시스템 프로그래밍  
attack lab

수업명 | 시스템프로그래밍

지도교수 | 박문주

작성날짜 | 2019.11.20

학번 | 201701524

이름 | 강은선

[1. Code Injection Attack 3](#_Toc25179993)

[(1) Level1 3](#_Toc25179994)

[(2) Level2 3](#_Toc25179995)

[(3) Level3 4](#_Toc25179996)

[2. ROP 4](#_Toc25179997)

[3. 첨부 4](#_Toc25179998)

## Code Injection Attack

Code Injection Attack은 target81폴더의 ctarget의 어셈블리어 코드와 레지스터의 값을 이용하여 푸는 문제로 스택 오버플로우를 발생시켜 해결한다. 해당 어택은 Level1부터 Level3까지 총 3문제로 이루어져 있다.

우선, <첨부1>의 어셈블리어 첫번째 줄 sub $0x18, %rsp를 보면, 스택에서 0x18만큼의 공간을 설정하는 것을 볼 수 있다. 이때, getbuf는 스택을 버퍼로 gets를 호출하는 함수이므로 사용자가 입력을 줄 때, 0x18바이트보다 크게 주면 리턴 어드레스를 원하는 값으로 덮어 씌워, 그 값에 해당하는 주소로 점프할 수 있다. 따라서, Level1에서 Level3까지의 과제를 수행할 때, 공통적으로 0x18크기의 패딩을 주도록 할 것이다. 이때, 패딩은 아무 값으로 넣어도 상관은 없지만 임의로 0으로 지정하여 입력한다. 또한, 컴퓨터가 리틀엔디안이므로 이를 고려하여 답안을 작성한다.

입력을 표준입력으로 줘야 하므로 제공되는 hex2raw 프로그램을 사용하여 txt파일을 raw파일로 변환하여 사용하며, “./hex2raw < (solution.txt) > (solution.raw)”형식을 이용하여 변환한 뒤, raw파일을 인자로 넘겨 실행한다.

### Level 1

우선 touch1에 접근하기 위해 앞에 0x18크기의 패딩을 주고 리턴 어드레스로 touch1의 시작주소인 0x401932(<첨부2> 참조)를 넣는다.

위의 과정으로 작성한 정답 텍스트 파일은 <첨부3>과 같다.

### Level 2

<첨부4>의 <touch2+16>을 보면 <cookie>와 입력 받은 값을 비교하여 같을 때만 다음 코드로 넘어가는 것을 알 수 있다. cookie의 값은 target81폴더의 cookie.txt파일에서 확인할 수 있으며 그 값은 0x6ea28ad7이다. 해당 값을 인자로 전달하려면 rdi/edi레지스터에 값을 넣어야 하는데, 이를 위해, touch2로 가기 전에 rdi/edi레지스터에 쿠키 값을 넣고 리턴하는 코드가 실행되어야 한다. 이때, 스택의 아랫단을 건드리지 않도록 push 명령어를 사용하여 스택에 touch2의 시작주소를 전달하는 코드가 실행되어야 하므로, 정답 파일에 다음과 같은 순서로 들어가야 한다.  
① rdi/edi레지스터에 쿠키를 넣고 touch2의 시작주소를 스택에 넣은 뒤, 리턴하는 코드  
② 앞의 코드를 포함한 버퍼 크기만큼의 패딩  
③ 버퍼의 코드 주소 (첫 번째 리턴 어드레스)

이때, edi레지스터에 값을 대입하는 기계어 코드를 넣어주어야 하므로 해당 코드를 어셈블리어로 작성한 후 gcc로 어셈블하여 이용한다. <첨부5>가 앞서 얘기한 edi레지스터에 쿠키 값을 대입한 후 리턴하는 코드이고, 버퍼의 주소는 <첨부1>에서의 rsp레지스터 값(=0x55622be0)에 0x18을 뺀 값이므로 0x55622bc8이다.

따라서, 정답 파일은 <첨부6>으로 기계어코드(mov+pushq+ret) + 패딩 + 버퍼 주소로 이루어진 text파일이다.

