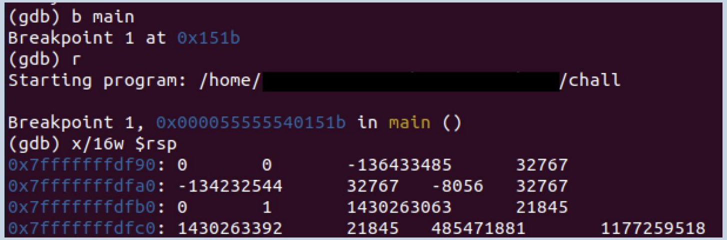
MWS Cup 事前課題 pwndbg-perceptor

Team: MONKY
小野大河　松尾和輝　刀塚敦子
河岡諒　谷田部和貴
（早稲田大学）

既存ツール：pwndbg

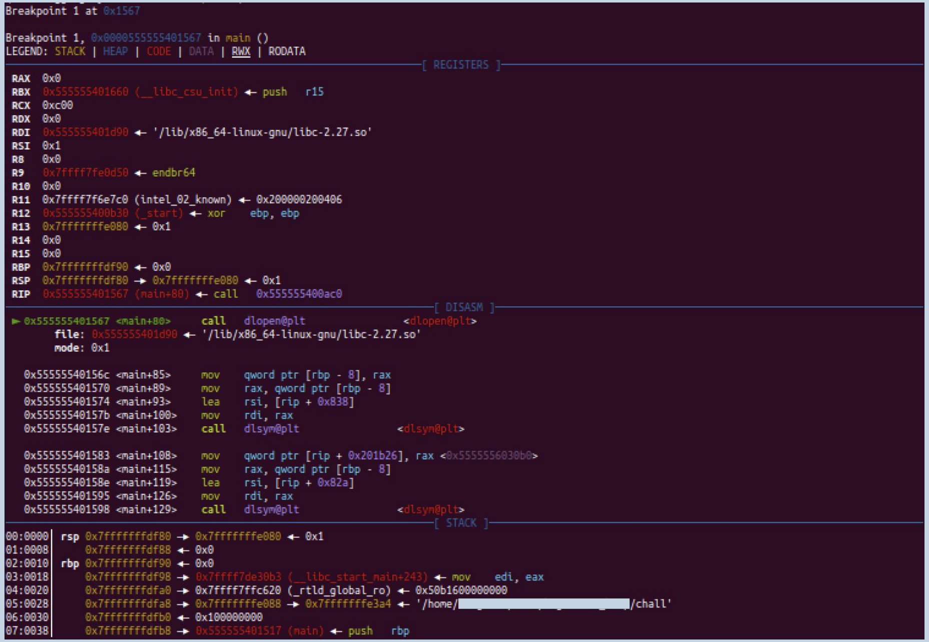
★ 初心者/入門者にとっての利便性/可読性の問題

GDB



```
(gdb) b main
Breakpoint 1 at 0x151b
(gdb) r
Starting program: /home/[redacted]/chall

Breakpoint 1, 0x000055555540151b in main ()
(gdb) x/16w $rsp
0x7fffffffdf90: 0       0       -136433485      32767
0x7fffffffdfa0: -134232544      32767   -8056   32767
0x7fffffffdfb0: 0       1       1430263063      21845
0x7fffffffdfc0: 1430263392      21845   485471881       1177259518
```

pwndbg



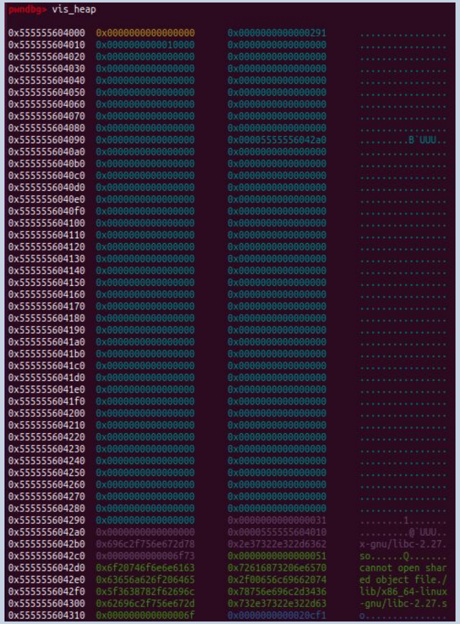
```
Breakpoint 1 at 0x1567

Breakpoint 1, 0x0000555555401567 in main ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
─────────────────────────────────────[ REGISTERS ]─────────────────────────────────────
 RAX  0x0
 RBX  0x555555401660 (__libc_csu_init) ◂— push   r15
 RCX  0xc00
 RDX  0x0
 RDI  0x555555401d90 ◂— '/lib/x86_64-linux-gnu/libc-2.27.so'
 RSI  0x1
 R8   0x0
 R9   0x7ffff7fe0d50 ◂— endbr64
 R10  0x0
 R11  0x7ffff7f6e7c0 (intel_02_known) ◂— 0x200000200406
 R12  0x555555400b30 (_start) ◂— xor    ebp, ebp
 R13  0x7fffffffe080 ◂— 0x1
 R14  0x0
 R15  0x0
 RBP  0x7fffffffdf90 ◂— 0x0
 RSP  0x7fffffffdf80 —▸ 0x7fffffffe080 ◂— 0x1
 RIP  0x555555401567 (main+80) ◂— call   0x555555400ac0
──────────────────────────────────────[ DISASM ]───────────────────────────────────────
 ► 0x555555401567 <main+80>     call   dlopen@plt                <dlopen@plt>
        file: 0x555555401d90 ◂— '/lib/x86_64-linux-gnu/libc-2.27.so'
        mode: 0x1

   0x55555540156c <main+85>     mov    qword ptr [rbp - 8], rax
   0x555555401570 <main+89>     mov    rax, qword ptr [rbp - 8]
   0x555555401574 <main+93>     lea    rsi, [rip + 0x838]
   0x55555540157b <main+100>    mov    rdi, rax
   0x55555540157e <main+103>    call   dlsym@plt                <dlsym@plt>

   0x555555401583 <main+108>    mov    qword ptr [rip + 0x201b26], rax <0x5555556030b0>
   0x55555540158a <main+115>    mov    rax, qword ptr [rbp - 8]
   0x55555540158e <main+119>    lea    rsi, [rip + 0x82a]
   0x555555401595 <main+126>    mov    rdi, rax
   0x555555401598 <main+129>    call   dlsym@plt                <dlsym@plt>
───────────────────────────────────────[ STACK ]───────────────────────────────────────
00:0000│ rsp 0x7fffffffdf80 —▸ 0x7fffffffe080 ◂— 0x1
01:0008│     0x7fffffffdf88 ◂— 0x0
02:0010│ rbp 0x7fffffffdf90 ◂— 0x0
03:0018│     0x7fffffffdf98 —▸ 0x7ffff7de30b3 (__libc_start_main+243) ◂— mov    edi, eax
04:0020│     0x7fffffffdfa0 —▸ 0x7ffff7ffc620 (_rtld_global_ro) ◂— 0x50b1600000000
05:0028│     0x7fffffffdfa8 —▸ 0x7fffffffe088 —▸ 0x7fffffffe3a4 ◂— '/home/[redacted]/chall'
06:0030│     0x7fffffffdfb0 ◂— 0x100000000
07:0038│     0x7fffffffdfb8 —▸ 0x555555401517 (main) ◂— push   rbp
```

