Security 리버싱

GDB 사용법

2017. 11. 27. 20:35 *by* 이만석

- GDB

GDB(GNU debugger) 는 GNU 소프트웨어 시스템을 위한 표준 debugger 이다.

C, C++, 포트란을 비롯한 수많은 프로그래밍 언어를 디버깅하도록 도와주는 이식성 높은 GNU debugger이다.

- 실행 방법

gdb [프로그램명]

gdb [프로그램명] [core파일명]

gdb [프로그램명] [실행중인 프로세스 pid]

- 종료 방법

q (quit_

Ctrl + d

- 소스 보기

(gdb 실행시 소스 폴더에서 구동해야 가능함)

l	main 함수를 기점으로 소스 출력
l 10	10행을 기준으로 출력
l func	func 함수의 소스를 출력
l -	출력 된 이전 행을 출력

file.c:10 file.c	파일의 10행에 브레이크 포인트 설정

- 브레이크 포인트

(프로그램을 브레이크 포인트에 멈춰서 그 때의 상황을 파악하면서 확인할 때 사용)

b func	func 함수의 시작 부분에 브레이크 포인트 설정
b 10	10행에 브레이크 포인트 설정
b file.c:func	file.c 파일의 func 함수에 브레이크 포인트 설정
b file.c:10 file.c	파일의 10행에 브레이크 포인트 설정
b +2	현재 행에서 2개행 이후 지점에 브레이크 포인트 설정
b -2	현재 행에서 2개행 이전 지점에 브레이크 포인트 설정
b *0x8049000	0x8049000 주소에 브레이크 포인트 설정
b 10 if var == 0	10행에 브레이크 포인트를 설정하되, var 변수 값이 0일 때 작동
tb	임시 중단점을 사용하는 것으로 한번만 설정되며, 그 이후엔 삭제된다.

disable 2	고유번호 2번인 브레이크 포인트 비활성화
enable 2	고유번호 2번인 브레이크 포인트 활성화

- 프로그램 실행, 종료 (run kill)

r	프로그램 수행(재시작)
r arg1 arg2	arg1과 arg2를 인자로 프로그램 실행
k	프로그램 수행 종료

- 역추적 하기

bt	오류가 발생한 함수를 역으로 찾아간다.

- 디버깅 하기

s	현재 출력된 행을 수행하고 멈추지만, 함수의 경우 함수의 내부로

n	현재 행을 수행하고 멈추지만, 함수의 경우 함수를 수행하고 넘어다.
n 5	n을 5번 입력한 것과 동일
c	다음 브레이크 포인트를 만날때 까지 계속 수행한다.
u	for 문에서 빠져나와서 다음 브레이크 포인트까지 수행한다.
finish	현재 함수를 수행하지 않고 빠져나감
return	현재 함수를 수행하지 않고 빠져나감
return 123	현재 함수를 수행하지 않고 빠져나감, 단, 리턴값은 123
si	현재의 익스트럭션을 수행, 함수 호출 시 내부로 들어간다.
ni	현재의 인스트럭션을 수행, 함수 호출 시 내부로 들어가지 않는다.

- 변수 정보 보기

info locals	현재 상태에서 어떤 지역변수들이 있으며, 값은 어떠한지를 알 수 다.
info variables	현재 상태에서의 전역변수 리스트를 확인 할 수 있다.
p lval	lval 값을 확인한다.
p func	func 함수의 주소값을 확인한다.
p pt	pt가 구조체라면 구조체의 주소를 확인한다.
p *pt	*pt가 구조체라면 구조체의 값을 확인한다.
p **pt	**pt가 구조체라면 구조체의 값을 확인한다.
info registers	레지스트 값 전체를 한번에 확인한다.

