call    runtime_writebarrierptr
jmp     short loc_806C81E
```

```
loc_806C81E:
add     esp, 18h
retn
```

- Runtime has versioning!

HEURISTICS BREAK ON NEW REVISIONS

Damn it, nothing is ever easy…



```
; void runtime_schedinit()
public runtime_schedinit
runtime_schedinit proc near
```

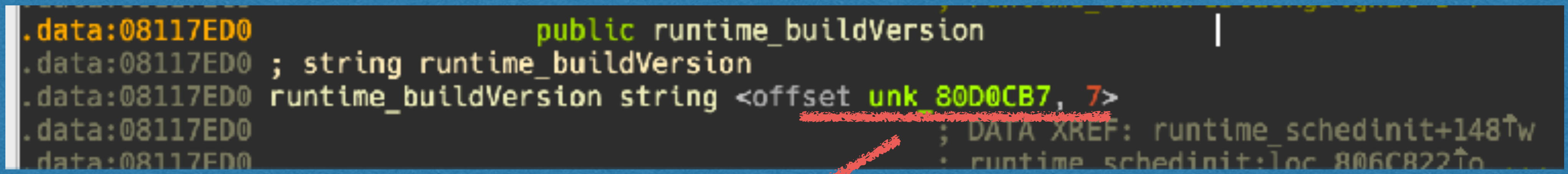
```
sub     esp, 18h
mov     eax, large gs:0
mov     eax, [eax-4]
mov     [esp+18h+_g_], eax
mov     dword ptr ds:unk_81191A4, 2710h
call    runtime_tracebackinit
```

```
loc_806C85D:
call    runtime_morestack_noctxt
jmp     runtime_schedinit
runtime_schedinit endp
```

```
.enabled
```

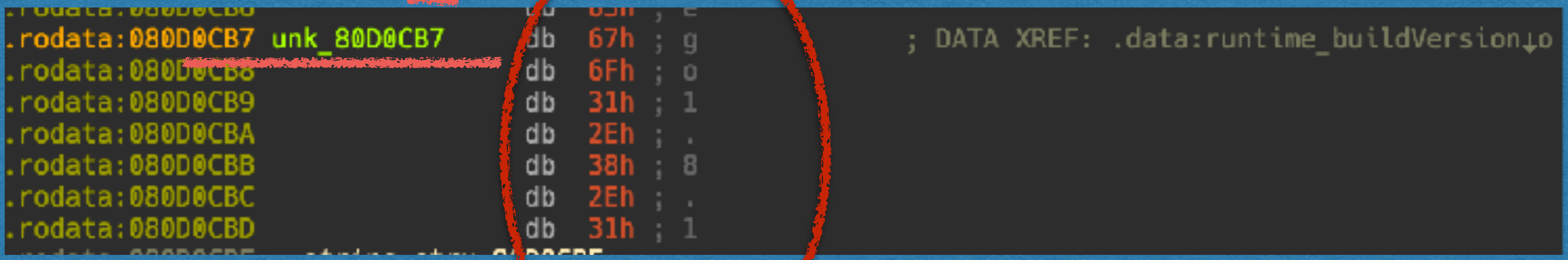
```
time_buildVersion
        18h+mp], eax ; dst
lea     eax, dword_80D0CF6
mov     [esp+18h+_r1], eax ; src
call    runtime_writebarrierptr
jmp     short loc_806C81E
```

```
loc_806C81E:
add     esp, 18h
retn
```



```
.data:08117ED0                 public runtime_buildVersion
.data:08117ED0 ; string runtime_buildVersion
.data:08117ED0 runtime_buildVersion string <offset unk_80D0CB7, 7>
.data:08117ED0                                         ; DATA XREF: runtime_schedinit+148↑w
.data:08117ED0                                         ; runtime_schedinit:loc_806C822↑o ...
```

- Runtime has versioning!

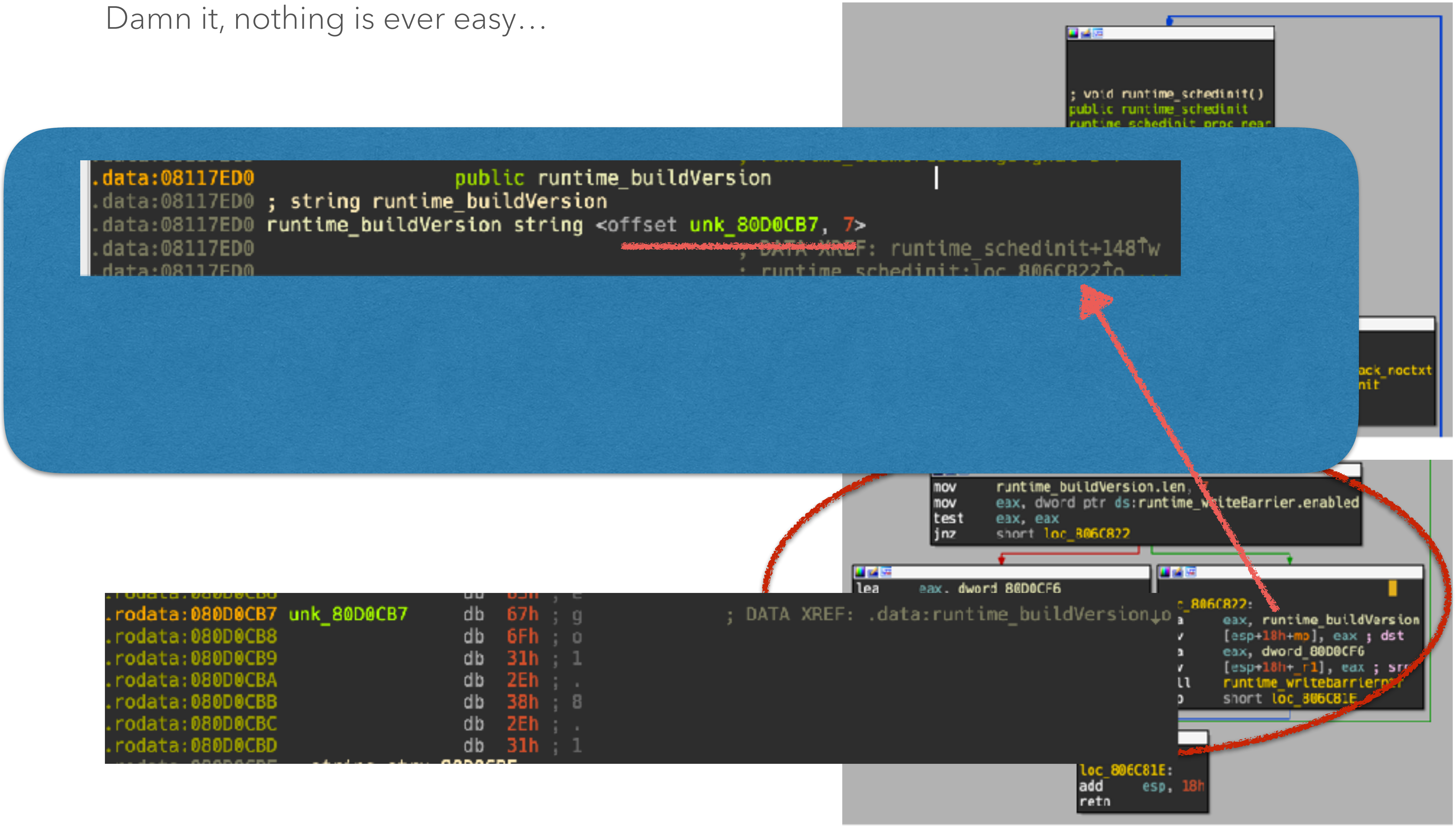