既存ツールの課題：CLI操作

pwndbgにはすでに可視化ようのコマンドは実装されている

しかし、CLI上で可視化した内容の可読性には限度がある

結果、解析にかかる時間が増える

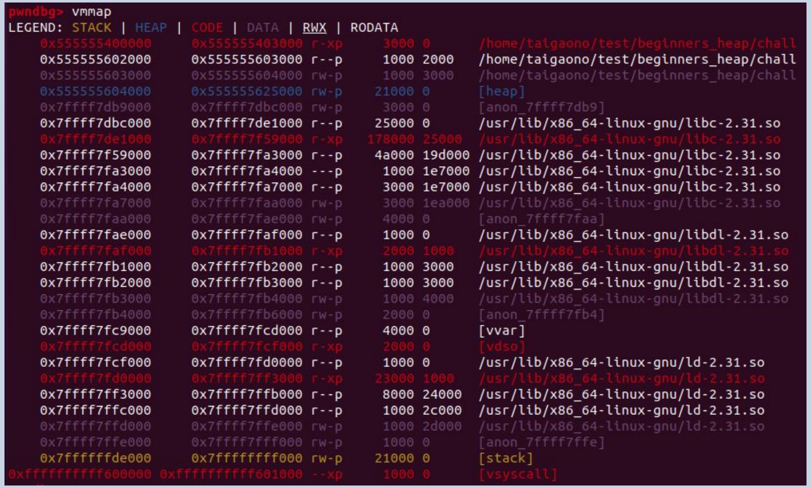


```
pwndbg> vis_heap

0x555555604000  0x0000000000000000      0x0000000000000291      ................
0x555555604010  0x0000000000010000      0x0000000000000000      ................
0x555555604020  0x0000000000000000      0x0000000000000000      ................
0x555555604030  0x0000000000000000      0x0000000000000000      ................
0x555555604040  0x0000000000000000      0x0000000000000000      ................
0x555555604050  0x0000000000000000      0x0000000000000000      ................
0x555555604060  0x0000000000000000      0x0000000000000000      ................
0x555555604070  0x0000000000000000      0x0000000000000000      ................
0x555555604080  0x0000000000000000      0x0000000000000000      ................
0x555555604090  0x0000000000000000      0x00005555556042a0      .........B`UUU..
0x5555556040a0  0x0000000000000000      0x0000000000000000      ................
0x5555556040b0  0x0000000000000000      0x0000000000000000      ................
0x5555556040c0  0x0000000000000000      0x0000000000000000      ................
0x5555556040d0  0x0000000000000000      0x0000000000000000      ................
0x5555556040e0  0x0000000000000000      0x0000000000000000      ................
0x5555556040f0  0x0000000000000000      0x0000000000000000      ................
0x555555604100  0x0000000000000000      0x0000000000000000      ................
0x555555604110  0x0000000000000000      0x0000000000000000      ................
0x555555604120  0x0000000000000000      0x0000000000000000      ................
0x555555604130  0x0000000000000000      0x0000000000000000      ................
0x555555604140  0x0000000000000000      0x0000000000000000      ................
0x555555604150  0x0000000000000000      0x0000000000000000      ................
0x555555604160  0x0000000000000000      0x0000000000000000      ................
0x555555604170  0x0000000000000000      0x0000000000000000      ................
0x555555604180  0x0000000000000000      0x0000000000000000      ................
0x555555604190  0x0000000000000000      0x0000000000000000      ................
0x5555556041a0  0x0000000000000000      0x0000000000000000      ................
0x5555556041b0  0x0000000000000000      0x0000000000000000      ................
0x5555556041c0  0x0000000000000000      0x0000000000000000      ................
0x5555556041d0  0x0000000000000000      0x0000000000000000      ................
0x5555556041e0  0x0000000000000000      0x0000000000000000      ................
0x5555556041f0  0x0000000000000000      0x0000000000000000      ................
0x555555604200  0x0000000000000000      0x0000000000000000      ................
0x555555604210  0x0000000000000000      0x0000000000000000      ................
0x555555604220  0x0000000000000000      0x0000000000000000      ................
0x555555604230  0x0000000000000000      0x0000000000000000      ................
0x555555604240  0x0000000000000000      0x0000000000000000      ................
0x555555604250  0x0000000000000000      0x0000000000000000      ................
0x555555604260  0x0000000000000000      0x0000000000000000      ................
0x555555604270  0x0000000000000000      0x0000000000000000      ................
0x555555604280  0x0000000000000000      0x0000000000000000      ................
0x555555604290  0x0000000000000000      0x0000000000000031      ........1.......
0x5555556042a0  0x0000000000000000      0x0000555555604010      .........@`UUU..
0x5555556042b0  0x696c2f756e672d78      0x2e37322e322d6362      x-gnu/libc-2.27.
0x5555556042c0  0x0000000000006f73      0x0000000000000051      so......Q.......
0x5555556042d0  0x6f20746f6e6e6163      0x72616873206e6570      cannot open shar
0x5555556042e0  0x63656a626f206465      0x2f00656c69662074      ed object file./
0x5555556042f0  0x5f3638782f62696c      0x78756e696c2d3436      lib/x86_64-linux
0x555555604300  0x62696c2f756e672d      0x732e37322e322d63      -gnu/libc-2.27.s
0x555555604310  0x000000000000006f      0x0000000000020cf1      o...............
```

