Reversing Basic

Reverse Engineering 이란?

인공적으로 만들어진 어떤것으로 부터 정보를 빼내거나 그것이 어떻게 설계 됐는지 알아내는 과정

- 과학자들이 원자나 인간의 마음에 대한 "설계도"를 알아내고자 연구하는 것과 흡사
- 지식, 아이디어, 설계 철학 등을 알아내는 과정

거시적인 관점의 System-level reversing 과 집중적으로 분석하는 Code-level reversing 이 있다.

- System-level reversing : OS 의 다양한 Service 를 이용해서 Program 실행 File 과 입출력 값 등을 조사해서 정보를 추출한다.
- Code-level reversing : Low-level 에서 Software 가 어떻게 동작하는지 세밀한 부분까지 분석한다.

Reverse Engineering 의 적용

악성코드 개발자	OS나 기타 Software의 취약점을 찾아내기 위해
Anti-Virus Software 개발자	악성코드가 실행되는 각 단계를 추적, 분석하여 악성코드에 의해 발생할 수 있는 피해의 정도, 전파율, 제거 방법과 전파를 방지할 수 있는 방법을 찾아내기 위해
기타	암호 Algorithm 검증, DRM(Digital Right Management) 무력화, 소유권이 있는 Software 의 보안 취약점 발견, Software 개발에서의 Reversing

Low Level Software 란?

Software System 기반 구조에 대한 일반적인 명칭이다.

- Low-level Data 처리 : High-level programming 언어는 Data 처리에 대한 세부 사항의 상당 부분을 은닉하지만, Low-level 에서는 상당히 원시적이다.
  - C 언어 Code ---------- Low-level 처리

```
int Multiply ( int x, int y)
{
        int z;

        z = x * y;
        return z;
}
```

1. 함수의 Code 를 실행하기 전에 이전 상태 정보를 저장
2. z 변수를 위한 Memory를 할당
3. 인자로 전달된 x, y 값을 Register 로 복사
4. x 와 y 를 곱해서 결과 값을 Register 에 저장
5. 곱한 결과 값을 이미 Memory 를 할당한 z 변수에 복사
6. 앞서 저장한 상태 정보를 복원
7. 함수 호출자에게 반환 값으로 z 변수의 값을 전달

Register	성능에 거의 영향을 미치지 않고 접근할 수 있는 내부 Memory
Stack	비교적 짧은 기간동안 유지되는 정보를 위한 보조 저장 공간
Heap	실행 중 Memory Block 을 동적 할당할 수 있게 관리되는 Memory 영역
Executable Data Section	Application Data 를 저장하기 위해서 사용

Assembly Language

IA-32 의 8개의 범용 Register

Register	설명
EAX,EBX,EDX	정수 연산, Bool 연산, 논리 연산, 메모리 연산 모두에 사용 가능
ECX	Counting 이 필요한 연속적인 명령에 대한 Counter 로 가끔 사용
ESI/EDI	Memory 복사 명령에서 복사될 대상의 주소와 복사가 수행될 목적지의 주소 Pointer 에서 자주 사용
EBP	범용 Register 로 사용할 수 있지만 Stack Base Pointer 로서 가장 많이 사용. • Base Pointer 로 사용하면서 Stack Pointer 와 조합을 이뤄 Stack Frame 을 형성 • Stack Frame 은 현재 함수의 Stack 영역을 나타내며 Stack Pointer (ESP) 와 Base Pointer (EBP) 사이의 공간을 의미 • Base Pointer 는 일반적으로 함수의 반환주소가 저장된 바로 다음 위치의 Stack 주소를 가짐 • Stack Frame 은 함수에 전달된 Parameter 값이나 지역변수에 쉽고 빠르게 접근하기 위해서 사용
ESP	CPU Stack Pointer 로 사용 • Stack Pointer 는 Stack 의 현재 위치를 가리키는 것이며, 어떤 값이 Stack 에 Push 되면 Stack Pointer Register 가 가리키는 주소값도 감소

Flag

EFLAGS 라고 부르는 Register 가 모든 종류의 상태 Flag 와 System Flag 를 포함한다.

