Rootkit Tools & Debugger

Deok9

http://deok9.org

I love hot girl(goal) & beat(bit)

개요

1. Rootkit Tools

2. Debugger

Rootkit Tools

- Development Tools
- Diagnostic Tools
- Reversing Tools
- Disk Imaging Tools

개발 툴

- **WDK(Windows Device Driver Kit)**
  - Kernel-mode 소프트웨어 개발에 필요한 모든 것을 제공 받을 수 있음
  - 공식적인 문서들을 제공 받기 용이

- **Windows SDK**
  - Rootkit이 유저레벨에 위치할 때 유용한 툴
  - 윈도우 API 와 COM 개발에 필요한 헤더, 라이브러리 문서들을 포함하고 있음
  - Resource Compiler(RC) 와 Dumpbin.exe 과 같은 핸디 툴 포함

Rootkit Tools

- **Visual Studio Express**
  - 가벼운 무료 소프트웨어
  - 표준적인 에디터 제공
  - 전체설치 시 C/C++ Reference를 제공 받을 수 있음
  - C 런타임 라이브러리 함수들을 상세하게 다루고 있음
  - cl.exe, link.exe, nmake.exe 와 같은 기본적인 개발 툴도 보유

- **Windows Server 2003 Device Driver Kit**
  - 16bit Real-mode에서의 개발을 하기 위해서는 이를 지원하는 개발 툴이 필요
  - Open Watcom 이라는 오픈소스 툴도 이와 같이 Real-mode 지원

진단 툴

- **효율적으로 Rootkit의 동작 패턴을 알 수 있음**
  - WDK 의 drivers.exe 를 통해 설치된 드라이버들을 확인 가능
  - netstat, task list 와 같은 빌트인 도구를 통해서도 확인 가능
    - ✓ 이와 같은 툴은 실시간 스냅샷 제공 보장에는 부족

- **1990 년대 중반 Sysinternals 등장**
  - 초기 개별적 이었던 프로그램들은 2006년 MS 에서 Suite 로 나오게 됨
  - regmon, filemon 과 같은 툴들의 소스 코드가 공개 되었고, 이에 따라 Undocumented 시스템 오브젝트 들에 접근이 가능하게 됨
  - 그에 따라, Rootkit 개발자들도 목적에 맞게 Rootkit을 재 디자인

리버싱 툴

- **조금 더 상세하게 raw 단에서 분석**
  - Kernel-mode Debugger(kd.exe, windbg.exe)를 통해 수행가능
    - ✓ 이를 통해 키 루틴을 Disassemble 가능하고, 메모리 분석 및 Kernel-object 조작이 가능

- **Disassembler**
  - 실시간 분석이 전제 조건이 아니라면, IDA Pro Disassembler를 활용
    - ✓ Disassembler는 라이브 메모리 이미지 보다 주로 비 활성 파일을 다룸
  - Dumpbin 의 "/disasm" 옵션을 통하여 코드 섹션 확인 가능
  - Raw 바이너리 폼 분석 시에는 Hex editor(Cygnus hex editor) 를 사용
    - ✓ 주로 리버싱보다 바이너리 패치에 사용

디스크 이미징 툴

- **포렌식 조사 가이드**
  - Air Gap Security : 네트워크와 같은 외부 연결을 끊어 버림
    - ✓ 다른 물리적인 미디어에 해당 디스크의 카피 본을 생성할 때 사용하는 용어
    - ✓ Sneakernet 으로 알려져 있음
  - 고립된 상태에서 모든 테스트를 수행

- **DBAN(Darik's Boot and Nuke)**
  - 자체 부팅환경을 가지고 있으며, 플로피, CD, Flash 드라이브에 설치가능
  - 데이터 삭제 툴로 잘 알려져 있음
  - 감지된 디스크의 내용을 자동으로 삭제 가능

- **대체 툴**
  - Windows Automated Installation Kit(WAIK) 는 900MB ISO 이미지를 생성
  - 리눅스 유저의 경우 dd 를 통해 디스크 이미지를 획득 가능하지만 매우 느림
  - 책의 저자는 PING(Partimage Is Not Host) 라는 오픈 소스 툴을 사용함

Debugger

- General
- Kernel Debugger

일반

- **디버거를 사용하는 이유?**
  - 윈도우는 리눅스와 달리 소스코드가 공개되지 않음
    - ✓ Kernel-mode 루틴에 대한 정보를 알기 위해서는 Kernel-mode Debugger를 통해 살펴보아야 함
  - Printf() 기능
    - ✓ 그러나 시스템 크래쉬, 에러 등 디버거 콘솔에 나타내기 힘든 것들도 많음