既存ツールの課題：表示される情報がわかりづらい

出力を「**見慣れる**」必要がある

メモリ全体を表示するコマンドの出力も、そもそもプログラム内のメモリのレイアウトのイメージができていないと理解しづらい



```
pwndbg> vmmap
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
    0x555555400000     0x555555403000 r-xp     3000 0      /home/taigaono/test/beginners_heap/chall
    0x555555602000     0x555555603000 r--p     1000 2000   /home/taigaono/test/beginners_heap/chall
    0x555555603000     0x555555604000 rw-p     1000 3000   /home/taigaono/test/beginners_heap/chall
    0x555555604000     0x555555625000 rw-p    21000 0      [heap]
    0x7ffff7db9000     0x7ffff7dbc000 rw-p     3000 0      [anon_7ffff7db9]
    0x7ffff7dbc000     0x7ffff7de1000 r--p    25000 0      /usr/lib/x86_64-linux-gnu/libc-2.31.so
    0x7ffff7de1000     0x7ffff7f59000 r-xp   178000 25000  /usr/lib/x86_64-linux-gnu/libc-2.31.so
    0x7ffff7f59000     0x7ffff7fa3000 r--p    4a000 19d000 /usr/lib/x86_64-linux-gnu/libc-2.31.so
    0x7ffff7fa3000     0x7ffff7fa4000 ---p     1000 1e7000 /usr/lib/x86_64-linux-gnu/libc-2.31.so
    0x7ffff7fa4000     0x7ffff7fa7000 r--p     3000 1e7000 /usr/lib/x86_64-linux-gnu/libc-2.31.so
    0x7ffff7fa7000     0x7ffff7faa000 rw-p     3000 1ea000 /usr/lib/x86_64-linux-gnu/libc-2.31.so
    0x7ffff7faa000     0x7ffff7fae000 rw-p     4000 0      [anon_7ffff7faa]
    0x7ffff7fae000     0x7ffff7faf000 r--p     1000 0      /usr/lib/x86_64-linux-gnu/libdl-2.31.so
    0x7ffff7faf000     0x7ffff7fb1000 r-xp     2000 1000   /usr/lib/x86_64-linux-gnu/libdl-2.31.so
    0x7ffff7fb1000     0x7ffff7fb2000 r--p     1000 3000   /usr/lib/x86_64-linux-gnu/libdl-2.31.so
    0x7ffff7fb2000     0x7ffff7fb3000 r--p     1000 3000   /usr/lib/x86_64-linux-gnu/libdl-2.31.so
    0x7ffff7fb3000     0x7ffff7fb4000 rw-p     1000 4000   /usr/lib/x86_64-linux-gnu/libdl-2.31.so
    0x7ffff7fb4000     0x7ffff7fb6000 rw-p     2000 0      [anon_7ffff7fb4]
    0x7ffff7fc9000     0x7ffff7fcd000 r--p     4000 0      [vvar]
    0x7ffff7fcd000     0x7ffff7fcf000 r-xp     2000 0      [vdso]
    0x7ffff7fcf000     0x7ffff7fd0000 r--p     1000 0      /usr/lib/x86_64-linux-gnu/ld-2.31.so
    0x7ffff7fd0000     0x7ffff7ff3000 r-xp    23000 1000   /usr/lib/x86_64-linux-gnu/ld-2.31.so
    0x7ffff7ff3000     0x7ffff7ffb000 r--p     8000 24000  /usr/lib/x86_64-linux-gnu/ld-2.31.so
    0x7ffff7ffc000     0x7ffff7ffd000 r--p     1000 2c000  /usr/lib/x86_64-linux-gnu/ld-2.31.so
    0x7ffff7ffd000     0x7ffff7ffe000 rw-p     1000 2d000  /usr/lib/x86_64-linux-gnu/ld-2.31.so
    0x7ffff7ffe000     0x7ffff7fff000 rw-p     1000 0      [anon_7ffff7ffe]
    0x7ffffffde000     0x7ffffffff000 rw-p    21000 0      [stack]
0xffffffffff600000 0xffffffffff601000 --xp     1000 0      [vsyscall]
```

