연습문제

01 다음 중 리버스 엔지니어링이 적용되는 부분이 아닌 것은? (3)

1 이미 만들어진 프로그램의 동작 원리를 이해하고, 유사한 프로그램을 단들기 위해 리버스 엔

지니어링을 이용한다.

2 프로그램의 보안상 문제, 동작 문제, 오류 등을 이용하거나 제품 출시 전에 그런 둔제나 오류

동을 제거 · 검토하기 위해 리버스 엔지니어링을 이용한다.

3 성능이 좋은 프로그램을 만들기 위해 리버스 엔지니어링으로 설계하고 프로그래밍한다.

4 리버스 엔지니어링을 이용하여 바이러스(또는 웜)를 만들거나 반신을 만든다.

03 다음 중 이뮤니티 디버거에 존재하지 않는 창은? (4)

1 기계어 코드와 어셈블리어 코드

2 레지스터와 플래그 창

3 HEX 덤프 창

4변경 내역 창

04 이뮤니티 디버거에서 F9로 프로그램을 실행시키면 ( )에서 프로그램이 멈춘다.

( )는 운영체제에서 프로그램으로 처음 진입하는 지점인데, 일반적으로는 프로그램의 main 함수와 의미가 같지만, 실행하는 프로그램의 종류 및 환경에 따라 라이브러리의 한 지점일 수도 있다.

(Module Entry Point)

05 반복 작업을 수행하는 함수를 디버깅할 때 카운터로 사용되는 레지스터는? (3)

1 EAX

2 EBX

3 ECX

4 EDX

06 디버거를 이용하여 (레지스터값)이나 (어셈블리어)를 임의로 변경해서 프로그램이 수행되는

흐름을 바꾸어 수행할 수도 있다.

08 원본 프로그램을 새로운 PE 파일에 패킹된 형태로 압축하고 암호화하여 저장하는 기술은? (3)

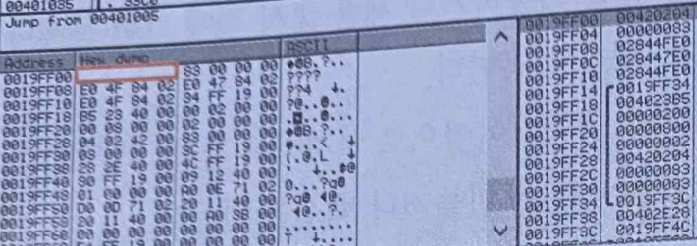
1 안티 디버거

2 타이밍 체크

3 패킹

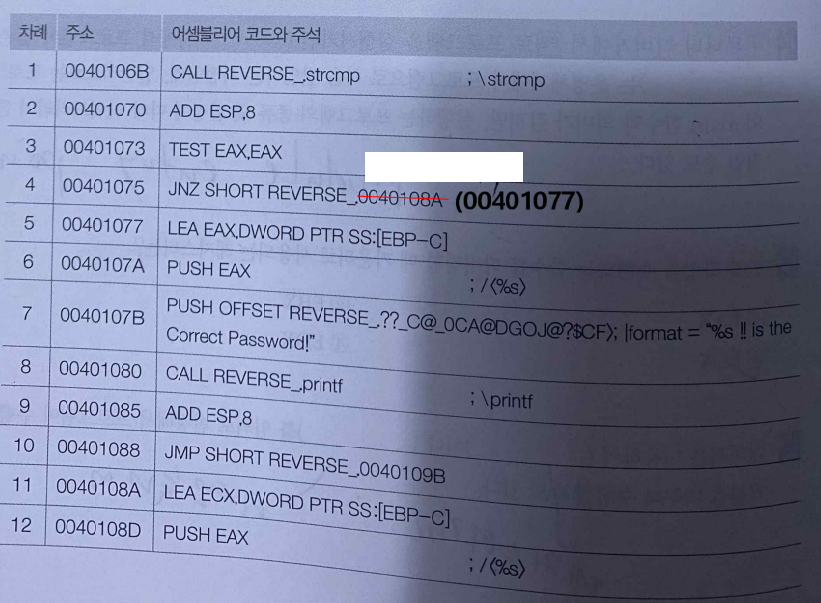
4 코드 치환

07 다음은 프로그램을 디버거에서 수행하는 중간 단계의 화면이다. 파일 덤프 장에서 흰색 사각형 박스에 들어갈 값은? 단 파일 덤프 창은 리틀 엔디안 방식이다.



04 02 42 00

09 다음과 같이 실행되는 프로그램에서 무조건 7번줄을 실행한 후 프로그램을 종료하도록 어셈블리어 코드 한 줄을 고치시오.



10 다음 명령이 실행되면 스택에는 어떤 변화가 생기는가?

