2022 CAT-CERT 스터디 보고서

컴퓨터정보공학부 202221168 김지혜

2022.05.30#Reversing\_study-1

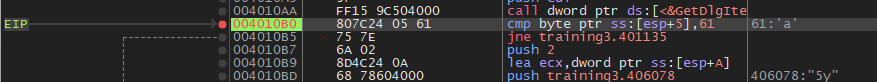
실습 #3

텍스트, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. DialogBoxParamA 라는 함수를 통해 저 부분이 대화상자를 여는 함수 인 것을 알 수 있음

텍스트, 검은색, 점수판, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. ‘1aE5’를 입력했을 때 두번째 자리인 ‘a’부터 비교하고 있고, 입력한 a와 16진수로 61인 ‘a’를 비교하고 있는 것을 알 수 있음

텍스트, 실내, 전자기기, 화면이(가) 표시된 사진

자동 생성된 설명텍스트, 모니터, 실내, 화면이(가) 표시된 사진

자동 생성된 설명

1. 그 뒤로 ‘5y’라는 수를 ‘traning3.401150’ 함수의 인수로 받고 내가 입력한 값과 ‘5y’를 비교하고 있음

-> 3,4번째자리의 값이 ‘5y’인 것을 알 수 있음

텍스트, 실내, 검은색, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. ‘yaE5’를 입력했을 때 ‘y’와 16진수로 45인 ‘E’를 비교하는 것을 보고 첫번째 자리값이 ‘E’인 것을 알 수 있음

텍스트, 점수판, 화면이(가) 표시된 사진

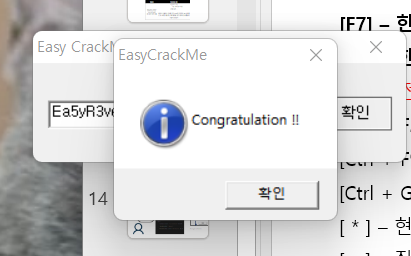
자동 생성된 설명

<’Ea5y234567’를입력할때>텍스트, 실내, 디스플레이, 스크린샷이(가) 표시된 사진

자동 생성된 설명

<’Ea5yR3versing’을 입력할 때 ‘s’을 비교하는과정>

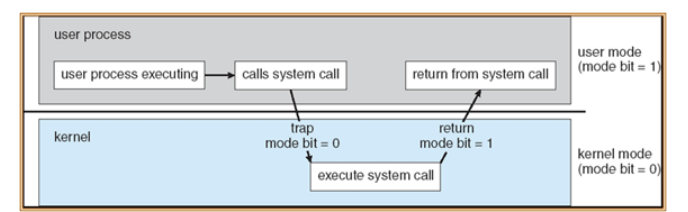
1. ‘R3versing’이라는 값을 esi에 넣었고 esi의 값(R3versing)을 bl에 넣어 dl과 비교하고 있음. 여기서 dl은 내가 입력한 값이고(234567) 주소값을 하나씩 더하면서 반복문을 통해 하나하나 비교하는 것을 알 수 있다.



1. 이 과정을 통해 비밀번호는 ‘Ea5yR3versing’인 것을 알 수 있다!

**- 커널영역과 유저영역 차이**

|  |  |
| --- | --- |
| 커널 영역 | 유저영역 |
| -컴퓨터 운영체제의 핵심  -유저영역을 제외한 영역  -시스템의 모든 메모리에 접근할 수 있고 모든 cpu 명령을 실행할 수 있음  -시스템의 모든 것을 완전히 통제하기 때문에 사용자(유저모드)가 직접 접근할 수 없고 접근 시 system call을 통한 커널 모드로 전환이 필요함 | -스택영역, 힙영역, 데이터영역, 코드영역  -시스템데이터에 제한된 접근만이 허용되고 하드웨어를 직접접근할 수 없음  - CPU유저모드특권수준으로 코드를 실행함  -사용자는 유저모드에서 코드를 작성하고 프로세스를 실행하는 등의 행동을 할 수 있음 |



모드 비트가 1이면 보통 유저모드 0이면 커널모드를 가리킴

코드가 수행될때는 유저모드에서 돌고, 프로그램을 수행하다 인터럽트가 걸려서 운영체제가 수행될때는 비트모드를 바꾸고 커널모드로 전환된다

\*\*CPU에 두가지 모드가 있는 이유 : I/O장치를 보호하기위해

->어플리케이션 프로그램들은 직접 I/O장치에 접근할 수 없게 만들고 운영체제를 통해서만 사용할 수 있도록 함

- 가상메모리 물리모메모리(상세)

**물리 메모리**

: 물리메모리는 우리가 흔히 알고 있는 ‘RAM’에 올려져 있는 것을 말함

실제로 프로그램이 동작하기 위해서는 물리메모리에 로딩되어야함

**가상메모리**

:가상메모리는 물리메모리가 존재하는 RAM을 관리하는 방법에서 만들어진 주소

**실제 필요한 내용의 가상메모리만 실제 물리 메모리로 변형시켜 일부분만 사용함**

**(요구 페이징 기술)**

**프로세스간 공간 분리로 프로세스 이슈가 전체 시스템에 영향을 주지 않음**

가상메모리의 실제 공간은 ‘하드디스크 + RAM’