### Level 3

touch3는 문자열을 입력으로 받아 cookie와 같은지 비교하므로 쿠키 값(문자형태)을 <첨부7>의 아스키 코드표를 참조하여 16진수 형태로 바꾸어 주어야 한다. 바꾼 쿠키 문자열의 16진수 값은 26 65 61 32 38 61 64 37이다.

Level3에서는 touch3의 인자로 문자열의 주소를 주어야 하는데, 위에서 구한 rsp레지스터의 값, 즉, 0x55622be0에 첫 번째 리턴 어드레스가 오므로 +0x8 자리에 쿠키 문자열을 넣어주면 된다. 그러므로 level3.s 파일을 생성하여 0x55622be0에 0x08을 더한 0x55622be8을 edi레지스터에 대입하는 코드를 작성한다.

또한, 이번에도 Level2와 같이 touch3의 시작주소를 pushq명령어를 이용하여 넘겨준 뒤, 리턴한다. 따라서 level3.s를 어셈블한 level3.o 코드는 <첨부8>과 같이 나온다.

정답 파일에는 기계어 코드를 포함한 패딩, 코드 주소, 쿠키 문자열 순으로 작성한다. 정답은 <첨부9>와 같다.

## ROP

ROP에서는 Code Injection Attack에서와 달리 버퍼에 직접 코드를 삽입하는 것이 아닌 이미 존재하는 gadget을 이용하여 원하는 명령들의 기계어 코드를 조합하는 방법으로 문제를 해결한다. 이때, Level2에서와 마찬가지로 0x18크기의 패딩을 주도록 할 것이다. 역시, 패딩은 아무 값으로 넣어도 상관은 없지만 임의로 0으로 지정하여 입력한다. 또한, 컴퓨터가 리틀엔디안이므로 이를 고려하여 답안을 작성한다.

입력을 표준입력으로 줘야 하므로 제공되는 hex2raw 프로그램을 사용하여 txt파일을 raw파일로 변환하여 사용하며, “./hex2raw < (solution.txt) > (solution.raw)”형식을 이용하여 변환한 뒤, raw파일을 인자로 넘겨 실행한다.

우선, 이번 어택에서 사용할 start\_farm과 end\_farm 사이의 gadget들을 중복을 무시하고 모두 적으면 다음과 같다. (ret과 al, cl, dl, bl레지스터 명령어는 기계어 코드만 작성하고 어셈블리어로는 작성하지 않는다.)

① 0x401b16: 58 90 c3 popq %rax

② 0x401b1b: 48 89 c7 90 c3 movq %rax, %rdi

③ 0x401b25: 58 c3 popq %rax

④ 0x401b48: 48 8d 04 37 c3 lea (%rdi, %rsi, 1), %rax =><add\_xy>

⑤ 0x401b4f: 48 89 e0 90 c3 movq %rsp, %rax

⑥ 0x401b72: 89 c1 38 d2 c3 movl %eax, %ecx

⑦ 0x401ba1: 89 d6 20 db c3 movl %edx, %esi

⑧ 0x401bc1: 89 ca 08 c9 c3 movl %ecx, %edx

⑨ 0x401bf4: 89 c1 08 db c3 movl %eax, %ecx

⑩ 0x401c0e: 89 ca 20 db c3 movl %ecx, %edx

이때, ③과 ④ 사이에 mid\_farm이 존재한다.

### Level 2

Level2에서 Gadget들은 rtarget의 start\_farm과 mid\_farm사이의 함수들에 존재하며 2개의 gadget만 사용하여 attack이 가능하다. 이때, 참고할 명령들은 movq, popq, ret, nop으로 ret은 0xc3, nop은 0x90이고, 나머지 movq, popq는 인자가 되는 레지스터 별 기계어 코드가 다르니 <첨부10>과 <첨부11>을 참고한다.