```
.rodata:080D0CB7 unk_80D0CB7     db  67h ; g            ; DATA XREF: .data:runtime_buildVersion↓o
.rodata:080D0CB8                 db  6Fh ; o
.rodata:080D0CB9                 db  31h ; 1
.rodata:080D0CBA                 db  2Eh ; .
.rodata:080D0CBB                 db  38h ; 8
.rodata:080D0CBC                 db  2Eh ; .
.rodata:080D0CBD                 db  31h ; 1
```

Can now programmatically tell the Go version!

HEURISTICS BREAK ON NEW REVISIONS

Damn it, nothing is ever easy…



```
.data:08117ED0                 public runtime_buildVersion
.data:08117ED0 ; string runtime_buildVersion
.data:08117ED0 runtime_buildVersion string <offset unk_80D0CB7, 7>
.data:08117ED0                                         ~~; DATA XREF~~: runtime_schedinit+148↑w
.data:08117ED0                                         ; runtime_schedinit:loc_806C822↑o ...
```

```
; void runtime_schedinit()
public runtime_schedinit
runtime_schedinit proc near
```

```
ack_noctxt
nit
```

```
mov     runtime_buildVersion.len, 7
mov     eax, dword ptr ds:runtime_writeBarrier.enabled
test    eax, eax
jnz     short loc_806C822
```

```
lea     eax, dword 80D0CF6
```

```
loc_806C822:
        eax, runtime_buildVersion
        [esp+18h+mp], eax ; dst
        eax, dword_80D0CF6
        [esp+18h+_r1], eax ; src
        runtime_writebarrierptr
        short loc_806C81E
```

