2025 SWING REVERSING

32기 리버싱 스터디

2025

SWING

담당자 육은서

진행 순서를 알아봅니다.

Chapter.1 ptrace 안티 디버깅

Chapter.2 anti what?

reversing week 6

- 진행 순서
- **ptrace 안티 디버깅**
- anti whar? 풀이

SWING 32nd study

ptrace란?

저번 시간에는 windows에서 쓰는 안티 디버깅에 대해 알아보았다
리눅스에서는 어떤 안티 디버깅을??

SWING 32nd study

ptrace란?

저번 시간에는 windows에서 쓰는 안티 디버깅에 대해 알아보았다
리눅스에서는 어떤 안티 디버깅을??

ptrace	다른 프로세스를 추적하는 system call

ptrace란?

저번 시간에는 windows에서 쓰는 안티 디버깅에 대해 알아보았다
리눅스에서는 어떤 안티 디버깅을??

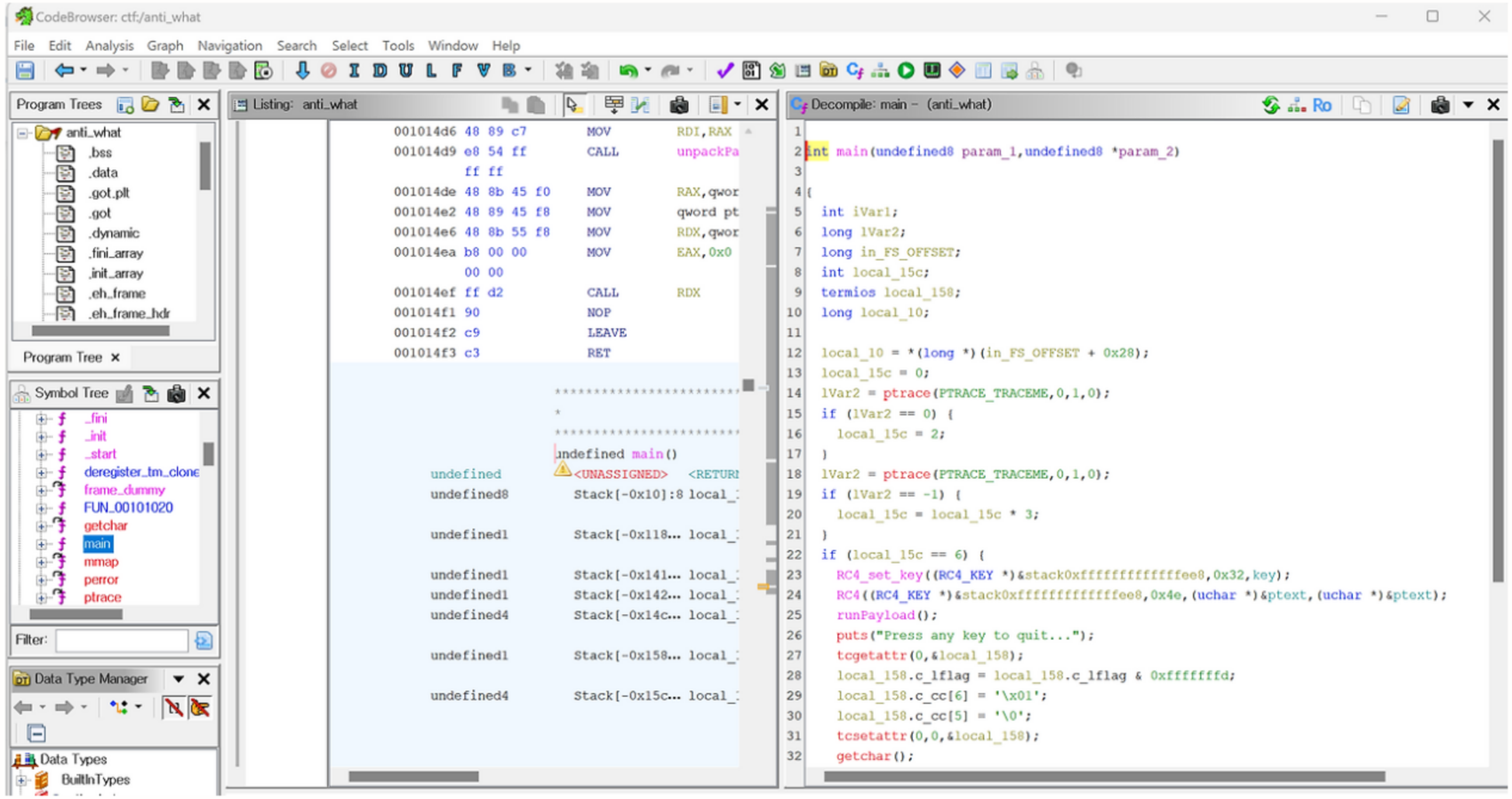
ptrace	다른 프로세스를 추적하는 system call

보통 이 함수를 써서 해당 프로세스가 추적이 되고 있다 확인 가능
= 디버깅 중이라는 거니까
if문 써서 exit 하는 등 프로그램을 종료 시켜버림 -> 안티 디버깅

reversing week 6

- 진행 순서
- ptrace 안티 디버깅
- **anti what? 풀이**

