Blablabla чтот про кучу

Спикер: @thsage

definitely not vulnerability researcher

whoami

- Main SPRUSH heaper
- Former Cyber Threat Researcher
- Senior Vulnerability Researcher

Терминология

- Куча (heap) - регионы в адресном пространстве программы, использующиеся аллокатором ptmalloc.

```
malloc(SIZE);
```

```
0x405000           0x426000 rw-p    21000      0 [heap]
```

Терминология

- Чанк (chunk) - структура внутри адресного пространства, принадлежащего куче, используемая для управления памятью библиотекой libc.

```
Allocated chunk | PREV_INUSE
Addr: 0x405290
Size: 0x30 (with flag bits: 0x31)
```

Терминология

- Бины (heap bins) - библиотечные структуры данных, используемые для управления освобожденными чанками внутри кучи.

```
tcachebins
0x30 [  1]: 0x4052a0 ◂— 0
fastbins
empty
unsortedbin
empty
smallbins
empty
largebins
empty
pwndbg>
```

malloc

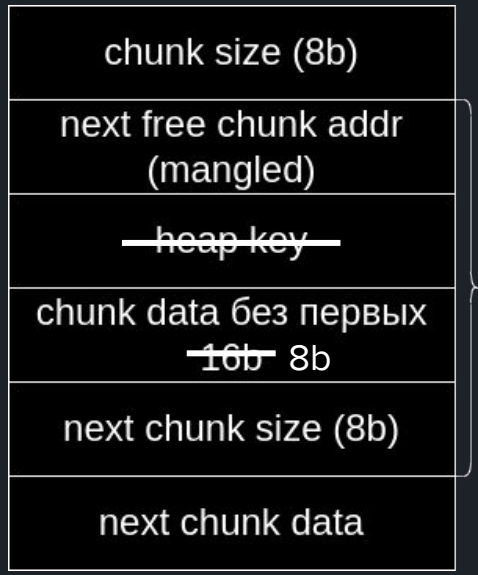

```
Allocated chunk | PREV_INUSE
Addr: 0x555555559290
Size: 0x40 (with flag bits: 0x41)
```

Размер чанка всегда кратен 16
Прибавляем 8 байт (next chunk size) + округляем до 16 в большую сторону

```
pwndbg> x/10xg 0x555555559290
0x555555559290: 0x0000000000000000      0x0000000000000041
0x5555555592a0: 0x0807060504030201      0x100f0e0d0c0b0a09
0x5555555592b0: 0x1817161514131211      0x201f1e1d1c1b1a19
0x5555555592c0: 0x2827262524232221      0x302f2e2d2c2b2a29
0x5555555592d0: 0x0000000000000000      0x0000000000020d31
```

size

free in tcache and fastbins



Попадают размеры от 0x20 до 0x420
До 7 штук каждого размера

Попадают размеры до 0x80 при заполнении tcache

mangled 0

```
pwndbg> x/10xg 0x555555559290
0x555555559290: 0x0000000000000000      0x0000000000000041
0x5555555592a0: 0x0000000555555559      0x8e216298b4601cac
0x5555555592b0: 0x1817161514131211      0x201f1e1d1c1b1a19
0x5555555592c0: 0x2827262524232221      0x302f2e2d2c2b2a29
0x5555555592d0: 0x0000000000000000      0x0000000000020d31
```

pointer mangling

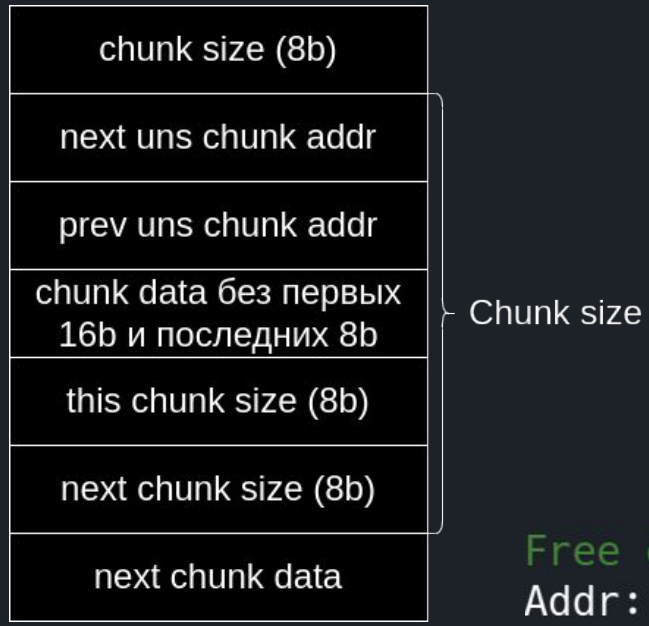
Входные данные:
- адрес чанка в куче (value)
- адрес следующего чанка в цепочке (addr)
- адрес страницы чанка в куче (value & 0xfffffffffffff000)

