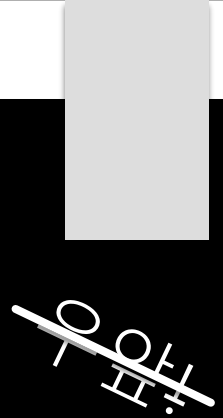


UAF

UseAfterFree

2019/01/07

TEAMSCP

정재훈

Agenda

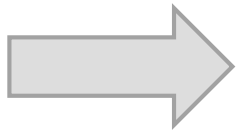
- UAF Example code
- new/delete malloc/free
- virtual function
- pwnable.kr uaf

Example Code

```
printName(test* t)

printf("%s\n",t->name);
```

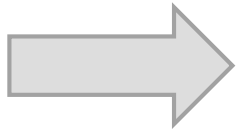
```
shell(void)

printf("This is Shell\n");
```

```
struct test

char name[10];
void (*print)(void*);
```



*t1

```
struct string

char name[128];
```



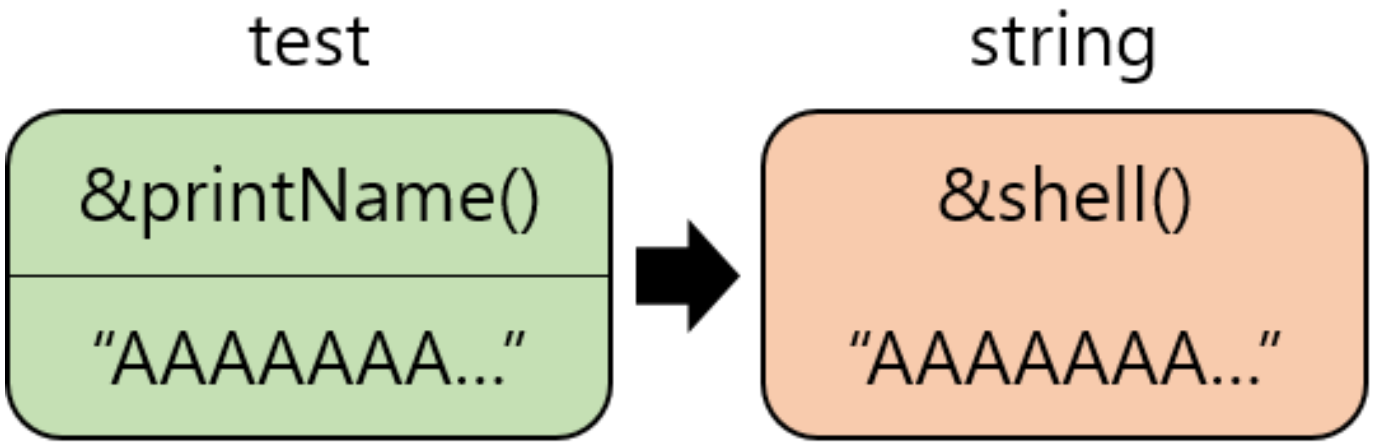
*s1

Example Code

```
int main(void)
{
    test* t1;
    string* s1;

    t1 = malloc(256);

    strcpy(t1->name, "DOG");
    t1->print = (void*)printName;

    t1->print(t1);

    free(t1);

    s1 = malloc(256);

    scanf("%128s",s1->name);

    t1->print(t1);
    return 0;
}
```

Example Code

```
    s1 = malloc(256);

    scanf("%128s",s1->name);

    t1->print(t1);
    return 0;
}
```



Difference

C

- struct

- malloc()
- free()

C++

- class

- new
- delete()

malloc/free new/delete

공통점

- malloc과 new는 힙의 일부 메모리 주소를 지정하는 포인터를 반환함. 즉, 메모리를 할당함

차이점

- new/delet는 생성자, 소멸자 및 연산자 오버로드를 통해 임의의 다른 작업을 수행함
- malloc/free는 메모리만 할당하고 해제

...

new와 malloc의 주요 차이점은 new가 객체의 생성자를 호출하고 delete에 해당하는 호출이 객체의 소멸자를 호출한다는 것입니다.
다른 차이점은 다음과 같습니다.
•new는 type-safe이고, malloc은 void*형의 객체를 반환합니다.
•new는 에러시 예외를 던집니다. malloc은 NULL반환하고 errno를 설정합니다.
•new는 연산자이며 오버로드 될 수 있습니다. malloc은 함수이며 오버로드 할 수 없습니다.
•배열을 할당하는 new[]는 malloc보다 직관적이며 유형 안전합니다.
•malloc파생 된 할당은 realloc통해 크기를 조정할 수 있으며 new파생 된 할당은 크기를 조정할 수 없습니다.
•malloc은 N 바이트의 메모리 덩어리를 할당 할 수 있습니다. new것은 char형의 배열을 할당해야합니다.
차이점을 살펴보면, malloc은 C-esque이고, 새로운 것은 C ++ - esque입니다. 코드 기반에 맞는 것으로 사용하십시오.
새로운 메모리 할당 알고리즘을 사용하여 new와 malloc을 구현하는 것은 합법적이지만, 대부분의 시스템에서 new는 내부적으로 malloc을 사용하여 구현되며 시스템 수준의 차이는 없습니다.

...

new 와 delete 는 클래스의 새로운 인스턴스를 선언하거나 delete 하는 C ++ *프리미티브* 이다 (따라서 인스턴스의 클래스의 소멸자를 호출한다).
malloc 과 free 는 C *함수* 이며 메모리 블록을 할당하고 해제합니다 (크기로).
두 가지 모두 힙을 사용하여 할당합니다. 그럼에도 불구하고 malloc 과 free 는 아마도 포인터와 연관된 메모리 공간의 덩어리를 예약하기 때문에 더 많은 "로우 레벨"이됩니다. C 배열을 구조체로 간주하지 않는 한 해당 메모리 주위에 구조체가 생성되지 않습니다.

