Lazenca.net Technote 요약

윤준혁

- The basics technic of Shellcode

shellcode는 machine code로 작은 크기의 프로그램이고 일반적으로 어셈블리어로 작성 후 기계어로 변경한다. 따라서 어셈블리어에 대한 지식이 필요하며 고급언어에 대한 지식 또한 필요하다.

자주쓰는 asm 명령어 : POP, PUSH, mov, add, sub…. etc.

Assembly code에서 시스템 함수를 호출하기 위해 int 0x80, syscall 명령어를 사용한다. int명령어의 피연산자 값으로 0x80을 전달하면 EAX에 저장된 시스템 함수를 호출하고 syscall 명령어를 호출하면 RAX에 저장된 시스템 함수를 호출한다.

C에서는 편의성과 호환성을 위해 표준 라이브러리가 제공되므로 여러 시스템에서 컴파일이 가능하지만 어셈블리 언어는 호환성 있는 표준 라이브러리가 없으며, 커널 시스템 콜을 직접 호출 할 수 있다.

(어셈블리로 시스템 함수를 호출할 때는 시스템 콜 번호를 이용) 시스템 콜 정보는 운영체제, 프로세서의 아키텍쳐마다 다르다. 시스템 콜 번호는 EAX, RAX에 저장한다.

어셈블리 코드의 앞부분에는 메모리 세그먼트를 선언하고 텍스트 세그먼트에 실제 어셈블리 명령이 있다. 따라서 ELF 바이너리를 생성하려면 링커에게 어셈블리 명령이 어디서부터 시작하는지 알려주는 global_start줄이 필요하다.

call, ret명령어를 사용해 메모리 세그먼트를 사용하지 않고 완전히 위치 독립적인 방법으로 코드를 생성 할 수 있다.

➔ call 명령어에 의해 함수가 호출될 때 call 명령어 다음 명령어의 주소를 스택에 저장

➔ 함수의 사용이 끝난 후에 ret 명령어를 이용해 스택에 저장된 주소를 EIP 레지스터에 저장

➔ call 명령어 뒤에 출력할 메시지를 저장하면 pop 명령어를 이용해 ecx 레지스터에 주소값을 전달 할 수 있다.

shellcode에 포함된 null byte(0x00) 때문에 shellcode의 내용이 code 영역에 복사되지 않아 문제가 발생하므로 shellcode에 포함된 null byte를 제거해야한다. 다음 코드를 통해 제거해야 할 null byte를 확인 할 수 있다.

0xE80F000000 : call dword 0x14

0xB804000000 : mov eax, 4

0xBB01000000 : mov ebx, 1

0xBA0F000000 : mov edx, 15

0xB801000000 : mov eax,1

0xBB00000000 : mov ebx,0

call 명령어에 첫번째 null byte가 존재하므로 jmp 명령어를 사용해 helloworld 함수를 지나 last함수로 이동한 뒤 helloworld 함수를 호출하고, 출력할 메시지를 저장하는 방법으로 제거한다.

레지스터에 값을 저장하는 부분에도 null byte가 존재하므로 xor명령어를 이용하여 초기화 해주는 것이 좋다. (XOR 명령어가 sub 명령어보다 flag에 영향을 덜 주기 때문에 XOR 을 이용해 레지스터 값을 초기화 하는 것이 효율적이다.)

- Return to Shellcode

Review

call 명령어는 return address (call 명령어 다음 명령어의 위치)를 stack에 저장하고, 피연산자 주소로 이동한다.

ret 명령어는 pop 명령어를 이용해 RSP레지스터가 가리키는 stack 영역에 저장된 값을 RIP(EIP)에 저장 후, 해당 주소로 이동한다.

permissions in memory 에는 read(r) – 메모리 영역의 값을 읽을 수 있다, write(w) – 메모리 영역에 값을 저장할 수 있다, excute(x) – 메모리 영역에서 코드를 실행 할 수 있다 가 있다.

GCC는 기본적으로 DEP가 적용되기 때문에, 코드가 저장된 영역에만 실행권한이 설정되며, 데이터가 저장되는 영역에는 실행권한이 설정되지 않는다. 즉 shellcode를 실행하기 위해 shellcode가 저장된 영역은 execute 권한이 설정되어 있어야 한다.

DEP를 해제하기 위해 GCC옵션으로 "-z execstack"을 추가한다.

메모리 영역에서 코드가 실행되는 것을 차단하기 위해 NX Bit(DEP)를 적용 할 수 있다.