Mangled address:
((value & 0xfffffffffffff000) >> 12) ^ addr

```
0x5555555592d0: 0x0000000000000000
0x5555555592e0: 0x0000000555555559
0x5555555592f0: 0x0000000000000000
```

addr: 0
value: 0x5555555592e0
mangled:
- value & 0xfffffffffffff000: 0x555555559000
- * >> 12: 0x555555559
- * ^ 0: 0x555555559

free in unsorted



- Попадают размеры от 0x420
- Попадают размеры от 0x80 при заполнении tcache
- Изначально next и prev указывают на адрес в libc
- Если лежат несколько подряд, то сливаются в один (consolidate)
- бит PREV IN USE (& 0b001) размера следующего чанка становится 0

```
Free chunk (unsortedbin) | PREV_INUSE
Addr: 0x5555555596f0
Size: 0xa0 (with flag bits: 0xa1)
fd: 0x7ffff7f8eb20
bk: 0x7ffff7f8eb20
```

Правила выделения

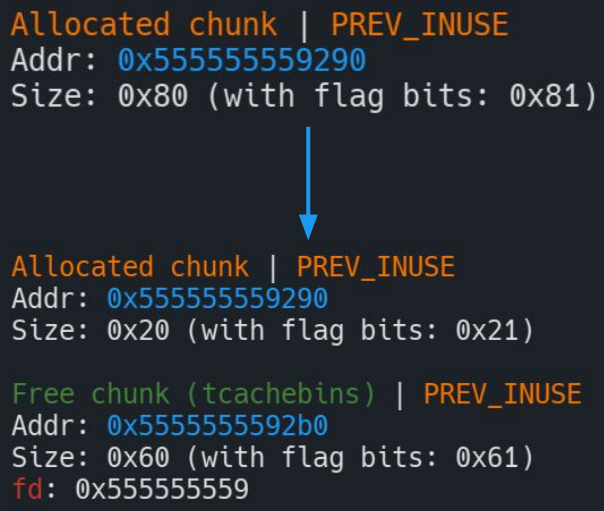
1. если есть в tcache - берётся из tcache
2. если есть в fastbin - берётся из fastbin + 7 чанков из fastbin переносятся в tcache
3. если есть место в unsorted - выделяется внутри него, размер unsorted уменьшается
4. выделяется из top чанка

**изначально почти вся куча – большой top chunk*

realloc

```
realloc(pointer, NEW_SIZE);
```

- Если чанк находится на границе с top-чанком и размер увеличивается, то расширяет чанк до требуемого значения.
- Если размер чанка должен уменьшится, то разбивает чанк на два, и освобождает второй чанк.
- Для большинства других случаев: выделяет новый чанк, копирует данные из старого, а затем высвобождает его.
- Не использует TCACHE для аллоцирования чанков.



calloc

```
calloc(SIZE);
```

- Выделяет чанк, аналогично функции malloc.
- Зануляет данные внутри чанка.
- Не использует TCACHE для аллоцирования чанков.

Структура tcache

1.

Для каждого размера от 0x20 до 0x420 хранятся количество чанков и адрес следующего выделяемого ПРЯМ АДРЕС, БЕЗ МАНГЛИНГА

2.

Структура - количество каждого от 0x20 до 0x420 по 2 байта
Затем адрес следующего для каждого размером 8 байт

3.

Структура находится в начале кучи и занимает размер 0x290

Команды-помощники pwndbg:

- bins
- heap
- tcache

```
Allocated chunk | PREV_INUSE
Addr: 0x555555559000
Size: 0x290 (with flag bits: 0x291)
```

BI.ZONE

НИЯУ МИФИ

Idea

Вектор интересов

1. Лик адресов libc
2. Выделение где угодно (ну разумеется в rw регионах, преимущественно libc)
3. Лик stack-а
4. Запись на стек rop-чика
5. Запись в got, если есть адреса программы
6. Запись в структуры внутри libc

Вектор интересов

1. Куча также используется многими библиотеками, например libstdc++.
2. В куче могут хранится другие критичные структуры данных, не контролируемые пользователем.
3. С помощью уязвимостей в программе могут появиться возможности изменения структур, например, адресов на вызываемые виртуальные и перегруженные функции языка c++.