SUB ESP, 40

PUSH EBX

PUSH ESI

PUSH EDI

(ESP에서 64를 빼는 것으로 스택에 0x40(64) 바이트의 공간을 할당하고, 스택에 EBX, ESI, EDI 값을 저장한다.)

01 다음 증 a.txt와 link 파일을 하드 링크하는 올바른 명령은? (4)

1 In -hard a.txt link 2 In -s a.txt link

3 hardlink a,txt link 4 In a.txt link

02 원본 파일을 삭제할 때, 그 원래의 내용을 확인할 수 없는 링크는? (4)

1 하드 링크 2 디바이스 링크

3 데몬 링크 4 심볼릭 링크

03 특정 프로세스를 백그라운드로 실행하는 명령과 포그라운드로 실행하는 명령은?

프로세스를 백그라운드로 실행하는 명령 : "프로세스명 &"

백그라운드로 동작중인 프로세스를 포그라운드로 전환시키는 명령 : fg

04 프로그램이 생성하는 임시 파일을 사용하여 root 수준의 권한이 필요한 파일에 대한 레이스 컨디션 공격을 수행할 때, 공격에 필요한 세 가지 조건이 아닌 것은? (4)

1 공격 대상 프로그램의 소유자가 root일 것

2 공격 대상 프로그램이 SetUID 비트를 가진 것

3 생성되는 임시 파일의 이름을 알고 있어야 할 것

4 생성되는 임시 파일이 SetUID 비트를 가질 것

05 다음 임시 파일에 대한 레이스 컨디션 공격을 할 때 수행되는 항목들을 순서에 맞게 번호를 매기시오. (1-2-3-4-5-6)

1 정상 프로세스 - 임시 파일 존재 여부 확인

5 공격 프로세스 - 임시 파일을 심볼릭 링크로 생성

2 정상 프로세스 - 임시 파일이 이미 있다면 삭제하고 재생성

4 공격 프로세스 - 심볼릭 링크가 아닐 경우 임시 파일을 삭제

6 정상 프로세스 - 임시 파일에 접근하고 처리

3 공격 프로세스 - 임시 파일이 존재하는 경우 심볼릭 링크 파일인지 확인

06 레이스 컨디션의 목적이 임시 파일의 데이터일 때, 프로그램이 임시 파일을 생성하는 다음 과정중 '임시 파일에 접근하고 복사'하는 단계는?

(프로그램 실행 -1> 암호화된 파일 접근 -2> 복호화된 임시 파일 생성 -3> 메모리 업로드 후 임시 파일 삭제 -4> 프로세스 처리 -> 프로그램 종료)

07 심볼릭 링크와 하드 링크를 설명하시오.

• 하드 링크 : 두 파일을 In(link)으로 연결시켜주며, 링크 후에는 링크 수가 증가하여 2가 된다. 하드 링크된 파일을 수정하면 원본 파일 역시 수정되나, 원본 파일을 삭제하더라도 링크된 파일은 삭제되지 않는다.

• 심볼릭 링크 : 심볼릭 링크는 In 명령에 '-s'옵션을 이용하며, 링크 수는 변하지 않는다. 또한 원본 파일이 삭제될 경우 심볼릭 파일에서는 아무런 데이터도 얻을 수 없다.

01 다음과 같이 bugfile을 실행할 경우 argc, argy[0], argv[1], argv[2]의 값은?

# ./bugfile 12 /bin/sh

argc : 3 argv[0] : bugfile argv[1] : 12 argv[2] : /bin/sh

02 다음은 gdb에서 사용하는 명령들이다. 각 명령의 동작을 간단히 설명하시오.

• list 3, 7: 프로그램의 3~7행을 출력한다.

• break 5 : 프로그램의 5행에 브레이크 포인트를 설정한다.

• disass main : main 함수의 어셈블리어 코드를 출력한다.

• ×/16xw $esp : esp를 기준으로 16개의 워드(word) 만큼 메모리 내용을 출력한다.

• s(tep) : 한 행씩 실행하는데, 함수를 포함하고 있으면 함수 내부로 이동해 실행한다,

• n(ext) : 한 행씩 실행하는데, 함수를 포함하고 있으면 함수를 완전히 실행하고

다음 행으로 이동한다.

• q(ontinue) : 현재 위치에서 프로그램의 끝(또는 브레이크 포인트)까지 실행한다.

03 다음 중 버퍼 오버플로 취약점이 존재하는 함수가 아닌 것은? (4)

1 strcpy 2 strcat

3 scanf 4 printf

04 다음 중 스택 오버플로 공격을 이용하여 관리자 권한을 얻는 조건이 아닌 것은? (4)

1 취약 파일이 관리자 소유여야 한다.

2 취약 파일 함수 안에 입력 값을 체크하지 않는 함수가 존재하야 한다.

3 취약 파일의 권한에 SetUID가 설정되어 있어야 한다.

4 취약 파일에 실행 권한이 없어야 한다.

