Lazenca - Technote

Return To Shellcode

- Return address 영역을 Shellcode가 저장된 주소로 변경하여, Shellcode를 호출하는 것
- CALL : 다음 명령어의 주소 값을 Stack에 저장하고, Operation의 주소로 이동
- RET : POP을 이용해 RSP가 가리키는 Stack 영역에 저장된 값을 RIP에 저장 후, 그 주소로 이동
- 따라서, Return Address의 값을 바꾸면 프로그램의 실행을 바꿀 수 있다!

Permissions in memory

- 메모리 권한
  - read(r) : 메모리 영역의 값을 읽을 수 있음.
  - write(w) : 메모리 영역에 값을 저장 할 수 있음.
  - excute(x) : 메모리 영역에서 코드를 실행 할 수 있음.
- GCC는 기본적으로 DEP가 적용되어 코드가 저장된 영역에만 실행권한이 설정되며, 데이터가 저장되는 영역에는 실행권한이 설정되지 않음
- 따라서 Shellcode를 실행시키기 위해서는 Shellcode가 저장된 영역에 execute 권한이 설정되어 있어야 함

NX Bit(MS : DEP)

- Never eXecute bit, 실행 방지 비트
- 프로세스 명령어나 코드 또는 데이터 저장을 위한 메모리 영역을 따로 분리하는 CPU의 기술
- NX 특성으로 지정된 모든 메모리 구역은 데이터 저장을 위해서만 사용되며, 프로세서 명령어가 그 곳에 상주하지 않음으로써 실행되지 않도록 만듦
- DEP : 마이크로소프트 윈도우 운영 체제에 포함된 보안 기능
  - 악의적인 코드가 실행되는 것을 방지하기 위해 메모리를 추가로 확인하는 하드웨어 및 소프트웨어 기술

- 하드웨어 DEP: 메모리에 명시적으로 실행 코드가 포함되어 있는 경우를 제외하고 프로세스의 모든 메모리 위치에서 실행할 수 없도록 표시
- 소프트웨어 DEP: CPU가 하드웨어 DEP를 지원하지 않을 경우 사용

Binary

```
# check for NX support
if readelf -W -l $1 2>/dev/null | grep 'GNU_STACK' | grep -q 'RWE'; then
  echo -n -e '\033[31mNX disabled\033[m   '
else
  echo -n -e '\033[32mNX enabled \033[m   '
fi
```

- readelf 명령어를 이용해 파일의 세그먼트 헤더 정보에서 NX 여부를 확인
- 파일의 세그먼트 헤더 정보에서 'GNU_STACK'의 Flg 값이 'RWE'이라면 NX가 활성화되었다고 판단함
  - NX가 적용된 바이너리의 Flg 값은 'RW'
  - NX가 적용되지 않은 바이너리의 Flg 값은 'RWE'

Process

```
# fallback check for NX support
elif readelf -W -l $1/exe 2>/dev/null | grep 'GNU_STACK' | grep -q 'RWE'; then
  echo -n -e '\033[31mNX disabled\033[m   '
else
  echo -n -e '\033[32mNX enabled \033[m   '
fi
```

- Binary의 확인 방식과 동일하며, 전달되는 파일의 경로가 다름
  - Ex) /proc/<PID>/exe

CPU

```
# check cpu nx flag
nxcheck() {
```

```
  if grep -q nx /proc/cpuinfo; then
    echo -n -e '\033[32mYes\033[m\n\n'
  else
    echo -n -e '\033[31mNo\033[m\n\n'
  fi
}
```

- "/proc/cpuinfo" 파일에서 'nx' 문자가 있는지 확인함