- 중복된 변수명이 있는 경우 특정 변수를 지정해서 출력

p 'main.c'::var	main.c 파일에 있는 전역변수인 var 변수의 값을 출력
p hello::var	hello 함수에 포함된 static 변수인 var 변수의 값 출력

- 출력명령 요약

p [변수명]	변수 값을 출력
p [함수명]	함수의 주소를 출력

p [함수명]::[변수명]	함수에 있는 변수 값을 출력
p [변수명]@[배열크기]	변수의 내용을 변수 배열의 크기 형태로 출력

- 스택 프레임 관련 명령

frame [N]	n번 스택 프레임으로 변경
up	상위 프레임으로 이동
up [N]	n번 상위 스택 프레임으로 이동
down	하위 프레임으로 이동
down [N]	n번 하위 스택 프레임으로 이동
info frame	현재 스택 프레임 정보를 출력
info args	현재 스택 프레임의 함수가 호출될 때 인자를 출력
info locals	현재 스택 프레임의 함수내의 지역변수를 출력
info catch	현재 스택 프레임의 함수내의 예외 핸들러를 출력

- 메모리 상태 검사

x/[범위] [출력형식] [범위의 단위] : 메모리의 특정 값들을 확인 할 수 있다.

출력형식

x/10 main	main 함수 시작부터 40바이트를 출력한다.
x/10t main	2진수로 출력
x/10o main	8진수로 출력
x/10d main	부호가 있는 10진수로 출력 (int)
x/10u main	부호가 없는 10진수로 출력 (unsigned int)
x/10x main	16진수로 출력
x/10c main	최초 1바이트 값을 문자형으로 출력
x/10f main	부동 소수점 값 형식으로 출력
x/10a main	가장 가까운 심볼의 오프셋을 출력
x/10s main	문자열로 출력
x/10i main	어셈블리 형식으로 출력

x/10b main	byte 1바이트 단위 - 10바이트 출력
x/10h main	halfword 2바이트 단위 - 20바이트 출력
x/10w main	word 4 바이트 단위 - 40바이트 출력
x/10g main	giant word 8바이트 단위 - 80바이트 출력

- 디스어셈블링

어셈블리 코드를 좀 보편적으로 보기 위한 명령어

disas func : func 함수 어셈블리 코드 보기

- 함수 호출

call func(arg1, arg2) : 특정 함수 func를 arg1, arg2 파라미터를 포함하여 호출하고, 반환값을 출력

- 점프

jump *0x0848321	해당 주소로 무조건 분기하여 인스트럭션 수행
jump 10	무조건 10행으로 분기하여 수행
jump func	func 함수로 무조건 분기하여 수행

- 시그널 전송

info signals : 보낼 수 있는 시그널의 종류를 확인 할 수 있다.

signal SIGKILL : 디버깅 대상의 프로세스에게 KILL 시그널을 보낼 수 있다.

- 메모리 특정 영역에 값을 설정

set {타입} [주소] = [값] : p 명령 대신에 set을 통해 메모리의 특정 주소에 저장하는 것이 더 일반적이다.

set {int}0x8048300 = 100 : 해당 주소에 100의 값을 입력

TAG GDB

Security / 리버싱 카테고리의 다른글

level 1 문제 해결 (/bin/ExecuteMe 해석)

level 1 문제 해결

GDB 사용법

리버싱 실습환경 구성

어셈블리언어

CPU와 매모리의 구조

댓글 1

댓글을 남겨주세요

tky7068
2020.07.14 16:35 신고

참고가 되었습니다. 감사합니다. ^^

↘답글달기

이전 76 / 218 다음

최근 글 최근 댓글

1 FreeRadius OTP 인증 서버 만들기

2 raw 디스크 부팅시 자동 연결

4 Network Summary (네트워크 축약)

5 HAproxy & Keepalived

6 RHCSA

7 평창올림픽 첫 멀웨어 공격 ' 파괴형 멀…

8 BitLocker 드라이브 암호화

9 윈도우 7 암호 복구 방법

10 [네이버 블로그]녹아내리는 귀신? 사상…

홈으로 방명록 로그아웃 맨위로