

How to debug instructions mixed with data using GDB-GEF

 medium.com/@therealdreg/how-to-debug-instructions-mixed-with-data-using-gdb-gef-5acffce3bb18

Dreg

June 29, 2022



Dreg

Jun 28

.

3 min read

Celebrating GDB-GEF 10 year anniversary, **2022.06 — Upset Constant** version has been released. So, I will introduce how to debug instructions mixed with data using GDB-GEF.

Scenario

You want debug a shellcode/malware/packer code like this:

```
; nasm -felf64 -o poc.o poc.asm && ld -m elf_x86_64 poc.o -o poc
section .text
    global _start
_start:
    times 2 db 90h
    call first
    msg_one db `'/bin/sh`,0
    first:
    xor ebx, ecx
    add ebx, 1
    call second
    msg_two db `dregishot`,0
    second:
    mov rax, 60 ; exit
    mov rbx, 0
    syscall
```

With IDA PRO, GHIDRA or RADARE its very easy, just selecting the strings as data the disas will be fixed.

Some years ago... I made a simple radare crappy-script-poc to auto-mark strings in .text section: call_trick_r2pipe

before call_trick_r2pipe execution:

```

File  Settings  Edit  View  Tools  Search  Emulate  Debug Tab [1] [0x00401050]
-----
[X] Disassembly (pd) [Cache] Off
|
| ; CODE XREF from entry0 @ 0x401012
| ;-- child:
| 0x00401050      6800000000      push 0
| 0x00401055      e80d020000      call loc.fpu ;[1]
| ;-- arg3:
| ,=< 0x0040105a      657865      js 0x4010c2
| | 0x0040105d      6328      movsxd rbp, dword [rax]
| | 0x0040105f      2222      and ah, byte [rdx]
| | 0x00401061      220a      and cl, byte [rdx]
| | 0x00401061      706f7274.    imul ebp, dword [rbp + 0x70], 0x20
| | 0x00401065      80000000      jae 0x4010db
| | 0x00401065      80000000      movsxd rbp, dword [rbx + 0x65]
| ,===< 0x0040106f      742c      je 0x40109d
| \ ,===< 0x00401071      7375      jae 0x4010e8
| | | | 0x00401073      62      invalid
| ,=====< 0x00401074      7072      jo 0x4010e8
| | | | 0x00401076      6f      outsd dx, dword [rsi]
| | | | 0x00401077      636573      movsxd rsp, dword [rbp + 0x73]
| ,=====< 0x0040107a      732c      jae 0x4010a8
| | | | 0x0040107c      6f      outsd dx, dword [rsi]

```

DISAS BROKEN

After `call_trick_r2pipe` execution:

```

File  Settings  Edit  View  Tools  Search  Emulate  Debug Tab [1] [0x00401050]
-----
[X] Disassembly (pd)
|
| ; CODE XREF from entry0 @ 0x401012
| ;-- child:
| 0x00401050      6800000000      push 0
| 0x00401055      e80d020000      call loc.fpu ;[1]
| ;-- arg3:
| 0x0040105a      .string "exec(\"\"\"\\nimport socket,subprocess,os,
| 0x0040105d      .string "exec(\"\"\"\\nimport socket,subprocess,os,
| ; CALL XREF from entry0 @ 0x401055
| / 14: loc.fpu ();
| 0x00401267      e803000000      call loc.lxz ;[2]
| ;-- arg2:
| 0x0040126c ~ 2d6300e80c      sub eax, 0xce80063
| ; CALL XREF from loc.fpu @ 0x401267
| ;-- lxz:
| 0x0040126f      e80c000000      call loc.drgs ;[3]
| ; CODE XREF from loc.arg3 @ +0x1b4
| ;-- arg1:
| 0x00401274      .string "/bin/python" ; len=12
| ;-- drgs:
| 0x00401280      488d05300000.    lea rax, loc.msg ; 0x40

```

Problem

GDB-GEF TEAM (hugsy & contributors) are awesome. But due to GDB-GEF keep-minimal-as-possible philosophy (IMO) this feature will not be included. Maybe in a near future will be supported via gef-extras (I will try it, I promise).

gdis

GDB-GEF API is very powerful and easy to use.