既存ツールの課題：まとめ

★ pwndbgの現状課題まとめ
  ○ CLIでの可視化機能が使いづらい
  ○ 出力情報が理解しづらい

バイナリ解析入門者は最初使いづらいツールの断片的な使い方を理解することが強いられ、効率的に知識をつけることが難しい

（近寄り難い分野となっている）

pwndbgの改良

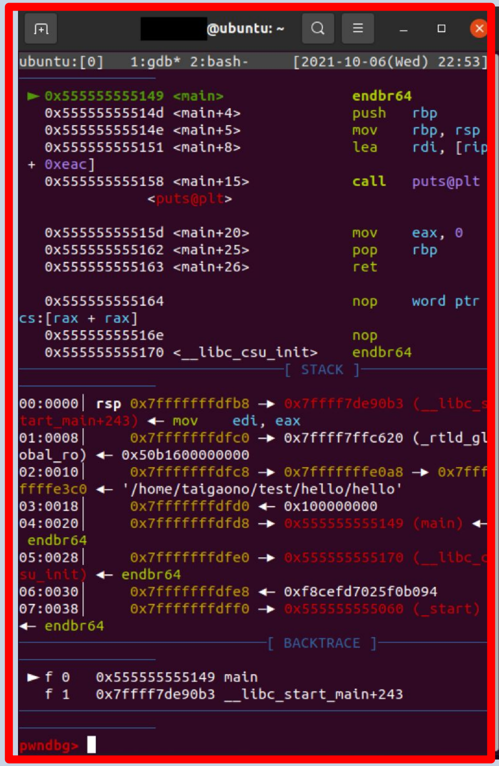
- ★ 「**pwndbg-perceptor**」を実装
  - ○ メモリ状況の可視化をベースとした、使い勝手/直感性に特化した拡張
  - ○ 「知覚」（perception）の面でpwndbg改良する
- ★ pwndbgはGithubで提供されている

  → 拡張はforkとして実装（既存のpwndbgの機能には干渉せず実装）

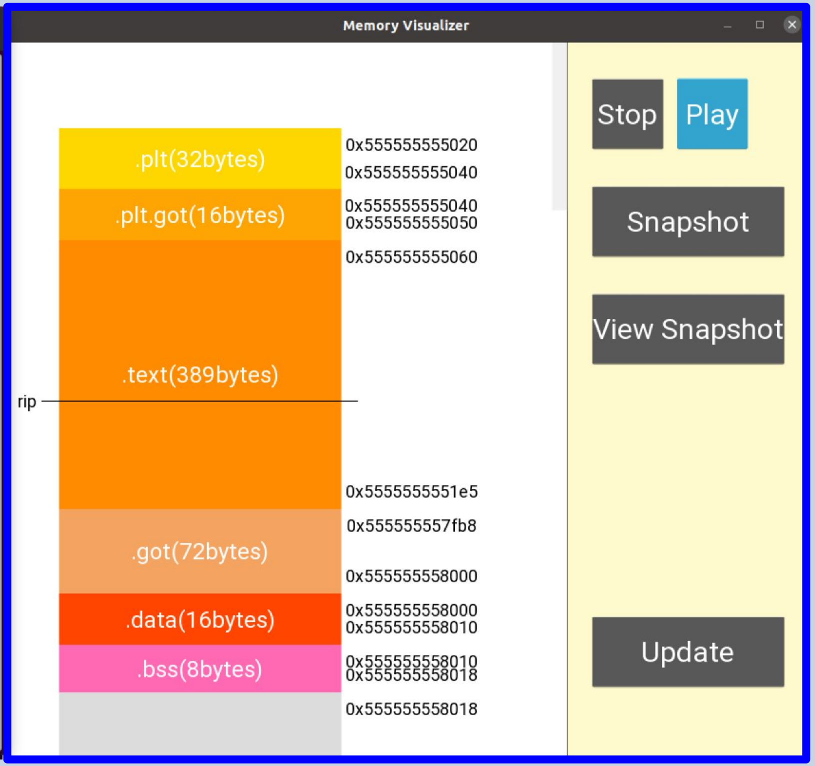
- ★ binary解析に重要なメモリ状況を把握しやすくしつつ、pwndbg自体の使い方の学習も支援する