•new는 연산자이고 malloc ()은 함수입니다.
•new는 정확한 데이터 타입을 반환하고, malloc ()은 void * (void 타입의 포인터)를 반환합니다.
•malloc ()은 메모리가 초기화되지 않았고 기본값은 가비지 인 반면, new의 경우 메모리가 기본값으로 초기화됩니다. int의 경우 '0'과 같습니다.
•delete와 free () 둘 다 'NULL'포인터로 사용될 수 있습니다.

Virtual fuction

```
class First

MyFunc()
"1"
```

```
class Second : public First

MyFunc()
"2"
```

```
class Third : public Second

MyFunc()
"3"
```

```
Third*   third  =  new Third;
Second* second=  third;
First*   first  =  second;
```

```
first   -> MyFunc();
second  -> MyFunc();
third   -> MyFunc();
```

Result

1
2
3

Virtual fuction

```
class First

virtual MyFunc()
"1"
```

```
class Second : public First

virtual MyFunc()
"2"
```

```
class Third : public Second

virtual MyFunc()
"3"
```

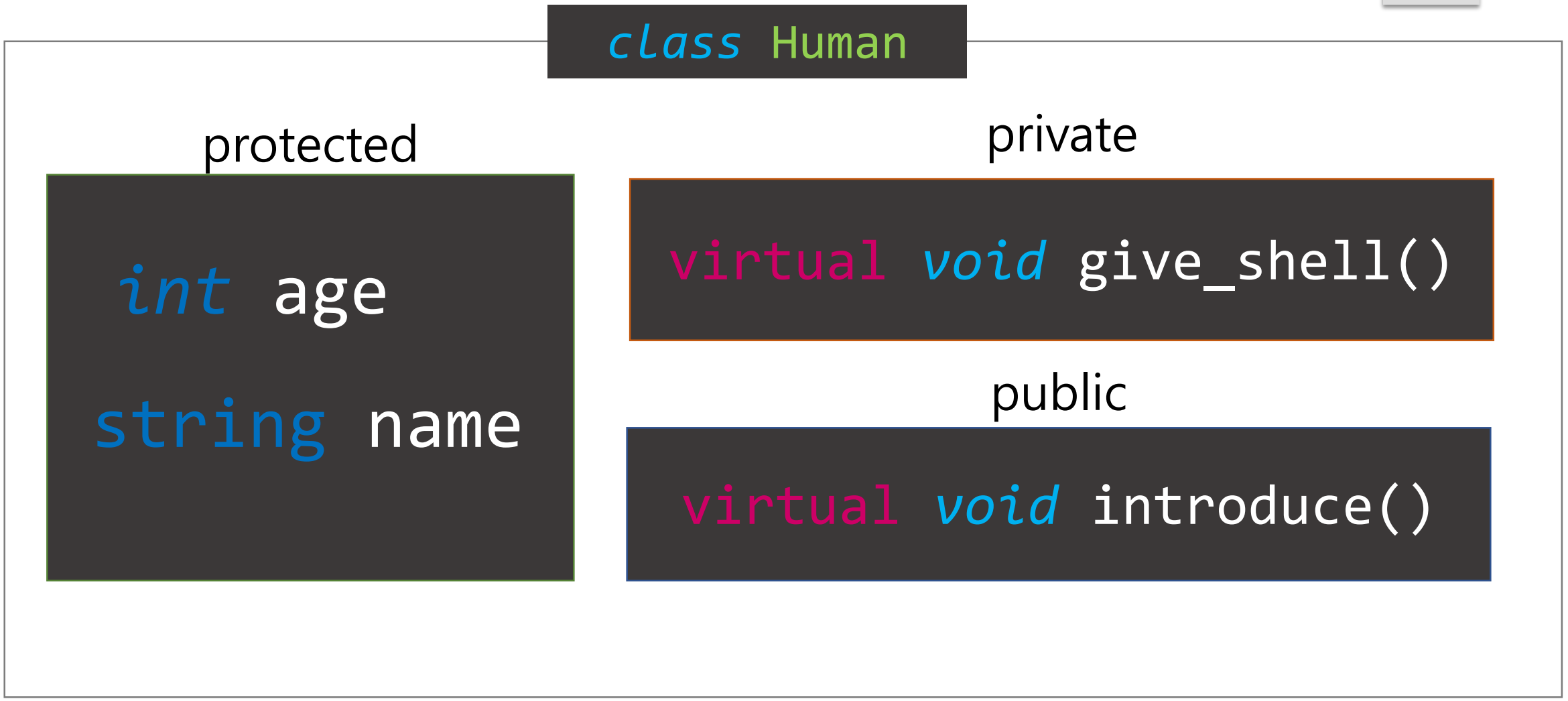
```
Third*   third  =  new Third;
Second* second=  third;
First*   first  =  second;
```

```
first  -> MyFunc();
second -> MyFunc();
third  -> MyFunc();
```

Result

3
3
3

Pwnable.kr uaf


class Human
protected
int age
string name
private
virtual void give_shell()
public
virtual void introduce()

```
class Man : public Human
class Woman : public Human
```

Man 클래스에서
오버라이딩한 함수 주소
=
Woman 클래스에서
오버라이딩한 함수 주소

```
virtual void give_shell()
```

```
virtual void introduce()
```