```
loc_806C81E:
add     esp, 18h
retn
```

```
.rodata:080D0CB7 unk_80D0CB7     db  67h ; g              ; DATA XREF: .data:runtime_buildVersion↓o
.rodata:080D0CB8                 db  6Fh ; o
.rodata:080D0CB9                 db  31h ; 1
.rodata:080D0CBA                 db  2Eh ; .
.rodata:080D0CBB                 db  38h ; 8
.rodata:080D0CBC                 db  2Eh ; .
.rodata:080D0CBD                 db  31h ; 1
```

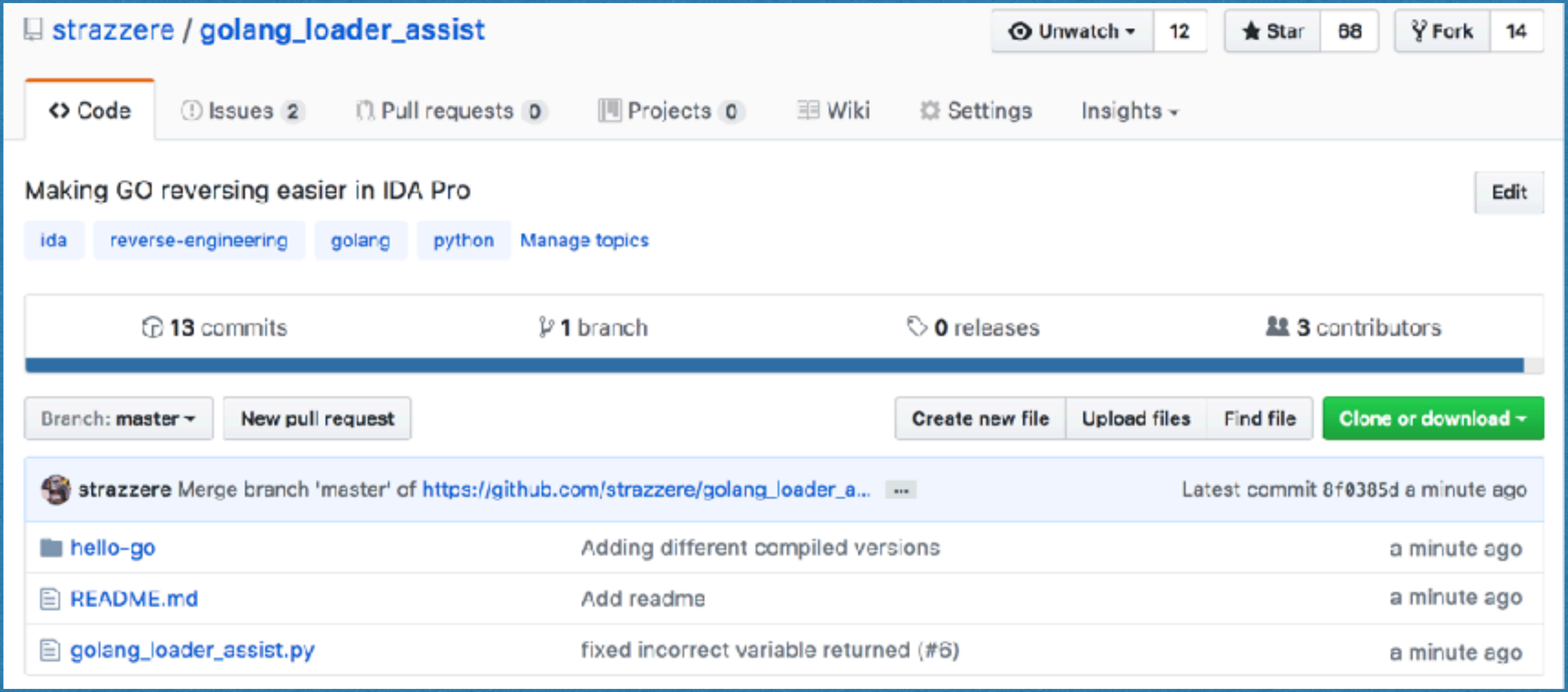
FIXING OUR TOOLS…

Issues Identified

- ~~Functions are not all easily defined~~
- ~~Functions do not retain their name when stripped~~
- ~~String loads can be funky - dependent on architecture and Go version~~
- ~~For above, we need to easily identify Go version!~~

OPEN SOURCE, YAY!

Issues, handled :D


strazzere / golang_loader_assist

Making GO reversing easier in IDA Pro

13 commits · 1 branch · 0 releases · 3 contributors

strazzere Merge branch 'master' of https://github.com/strazzere/golang_loader_a... Latest commit 8f0385d a minute ago

hello-go	Adding different compiled versions	a minute ago
README.md	Add readme	a minute ago
golang_loader_assist.py	fixed incorrect variable returned (#6)	a minute ago

MAKING ISSUES…

Issues, handled :D

- If stripping doesn't protect binaries…
  How can we make life hard?
- Let's prepare for the next "Bear" APT
- Or at least make people
  step up their game…

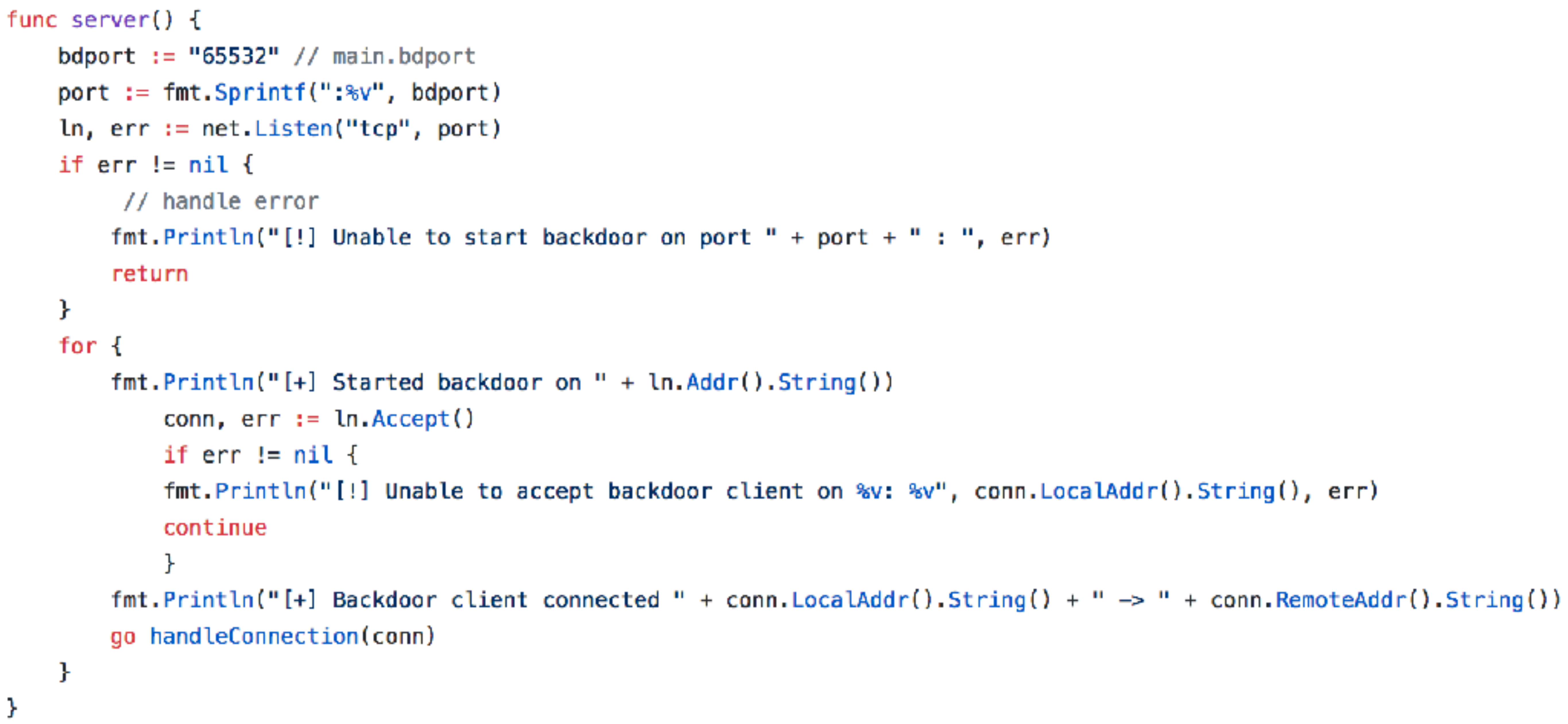
GOGUARD!

Oh yay… Only seems fitting since Go acts like Java :D

- Preprocess source using github.com/lunixbochs/og
- Build AST of all classes, functions and variables
- Obfuscate!
- Next release will contain encryption

GOGUARD!

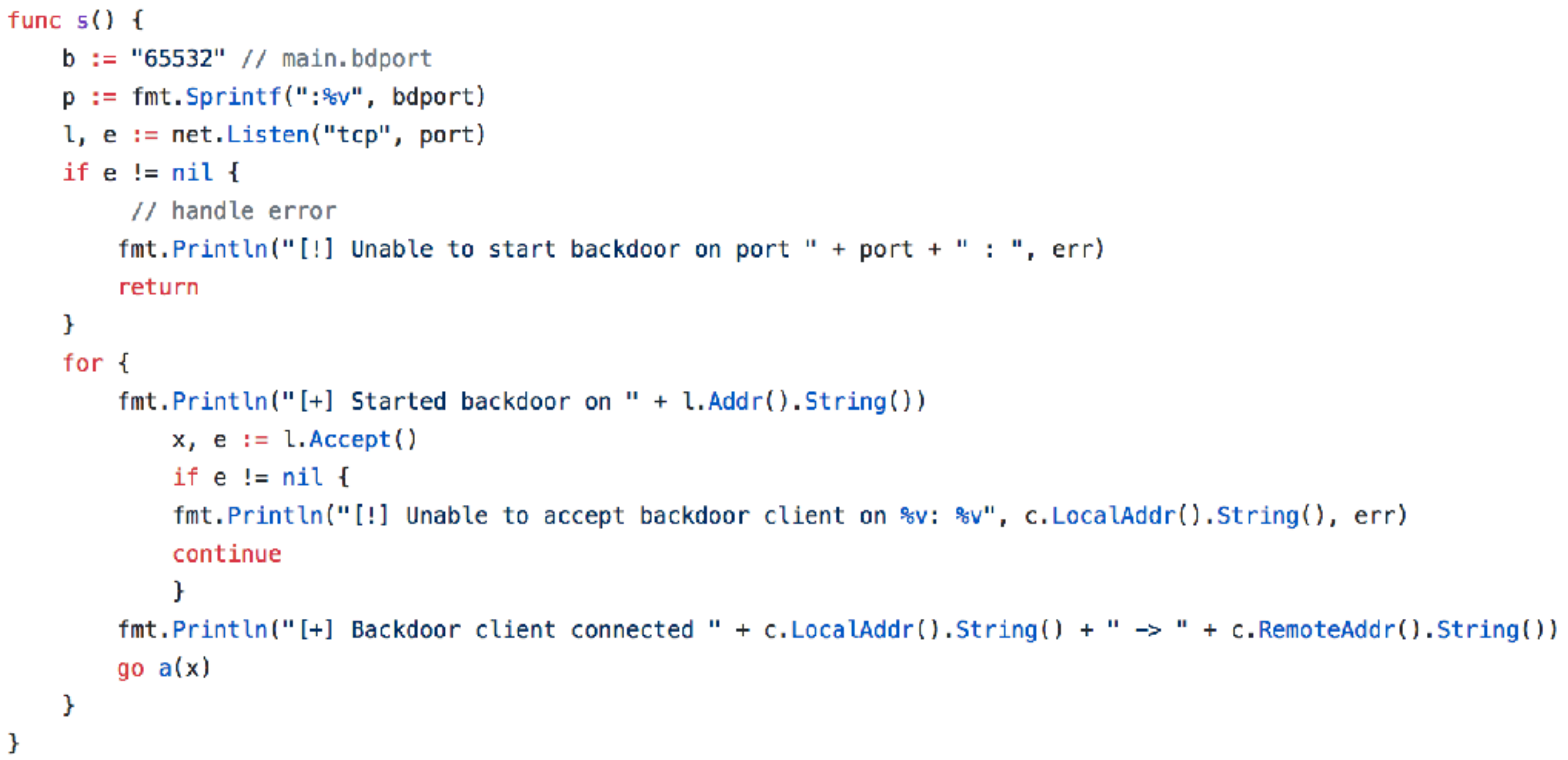
Oh yay… Only seems fitting since Go acts like Java :D



