Auth Or Out

1) Checked Security

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Auth or out/challenge]
└─$ checksec auth-or-out
[!] Could not populate PLT: Invalid argument (UC_ERR_ARG)
[*] '/home/vigneswar/Pwn/Auth or out/challenge/auth-or-out'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Auth or out/challenge]
└─$
```

2) Decompiled the code

Decompile: main - (auth-or-out)

```
/* WARNING: Unknown calling convention */

int main(void)

{
  _Bool _Var1;
  ulonglong uVar2;
  long in_FS_OFFSET;
  uchar CustomHeap [14336];
  undefined auStack_18 [8];
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  setvbuf(stdin,(char *)0x0,2,0);
  setvbuf(stdout,(char *)0x0,2,0);
  memset(CustomHeap,0,0x3800);
  _Var1 = ta_init(CustomHeap,auStack_18,10,0x10,8);
  if (_Var1) {
    puts("*** Welcome to DZONERZY authors editor v0.11.2 ***");
switchD_00101a4f_caseD_0:
    uVar2 = print_menu();
    switch(uVar2) {
    case 1:
      add_author();
      goto switchD_00101a4f_caseD_0;
    case 2:
      modify_author();
      goto switchD_00101a4f_caseD_0;
    case 3:
      print_author();
      goto switchD_00101a4f_caseD_0;
    case 4:
      delete_author();
      goto switchD_00101a4f_caseD_0;
    case 5:
      goto switchD_00101a4f_caseD_5;
    }
  }
LAB_00101a9b:
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return 0;
switchD_00101a4f_caseD_5:
  puts("bye bye!");
  goto LAB_00101a9b;
}
```

Decompile: ta_init - (auth-or-out)

```
_Bool ta_init(void *base,void *limit,size_t heap_blocks,size_t split_thresh,size_t alignment)

{
  size_t alignment-local;
  size_t split_thresh-local;
  size_t heap_blocks-local;
  void *limit-local;
  void *base-local;
  Block *block;
  size_t i;

  heap = (Heap *)base;
  heap_limit = limit;
  heap_split_thresh = split_thresh;
  heap_alignment = alignment;
  heap_max_blocks = heap_blocks;
  *(undefined8 *)base = 0;
  heap->used = (Block *)0x0;
  heap->fresh = (Block *)(heap + 1);
  heap->top = (size_t)(heap->fresh + heap_blocks);
  block = heap->fresh;
  i = heap_max_blocks;
  while (i = i - 1, i != 0) {
    block->next = block + 1;
    block = block + 1;
  }
  block->next = (Block *)0x0;
  return true;
}
```

Structure Editor - Block (auth-or-out) [CodeBrowser: Pwn:/auth-or-out]

Edit Help

Structure Editor - Block (auth-or-out)

Offset	Length	Mnemonic	DataType	Name	Comment
0x0	0x8	void *	void *	addr	
0x8	0x8	Block *	Block *	next	
0x10	0x8	size_t	size_t	size	

Structure Editor - Heap (auth-or-out) [CodeBrowser: Pwn:/auth-or-out]

Edit Help

Structure Editor - Heap (auth-or-out)

Offset	Length	Mnemonic	DataType	Name	Comment
0x0	0x8	Block *	Block *	free	
0x8	0x8	Block *	Block *	used	
0x10	0x8	Block *	Block *	fresh	
0x18	0x8	size_t	size_t	top	

Decompile: print_menu - (auth-or-out)

```c

/* WARNING: Unknown calling convention */

ulonglong print_menu(void)

{
  ulonglong uVar1;
  ulonglong choice;

  do {
    do {
      puts("1 - Add Author");
      puts("2 - Modify Author");
      puts("3 - Print Author");
      puts("4 - Delete Author");
      puts("5 - Exit");
      putchar(10);
      printf("Choice: ");
      uVar1 = get_number();
      putchar(10);
    } while (uVar1 == 0);
  } while (5 < uVar1);
  return uVar1;
}

```

Decompile: add_author - (auth-or-out)