Anti What?



SWING 32nd study

SWING 32nd study

Anti What?

```
lVar2 = ptrace(PTRACE_TRACEME,0,1,0);
if (lVar2 == 0) {
local_15c = 2;
}
lVar2 = ptrace(PTRACE_TRACEME,0,1,0);
if (lVar2 == -1) {
local_15c = local_15c * 3;
}
if (local_15c == 6) {
RC4_set_key((RC4_KEY *)&stack0xffffffffffffee8,0×32,key);
RC4((RC4_KEY*)&stack0xffffffffffffee8,0×4e,(uchar*)&ptext,(uchar*)&ptext);
runPayload();
```

Anti What?

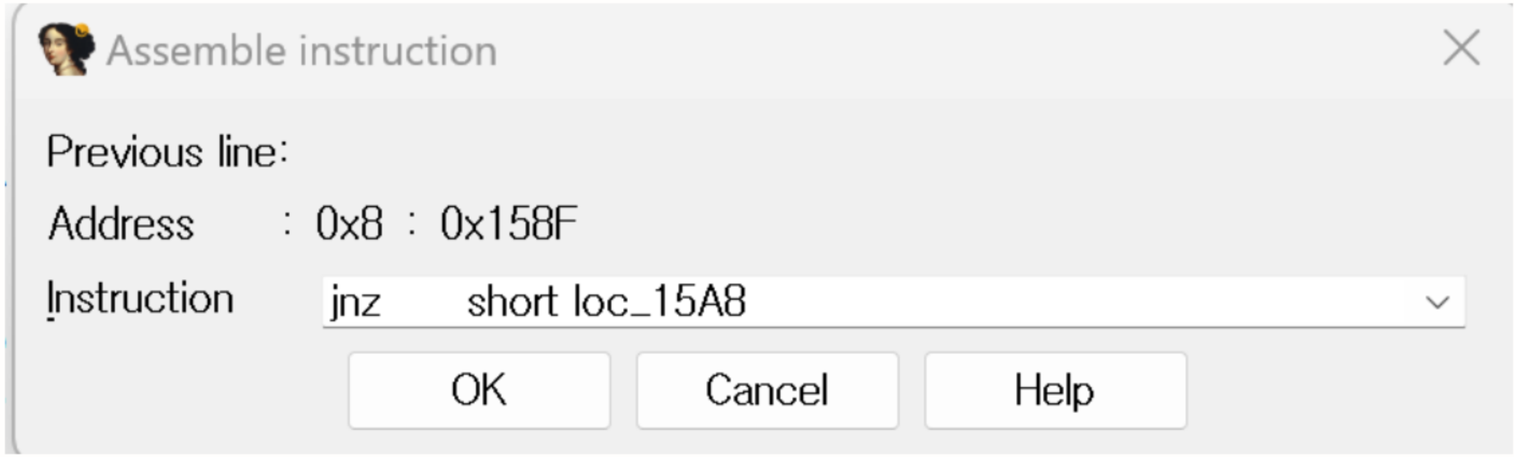
```
lVar2 = ptrace(PTRACE_TRACEME,0,1,0);
if (lVar2 == 0) {
    local_15c = 2;
}
lVar2 = ptrace(PTRACE_TRACEME,0,1,0);
if (lVar2 == -1) {
    local_15c = local_15c * 3;
}
if (local_15c == 6) {
    ...
    runPayload();
```