우선, touch2를 해결하기 위해서는 쿠키 값을 인자로 전달해야 하므로 edi레지스터에 쿠키 값을 넣어주어야 한다. 이때, 원하는 값을 대입하기 위해서는 스택에 쿠키 값을 넣고 스택에서 popq로 레지스터에 옮겨주는 방법으로 풀 수 있다. 그러기 위해 gadget①과 ③ 중 한가지와 gadget②를 사용해야 한다. 또한, popq과 movq를 수행하는 중간에 쿠키 값을 넣어 popq가 수행될 때, 해당 값이 rax레지스터로 들어가게 해야 하므로, 정답 파일에 들어갈 순서는 패딩, popq %rax, ret의 주소, 쿠키 값, movq %rax, %rdi의 주소, touch2함수의 주소 순이다. popq 명령어의 기계어 코드로 gadget③을 사용한 답안이 <첨부12>와 같다.

### Level 3

Level 3에서는 start\_farm과 end\_farm사이의 8개의 gadget을 이용하여 풀이를 진행한다. 우선, 위에서 찾은 명령어를 중복을 제외하여 정리하면 아래의 총 7개의 명령어가 나온다.

① popq %rax

② lea (%rdi, %rsi, 1), %rax =>add\_xy

③ movq %rsp, %rax

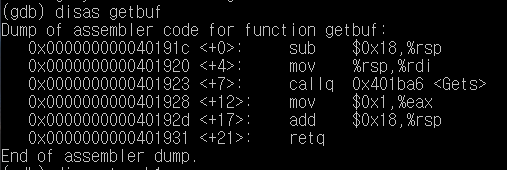
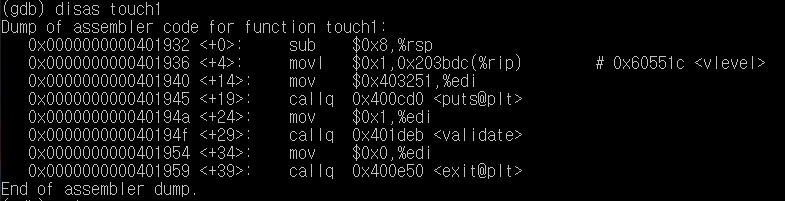
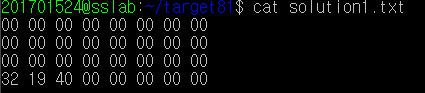
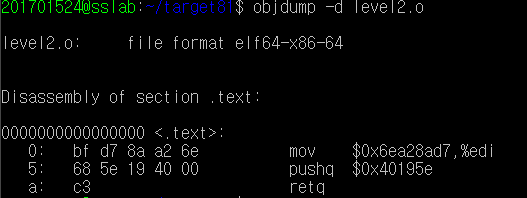