```
 4 void add_author(void)
 5
 6 {
 7   PAuthor p_Var1;
 8   ulonglong uVar2;
 9   char *pcVar3;
10   size_t size;
11   ulonglong i;
12   ulonglong NoteSize;
13
14   i = 0;
15   while( true ) {
16     if (9 < i) {
17       puts("MAX AUTHORS REACHED!");
18       return;
19     }
20     if (authors[i] == (PAuthor)0x0) break;
21     i = i + 1;
22   }
23   p_Var1 = (PAuthor)ta_alloc(0x38);
24   authors[i] = p_Var1;
25   if (authors[i] != (PAuthor)0x0) {
26     authors[i]->Print = PrintNote;
27     printf("Name: ");
28     get_from_user(authors[i]->Name,0x10);
29     printf("Surname: ");
30     get_from_user(authors[i]->Surname,0x10);
31     printf("Age: ");
32     p_Var1 = authors[i];
33     uVar2 = get_number();
34     p_Var1->Age = uVar2;
35     printf("Author Note size: ");
36     uVar2 = get_number();
37     if (uVar2 != 0) {
38       p_Var1 = authors[i];
39       pcVar3 = (char *)ta_alloc(uVar2 + 1);
40       p_Var1->Note = pcVar3;
41       if (authors[i]->Note == (char *)0x0) {
42         printf("Invalid allocation!");
43                     /* WARNING: Subroutine does not return */
44         exit(0);
45       }
46       printf("Note: ");
47       if (uVar2 < 0x101) {
48         size = uVar2 + 1;
49       }
50       else {
```

```
51          size = 0x100;
52        }
53        get_from_user(authors[i]->Note,size);
54      }
55      printf("Author %llu added!\n\n",i + 1);
56      return;
57    }
58    printf("Invalid allocation!");
59                        /* WARNING: Subroutine does not return */
60    exit(0);
61  }
62
```

Decompile: ta_alloc - (auth-or-out)

```
1
2  void * ta_alloc(size_t num)
3
4  {
5    Block *pBVar1;
6    void *pvVar2;
7    size_t num-local;
8    Block *block;
9
10   pBVar1 = alloc_block(num);
11   if (pBVar1 == (Block *)0x0) {
12     pvVar2 = (void *)0x0;
13   }
14   else {
15     pvVar2 = pBVar1->addr;
16   }
17   return pvVar2;
18 }
19
```

```
Block * alloc_block(size_t num)

{
  void *pvVar1;
  bool bVar2;
  _Bool _Var3;
  ulong uVar4;
  ulong uVar5;
  Block *pBVar6;
  size_t num-local;
  int is_top;
  Block *ptr;
  Block *prev;
  size_t top;
  size_t new_top;
  size_t excess;
  Block *split;

  ptr = heap->free;
  prev = (Block *)0x0;
  pvVar1 = (void *)heap->top;
  uVar4 = -heap_alignment & (num + heap_alignment) - 1;
  while( true ) {
    if (ptr == (Block *)0x0) {
      if ((heap->fresh == (Block *)0x0) || (heap_limit < (void *)(uVar4 + (long)pvVar1))) {
        pBVar6 = (Block *)0x0;
      }
      else {
        pBVar6 = heap->fresh;
        heap->fresh = pBVar6->next;
        pBVar6->addr = pvVar1;
        pBVar6->next = heap->used;
        pBVar6->size = uVar4;
        _Var3 = ta_safe(pBVar6->addr);
        if (_Var3) {
          heap->used = pBVar6;
          heap->top = (size_t)(void *)(uVar4 + (long)pvVar1);
        }
        else {
          pBVar6 = (Block *)0x0;
        }
      }
      return pBVar6;
    }
    if (((void *)(ptr->size + (long)ptr->addr) < pvVar1) ||
       (heap_limit < (void *)((long)ptr->addr + uVar4))) {
      bVar2 = false;
    }
    else {
      bVar2 = true;
    }
    if ((bVar2) || (uVar4 <= ptr->size)) break;
    prev = ptr;
    ptr = ptr->next;
  }
  if (prev == (Block *)0x0) {
    heap->free = ptr->next;
  }
  else {
    prev->next = ptr->next;
  }
  ptr->next = heap->used;
  _Var3 = ta_safe(ptr->addr);
  if (!_Var3) {
    return (Block *)0x0;
  }
  heap->used = ptr;
  if (bVar2) {
    ptr->size = uVar4;
    heap->top = uVar4 + (long)ptr->addr;
    return ptr;
  }
  if (heap->fresh == (Block *)0x0) {
    return ptr;
  }
  uVar5 = ptr->size - uVar4;
  if (uVar5 < heap_split_thresh) {
    return ptr;
  }
  ptr->size = uVar4;
  pBVar6 = heap->fresh;
  heap->fresh = pBVar6->next;
  pBVar6->addr = (void *)(uVar4 + (long)ptr->addr);
  pBVar6->size = uVar5;
  insert_block(pBVar6);
  compact();
  return ptr;
}
```

Decompile: ta_safe - (auth-or-out)