Anti What?

```
lVar2 = ptrace(PTRACE_TRACEME,0,1,0);
if (lVar2 == 0) {
    local_15c = 2;
}
lVar2 = ptrace(PTRACE_TRACEME,0,1,0);
if (lVar2 == -1) {
    local_15c = local_15c * 3;
}
if (local_15c != 6) {
    ...
    runPayload();
```

Anti What?

바꿔야 하는 어셈블리 명령줄로 커서 위치 옮기고
Edit > patch program > assemble...



Edit > patch program > apply patches 머시기 > OK

Anti What?

```
──────────────────[ DISASM / x86-64 / set emulate on ]──────────────────
► 0x5555555555ea <main+246>    call   runPayload                  <runP
ayload>
        rdi: 0x7fffffffe130 ◂— 0x6d0000004e /* 'N' */
        rsi: 1
        rdx: 0x6d
        rcx: 0x3d
```

실행시켜보면..

```
──────────[ REGISTERS / show-flags off / show-compact-regs off ]──────────
*RAX  0x7ffff7ffa027 ◂— outsd dx, dword ptr [rsi] /* 0x6f4379746e756f42;
 'BountyCon{JUDDpJ4K56wvdSH9tKFnbRzYeqT}' */
*RBX  0x7ffff7ffa04e ◂— 0
```

ptrace란?

저번 시간에는 windows에서 쓰는 안티 디버깅에 대해 알아보았다
리눅스에서는 어떤 안티 디버깅을??

ptrace(PTRACE_TRACEME, pid, ...)

PTRACE_TRACEME -> debugger에 의해 추적되고 있는 프로세스임을 명시

ptrace란?

PTRACE_TRACEME -> debugger에 의해 추적되고 있는 프로세스임을 명시
PTRACE_GETREGS -> 추적되는 프로세스의 레지스터 값을 복사
<sys/user.h> 의 user_regs_struct 활용
PTRACE_PEEKTEXT -> 추적되는 프로세스 주소의 값을 복사(명령어, opcode)
PTRACE_SINGLESTEP -> 한 instr 실행 후에 멈춤 (ni)
PTRACE_CONT -> 프로세스를 재시작 (c)

SWING 32nd study

부모자식 프로세스와 fork

하나의 프로그램 -> 여러개의 프로세스
fork -> 자식 프로세스를 생성하는 system call
이때 반환값이 0이면 child process -> 초기값/환경은 parent에게서 copy하고.
exec 같은 함수를 실행할 예정
반환값이 양수이면 parent process -> 얘가 없어지지 않게 wait를 넣어줘야 함

SWING 32nd study

부모자식 프로세스와 fork

하나의 프로그램 -> 여러개의 프로세스
fork -> 자식 프로세스를 생성하는 system call
이때 반환값이 0이면 child process -> 초기값/환경은 parent에게서 copy하고.
exec 같은 함수를 실행할 예정
반환값이 양수이면 parent process -> 얘가 없어지지 않게 wait를 넣어줘야 함

순서를 정리하면

1. fork로 자식 프로세스 생성
2. 자식 프로세스는 exec류의 함수로 다른 작업 실행
3. 실행이 끝나면 부모 프로세스에 시그널 전달
4. wait하고 있던 부모 프로세스 작업 실행