機能概要：メモリ内の可視化

従来のpwndbg CLI



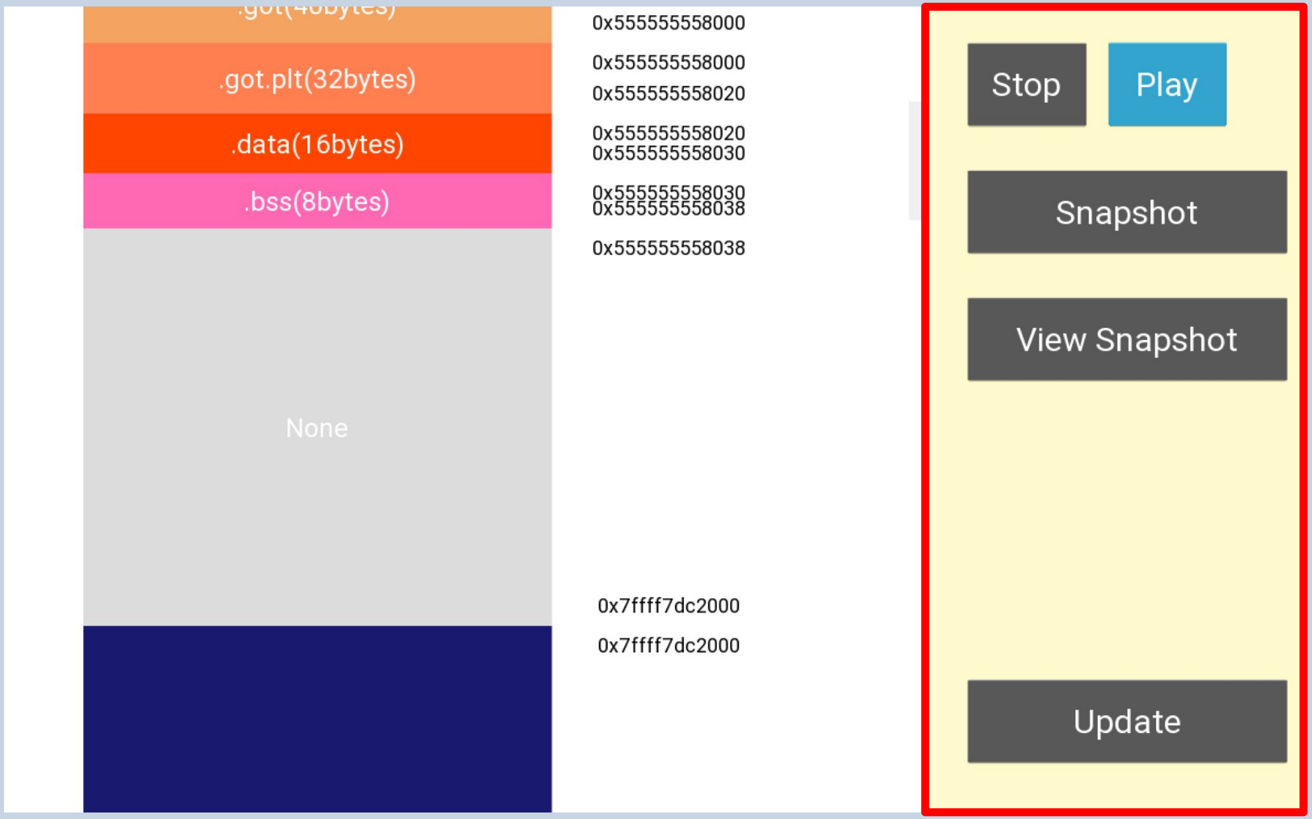
```
[redacted]@ubuntu: ~
ubuntu:[0]   1:gdb* 2:bash-      [2021-10-06(Wed) 22:53]

 ► 0x555555555149 <main>              endbr64
   0x55555555514d <main+4>            push   rbp
   0x55555555514e <main+5>            mov    rbp, rsp
   0x555555555151 <main+8>            lea    rdi, [rip
 + 0xeac]
   0x555555555158 <main+15>           call   puts@plt
                <puts@plt>

   0x55555555515d <main+20>           mov    eax, 0
   0x555555555162 <main+25>           pop    rbp
   0x555555555163 <main+26>           ret

   0x555555555164                     nop    word ptr
cs:[rax + rax]
   0x55555555516e                     nop
   0x555555555170 <__libc_csu_init>   endbr64
───────────────────────────────────[ STACK ]───────────
──────────────
00:0000│ rsp 0x7fffffffdfb8 —▸ 0x7ffff7de90b3 (__libc_s
tart_main+243) ◂— mov    edi, eax
01:0008│     0x7fffffffdfc0 —▸ 0x7ffff7ffc620 (_rtld_gl
obal_ro) ◂— 0x50b1600000000
02:0010│     0x7fffffffdfc8 —▸ 0x7fffffffe0a8 —▸ 0x7fff
ffffe3c0 ◂— '/home/taigaono/test/hello/hello'
03:0018│     0x7fffffffdfd0 ◂— 0x100000000
04:0020│     0x7fffffffdfd8 —▸ 0x555555555149 (main) ◂—
 endbr64
05:0028│     0x7fffffffdfe0 —▸ 0x555555555170 (__libc_c
su_init) ◂— endbr64
06:0030│     0x7fffffffdfe8 ◂— 0xf8cefd7025f0b094
07:0038│     0x7fffffffdff0 —▸ 0x555555555060 (_start)
◂— endbr64
────────────────────────────────[ BACKTRACE ]──────────
──────────────

 ► f 0   0x555555555149 main
   f 1   0x7ffff7de90b3 __libc_start_main+243
───────────────────────────────────────────────────────
──────────────
pwndbg> █
```