```c

_Bool ta_safe(void *addr)

{
  void *addr-local;
  Block *used;

  used = heap->used;
  while( true ) {
    if (used == (Block *)0x0) {
      return true;
    }
    if ((used->addr <= addr) && (addr < (void *)((long)used->addr + used->size))) break;
    used = used->next;
  }
  return false;
}

```

Decompile: get_from_user - (auth-or-out)

```c

int get_from_user(char *buffer,size_t size)

{
  int iVar1;
  ssize_t sVar2;
  long in_FS_OFFSET;
  size_t size-local;
  char *buffer-local;
  char c;
  int fd;
  size_t cnt;
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  cnt = 0;
  if ((buffer == (char *)0x0) || (size == 0)) {
    iVar1 = 0;
  }
  else {
    fd = 0;
    while( true ) {
      sVar2 = read(fd,&c,1);
      if ((sVar2 != 1) || (size - 1 <= cnt)) break;
      if (c == '\n') {
        iVar1 = 1;
        goto LAB_00101359;
      }
      buffer[cnt] = c;
      cnt = cnt + 1;
    }
    iVar1 = 1;
  }
LAB_00101359:
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return iVar1;
}

```

Decompile: get_number - (auth-or-out)

```
ulonglong get_number(void)

{
  long lVar1;
  ulonglong uVar2;
  long in_FS_OFFSET;
  char *pEnd;
  char choice [32];

  lVar1 = *(long *)(in_FS_OFFSET + 0x28);
  choice[0] = '\0';
  choice[1] = '\0';
  choice[2] = '\0';
  choice[3] = '\0';
  choice[4] = '\0';
  choice[5] = '\0';
  choice[6] = '\0';
  choice[7] = '\0';
  choice[8] = '\0';
  choice[9] = '\0';
  choice[10] = '\0';
  choice[11] = '\0';
  choice[12] = '\0';
  choice[13] = '\0';
  choice[14] = '\0';
  choice[15] = '\0';
  choice[16] = '\0';
  choice[17] = '\0';
  choice[18] = '\0';
  choice[19] = '\0';
  choice[20] = '\0';
  choice[21] = '\0';
  choice[22] = '\0';
  choice[23] = '\0';
  choice[24] = '\0';
  choice[25] = '\0';
  choice[26] = '\0';
  choice[27] = '\0';
  choice[28] = '\0';
  choice[29] = '\0';
  choice[30] = '\0';
  choice[31] = '\0';
  __isoc99_scanf(&DAT_00101b54,choice);
  uVar2 = strtoull(choice,&pEnd,10);
  if (lVar1 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return uVar2;
}
```

Decompile: modify_author - (auth-or-out)

```c

/* WARNING: Unknown calling convention */

void modify_author(void)

{
  PAuthor buffer;
  ulonglong uVar1;
  ulonglong authorid;
  PAuthor a;

  do {
    printf("Author ID: ");
    uVar1 = get_number();
    putchar(10);
  } while (10 < uVar1);
  buffer = authors[uVar1 - 1];
  if (buffer == (PAuthor)0x0) {
    printf("Author %llu does not exists!\n\n",uVar1);
  }
  else {
    printf("Name: ");
    get_from_user(buffer->Name,0x10);
    printf("Surname: ");
    get_from_user(buffer->Surname,0x11);
    printf("Age: ");
    uVar1 = get_number();
    buffer->Age = uVar1;
    putchar(10);
  }
  return;
}

```

Decompile: print_author - (auth-or-out)