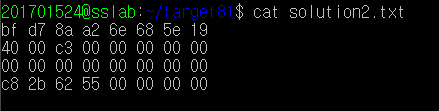
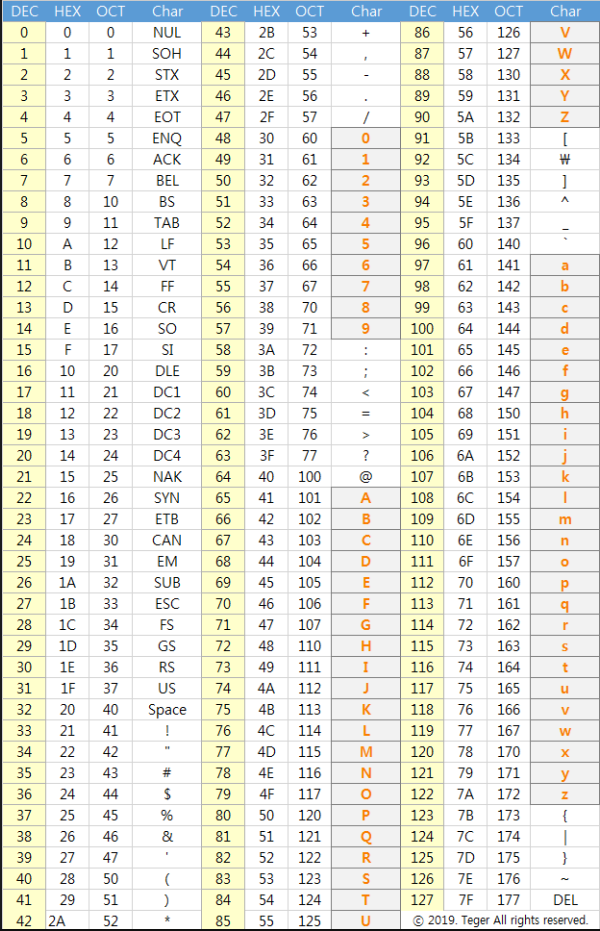
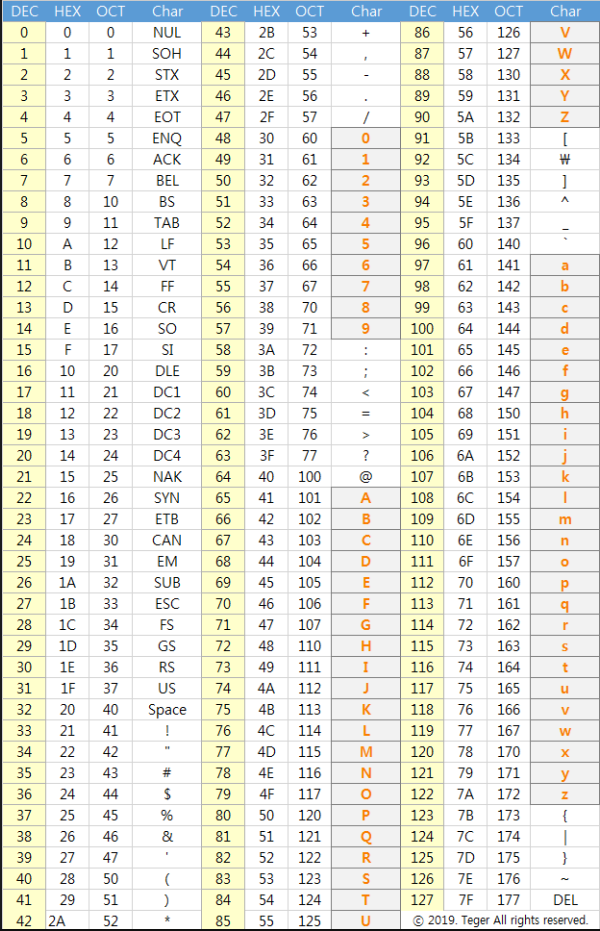
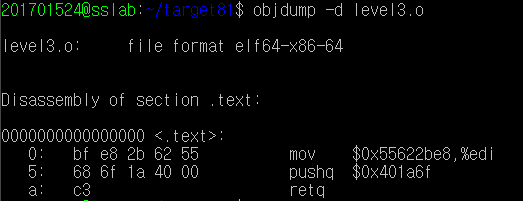
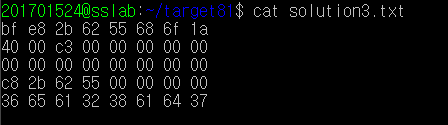
④ movq %rax, %rdi

⑤ movl %eax, %ecx

⑥ movl %ecx, %edx

⑦ movl %edx, %esi

## 첨부

1. ctarget의 getbuf 함수 어셈블리어  
   
2. ctarget의 touch1함수 어셈블리어  
   
3. Code Injection Attack Level1의 정답 txt  
   
4. ctarget의 touch2함수 어셈블리어  
   
5. cookie값을 edi에 대입하고 touch2의 시작주소를 스택에 넣은 뒤, 리턴하는 코드  
   
6. Code Injection Attack Level2 정답 txt  
   
7. 아스키 코드표(16진수에 사용되는 부분만 발췌)
8. 쿠키 문자열의 주소를 edi레지스터에 저장하고 touch3의 시작 주소를 스택에 넣은 뒤, 리턴하는 코드  
   
9. Code Injection Attack Level3 정답 txt  
   
10. movq 명령어의 인자 레지스터 별 기계어 코드  
    
11. popq 명령어의 인자 레지스터 별 기계어 코드  
    
12. ROP Level2 정답  
    