Buffer overflow 달고나문서

1. 8086 Memory Architecture

8086의 기본적인 메모리 구조는 커널과 가용 메모리로 이루어져 있다. 하나의 프로세스를 실행시키면 이는 segment라는 단위로 묶여서 가용 메모리안에 할당이 된다. 이 segment는 code segment, data segment, stack segment로 구성되어 있고 최대 2^32byte만큼의 크기를 가진다. Code segment에는 명령어들이 있고 명령을 수행하다 보면 다른 주소로 갔다가 되돌아와야 하는 경우도 생긴다. 따라서 정확한 주소가 필요한데, segment는 logical address라는 것을 사용한다. Segment는 segment selector로 시작 위치(offset)을 찾을 수 있고, 시작 위치로부터의 위치에 있는 명령어를 수행할 수 있게 된다. 따라서 실제 메모리 주소(physical address)는 offset + segment시작주소로 구성된다. Data segment에는 프로그램 실행 시 필요한 전역 변수들이 들어간다. Stack segment에는 handler, task, program이 저장하는 데이터 영역으로 버퍼들이 저장되는 공간이다. 또 스택들간 switch가 가능하고 지역 변수들이 자리잡는 공간 이기도 하다. 스택은 필요한 크기만큼 만들어지고 명령에 의해 저장해 나가는 과정을 거치는데 stack pointer라고 하는 레지스터가 스택 맨 위쪽에서 POP과 PUSH를 수행한다.

2. 8086 CPU Register

CPU가 빨리 읽고 쓰기 위해서는 CPU내에 존재하는 메모리를 사용해야 하는데 이 메모리를 Register라고 한다. 레지스터는 각각 목적이 있는데 범용 레지스터는 연산에 사용되는 피연산자, 메모리 포인터가 저장되는 공간이고 세그먼트 레지스터는 code, data, stack segment의 주소가 저장된 공간이고 플래그 레지스터는 플래그들이, 명령 포인터는 명령어가 있는 메모리상의 주소가 들어가 있는 공간이다.

범용 레지스터는 4개의 32bit변수라고 생각하면 되는데 EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP가 있고 EAX~EBX는 다시 상위 부분과 하위 부분 레지스터로 나뉜다.

세그먼트 레지스터는 특정 세그먼트를 가리키는 포인터 역할을 하며 데이터, 명령어들을 정확히 찾을 수 있게 해준다.

플래그 레지스터에는 상태 플래그, 컨트롤 플래그, 시스템 플래그가 있고 초기화 값으로 0x00000002를 갖는다.

상태 플래그에는 연산을 수행할 때 carry나 borrow가 발생하면 1이되는 carry flag, 연산 후 최하위 바이트가 1이 될 때 1인 parity flag, 결과가 0임을 알리는 zero flag, 음수 일 때 알리는 sign flag, 너무 큰 양수나 작은 음수여서 데이터 타입에서 넘침을 알리는 overflow flag등이 있다.

시스템 플래그에는 IF, TF, IOPL, NT 등이 있고, 명령어 포인터 레지스터는 현재 code segment의 offset값을 가지며 JMP, Jcc, CALL, RET와 IRET instrucion의 주소 값을 가진다. EIP 레지스터를 읽을 수 있는 방법을 procedure stack으로부터 리턴하는 instruction의 주소를 읽는 것이다.

3. 프로그램 구동 시 segment변화

코드를 gdb로 disassemble하게 되면 주소<메인 함수로부터의 거리>명령어의 형태로 나오게 된다. 프로그램이 시작되면 EIP 레지스터는 메인 함수가 시작되는 코드를 가리킨다. 이 때 ESP 또한 스택의 맨 꼭대기를 가리키고 있는데 PUSH와 POP을 하기 위한 것이다. EBP의 함수의 데이터를 보존하기 위해 EBP를 저장하는데 이것을 base pointer라고 부른다. 함수가 시작될 때 stack pointer와 base pointer를 재지정하는 것을 함수 프롤로그라고 한다.