```go
func server() {
    bdport := "65532" // main.bdport
    port := fmt.Sprintf(":%v", bdport)
    ln, err := net.Listen("tcp", port)
    if err != nil {
         // handle error
        fmt.Println("[!] Unable to start backdoor on port " + port + " : ", err)
        return
    }
    for {
        fmt.Println("[+] Started backdoor on " + ln.Addr().String())
            conn, err := ln.Accept()
            if err != nil {
            fmt.Println("[!] Unable to accept backdoor client on %v: %v", conn.LocalAddr().String(), err)
            continue
            }
        fmt.Println("[+] Backdoor client connected " + conn.LocalAddr().String() + " -> " + conn.RemoteAddr().String())
        go handleConnection(conn)
    }
}
```

GOGUARD!

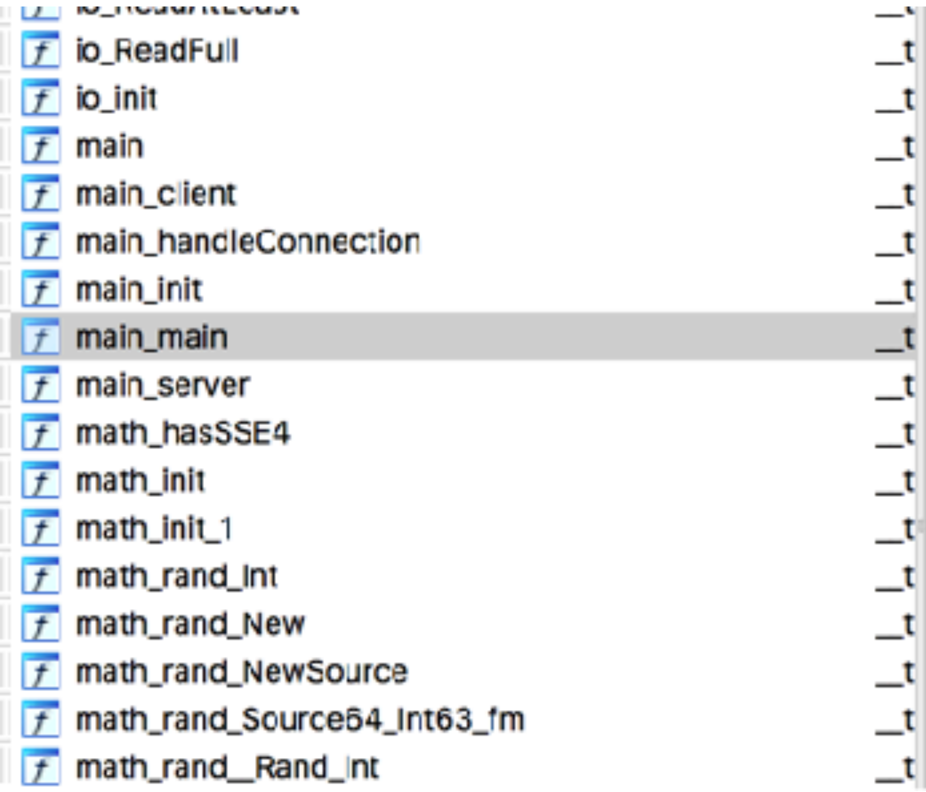
Oh yay… Only seems fitting since Go acts like Java :D



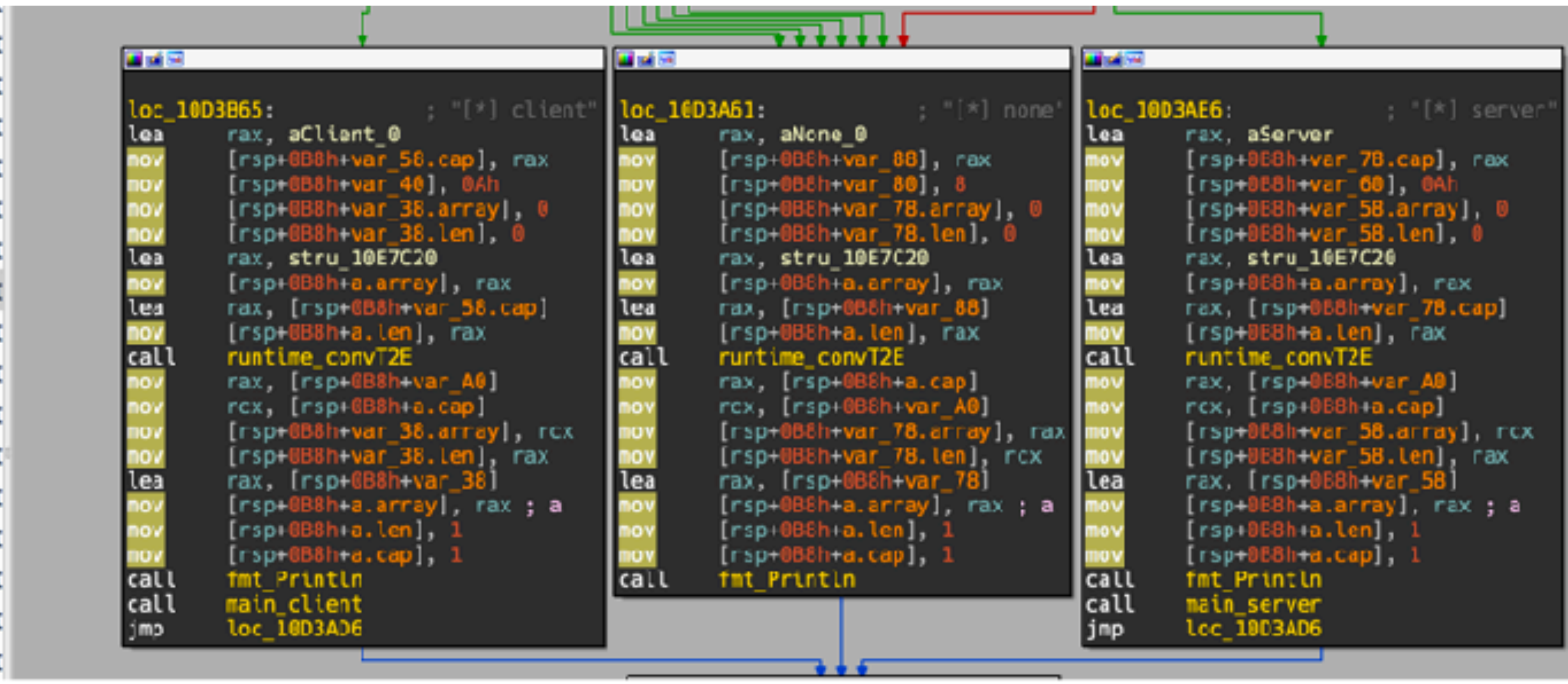
```
func s() {
    b := "65532" // main.bdport
    p := fmt.Sprintf(":%v", bdport)
    l, e := net.Listen("tcp", port)
    if e != nil {
         // handle error
        fmt.Println("[!] Unable to start backdoor on port " + port + " : ", err)
        return
    }
    for {
        fmt.Println("[+] Started backdoor on " + l.Addr().String())
            x, e := l.Accept()
            if e != nil {
            fmt.Println("[!] Unable to accept backdoor client on %v: %v", c.LocalAddr().String(), err)
            continue
            }
        fmt.Println("[+] Backdoor client connected " + c.LocalAddr().String() + " -> " + c.RemoteAddr().String())
        go a(x)
    }
}
```

GOGUARD!

Oh yay… Only seems fitting since Go acts like Java :D