- System Flag : 다양한 Processor mode 와 상태를 관리하기 위해 사용
- 상태 Flag : 여러가지 논리 명령이나 정수 명령의 수행 결과에 따라서 Flag 값이 갱신 (조건문에 사용)

명령 Format

일반적으로 opcode(operation code) 와 하나 또는 두개의 operand 로 구성된다.

- opcode 는 mov 와 같은 명령의 이름, operand 는 명령에 전달되는 인자를 나타낸다.
  - operand 의 전형적인 예와 의미

Operand	설명
EAX	읽거나 쓰기 위해서 단순히 EAX 를 참조
0x30004040	Code 안에 삽입된(상수처럼) 직접적인 값
[0x4000349e]	Hard Coding 된 Memory 주소 값 - 전역변수 접근시 사용 가능

Compiler

High-level 언어로 작성된 Program 을 기계가 읽을 수 있는 형태인 Low-level 언어로 변환

Front end	Program Text 의 구문이 올바른지, 규약대로 작성됐는지 확인
Optimizer	Program Code 의 원래 의미를 유지시키면서 여러가지 방법으로 Program 을 향상 • Ex) 제거 가능한 Code 를 찾아내서 제거
Back end (Code generator)	Optimizer 에 의해 변경된 Code 를 가지고 해당 Platform 에 맞는 Binary 를 생성 • 명령 선택, Register 할당, 명령 Schedule 을 수행

Win32 API

상당히 많은 함수들의 모음으로서 Window Application 을 위한 공식적인 Low-level Programming Interface 이다.

Kernel API	KERNEL32.DLL Module 안에 구현되어 있으며 GUI 와 관련되지 않은 Service (File I/O, Memory 관리, 객체관리, Process 와 Thread 관리 등) 제공 • KERNEL32.DLL 은 다양한 Service 구현을 위해 NTDLL.DLL 의 Native API 를 호출
USER API	USER32.DLL Module 안에 구현되어 있으며 High-level 의 GUI 관련 Service 를 제공 • GUI 객체를 그리기 위해서 GDI 를 호출
GDI API	GDI32.DLL 안에 구현되어 있으며 Line 이나 Bitmap Image 를 그려주는 등 Low-level Graphic Service 를 제공

구조화된 예외 처리

예외는 Exception Handler 라고 불리는 특별한 함수로 즉각 JMP 하는 Program 안에서의 특별한 조건이다.

- Hardware 예외 : Processor 가 만들어 내는 예외
- Software 예외 : Program 이 Error 를 보고하기 위해서 예외를 생성할 때 발생

다양한 Reversing 방법

Dead-Listing	실행 Binary 를 Disassembler 나 Decompiler 를 이용해서 인간이 읽을 수 있는 형태로 변환해서 직접 분석 • Program 의 윤곽을 제공하고 관심있는 특정 함수를 쉽게 찾을 수 있는 강력한 장점
Live Code Analysis	실행 Binary 를 Debugger 로 실행시켜 동작을 관찰 • 내부 Data 를 관찰할 수 있고 Data 가 Code 의 흐름에 어떤 영향을 주는지 알 수 있으므로 더 많은 정보 획득 가능

Reversing Tool

Disassembler	Binary 기계 Code 를 해석해서 읽을 수 있는 Assembly 언어로 변환 • IDA Pro : 상당히 자세한 Disassembly 정보 제공과, Flowchart, hexray 기능을 제공 • 현재 선택한 연산자와 동일한 것들을 모두 강조해서 보여주는 것과 같은 세세한 기능도 제공
Debugger	현재 실행중인 함수의 Disassemble 된 Code 를 보여주고 사용자는 해당 Code 를 따라가며 실행과정을 파악 가능 • Reverser 에게 필요한 Debugger 의 핵심적인 기능 : 강력한 Disassembler, Software/Hardware Breakpoint, Register & Memory View, Process 정보 ▸User-mode Debugger : OllyDbg, Kernel-mode Debugger : WinDbg
Dump Tool	DUMPBIN, PEView, PEBrowse Professional 등
System Monitoring Tool	OS의 특정 Low-level Component 와 Application 이 만들어내는 적당한 Call 을 Hooking 함으로써 수행 • FileMon, TCPView, TDIMon, RegMon, PortMon, WinObj, Process Explorer 등