1. 라이브러리가 여러 프로세스 사이에 공유가 가능함

-> 실제로 라이브러리가 들어있는 물리메로리는 하나로 모든 프로세스에 공유됨

2. 프로세스가 만들어질 때 훨씬 더 효율적임

3. 실제 물리메모리보다 더 큰 메모리를 요구하는 프로세스 구동가능

**- DLL이란 (Dynamic Link Library)**

|  |  |
| --- | --- |
| **라이브러리**  **(library)** | -자주쓰고 기초적인 함수들을 중복 개발하는 것을 피하기 위해 표준화된 함수 및 데이터 타입을 만들어서 모아 놓은 것  -한번 구축해놓으면 다시 만들 필요없어서 개발속도도 빠르고 신뢰성도 확보할 수 있음 |

DLL : 동적링크; 실행파일에서 해당라이브러리의 기능을 사용 시에만 라이브러리 파일을 참조하여 기능을 호출함

- 정적링크와 다르게 컴파일 시점에 실행파일에 함수를 복사하지 않고 함수의 위치정보만 가지고 그 함수를 호출할 수 있게 함

|  |  |
| --- | --- |
| 좋은점 | 1. **더 적은 리소스 사용**   -한 코드를 여러 프로그램이 동시에 사용하기에 메모리가 절약됨  -사용하는 디스크 공간을 줄일 수 있음  🡪 정적링크는 함수모두가 포함되지만 DLL을 사용하는 프로그램은 크기가 작음   1. **모듈식 아키텍쳐 사용**   -모듈식 프로그램을 효율적으로 개발  -여러 언어버전이 필요한 프로그램,모듈식 아키텍처가 필요한 프로그램개발 가능   1. **손쉬운 배포와 설치**   -DLL 함수를 업데이트나 수정해야하는 경우 설치할 때 따로 다시 연결안해도 됨  -여러 프로그램이 같은 DLL을 사용하는 경우 모든 프로그램에 업데이트나 수정내용이 적용됨   1. **프로그래머들의 분담 작업에 용이, 뛰어난 재사용성** |

- 멀티바이트 유니코드 차이점

|  |  |  |
| --- | --- | --- |
|  | 멀티바이트 | 유니코드 |
| 개념 | 하나의 문자세트에 부여된 글자값이 두 바이트(0-65535 범위 내에 있음) | 전 세계의 모든 문자를 컴퓨터에서 일관되게 표현하고 다룰수 있도록 설계된 표준  Ex) UTF-8 |
| byte크기 | 한글 : 2byte  영어 : 1byte | 모든 문자 2byte |
| 장점 | 멀티바이트 문자 집합은 특정 문자 집합마다의 코드페이지가 존재하여 영어 이외의 나라 문자 표현 가능함 | 지구상에서 통용되는 대부분의 문자를 담고 있음  2byte를 사용하므로 아스키코드가 가진 한계를 극복함 |
| 단점 | ASCII를 제외하고는 DBCS들이 모두 다름  (원래 8비트에서 또 다른 8비트를 단순히 확장한 것을 의미함) | Printf, scanf 함수는 유니코드와 맞지 않음  ->아스키코드의 char는 1byte 이지만 유니코드의 char는 2byte이기 때문임  함수사용을 위해서는 wprint등의 새로 정의된 함수를 사용해야함 |
| 변수 | 일반 문자열과 변수를 사용함 | 문자열의 경우 ‘L”@@”’를 사용함  변수의 경우 w가 붙음 |

\*코드페이지

: 특정 문자 인코딩 테이블(정수를 표현하는 일련의 비트들이 특정한 문자와 결합하여 도표화한 것)을 위해 쓰이는 전통적인 IBM 용어

**PE에서 일반적인 섹션 4가지에 대해 조사해오고, 각 섹션마다의 권한 조사**

|  |  |
| --- | --- |
| .text 섹션 | 실행파일의 소스코드가 들어있는 섹션, 초기화되거나 초기화 되지 않는 변수, import와 export 테이블 정보를 제외한 나머지 소스코드들이 저장됨  <기본 속성정보>  -IMAGE\_SCN\_CNT\_CODE  -IMAGE\_SCN\_MEN\_EXCUTE  -IMAGE\_SCN\_MEN\_READ |
| .data  섹션 | **읽기/쓰기 가 가능한 일반적인 데이터를 위한 섹션**.  정적변수와 전역변수가 저장됨  <기본 속성정보>  -IMAGE\_SCN\_CNT\_INITIALIZED\_DATA  -IMAGE\_SCN\_MEN\_READ  -IMAGE\_SCN\_MEN\_WRITE |
| .radata  섹션 | r.data 섹션은 **읽기 전용 섹션으로** 상수와 배열로 정의된 문자열과 런타임 라이브러리에서 사용하는 에러 메세지등이 저장됨  가끔 섹션의 크키가 작고, 읽기전용속성을 가지면 .rdata섹션으로 통합되는 경우도 잇음  <기본 속성정보>  -IMAGE\_SCN\_CNT\_INITIALIZED\_DATA  -IMAGE\_SCN\_MEN\_READ |
| .idata,edata  섹션 | .idata섹션은 import table이 들어있고, .edata 섹션은 export table이 들어있음  ->실행하는데 꼭 필요한 섹션은 아님  <.idata 기본 속성정보>  -IMAGE\_SCN\_CNT\_INITIALIZED\_DATA  -IMAGE\_SCN\_MEN\_READ  -IMAGE\_SCN\_MEN\_WRITE  <.edata 기본 속성 정보>  -IMAGE\_SCN\_CNT\_INITIALIZED\_DATA  -IMAGE\_SCN\_MEN\_READ |