```
io_ReadFull
io_init
main
main_client
main_handleConnection
main_init
main_main
main_server
math_hasSSE4
math_init
math_init_1
math_rand_Int
math_rand_New
math_rand_NewSource
math_rand_Source64_Int63_fm
math_rand__Rand_Int
```



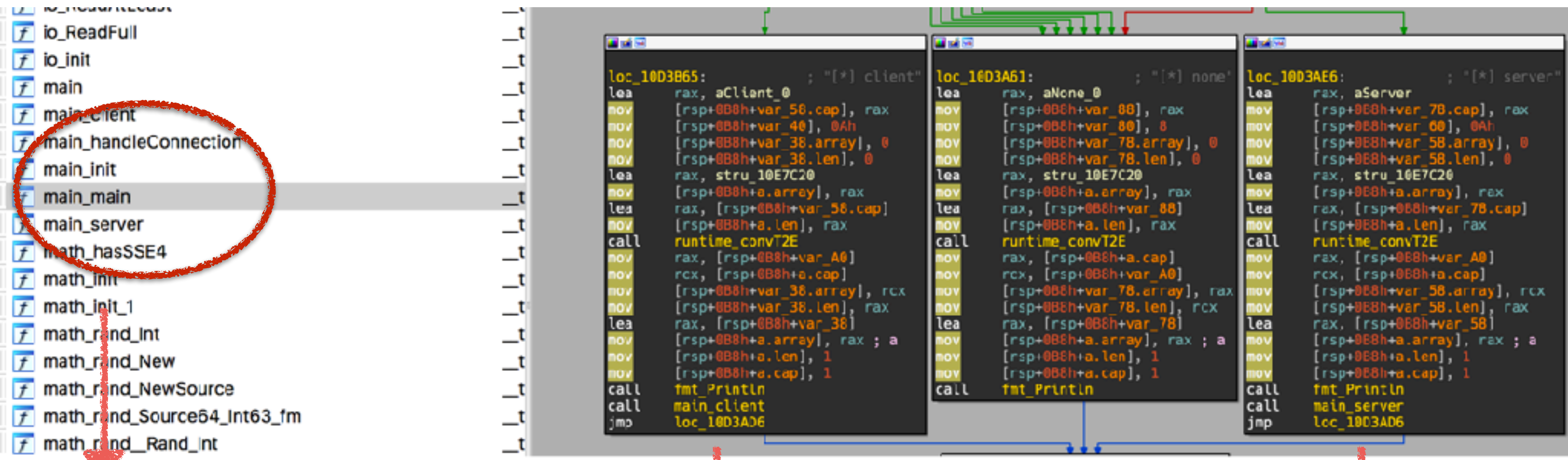
```
loc_10D3B65:                          ; "[*] client"
lea     rax, aClient_0
mov     [rsp+0B8h+var_58.cap], rax
mov     [rsp+0B8h+var_40], 0Ah
mov     [rsp+0B8h+var_38.array], 0
mov     [rsp+0B8h+var_38.len], 0
lea     rax, stru_10E7C20
mov     [rsp+0B8h+a.array], rax
lea     rax, [rsp+0B8h+var_58.cap]
mov     [rsp+0B8h+a.len], rax
call    runtime_convT2E
mov     rax, [rsp+0B8h+var_A0]
mov     rcx, [rsp+0B8h+a.cap]
mov     [rsp+0B8h+var_38.array], rcx
mov     [rsp+0B8h+var_38.len], rax
lea     rax, [rsp+0B8h+var_38]
mov     [rsp+0B8h+a.array], rax ; a
mov     [rsp+0B8h+a.len], 1
mov     [rsp+0B8h+a.cap], 1
call    fmt_Println
call    main_client
jmp     loc_10D3AD6
```

```
loc_10D3A61:                          ; "[*] none"
lea     rax, aNone_0
mov     [rsp+0B8h+var_88], rax
mov     [rsp+0B8h+var_80], 8
mov     [rsp+0B8h+var_78.array], 0
mov     [rsp+0B8h+var_78.len], 0
lea     rax, stru_10E7C20
mov     [rsp+0B8h+a.array], rax
lea     rax, [rsp+0B8h+var_88]
mov     [rsp+0B8h+a.len], rax
call    runtime_convT2E
mov     rax, [rsp+0B8h+a.cap]
mov     rcx, [rsp+0B8h+var_A0]
mov     [rsp+0B8h+var_78.array], rax
mov     [rsp+0B8h+var_78.len], rcx
lea     rax, [rsp+0B8h+var_78]
mov     [rsp+0B8h+a.array], rax ; a
mov     [rsp+0B8h+a.len], 1
mov     [rsp+0B8h+a.cap], 1
call    fmt_Println
```