- **콘솔 디버거**
  - 유저 모드 : Cdb.exe, Ntsd.exe
  - 커널 모드 : Kd.exe

- **GUI 디버거**
  - 유저 & 커널 모드 : Windbg.exe

- **심볼 파일**
  - 심볼 파일은 프로그램 데이터 베이스 포맷으로 만들어 진 메타 데이터
  - 개발 툴이 심볼 파일 생성을 지원하면 프로그램 이름과 같은 이름으로 .pdb 확장자를 가진 파일이 생성됨
  - 종류
    - ✓ Public Symbol Information : 프로그램 함수의 주소, 이름, 전역변수, 데이터 타입, 소스 코드에서 정의된 클래스를 가짐
    - ✓ Private Symbol Information : 지역변수, 기계어와 소스코드를 매핑 가능하게 함
    - ✓ Full Symbol File : Public + Private Symbol Information
    - ✓ Stripped Symbol File : 오직 Public Symbol Information
    - ✓ Exported Symbols : .pdb 와 같은 raw 바이너리 파일

- **symchk**
  - /r : 심볼파일이 체크 가능한지 여부 확인
  - /s : 심볼파일을 포함하고 있는 디렉토리 위치 정보
  - /ps : 심볼파일이 Stripped 인지 확인

- **윈도우 심볼**
  - MS 는 OS 심볼 파일을 무료로 공개 및 다운로드 가능하게 제공
  - Retail symbols : 쓸데 없는 것들을 제외한 최적화된 심볼
  - Checked symbols : Retail symbols 보다 더 많은 양을 가지고 있으며, 주로 디바이스 드라이버 개발자들이 사용

- **Cdb.exe 디버깅 환경**

1. _NT_SOURCE_PATH : 타겟 바이너리 소스 코드 파일
2. _NT_SYMBOL_PATH : 심볼 파일 디렉터리 루트 노드
3. _NT_DEBUG_LOG_FILE_OPEN : 디버깅 세션 로그 파일
   - ✓ 1, 2의 경우 복수 디렉토리가 가능하며 세미콜론으로 구분
   - ✓ 1의 경우 소스코드가 없으면 무시해도 됨
   - ✓ 2의 경우 반드시 필요
   - ✓ 3의 경우 로그 파일이 이미 있다면 덮어쓰기 됨

- 배치 파일 예제

```
setlocal
set PATH=%PATH%;C:\Program Files\Debugging Tools For Windows
set LOG_PATH=-logo .\DBG_LOG.txt
set DBG_OPTS=-v
set SYMS=-y symsrv*symsrv.dll*.\*http://msdl.microsoft.com/download/symbols
set SRC_PATH=-srcpath .\
cdb.exe %LOG_PATH& %DBG_OPTS% %sYMS% %SRC_PATH% MyWinApp.exe
endlocal
```

- **Cdb.exe 호출**
  - 실행 : Cdb.exe 파일명
  - Attach : Cdb.exe –p 또는 –pn 파일명 또는 프로세스 ID
  - 읽기 전용 : Attach 에서 –pv 옵션 추가
    - ✓ 영향을 주지 않고 읽기만 수행할 때 사용

- **Cdb.exe 제어**
  - __asm{ int 0x03; } 과 같은 Code 를 통해 브레이크 포인트를 삽입 가능함
    - ✓ 디버깅 중일 때 동적으로 브레이크 포인트를 설정하기에는 무리가 있음
    - ✓ Cdb.exe 에서는 브레이크 포인트를 조작가능한 명령어가 있음

- **브레이크 포인트 명령어**
  - bl : 현재 브레이크 포인트 들의 리스트 출력
  - bc breakpointID : 특정 브레이크 포인트 삭제
  - bp functionName : 특정 루틴 (함수)의 첫번째 바이트에 브레이크 포인트 설정
  - bp : 현재 IP 레지스터가 가리키는 지점에 브레이크 포인트 설정
  - Cdb.exe 를 통해 프로그램을 실행할 때 2개의 브레이크 포인트가 자동으로 들어가 있음
    - ✓ 프로그램 이미지가 로드 될 때, 디버깅이 종료 되었을 때

Debugger

- **실행 명령어**
  - g(go) : 다음 브레이크 포인트 까지 실행
  - t(trace) : 다음 명령어 실행(step into)
  - p(step) : 다음 명령어 실행(step over)
  - gu(go up) : 현재 함수 리턴 지점 까지 실행
  - q(quit) : 디버거와 디버기 종료

- **로드된 모듈 리스트 확인 명령어**
  - lm, !lmi
  - lm –v 또는 !lmi 의 경우 더 Detail 하고 정형화 하여 출력 시켜 줌

- **심볼 명령어(x)**

Command	Description
x moduleName!Symbol	Report the address of the given symbol (if it exists).
x *!	List all of the modules currently loaded.
x moduleName!*	List all of the symbols and their addresses in the specified module.
x moduleName!arg*	List all of the symbols that match the "arg*" wildcard filter.