pwndbg-perceptorで追加されるGUIウィンドウ

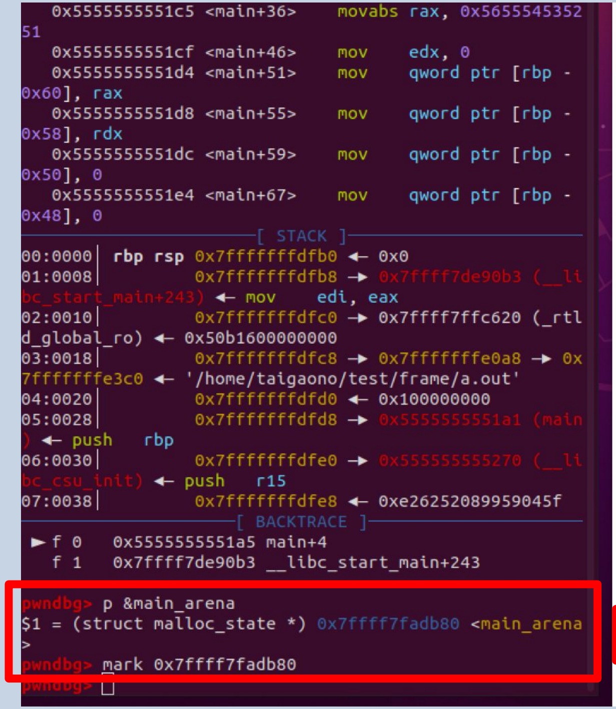
機能概要：メモリ状態の更新

ボタンを通して操作

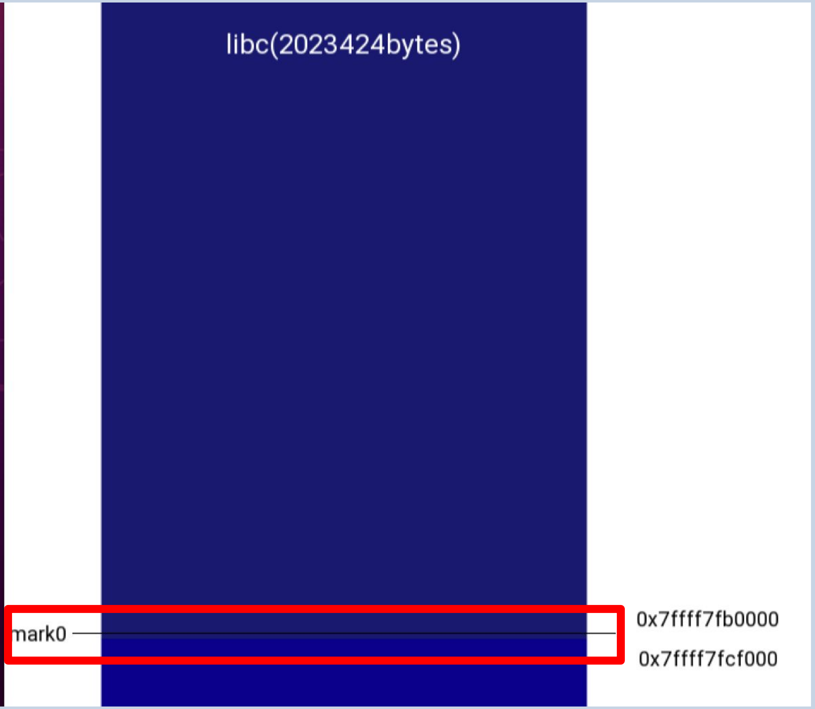


```
0x555555558000
0x555555558000
0x555555558020
0x555555558020
0x555555558030
0x555555558030
0x555555558038
0x555555558038
0x7ffff7dc2000
0x7ffff7dc2000
```

機能概要：任意のアドレス位置の可視化



```
   0x5555555551c5 <main+36>    movabs rax, 0x5655545352
51
   0x5555555551cf <main+46>    mov    edx, 0
   0x5555555551d4 <main+51>    mov    qword ptr [rbp -
0x60], rax
   0x5555555551d8 <main+55>    mov    qword ptr [rbp -
0x58], rdx
   0x5555555551dc <main+59>    mov    qword ptr [rbp -
0x50], 0
   0x5555555551e4 <main+67>    mov    qword ptr [rbp -
0x48], 0
──────────────────────────[ STACK ]──────────────────────
00:0000│ rbp rsp 0x7fffffffdfb0 ◂— 0x0
01:0008│         0x7fffffffdfb8 —▸ 0x7ffff7de90b3 (__li
bc_start_main+243) ◂— mov    edi, eax
02:0010│         0x7fffffffdfc0 —▸ 0x7ffff7ffc620 (_rtl
d_global_ro) ◂— 0x50b1600000000
03:0018│         0x7fffffffdfc8 —▸ 0x7fffffffe0a8 —▸ 0x
7fffffffe3c0 ◂— '/home/taigaono/test/frame/a.out'
04:0020│         0x7fffffffdfd0 ◂— 0x100000000
05:0028│         0x7fffffffdfd8 —▸ 0x5555555551a1 (main
) ◂— push   rbp
06:0030│         0x7fffffffdfe0 —▸ 0x555555555270 (__li
bc_csu_init) ◂— push   r15
07:0038│         0x7fffffffdfe8 ◂— 0xe26252089959045f
────────────────────────[ BACKTRACE ]────────────────────
 ► f 0   0x5555555551a5 main+4
   f 1   0x7ffff7de90b3 __libc_start_main+243

pwndbg> p &main_arena
$1 = (struct malloc_state *) 0x7ffff7fadb80 <main_arena
>
pwndbg> mark 0x7ffff7fadb80
pwndbg> 
```