So, I have created gdis, a GDB plugin to solve this problem. It works with GDB-GEF and raw GDB (IMO raw GDB is ugly as hell).

GDB-GEF before gdis:

```
$rax : 0x0
$rbx : 0x0
$rcx : 0x0
$rdx : 0x0
$rsp : 0x007fffffe070 → 0x0000000000000001
$rbp : 0x0
$rsi : 0x0
$rdi : 0x0
$rip : 0x00000000401000 → <_start+0> nop
$r8 : 0x0
$r9 : 0x0
$r10 : 0x0
$r11 : 0x0
$r12 : 0x0
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
$eflags: [zero carry parity adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x33 $ss: 0x2b $ds: 0x00 $es: 0x00 $fs: 0x00 $gs: 0x00

0x007fffffe070|+0x000: 0x0000000000000001 ← $rsp
0x007fffffe078|+0x0008: 0x007fffffe3c5 → "/home/tmp/poc"
0x007fffffe080|+0x0010: 0x0000000000000000
0x007fffffe088|+0x0018: 0x007fffffe3d3 → "COLORFGBG=15;0"
0x007fffffe090|+0x0020: 0x007fffffe3e2 → "COLORTERM=truecolor"
0x007fffffe098|+0x0028: 0x007fffffe3f6 → "COMMAND_NOT_FOUND_INSTALL_PROMPT=1"
0x007fffffe0a0|+0x0030: 0x007fffffe419 → "DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/[ ... ]"
0x007fffffe0a8|+0x0038: 0x007fffffe44f → "DESKTOP_SESSION=lightdm-xsession"

[#0] Id 1, Name: "poc", stopped 0x401000 in _start (), reason: STOPPED

[#0] 0x401000 → _start()

   0x400ffa 0000      add     BYTE PTR [rax], al
   0x400ffc 0000      add     BYTE PTR [rax], al
   0x400ffe 0000      add     BYTE PTR [rax], al
→  0x401000 90          nop
   0x401001 90          nop
   0x401002 e808000000      call    0x40100f <first>
   0x401007 2f          <msg_one+0> (bad)
   0x401008 62          <msg_one+1> (bad)
   0x401009 696e2f73680031  <msg_one+2> imul    ebp, DWORD PTR [rsi+0x2f], 0x31006873
   0x401010 cb          <first+1> retf
   0x401011 83c301      <first+2> add     ebx, 0x1
   0x401014 e80a000000      <first+5> call    0x401023 <second>
   0x401019 647265      <msg_two+0> fs     jb  0x401081
```