UAF

1. Мы можем изменять чанк, который находится в bin-е
2. Переписываем первые 8 байт
3. Выделяем его
4. Указатель на следующий чанк взялся из того что мы изменили
5. Следующий чанк выделяется в нужном нам месте

Additional content 1: mmaped chunks

Чанки больше максимального размера кучи (обычно, 0x21000) выделяются в отдельном регионе, который прилегает к libc (или ld).

```
malloc(0x20fe0)

0x7ffff7fa0000     0x7ffff7fc1000 rw-p    21000      0 [anon_7ffff7fa0]
0x7ffff7fc1000     0x7ffff7fc5000 r--p     4000      0 [vvar]
0x7ffff7fc5000     0x7ffff7fc7000 r-xp     2000      0 [vdso]
0x7ffff7fc7000     0x7ffff7fc8000 r--p     1000      0 /usr/lib/ld-linux-x86-64.so.2
0x7ffff7fc8000     0x7ffff7ff1000 r-xp    29000   1000 /usr/lib/ld-linux-x86-64.so.2
0x7ffff7ff1000     0x7ffff7ffb000 r--p     a000  2a000 /usr/lib/ld-linux-x86-64.so.2
```

Additional content 2: PREV_INUSE

1. Первый бит в адресе.
2. Выставлен в 0, если предыдущий находится в unsorted bin, в остальных случаях - 1.
3. Используется для определения возможности слияния нескольких чанков в один.

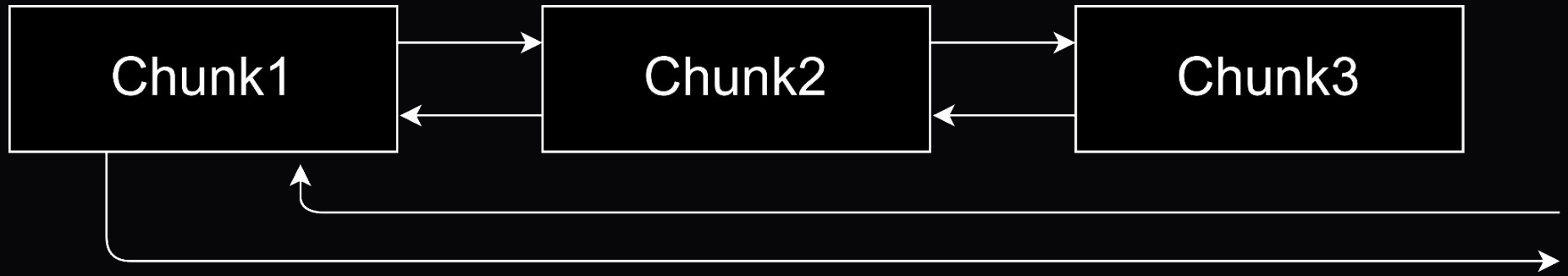
```
Free chunk (unsortedbin) | PREV_INUSE
Addr: 0x5555555592b0
Size: 0x440 (with flag bits: 0x441)
fd: 0x7ffff7f80b20
bk: 0x7ffff7f80b20

Allocated chunk
Addr: 0x5555555596f0
Size: 0x20 (with flag bits: 0x20)
```

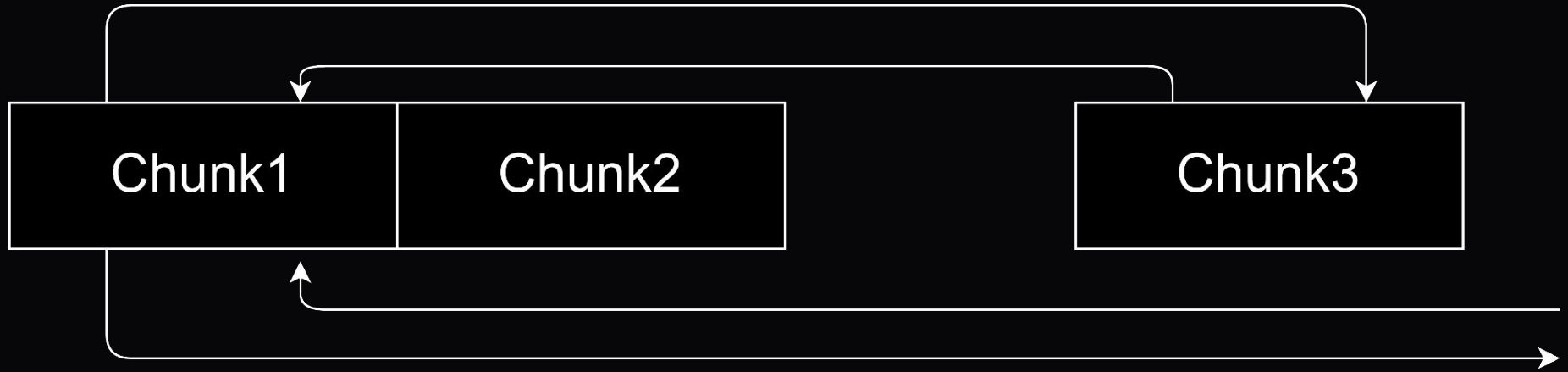
```
Allocated chunk | PREV_INUSE
Addr: 0x555555559000
Size: 0x290 (with flag bits: 0x291)

Allocated chunk | PREV_INUSE
Addr: 0x555555559290
Size: 0x20 (with flag bits: 0x21)
```