**VA, RVA, RAW 조사**

**VA:** 프로세스 가상 메모리의 절대 주소(실제 메모리에 로딩되는 주소)

파일을 실행하면 프로세스 어딘가에 존재하는 가상메모리주소공간에 실행한 파일이 올라가는데 **가상메모리이기 때문에 어디에 로딩되었는지 시작주소를 알고 있어야 접근이 가능함**

-> imageBase를 기준으로 만들어지고, 실제메모리에서는 이 주소를 보고 이 위치에 있는 명령을 실행함.

**\*\*\* imageBase주소 + RVA주소 = VA주소**

**RVA:** imageBase로 부터의 상대 주소(메모리에 로딩된 상태)

특정한 위치부터 얼만큼 떨어진 곳에 위치하는지 알려주는 주소임

imageBase는 NT헤더 중에서 IMAGE\_OPTIONAL\_HEADER32 구조체 안에 있으며, **메모리에서 PE파일이 어디에 로딩이 되는지 메모리에 로딩되는 시작 주소를 알려줌**

-> PE파일이 지정된 베이스 위치를 기준으로 로딩된다는 보장이 없기 때문에 RNA로 나타냄. 파일의 경우 일반적으로 파일이 지정된 위치에 로드 되지만 하나의 프로세스에는 여러 개의 DLL이 존재하고 있기 때문에 중첩이 됨. 중첩을 방지하기 위해 절대주소가 아닌 상대주소를 사용함

**RAW :** PE파일이 로딩되기 전에 파일이 실행되기 전의 주소

->실행되기 전의 주소이기 때문에 파일 자체를 분석해야할 때 주로 이 주소를 기반으로 데이터를 확인함

**\*\*\*RAW(파일의 임의 위치) = RVA (메모리의 임의 위치) -VA(메모리 해당 시작위치) +PointerToRawData(파일의 해당 섹션 시작위치)**

**하드웨어 , 소프트웨어 브레이크포인트 차이**

브레이크 포인트

:디버깅을 목적으로 실행중인 디버깅 대상 프로세스를 의도적으로 멈추게 하는 장소

->멈춘 시점의 변수나 스택파라미터, 특정 메모리 지점의 값을 조사 할 수 있음

|  |  |  |
| --- | --- | --- |
|  | 소프트웨어 브레이크포인트 | 하드웨어 브레이크포인트 |
| 개념 | -명령을 실행하는 CPU를 일시 중지 하는데 사용됨.  -흔하게 사용하는 형태의 브레이크 포인트 | -설정할 브레이크포인트개수가 적을때나 소프트웨어이 코드가 변경되면 안될 때 사용함  -디버그 레지스터(8개)를 이용하여 CPU레벨에서 설정함 |
| 종류 | 1. 일회성 브레이크포인트   : 한번 이벤트가 발생하면 디버거의 내부 브레이크 포인트 리스트에서 해당 정보가 제거됨   1. 지속적인 브레이크포인트   : 브레이크포인트가 발생하고 CPU가 원래 상태로 돌려놓은 다음 다시 브레이크포인트 설정을 수행하기 때문에 리스트에 정보가 유지됨 | 1. DR0 ~ DR3   : 브레이크포인트주소 저장   1. DR4 ~ DR5   :예약된레지스터 저장(사용안함)   1. DR6   :브레이크포인트에 의해 발생되는 디버깅 이벤트의 종류저장   1. DR7   : 하드웨어 브레이크포인트의 on/off 스위치, 서로 다른 브레이크포인트의 조건도 저장 |
| 장점 | 브레이크 포인트 횟수에 제한이없고 사용하기 쉬움 | 디버깅할 때 소프트웨어의 코드가 변경되지 않음 |
| 단점 | 메모리 상의 실행 바이너리의 바이트를 변경하기 때문에 CRC값이 변경됨  변경된 것이 확인이 되면 스스로 종료해버려서 악성코드 분석이 어려워짐  \*CRC(순환중복검사): 데이터가 변경되었는지 확인하기 위해서 사용되는 방법 | 브레이크포인트의 주소를 저장할 디버그가 4개이기 때문에 한번에 4개만 설정함  브레이크포인트를 설정할 수 있는 데이터의 최대 크기가 4바이트여서 메모리 영역을 다루기가 어려움 |