After gdis:

```
[ Legend: Modified register | Code | Heap | Stack | String ]

$rax : 0x0
$rbx : 0x0
$rcx : 0x0
$rdx : 0x0
$rsp : 0x007fffffe070 → 0x0000000000000001 help() Command:
$rbp : 0x0
$rsi : 0x0
$rdi : 0x0
$rip : 0x00000000401000 → <_start+0> nop
$r8 : 0x0
$r9 : 0x0
$r10 : 0x0
$r11 : 0x0
$r12 : 0x0
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
$eflags: [zero carry parity adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x33 $ss: 0x2b $ds: 0x00 $es: 0x00 $fs: 0x00 $gs: 0x00

0x007fffffe070|+0x0000: 0x0000000000000001 ← $rsp
0x007fffffe078|+0x0008: 0x007fffffe3c5 → "/home/tmp/poc"
0x007fffffe080|+0x0010: 0x0000000000000000
0x007fffffe088|+0x0018: 0x007fffffe3d3 → "COLORFGBG=15;0"
0x007fffffe090|+0x0020: 0x007fffffe3e2 → "COLORTERM=truecolor"
0x007fffffe098|+0x0028: 0x007fffffe3f6 → "COMMAND_NOT_FOUND_INSTALL_PROMPT=1"
0x007fffffe0a0|+0x0030: 0x007fffffe419 → "DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/[ ... ]"
0x007fffffe0a8|+0x0038: 0x007fffffe44f → "DESKTOP_SESSION=lightdm-xsession"

[#0] Id 1, Name: "poc", stopped 0x401000 in _start (), reason: STOPPED == PRINT

[#0] 0x401000 → _start()

⇒ 0x401000 <_start> : nop | 0x90 → b'\x90'
0x401001 <_start+1> : nop | 0x90 → b'\x90'
0x401002 <_start+2> : call 0x40100f <first> | 0xE8 0x08 0x00 0x00 0x00 → b'\xe8\x08\x00\x00\x00'
***** skipped data from 0x401007 to 0x40100f → b'/bin/sh\x00\xcb\x83\xc3\x01\xe8\n\x00\x00\x00dr'....
0x40100f <first> : xor ebx,ecx | 0x31 0xCB → b'1\xcb'
0x401011 <first+2> : add ebx,0x1 | 0x83 0xC3 0x01 → b'\x83\xc3\x01'
0x401014 <first+5> : call 0x401023 <second> | 0xE8 0x0A 0x00 0x00 0x00 → b'\xe8\n\x00\x00\x00'
***** skipped data from 0x401019 to 0x401023 → b'dregishot\x00\xb8<\x00\x00\x00\xbb\x00\x00\x00'....
0x401023 <second> : mov eax,0x3c | 0xB8 0x3C 0x00 0x00 0x00 → b'\xb8<\x00\x00\x00'
0x401028 <second+5> : mov ebx,0x0 | 0xBB 0x00 0x00 0x00 0x00 → b'\xbb\x00\x00\x00\x00'
0x40102d <second+10> : syscall | 0x0F 0x05 → b'\x0f\x05'
0x40102f : add BYTE PTR [rax],al | 0x00 0x00 → b'\x00\x00'
```

Nice! you can debug and disas the correct code :-)

How to use gdis in GDB-GEF

git clone

Edit ~/.gdbinit

Add gdis.py:

source /home/dreg/gef/gef.py

source /home/dreg/gdis/gdis.py

Open gdb

Type: gef save

Type: quit

Edit ~/.gef.rc

(Optional step) Remove original “code” panel from layout:

layout = legend regs **code** stack args source memory threads trace extra

layout = legend regs stack args source memory threads trace extra

POC example for Linux x86_64

open **gdb gdis/poc**

Type: **starti**

```
[#0] 0x401000 → _start()
⇒ 0x401000 <_start> : nop      | 0x90 → b'\x90'
0x401001 <_start+1> : nop      | 0x90 → b'\x90'
0x401002 <_start+2> : call    0x40100f <first> | 0xE8 0x08 0x00 0x00 0x00 → b'\xe8\x08\x00\x00'
0x401007 <msg_one> : (bad)   | 0x2F → b'/'
0x401008 <msg_one+1> : (bad)   | 0x62 → b'b'
0x401009 <msg_one+2> : imul    ebp,DWORD PTR [rsi+0x2f],0x31006873 | 0x69 0x6E 0x2F 0x73 0x68 0x00 0x31 → b'in/sh\x001'
0x401010 <first+1> : retf    | 0xCB → b'\xcb'
0x401011 <first+2> : add     ebx,0x1 | 0x83 0xC3 0x01 → b'\x83\xc3\x01'
0x401014 <first+5> : call    0x401023 <second> | 0xE8 0x0A 0x00 0x00 0x00 → b'\xe8\n\x00\x00\x00'
0x401019 <msg_two> : fs jb   0x401081 | 0x64 0x72 0x65 → b'dre'
0x40101c <msg_two+3> : imul    esi,DWORD PTR [ebx+0x68],0xb800746f | 0x67 0x69 0x73 0x68 0x6F 0x74 0x00 0xB8 → b'gishot\x00\xb8'
0x401024 <second+1> : cmp     al,0x0 | 0x3C 0x00 → b'\x00'
```