```c

/* WARNING: Unknown calling convention */

void print_author(void)

{
  PAuthor p_Var1;
  ulonglong uVar2;
  ulonglong authorid;
  PAuthor a;

  do {
    printf("Author ID: ");
    uVar2 = get_number();
    putchar(10);
  } while (10 < uVar2);
  p_Var1 = authors[uVar2 - 1];
  if (p_Var1 == (PAuthor)0x0) {
    printf("Author %llu does not exists!\n\n",uVar2);
  }
  else {
    puts("----------------------");
    printf("Author %llu\n",uVar2);
    printf("Name: %s\n",p_Var1);
    printf("Surname: %s\n",p_Var1->Surname);
    printf("Age: %llu\n",p_Var1->Age);
    (*p_Var1->Print)(p_Var1->Note);
    puts("----------------------");
    putchar(10);
  }
  return;
}

```

Decompile: delete_author - (auth-or-out)

```c

/* WARNING: Unknown calling convention */

void delete_author(void)

{
  PAuthor free;
  ulonglong uVar1;
  ulonglong authorid;
  PAuthor a;

  do {
    printf("Author ID: ");
    uVar1 = get_number();
    putchar(10);
  } while (10 < uVar1);
  free = authors[uVar1 - 1];
  if (free == (PAuthor)0x0) {
    printf("Author %llu does not exists!\n\n",uVar1);
  }
  else {
    ta_free(free->Note);
    ta_free(free);
    authors[uVar1 - 1] = (PAuthor)0x0;
    printf("Author %llu deleted!\n\n",uVar1);
  }
  return;
}

```

Decompile: ta_free - (auth-or-out)

```
1
2 _Bool ta_free(void *free)
3
4 {
5   void *free-local;
6   Block *block;
7   Block *prev;
8
9   block = heap->used;
10  prev = (Block *)0x0;
11  while( true ) {
12    if (block == (Block *)0x0) {
13      return false;
14    }
15    if (free == block->addr) break;
16    prev = block;
17    block = block->next;
18  }
19  if (prev == (Block *)0x0) {
20    heap->used = block->next;
21  }
22  else {
23    prev->next = block->next;
24  }
25  insert_block(block);
26  compact();
27  return true;
28 }
29
```

Decompile: insert_block - (auth-or-out)

```
void insert_block(Block *block)

{
  Block *block-local;
  Block *ptr;
  Block *prev;

  ptr = heap->free;
  prev = (Block *)0x0;
  for (; (ptr != (Block *)0x0 && (ptr->addr < block->addr)); ptr = ptr->next) {
    prev = ptr;
  }
  if (prev == (Block *)0x0) {
    heap->free = block;
  }
  else {
    prev->next = block;
  }
  block->next = ptr;
  return;
}

```

Decompile: compact - (auth-or-out)

```
/* WARNING: Unknown calling convention */

void compact(void)

{
  Block *pBVar1;
  Block *ptr;
  Block *prev;
  Block *scan;
  size_t new_size;
  Block *next;

  for (ptr = heap->free; ptr != (Block *)0x0; ptr = ptr->next) {
    prev = ptr;
    for (scan = ptr->next;
        (scan != (Block *)0x0 && ((void *)((long)prev->addr + prev->size) == scan->addr));
        scan = scan->next) {
      prev = scan;
    }
    if (prev != ptr) {
      ptr->size = (long)prev->addr + (prev->size - (long)ptr->addr);
      pBVar1 = prev->next;
      release_blocks(ptr->next,prev->next);
      ptr->next = pBVar1;
    }
  }
  return;
}

```

Decompile: release_blocks - (auth-or-out)

```
1
2 void release_blocks(Block *scan,Block *to)
3
4 {
5   Block *pBVar1;
6   Block *to-local;
7   Block *scan-local;
8   Block *scan_next;
9
10  scan-local = scan;
11  while (scan-local != to) {
12    pBVar1 = scan-local->next;
13    scan-local->next = heap->fresh;
14    heap->fresh = scan-local;
15    scan-local->addr = (void *)0x0;
16    scan-local->size = 0;
17    scan-local = pBVar1;
18  }
19  return;
20 }
21
```