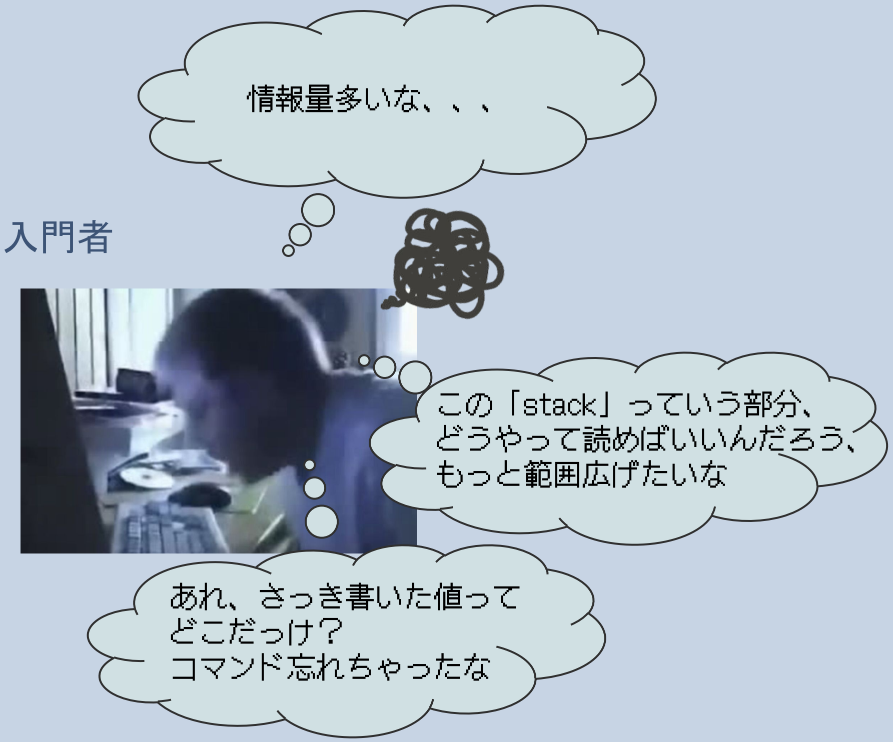
main_arenaの位置をマーキング

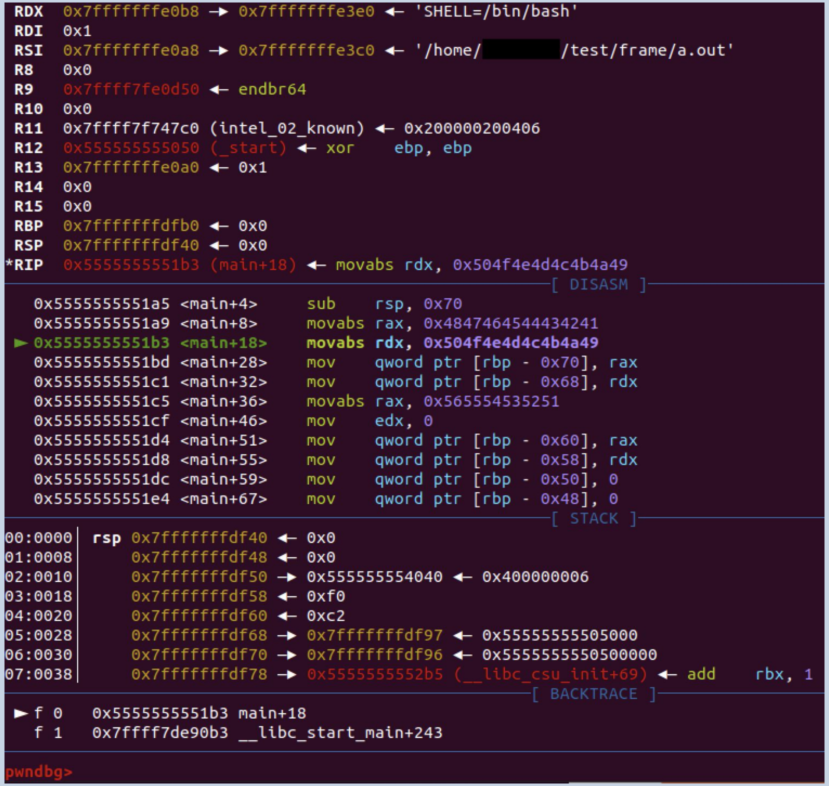
main_arenaの位置が可視化

使用例：Buffer Overflow

従来のpwndbg

入門者





```
 RDX  0x7fffffffe0b8 —▸ 0x7fffffffe3e0 ◂— 'SHELL=/bin/bash'
 RDI  0x1
 RSI  0x7fffffffe0a8 —▸ 0x7fffffffe3c0 ◂— '/home/[redacted]/test/frame/a.out'
 R8   0x0
 R9   0x7ffff7fe0d50 ◂— endbr64
 R10  0x0
 R11  0x7ffff7f747c0 (intel_02_known) ◂— 0x200000200406
 R12  0x555555555050 (_start) ◂— xor    ebp, ebp
 R13  0x7fffffffe0a0 ◂— 0x1
 R14  0x0
 R15  0x0
 RBP  0x7fffffffdfb0 ◂— 0x0
 RSP  0x7fffffffdf40 ◂— 0x0
*RIP  0x5555555551b3 (main+18) ◂— movabs rdx, 0x504f4e4d4c4b4a49
─────────────────────────────────────────[ DISASM ]──────────────────
   0x5555555551a5 <main+4>     sub    rsp, 0x70
   0x5555555551a9 <main+8>     movabs rax, 0x4847464544434241
 ► 0x5555555551b3 <main+18>    movabs rdx, 0x504f4e4d4c4b4a49
   0x5555555551bd <main+28>    mov    qword ptr [rbp - 0x70], rax
   0x5555555551c1 <main+32>    mov    qword ptr [rbp - 0x68], rdx
   0x5555555551c5 <main+36>    movabs rax, 0x565554535251
   0x5555555551cf <main+46>    mov    edx, 0
   0x5555555551d4 <main+51>    mov    qword ptr [rbp - 0x60], rax
   0x5555555551d8 <main+55>    mov    qword ptr [rbp - 0x58], rdx
   0x5555555551dc <main+59>    mov    qword ptr [rbp - 0x50], 0
   0x5555555551e4 <main+67>    mov    qword ptr [rbp - 0x48], 0