As you can see the disas is broken, for gdis help just type: **ghelp**

Fixing disas

POC contains two strings between instructions: `msg_one` and `msg_two`. Mark them as data with **strz address/symbol** command:

strz msg_one

strz msg_two

Type: **context**

```
[#0] 0x401000 → _start()
⇒ 0x401000 <_start> : nop      | 0x90 → b'\x90'
0x401001 <_start+1> : nop      | 0x90 → b'\x90'
0x401002 <_start+2> : call    0x40100f <first> | 0xE8 0x08 0x00 0x00 0x00 → b'\xe8\x08\x00\x00\x00'
***** skipped data from 0x401007 to 0x40100F → b'/bin/sh\x001\xcb\x83\xc3\x01\xe8\n\x00\x00\x00dr'....
0x40100f <first> : xor     ebx,ecx | 0x31 0xCB → b'1\xcb'
0x401011 <first+2> : add     ebx,0x1 | 0x83 0xC3 0x01 → b'\x83\xc3\x01'
0x401014 <first+5> : call    0x401023 <second> | 0xE8 0x0A 0x00 0x00 0x00 → b'\xe8\n\x00\x00\x00'
***** skipped data from 0x401019 to 0x401023 → b'dregishot\x00\xb8<\x00\x00\x00\xbb\x00\x00\x00'....
0x401023 <second> : mov     eax,0x3c | 0xB8 0x3C 0x00 0x00 0x00 → b'\xb8<\x00\x00\x00'
0x401028 <second+5> : mov     ebx,0x0 | 0xBB 0x00 0x00 0x00 0x00 → b'\xbb\x00\x00\x00\x00'
0x40102d <second+10> : syscall | 0x0F 0x05 → b'\x0f\x05'
0x40102f : add     BYTE PTR [rax],al | 0x00 0x00 → b'\x00\x00'
```

hooray! the disas is fixed!

How to mark N bytes as data?

easy, just use **awd address/symbol 0xSIZE** command:

```
[#0] 0x401000 → _start()
⇒ 0x401000 <_start> : nop      | 0x90 → b'\x90'
0x401001 <_start+1> : nop      | 0x90 → b'\x90'
0x401002 <_start+2> : call    0x40100f <first> | 0xE8 0x08 0x00 0x00 0x00 → b'\xe8\x08\x00\x00\x00'
0x401007 <msg_one> : (bad)   | 0x2F → b'/'
0x401008 <msg_one+1> : (bad)   | 0x62 → b'b'
0x401009 <msg_one+2> : imul    ebp,DWORD PTR [rsi+0x2f],0x31006873 | 0x69 0x6E 0x2F 0x73 0x68 0x00 0x31 → b'in/sh\x001'
0x401010 <first+1> : retf    | 0xCB → b'\xcb'
0x401011 <first+2> : add     ebx,0x1 | 0x83 0xC3 0x01 → b'\x83\xc3\x01'
0x401014 <first+5> : call    0x401023 <second> | 0xE8 0x0A 0x00 0x00 0x00 → b'\xe8\n\x00\x00\x00'
0x401019 <msg_two> : fs jb   0x401081 | 0x64 0x72 0x65 → b'dre'
0x40101c <msg_two+3> : imul    esi,DWORD PTR [ebx+0x68],0xb800746f | 0x67 0x69 0x73 0x68 0x6F 0x74 0x00 0xB8 → b'gishot\x00\xb8'
0x401024 <second+1> : cmp     al,0x0 | 0x3C 0x00 → b'<\x00'

gef> awd 0x401007 0x8
gef> gdis
⇒ 0x401000 <_start> : nop      | 0x90 → b'\x90'
0x401001 <_start+1> : nop      | 0x90 → b'\x90'
0x401002 <_start+2> : call    0x40100f <first> | 0xE8 0x08 0x00 0x00 0x00 → b'\xe8\x08\x00\x00\x00'
**** skipped data from 0x401007 to 0x40100f → b'/bin/sh\x001\xcb\x83\xc3\x01\xe8\n\x00\x00\x00dr'...
0x40100f <first> : xor     ebx,ecx | 0x31 0xCB → b'\xcb'
0x401011 <first+2> : add     ebx,0x1 | 0x83 0xC3 0x01 → b'\x83\xc3\x01'
0x401014 <first+5> : call    0x401023 <second> | 0xE8 0x0A 0x00 0x00 0x00 → b'\xe8\n\x00\x00\x00'
0x401019 <msg_two> : fs jb   0x401081 | 0x64 0x72 0x65 → b'dre'
0x40101c <msg_two+3> : imul    esi,DWORD PTR [ebx+0x68],0xb800746f | 0x67 0x69 0x73 0x68 0x6F 0x74 0x00 0xB8 → b'gishot\x00\xb8'
0x401024 <second+1> : cmp     al,0x0 | 0x3C 0x00 → b'<\x00'
0x401026 <second+3> : add     BYTE PTR [rax],al | 0x00 0x00 → b'\x00\x00'
0x401028 <second+5> : mov     ebx,0x0 | 0xBB 0x00 0x00 0x00 0x00 → b'\xbb\x00\x00\x00\x00'
```