Additional content 3: consolidation



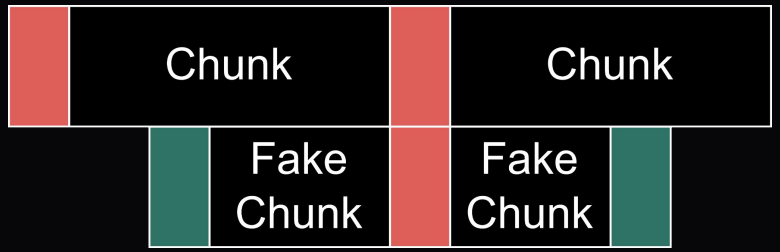
Additional content 3: consolidation



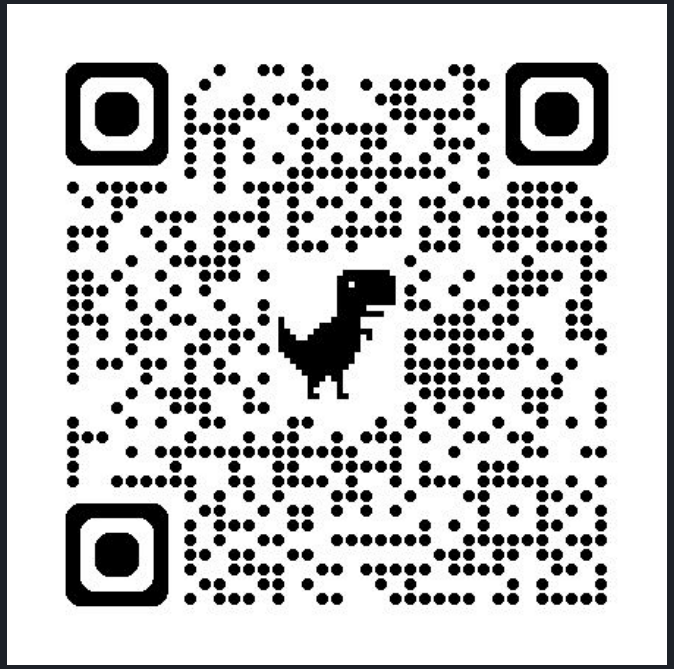
Additional content 3: consolidation

1. chunk_size == next_chunk->prev_size
2. chunk->fd->bk == chunk
3. chunk->bk->fd == chunk

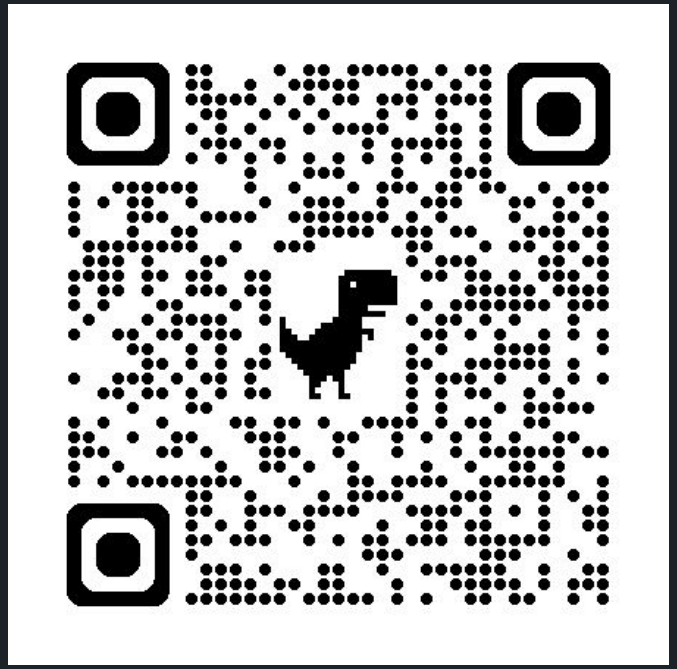
Additional content 4: fake chunks



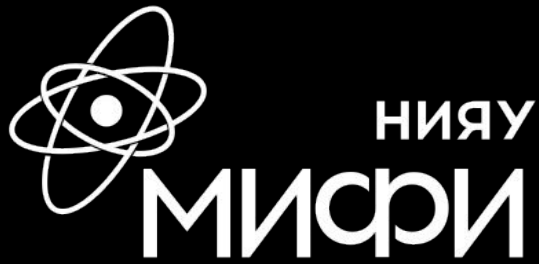
Референсы



Исходный код libc



how2heap



X

PONPONPON