```
loc_10D3AE6:                          ; "[*] server"
lea     rax, aServer
mov     [rsp+0B8h+var_78.cap], rax
mov     [rsp+0B8h+var_60], 0Ah
mov     [rsp+0B8h+var_58.array], 0
mov     [rsp+0B8h+var_58.len], 0
lea     rax, stru_10E7C20
mov     [rsp+0B8h+a.array], rax
lea     rax, [rsp+0B8h+var_78.cap]
mov     [rsp+0B8h+a.len], rax
call    runtime_convT2E
mov     rax, [rsp+0B8h+var_A0]
mov     rcx, [rsp+0B8h+a.cap]
mov     [rsp+0B8h+var_58.array], rcx
mov     [rsp+0B8h+var_58.len], rax
lea     rax, [rsp+0B8h+var_58]
mov     [rsp+0B8h+a.array], rax ; a
mov     [rsp+0B8h+a.len], 1
mov     [rsp+0B8h+a.cap], 1
call    fmt_Println
call    main_server
jmp     loc_10D3AD6
```

GOGUARD!

Oh yay… Only seems fitting since Go acts like Java :D

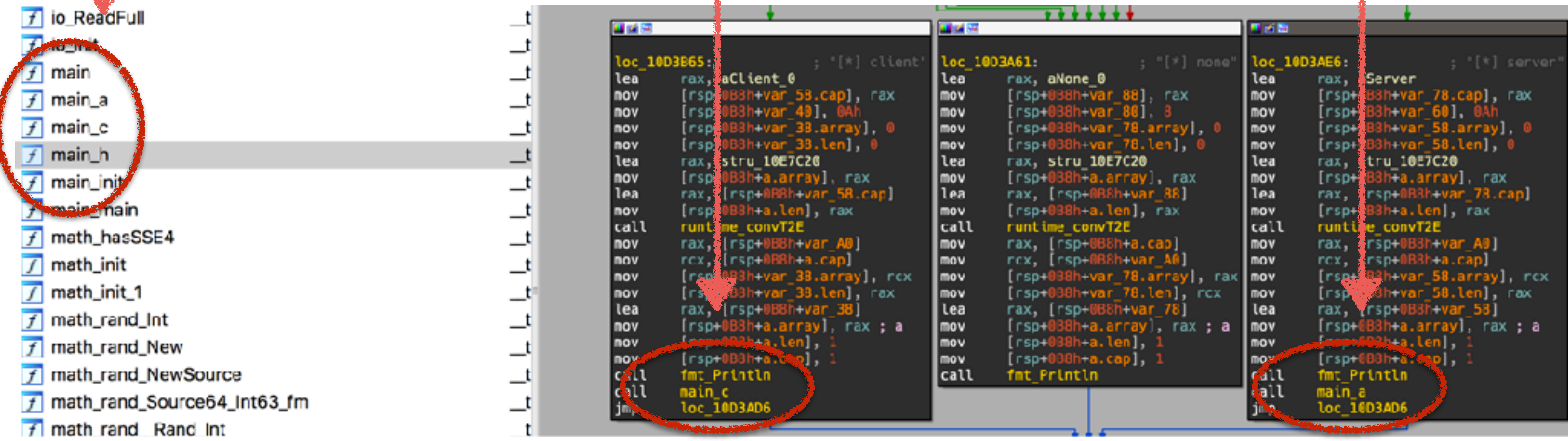


```
io_ReadFull
io_init
main
main_client
main_handleConnection
main_init
main_main
main_server
math_hasSSE4
math_init
math_init_1
math_rand_Int
math_rand_New
math_rand_NewSource
math_rand_Source64_Int63_fm
math_rand__Rand_Int
```

```
loc_10D3B65:                            ; "[*] client"
lea     rax, aClient_0
mov     [rsp+0B8h+var_58.cap], rax
mov     [rsp+0B8h+var_40], 0Ah
mov     [rsp+0B8h+var_38.array], 0
mov     [rsp+0B8h+var_38.len], 0
lea     rax, stru_10E7C20
mov     [rsp+0B8h+a.array], rax
lea     rax, [rsp+0B8h+var_58.cap]
mov     [rsp+0B8h+a.len], rax
call    runtime_convT2E
mov     rax, [rsp+0B8h+var_A0]
mov     rcx, [rsp+0B8h+a.cap]
mov     [rsp+0B8h+var_38.array], rcx
mov     [rsp+0B8h+var_38.len], rax
lea     rax, [rsp+0B8h+var_38]
mov     [rsp+0B8h+a.array], rax ; a
mov     [rsp+0B8h+a.len], 1
mov     [rsp+0B8h+a.cap], 1
call    fmt_Println
call    main_client
jmp     loc_10D3AD6
```

```
loc_10D3A61:                            ; "[*] none"
lea     rax, aNone_0
mov     [rsp+0B8h+var_88], rax
mov     [rsp+0B8h+var_80], 8
mov     [rsp+0B8h+var_78.array], 0
mov     [rsp+0B8h+var_78.len], 0
lea     rax, stru_10E7C20
mov     [rsp+0B8h+a.array], rax
lea     rax, [rsp+0B8h+var_88]
mov     [rsp+0B8h+a.len], rax
call    runtime_convT2E
mov     rax, [rsp+0B8h+a.cap]
mov     rcx, [rsp+0B8h+var_A0]
mov     [rsp+0B8h+var_78.array], rax
mov     [rsp+0B8h+var_78.len], rcx
lea     rax, [rsp+0B8h+var_78]
mov     [rsp+0B8h+a.array], rax ; a
mov     [rsp+0B8h+a.len], 1
mov     [rsp+0B8h+a.cap], 1
call    fmt_Println
```

```
loc_10D3AE6:                            ; "[*] server"
lea     rax, aServer
mov     [rsp+0B8h+var_78.cap], rax
mov     [rsp+0B8h+var_60], 0Ah
mov     [rsp+0B8h+var_58.array], 0
mov     [rsp+0B8h+var_58.len], 0
lea     rax, stru_10E7C20
mov     [rsp+0B8h+a.array], rax
lea     rax, [rsp+0B8h+var_78.cap]
mov     [rsp+0B8h+a.len], rax
call    runtime_convT2E
mov     rax, [rsp+0B8h+var_A0]
mov     rcx, [rsp+0B8h+a.cap]
mov     [rsp+0B8h+var_58.array], rcx
mov     [rsp+0B8h+var_58.len], rax
lea     rax, [rsp+0B8h+var_58]
mov     [rsp+0B8h+a.array], rax ; a
mov     [rsp+0B8h+a.len], 1
mov     [rsp+0B8h+a.cap], 1
call    fmt_Println
call    main_server
jmp     loc_10D3AD6
```