done!

How to use gdis in raw GDB (x86_64 Linux POC)

open gdb gdis/poc

type: **source /home/gdis/gdis.py**

type: **starti**

```
(gdb) starti
Starting program: /home/tmp/poc

Program stopped.
0x0000000000401000 in _start ()
rax            0x0
rbx            0x0
rcx            0x0
rdx            0x0
rsi            0x0
rdi            0x0
rbp            0x0
rsp            0x7fffffff550
r8             0x0
r9             0x0
r10            0x0
r11            0x0
r12            0x0
r13            0x0
r14            0x0
r15            0x0
rip            0x401000
eflags         0x200
cs             0x33
ss             0x2b
ds             0x0
es             0x0
fs             0x0
gs             0x0
0x7fffffff550: 0x1 0x7fffffff7ad
0x7fffffff560: 0x0 0x7fffffff7bb
0x7fffffff570: 0x7fffffff7cb 0x7fffffff7e6
0x7fffffff580: 0x7fffffff7fb 0x7fffffff814
0x7fffffff590: 0x7fffffff832 0x7fffffff850
0x7fffffff5a0: 0x7fffffff85e 0x7fffffff86a
0x7fffffff5b0: 0x7fffffff879 0x7fffffff89c
⇒ 0x401000 <_start> : nop      | 0x90 → b'\x90'
0x401001 <_start+1> : nop      | 0x90 → b'\x90'
0x401002 <_start+2> : call    0x40100f <first> | 0xE8 0x08 0x00 0x00 0x00 → b'\xe8\x08\x00\x00\x00'
0x401007 <msg_one> : (bad)   | 0x2F → b'/'
0x401008 <msg_one+1> : (bad)   | 0x62 → b'b'
0x401009 <msg_one+2> : imul    ebp,DWORD PTR [rsi+0x2f],0x31006873 | 0x69 0x6E 0x2F 0x73 0x68 0x00 0x31 → b'in/sh\x001'
0x401010 <first+1> : retf    | 0xCB → b'\xcb'
0x401011 <first+2> : add     ebx,0x1 | 0x83 0xC3 0x01 → b'\x83\xc3\x01'
0x401014 <first+5> : call    0x401023 <second> | 0xE8 0x0A 0x00 0x00 0x00 → b'\xe8\n\x00\x00\x00'
0x401019 <msg_two> : fs jb   0x401081 | 0x64 0x72 0x65 → b'dre'
0x40101c <msg_two+3> : imul    esi,DWORD PTR [ebx+0x68],0xb800746f | 0x67 0x69 0x73 0x68 0x6F 0x74 0x00 0xB8 → b'gishot\x00\xb8'
0x401024 <second+1> : cmp     al,0x0 | 0x3C 0x00 → b'<\x00'
```