부모자식 프로세스와 fork

```c
int main(int argc, char** argv){
    if(argc<2){
        printf("Usage: ./swingdbg <target program>\n");
        return -1;
    }
    int pid=fork();
    char command[100];

    if(pid==0){
        // run target (child)
        if(ptrace(PTRACE_TRACEME,0,0,0)<0){
            printf("err: ptrace error\n");
            return -1;
        }
        execl(argv[1],argv[1],0);
    }
    else if(pid>0){
        // run debugger (parent)
        int stat;
        wait(&stat);

        while(WIFSTOPPED(stat)){
            printf("SWINGdbg >>> ");
            fgets(command, sizeof(command), stdin);
            command[strcspn(command,"\n")]=0;
            commandmenu(pid,command);
        }
    }
    else{
        printf("err: fork error\n");
        return -1;
    }

    return 0;
}
```

```
int pid=fork();

if(pid==0){
    //child는 debuggee
    ptrace(PTRACE_TRACEME,0,0,0);
    execl(argv[1],argv[1],0);
}else if(pid>0){
    //parent는 debugger
    int stat;
    wait(&stat);
    while(WIFSTOPPED(stat)){
        //WIFSTOPPED는 자식이 중단 상태인지
        //시그널 확인하는 함수
```

부모자식 프로세스와 fork

```c
int main(int argc, char** argv){
    if(argc<2){
        printf("Usage: ./swingdbg <target program>\n");
        return -1;
    }
    int pid=fork();
    char command[100];

    if(pid==0){
        // run target (child)
        if(ptrace(PTRACE_TRACEME,0,0,0)<0){
            printf("err: ptrace error\n");
            return -1;
        }
        execl(argv[1],argv[1],0);
    }
    else if(pid>0){
        // run debugger (parent)
        int stat;
        wait(&stat);

        while(WIFSTOPPED(stat)){
            printf("SWINGdbg >>> ");
            fgets(command, sizeof(command), stdin);
            command[strcspn(command,"\n")]=0;
            commandmenu(pid,command);
        }
    }
    else{
        printf("err: fork error\n");
        return -1;
    }

    return 0;
}
```

```
int pid=fork();

if(pid==0){
    //child는 debuggee
    ptrace(PTRACE_TRACEME,0,0,0);
    execl(argv[1],argv[1],0);
}else if(pid>0){
    //parent는 debugger
    int stat;
    wait(&stat);
    while(WIFSTOPPED(stat)){
        //WIFSTOPPED는 자식이 중단 상태인지
        //시그널 확인하는 함수
```

과제 안내

./swingdbg 실행하면 아래처럼 출력

```
Usage: ./swingdbg <target program>
```

./swingdbg hello 로 hello 파일을 디버깅해보자

hello.c 코드

```c
#include <stdio.h>

int main(){
    printf("hello swing 32rd");
    printf("\nIs your ptrace running?");
}
```

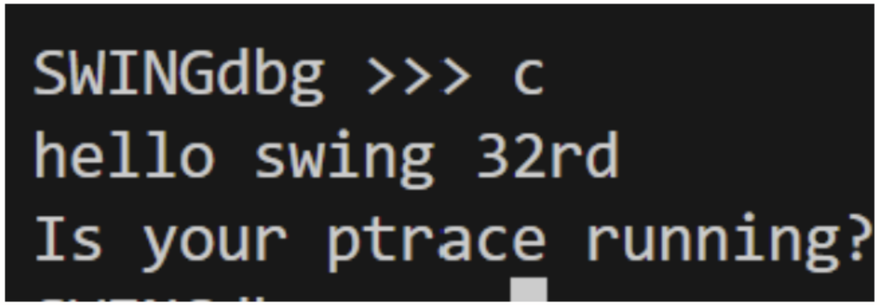
과제 안내

기능 구현 1. ni
아래처럼 출력되도록 하기 (tmi: instr이 어셈블리 문자열이 아니라 숫자인 이유는 opcode 디스어셈블리 기능이 없기 때문)

```
SWINGdbg >>> ni
[+] RIP = 0x70575e0ee543 | instr = 0xc4894900000c88e8
[+] Registers:
 RAX: 0x0
 RBX: 0x0
 RCX: 0x0
 RDX: 0x0
 RSI: 0x0
 RDI: 0x7ffd300c1e60
 RSP: 0x7ffd300c1e60
 RBP: 0x0
```

과제 안내

기능 구현 1. c
아래처럼 출력되도록 하기
이때 hello swing 32rd\n Is your ptrace running?은 자식 프로세스에서 온 문자열



깃허브에 업로드 후 깃허브 링크를 티스토리에 제출

수고하셨습니다~

질문사항은 카카오톡 주세요