─────────────────────────────────────────[ STACK ]───────────────────
00:0000│ rsp 0x7fffffffdf40 ◂— 0x0
01:0008│     0x7fffffffdf48 ◂— 0x0
02:0010│     0x7fffffffdf50 —▸ 0x555555554040 ◂— 0x400000006
03:0018│     0x7fffffffdf58 ◂— 0xf0
04:0020│     0x7fffffffdf60 ◂— 0xc2
05:0028│     0x7fffffffdf68 —▸ 0x7fffffffdf97 ◂— 0x55555555505000
06:0030│     0x7fffffffdf70 —▸ 0x7fffffffdf96 ◂— 0x5555555550500000
07:0038│     0x7fffffffdf78 —▸ 0x5555555552b5 (__libc_csu_init+69) ◂— add    rbx, 1
───────────────────────────────────────[ BACKTRACE ]─────────────────
 ► f 0   0x5555555551b3 main+18
   f 1   0x7ffff7de90b3 __libc_start_main+243
─────────────────────────────────────────────────────────────────────
pwndbg>
```

使用例：Buffer Overflow

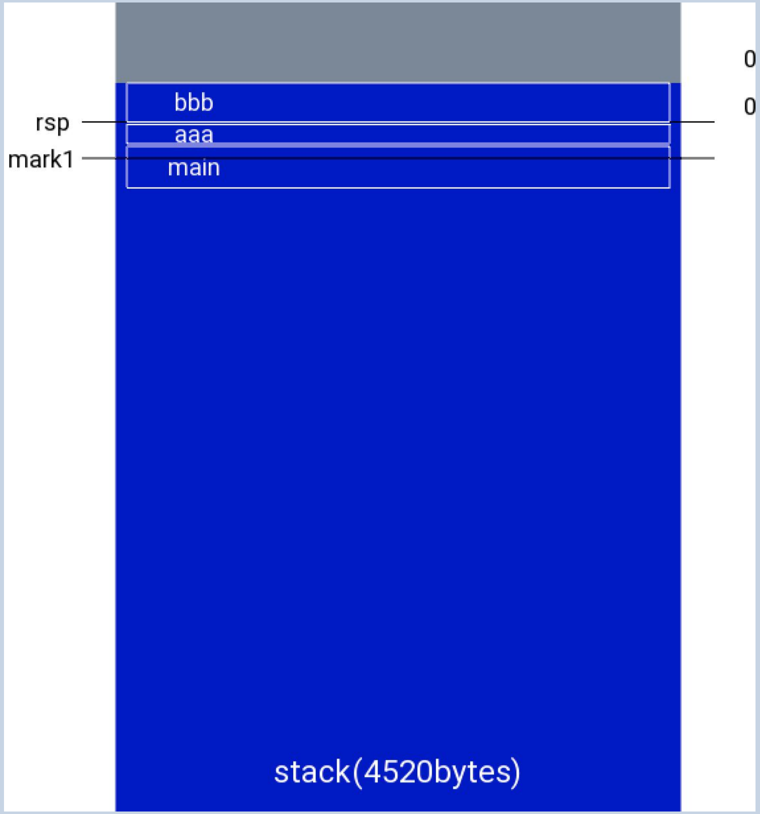
pwndbg-perceptor を使うと

なるほど、ここがstackか。
一番下にあるんだな

入門者

さっきデータ書き込んだところマークしたけど、stackのここにあるんだな

Stack frameって厳密にはこんな感じになってるのか



新規性/実用性

- ★ **新規性**
  - ○ 人間がイメージしやすいメモリ像の可視化ツールは初めて

- ★ **実用性**
  - ○ githubからcloneするだけでOK
  - ○ pwndbgの拡張なので使い勝手も良い

今後の展望

- **追加機能**
  - さらに多くのメモリ箇所の可視化
  - UI/UXの改善

- **今後の使用/開発**
  - 自身のCTFチームでも活用/改良する予定
  - pwndbg本家へのpull request

作業概要/役割分担

- ★ 作業期間：～１ヶ月
- ★ 打ち合わせ頻度：週１頻度でZoom MTG、常にslackでやりとり
- ★ アイディア出し：全員
- ★ 開発
  - ○ メモリの情報収集部分：小野、松尾
  - ○ GUI開発：刀塚、河岡、谷田部
- ★ 機能テスト/ドキュメント確認等：全員各自
- ★ 動画資料作成：全員

Thank you

MWS Cup 事前課題：pwndbg-perceptor

Team: MONKY
小野大河　松尾和輝　刀塚敦子　河岡諒　谷田部和貴