Done, just use **awd** and/or **strz** commands to mark data


```

⇒ 0x401000 <_start> : nop      | 0x90 → b'\x90'
0x401001 <_start+1> : nop      | 0x90 → b'\x90'
0x401002 <_start+2> : call    0x40100f <first> | 0xE8 0x08 0x00 0x00 0x00 → b'\xe8\x08\x00\x00\x00'
0x401007 <msg_one> : (bad)   | 0x2f → b'/'
0x401008 <msg_one+1> : (bad)   | 0x62 → b'b'
0x401009 <msg_one+2> : imul    ebp,DWORD PTR [rsi+0x2f],0x31006873 | 0x69 0x06 0x2F 0x73 0x68 0x00 0x31 → b'in/sh\x001'
0x401010 <first+1> : retf    | 0xCB → b'\xcb'
0x401011 <first+2> : add     ebx,0x1 | 0x83 0xC3 0x01 → b'\x83\xc3\x01'
0x401014 <first+5> : call    0x401023 <second> | 0xE8 0x0A 0x00 0x00 0x00 → b'\xe8\x0a\x00\x00\x00'
0x401019 <msg_two> : fs jb   0x401081 | 0x64 0x72 0x65 → b'dre'
0x40101c <msg_two+3> : imul    esi,DWORD PTR [ebx+0x68],0xb800746f | 0x67 0x69 0x73 0x68 0x6F 0x74 0x00 0xB8 → b'gishot\x00\xb8'
0x401024 <second+1> : cmp     al,0x0 | 0x3C 0x00 → b'<\x00'
(gdb) awd 0x401007 0x8
(gdb) awd 0x401019 0xA
(gdb) gdis
⇒ 0x401000 <_start> : nop      | 0x90 → b'\x90'
0x401001 <_start+1> : nop      | 0x90 → b'\x90'
0x401002 <_start+2> : call    0x40100f <first> | 0xE8 0x08 0x00 0x00 0x00 → b'\xe8\x08\x00\x00\x00'
**** skipped data from 0x401007 to 0x40100f → b'/bin/sh\x001\xcb\x83\xc3\x01\xe8\n\x00\x00\x00dr'....
0x40100f <first> : xor     ebx,ecx | 0x31 0xCB → b'l\xcb'
0x401011 <first+2> : add     ebx,0x1 | 0x83 0xC3 0x01 → b'\x83\xc3\x01'
0x401014 <first+5> : call    0x401023 <second> | 0xE8 0x0A 0x00 0x00 0x00 → b'\xe8\x0a\x00\x00\x00'
**** skipped data from 0x401019 to 0x401023 → b'dregishot\x00\xb8<\x00\x00\x00\xbb\x00\x00\x00'....
0x401023 <second> : mov     eax,0x3c | 0xB8 0x3C 0x00 0x00 0x00 → b'\xb8<\x00\x00\x00'
0x401028 <second+5> : mov     ebx,0x0 | 0xBB 0x00 0x00 0x00 0x00 → b'\xbb\x00\x00\x00\x00'
0x40102d <second+10> : syscall | 0x0F 0x05 → b'\x0f\x05'
0x40102f : add     BYTE PTR [rax],al | 0x00 0x00 → b'\x00\x00'

```

Disas fixed!

