__std_type_info_name

__std_type_info_name тот же typeid(*classAddress).name, это весьма подозрительный метод из-за того, что редко встречается необходимость получать имя класса игре. Рассмотрим использования этого метода на примере New World которая использует Lumberyard Engine.

В этой игре название классов хранятся в массиве, который используется __std_type_info_name, чтобы после передать название класса в метод для генерации хеша по имени класса для дальнейшего сопоставления, и если хеш совпадает метод возвращает адрес структуры.

```
3DF                     db    0
3E0 aAvplayerregist db '.?AVPlayerRegistry@Javelin@@',0
3FD                     db    0
3FE                     db    0
```

```
00000145EA312E                  db    0
00000145EA312F                  db    0
00000145EA3130 aAvspelldataman db '.?AVSpellDataManager@Javelin@@',0
00000145EA314F                  db    0
```

Пример:

В коде это выглядит так:

```
array = qword_14638EF90;
classNameHash = 0xCBF29CE484222325ui64; // default
classNameLenght = -1i64;
 do
   ++classNameLenght;
 while ( className[classNameLenght] );
 for ( ; classNameLenght; --classNameLenght ) {
   char = *className++;
   classNameHash = 0x100000001B3i64 * (char ^ classNameHash);
 }
 _classNameHash = classNameHash;
```

__std_type_info_name

Перепишем на Python для повышения читабельности:

```
def generateHashByClassName(className):
      classNameLenght = len(className) - 1
      classNameHash = 0xCBF29CE484222325
      for idx in range(classNameLenght):
            classNameHash = (0x100000001B3 * (ord(className[idx]) ^
classNameHash)) & 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

      return ((pair128(classNameHash, classNameHash) *
0xDE5FB9D2630458E9) >> 64) & 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

def pair128(high, low):
       return (high << 64 | low)
```

Для примера генерируем хеш для класса PlayerRegistry:

```
hex(generateHashByClassName("PlayerRegistry"))
>>> 0x668f0ec14ac79076fc6d4f6cf58bf59c
```

Если углубляться в код, то игра вычисляет адрес дальше по коду таким образом:

Где array->qword60 является указателем на массив адресов всех существующих классов. А array->qword58 ограничителем индекса

```
def getIndexByHash(a1):
  return (__PAIR128__(a1, a1) * 0xDE5FB9D2630458E9ui64) >> 64

v6 = (array->qword60[0x10 * (getIndexByHash(*p_classNameHash) &
array->qword58)]);
```