```
io_ReadFull
io_init
main
main_a
main_c
main_h
main_init
main_main
math_hasSSE4
math_init
math_init_1
math_rand_Int
math_rand_New
math_rand_NewSource
math_rand_Source64_Int63_fm
math_rand__Rand_Int
```

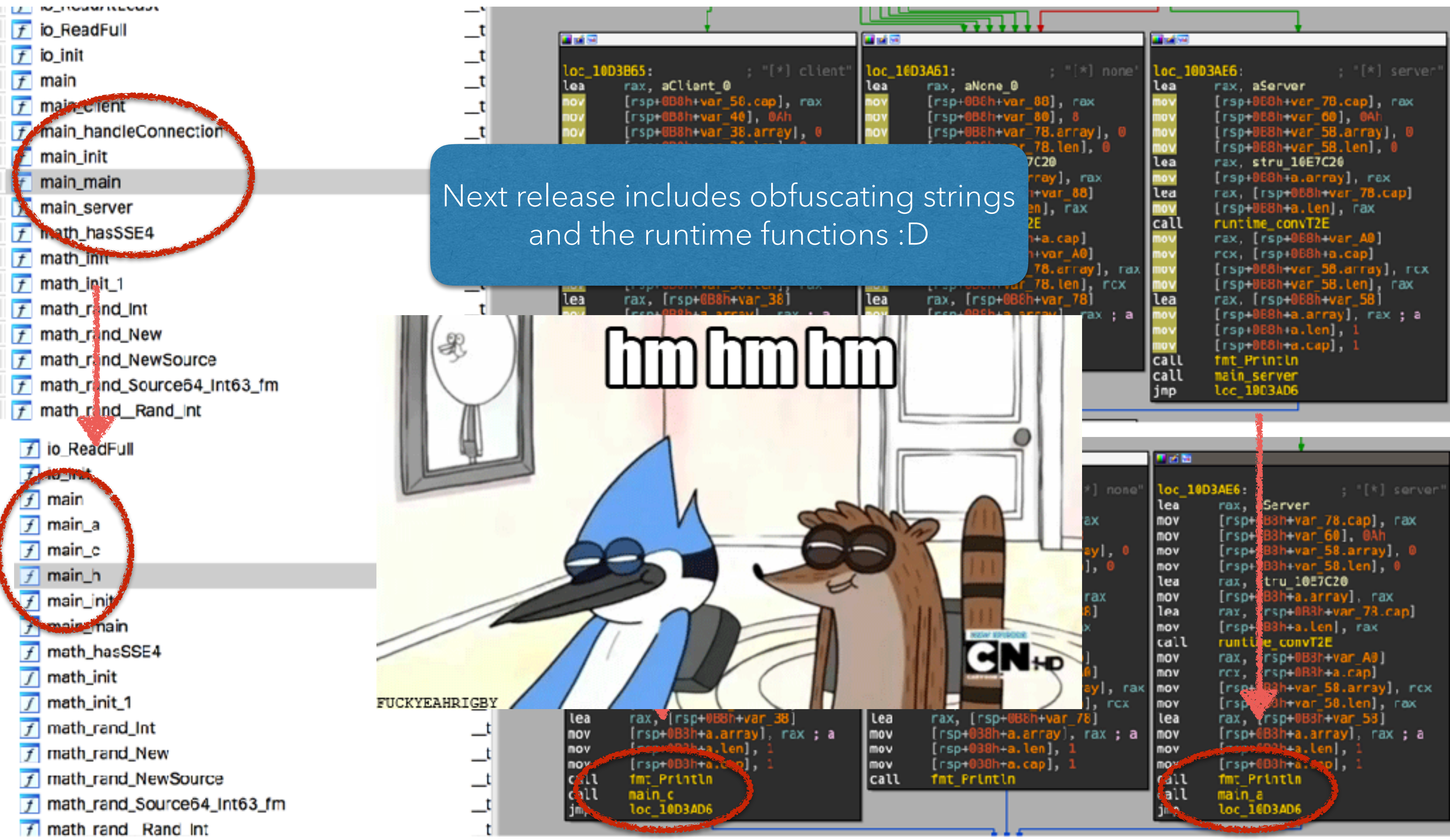
```
loc_10D3B65:                            ; "[*] client"
lea     rax, aClient_0
mov     [rsp+0B8h+var_58.cap], rax
mov     [rsp+0B8h+var_40], 0Ah
mov     [rsp+0B8h+var_38.array], 0
mov     [rsp+0B8h+var_38.len], 0
lea     rax, stru_10E7C20
mov     [rsp+0B8h+a.array], rax
lea     rax, [rsp+0B8h+var_58.cap]
mov     [rsp+0B8h+a.len], rax
call    runtime_convT2E
mov     rax, [rsp+0B8h+var_A0]
mov     rcx, [rsp+0B8h+a.cap]
mov     [rsp+0B8h+var_38.array], rcx
mov     [rsp+0B8h+var_38.len], rax
lea     rax, [rsp+0B8h+var_38]
mov     [rsp+0B8h+a.array], rax ; a
mov     [rsp+0B8h+a.len], 1
mov     [rsp+0B8h+a.cap], 1
call    fmt_Println
call    main_c
jmp     loc_10D3AD6
```

```
loc_10D3A61:                            ; "[*] none"
lea     rax, aNone_0
mov     [rsp+0B8h+var_88], rax
mov     [rsp+0B8h+var_80], 8
mov     [rsp+0B8h+var_78.array], 0
mov     [rsp+0B8h+var_78.len], 0
lea     rax, stru_10E7C20
mov     [rsp+0B8h+a.array], rax
lea     rax, [rsp+0B8h+var_88]
mov     [rsp+0B8h+a.len], rax
call    runtime_convT2E
mov     rax, [rsp+0B8h+a.cap]
mov     rcx, [rsp+0B8h+var_A0]
mov     [rsp+0B8h+var_78.array], rax
mov     [rsp+0B8h+var_78.len], rcx
lea     rax, [rsp+0B8h+var_78]
mov     [rsp+0B8h+a.array], rax ; a
mov     [rsp+0B8h+a.len], 1
mov     [rsp+0B8h+a.cap], 1
call    fmt_Println
```

```
loc_10D3AE6:                            ; "[*] server"
lea     rax, aServer
mov     [rsp+0B8h+var_78.cap], rax
mov     [rsp+0B8h+var_60], 0Ah
mov     [rsp+0B8h+var_58.array], 0
mov     [rsp+0B8h+var_58.len], 0
lea     rax, stru_10E7C20
mov     [rsp+0B8h+a.array], rax
lea     rax, [rsp+0B8h+var_78.cap]
mov     [rsp+0B8h+a.len], rax
call    runtime_convT2E
mov     rax, [rsp+0B8h+var_A0]
mov     rcx, [rsp+0B8h+a.cap]
mov     [rsp+0B8h+var_58.array], rcx
mov     [rsp+0B8h+var_58.len], rax
lea     rax, [rsp+0B8h+var_58]
mov     [rsp+0B8h+a.array], rax ; a
mov     [rsp+0B8h+a.len], 1
mov     [rsp+0B8h+a.cap], 1
call    fmt_Println
call    main_a
jmp     loc_10D3AD6
```

GOGUARD!

Oh yay… Only seems fitting since Go acts like Java :D



```
io_ReadFull
io_init
main
main_client
main_handleConnection
main_init
main_main
main_server
math_hasSSE4
math_init
math_init_1
math_rand_Int
math_rand_New
math_rand_NewSource
math_rand_Source64_Int63_fm
math_rand__Rand_Int
```