Auto mark null-end-strings as data (x86_64 Linux POC)

Just use `autostr 0xMIN_SIZE_STR address/symbol 0xSIZE_TO_SCAN` command:

`autostr 0x4 _start 0x200`

```

(gdb) gdis
⇒ 0x401000 <_start> : nop      | 0x90 → b'\x90'
0x401001 <_start+1> : nop      | 0x90 → b'\x90'
0x401002 <_start+2> : call    0x40100f <first> | 0xE8 0x08 0x00 0x00 0x00 → b'\xe8\x08\x00\x00\x00'
0x401007 <msg_one> : (bad)   | 0x2f → b'/'
0x401008 <msg_one+1> : (bad)   | 0x62 → b'b'
0x401009 <msg_one+2> : imul    ebp,DWORD PTR [rsi+0x2f],0x31006873 | 0x69 0x06 0x2F 0x73 0x68 0x00 0x31 → b'in/sh\x001'
0x401010 <first+1> : retf    | 0xCB → b'\xcb'
0x401011 <first+2> : add     ebx,0x1 | 0x83 0xC3 0x01 → b'\x83\xc3\x01'
0x401014 <first+5> : call    0x401023 <second> | 0xE8 0x0A 0x00 0x00 0x00 → b'\xe8\x0a\x00\x00\x00'
0x401019 <msg_two> : fs jb   0x401081 | 0x64 0x72 0x65 → b'dre'
0x40101c <msg_two+3> : imul    esi,DWORD PTR [ebx+0x68],0xb800746f | 0x67 0x69 0x73 0x68 0x6F 0x74 0x00 0xB8 → b'gishot\x00\xb8'
0x401024 <second+1> : cmp     al,0x0 | 0x3C 0x00 → b'<\x00'
(gdb) autostr 0x4 _start 0x200
mark as data:
0x401007 0x7 0x8 → b'/bin/sh'....
0x401019 0x9 0xA → b'dregishot'....
0x401121 0x7 0x8 → b'poc.asm'....
0x401129 0x7 0x8 → b'msg_one'....
0x401131 0x5 0x6 → b'first'....
0x401137 0x7 0x8 → b'msg_two'....
0x40113F 0x6 0x7 → b'second'....
0x401146 0x8 0xC → b'__bss_start'....
0x401152 0x6 0x7 → b'__edata'....
0x401159 0x4 0x5 → b'__end'....
0x40115F 0x7 0x8 → b'.symtab'....
0x401167 0x7 0x8 → b'.strtab'....
0x40116F 0x9 0xA → b'.shstrtab'....
0x401179 0x5 0x6 → b'.text'....
done!
(gdb) gdis
⇒ 0x401000 <_start> : nop      | 0x90 → b'\x90'
0x401001 <_start+1> : nop      | 0x90 → b'\x90'
0x401002 <_start+2> : call    0x40100f <first> | 0xE8 0x08 0x00 0x00 0x00 → b'\xe8\x08\x00\x00\x00'
**** skipped data from 0x401007 to 0x40100f → b'/bin/sh\x001\xcb\x83\xc3\x01\xe8\n\x00\x00\x00dr'....
0x40100f <first> : xor     ebx,ecx | 0x31 0xCB → b'l\xcb'
0x401011 <first+2> : add     ebx,0x1 | 0x83 0xC3 0x01 → b'\x83\xc3\x01'
0x401014 <first+5> : call    0x401023 <second> | 0xE8 0x0A 0x00 0x00 0x00 → b'\xe8\x0a\x00\x00\x00'
**** skipped data from 0x401019 to 0x401023 → b'dregishot\x00\xb8<\x00\x00\x00\xbb\x00\x00\x00'....
0x401023 <second> : mov     eax,0x3c | 0xB8 0x3C 0x00 0x00 0x00 → b'\xb8<\x00\x00\x00'
0x401028 <second+5> : mov     ebx,0x0 | 0xBB 0x00 0x00 0x00 0x00 → b'\xbb\x00\x00\x00\x00'
0x40102d <second+10> : syscall | 0x0F 0x05 → b'\x0f\x05'
0x40102f : add     BYTE PTR [rax],al | 0x00 0x00 → b'\x00\x00'

```

`gdis` can do more things, just explore `ghelp` command.

warning: code is pure crap, PRs are welcome! x)

btw, sorry foy my bad english and don't be polite in the comments!

RT @ therealdreg