05 segmentation fault는 버퍼 오버플로 공격 중 언제 일어나는가?

ret 주소 값을 다른 값이 불완전하게 덮어쓰면 Seqmentation fault가 일어난다

06 해당 프로그램에 취약한 함수가 사용되었는지 찾는 방법으로, 이미 컴파일되어 있는 프로그램

에서 사용한 함수를 확인할 수 있는 명령은? (3)

1 find 2 cmd

3 strings 4 string

07 다음 중 버퍼 오버플로를 막는 데 사용하는 방법이 아닌 것은? (3)

1 Non-Executable Stack 2 ASLR (Address Space Layout Randomization)

3 rtl (return to libc) 4 스택 가드 (Stack Guard)

08 다음 중 Terminator canary로 쓰이는 바이트가 아닌 것은? (1)

1 NOP (No OPeration(0x90)) 2 CR (Carriage Return(0x0d))

3 LF (Line Feed(0x0a)) 4 EOF (End Of File(0xff))

10 스택 버퍼 오버플로를 설명하시오.

입력 값을 확인하지 않는 입력 함수에 정상적인 크기보다 큰 입력 값을 입력하여

ret 값을 덮어쓰기 함으로써 임의의 코드를 실행하기 위한 공격이다.

11 다음 코드를 실행했을 때, 항목을 스택의 높은 주소부터 낮은 주소로 순서대로 배열하시오.(다

스택에 배열할 항목은 다음과 같다) char buffer[10], char \*argv[], sfp, int argc, ret(return address)

#include <stdio.h> // bugfile.c

int main(int argc, char \*argv[]) {

har buffer[10];

strcpy(buffer, argv[1]);

printf("&s\n", &buffer);

}

정답 : char \*argv[], int argc, ret(return address), sfp, char buffer[10]

12 힙은 스택과 반대로 메모리의 ( 하위 ) 주소에서 ( 상위 ) 주소로 영역이 커진다.

13 스택 가드Stack Guard를 설명하시오.

특정 변수를 입력받을 시에 해당 변수 값의 경계에 특정한 값을 입력해두어, 이 값이 변조되거나, 침범 당할 시에 시스템 프로세스의 오류를 발생시켜 버퍼 오버플로우등과 같은 공격으로 인해 권한을 빼앗기는 것을 막기 위한 것이다.

14 Non-Executable Stack을 설명하시오

스택에 존재하는 공격 코드가 실행되지 못하도록 스택에 해당하는 메모리 공간에 대한 실

행(Execution) 권한을 제거한 것이다.

17 ASLR(Address Space Layout Randomization)을 간단히 설명하시오.

메모리 공격을 방어하기 위해 주소 공간 배치를 난수화시키는 기법이다. 즉, 스택, 힙, 라이브러리 등의 데이터 영역 주소 등을 난수화시켜 프로세스의 주소 공간에 배치하여 공격자가 ret 주소를 추적하여 찾아내는 것을 어렵게 한다.

01 다음 중 각각의 포맷 스트링에 대한 설명이 틀린 것은? (2)

1 %d: 정수형 10신수 상수(integer) 2 %s: 문자 값(char)

3 %f: 실수형 상수(float) 4 %hn: %n의 반인 2바이트 단위

02 다음은 test2.c와 이 파일을 컴파일하여 실행한 결과이다. 실행 결과에서 8048440은 무엇을 의미하는가?

#include <stdio.h> // test2.c

main() {

char \*buffer = "wishfree\n%x\n";

printf(buffer);

}

[root@REDHAT62 /FS]# gcc -g -o test2 test2.c

[root@REDHAT62 /FS]# ./test2

wishfree

8048440

[root@REDHAT62/FS]#

(0x8048440은 wishfree 문자열이 저장된 메모리 주소 값을 가리킨다.)

03 다음 코드를 컴파일하여 실행할 때, 'Changed value of i' 값으로 출력되는 값은? ( 4 )

#include <stdio.h> // test4.c

main() {

int d=1234,i=0;

printf("Address of i: %x\n", &i);

printf("Value of i: %x\n", i);

printf("%d%n\n", d, &i);

printf("Changed value of i: %x\n", i); // 4

}

05 다음 중 포맷 스트링에 취약한 함수가 아닌 것은? (3)

1 printf() 2 fprintf()

3 scanf() 4 sprintf()

06 포맷 스트링을 설명하시오.

특정한 종류의 데이터를 표현하기 위해 사용하는 일종의 매개변수

07 포맷 스트링 사용 함수의 올바른 사용 예와 잘못된 사용 예를 설명하고, 잘못된 사용 예에서 일어날 수 있는 문제점을 설명하시오.

printf 함수 안에 인자를 포맷 스트링 문자를 사용하지 않고 포인터 값을 사용할 때 포맷 스트링 취약점이 발생한다.