```
io_ReadFull
io_init
main
main_a
main_c
main_h
main_init
main_main
math_hasSSE4
math_init
math_init_1
math_rand_Int
math_rand_New
math_rand_NewSource
math_rand_Source64_Int63_fm
math_rand__Rand_Int
```

```
loc_10D3B65:                    ; "[*] client"
lea     rax, aClient_0
mov     [rsp+0B8h+var_58.cap], rax
mov     [rsp+0B8h+var_40], 0Ah
mov     [rsp+0B8h+var_38.array], 0
lea     rax, [rsp+0B8h+var_38]
```

```
loc_10D3A61:                    ; "[*] none"
lea     rax, aNone_0
mov     [rsp+0B8h+var_88], rax
mov     [rsp+0B8h+var_80], 8
mov     [rsp+0B8h+var_78.array], 0
lea     rax, [rsp+0B8h+var_78]
```

```
loc_10D3AE6:                    ; "[*] server"
lea     rax, aServer
mov     [rsp+0B8h+var_78.cap], rax
mov     [rsp+0B8h+var_60], 0Ah
mov     [rsp+0B8h+var_58.array], 0
mov     [rsp+0B8h+var_58.len], 0
lea     rax, stru_10E7C20
mov     [rsp+0B8h+a.array], rax
lea     rax, [rsp+0B8h+var_78.cap]
mov     [rsp+0B8h+a.len], rax
call    runtime_convT2E
mov     rax, [rsp+0B8h+var_A0]
mov     rcx, [rsp+0B8h+a.cap]
mov     [rsp+0B8h+var_58.array], rcx
mov     [rsp+0B8h+var_58.len], rax
lea     rax, [rsp+0B8h+var_58]
mov     [rsp+0B8h+a.array], rax ; a
mov     [rsp+0B8h+a.len], 1
mov     [rsp+0B8h+a.cap], 1
call    fmt_Println
call    main_server
jmp     loc_10D3AD6
```

```
lea     rax, [rsp+0B8h+var_38]
mov     [rsp+0B8h+a.array], rax ; a
mov     [rsp+0B8h+a.len], 1
mov     [rsp+0B8h+a.cap], 1
call    fmt_Println
call    main_c
jmp     loc_10D3AD6
```

```
lea     rax, [rsp+0B8h+var_78]
mov     [rsp+0B8h+a.array], rax ; a
mov     [rsp+0B8h+a.len], 1
mov     [rsp+0B8h+a.cap], 1
call    fmt_Println
```

```
loc_10D3AE6:                    ; "[*] server"
lea     rax, Server
mov     [rsp+0B8h+var_78.cap], rax
mov     [rsp+0B8h+var_60], 0Ah
mov     [rsp+0B8h+var_58.array], 0
mov     [rsp+0B8h+var_58.len], 0
lea     rax, stru_10E7C20
mov     [rsp+0B8h+a.array], rax
lea     rax, [rsp+0B8h+var_78.cap]
mov     [rsp+0B8h+a.len], rax
call    runtime_convT2E
mov     rax, [rsp+0B8h+var_A0]
mov     rcx, [rsp+0B8h+a.cap]
mov     [rsp+0B8h+var_58.array], rcx
mov     [rsp+0B8h+var_58.len], rax
lea     rax, [rsp+0B8h+var_58]
mov     [rsp+0B8h+a.array], rax ; a
mov     [rsp+0B8h+a.len], 1
mov     [rsp+0B8h+a.cap], 1
call    fmt_Println
call    main_a
jmp     loc_10D3AD6
```

Next release includes obfuscating strings and the runtime functions :D

REFERENCES AND RESOURCES

Everything can be reversed…. It's also likely it already has been!

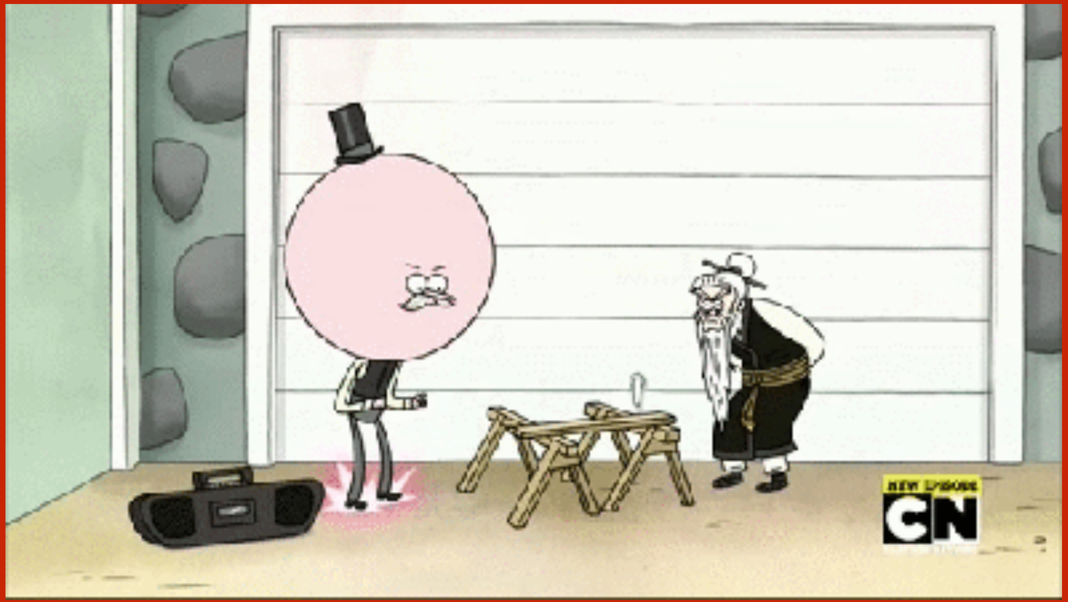
- Reversing Go Primer
    - https://www.rednaga.io
    - Go malware will be posted soon after talk along with slides

- golang_loader_assist
    - https://github.com/strazzere/golang_loader_assist
    - New version published after talk included all features with upgraded heuristics for Go 1.8+
    - Binary Ninja version will be dropped shortly in same repo!

- goguard
    - https://github.com/strazzere/goguard
    - Repo will be pushed after the talk
    - Currently obfuscates all custom code and vendor code
    - Working on runtime and adding encryption/obfuscation to strings

GOOD LUCK HUNTING!

TIM "DIFF" STRAZZERE
@TIMSTRAZZ



Good people to follow on Twitter for
Android / reversing / malware / hacking information:
@_jsoo_ @droidsec @jcase @marcwrogers @msolnik
@PatrickMcCanna @rotlogix @snare @tamakikusu @trimosx
@cryptax @virqdroid @WataShiva @againsthimself @collinrm
@michalmalik @utkan0s @malataz @LukasStefanko @ACKFlags
@bugcrowd @samhouston

07.25.2017

BugCrowd - BSides LV

REDNAGA

LAS VEGAS BSIDES