- **타입 디스플레이 명령어(dt)**
  - 주로 데이터 구조를 보는데 사용하며, 이를 통해 변수, 구조체 등을 확인 가능

- **레지스터 명령어(r)**

- 디스어셈블 명령어(u)

Command	Description
u	Disassemble eight instructions starting at the current address.
u Address	Disassemble eight instructions starting at the specified linear address.
u start end	Disassemble memory residing in the specified address range.
uf FunctionName	Disassemble the specified routine.

- 디스플레이 명령어(d*)

Command	Description
db addressRange	Display byte values both in hex and ASCII (default count is 128).
dW addressRange	Display word values both in hex and ASCII (default count is 64).
dd addressRange	Display double-word values (default count is 32).
dps addressRange	Display and resolve a pointer table (default count is 128) .
dg start End	Display the segment descriptors for the given range of selectors.

커널 디버거

- **호스트-타겟 설정 (비추)**
  - 타겟 머신과 호스트 머신을 상호 연결하여 커널 디버깅
  - 호스트 머신에서는 커널 디버거가 설치되어 있어야 함
  - 2개의 머신을 세팅해야 하는 불편함이 있음

- **호스트-타겟 설정 시 준비해야 할 하드웨어**
  - 널 모뎀 케이블, IEEE 1394, USB 2.0 디버그 케이블 중 하나로 가능
  - 널 모뎀의 경우 RS-232 시리얼 케이블을 통해 가능(둘다 수컷)
    - ✓ BIOS 또는 장치관리자에서 COM 포트가 활성화 되어 있는지 확인
  - 커뮤니케이션 포트인 COM 포트를 지정한 후, putty 와 같은 SSH 프로그램으로 접근하여 디버깅 가능

- **호스트-타겟 설정 시 준비해야 할 소프트웨어**
  - 타겟 머신의 부팅 설정을 커널 디버깅이 가능하도록 수정

```
BCDedit /debug ON
BCDedit /dbgsettings SERIAL DEBUGPORT:1 BAUDRATE:19200
BCDedit /enum all
```

    - ✓ /debug ON : 시스템 부트스트랩 동작 중 커널 디버깅이 가능하도록
    - ✓ /dbgsettings SERIAL DEBUGPORT:1 BAUDRATE:19200 : 글로벌 머신 파라미터 설정
    - ✓ /enum all : 부팅 설정을 리스트화 시킴

- **타겟 제어**
  - 타겟 머신이 시작되면 커널 디버거는 자동으로 사용자가 어떤 행위를 할 때까지 기다리며, 이 때 "Ctrl + C" 를 누르면 Break 모드가 되어서 커널 디버깅을 수행가능
  - "Ctrl + C" 명령은 디버거 명령어가 실행 중일 때 사용 시 해당 동작을 중지 시킬 때도 사용

- **명령어**
  - 대부분의 명령어는 Cdb.exe 와 동일
  - 타겟 머신을 중지 시키지 않고 종료 하려면?

```
kd> bc *
kd> g
kd> <Ctrl+B><Enter>
```

    - ✓ bc * : 모든 브레이크 포인트를 삭제
    - ✓ g : 중지 상태의 타겟 머신에게 실행을 계속 하도록 명령
    - ✓ <Ctrl + B><Enter> : 타겟 머신으로 부터 커널 디버거를 Detach 시키고 커널 디버거 종료

그 외 명령어

- !process 프로세스 플래그
  - ✓ 특정 프로세스 또는 모든 프로세스의 메타 정보를 나타낼 수 있음
  - ✓ 프로세스 인자는 PID 또는 해당 프로세스의 EPROCESS 구조체 베이스 주소
  - ✓ 플래그 인자는 얼마나 상세히 출력할 건지에 대한 레벨(5 비트 값)
    - 0 : 가장 작음, 31 : 가장 많음
    - !process 0 0 : 현재 실행되고 있는 모든 프로세스의 정보를 출력

- !peb
  - ✓ Processor Environment Block 으로서, 프로세스 이미지에 대한 정보 출력

Debugger

- **크래쉬 덤프 (비추)**
  - 일종의 머신 스냅샷 바이너리 파일
  - 버그가 일어났을 때 생성됨
  - 커널 디버거 명령어가 일부 되지 않음

- **크래쉬 덤프를 이용한 커널 디버깅**
  - 소프트웨어 엔지니어들이 시스템 상태를 사후 검토 하기 위한 목적
  - 2 머신의 설정 작업 필요 없이 단순히 툴로 분석 가능
    - ✓ Complete 메모리 덤프 : 크래쉬 덤프 파일 생성 당시 모든 물리 메모리 상태 덤프
    - ✓ 커널 메모리 덤프 : 유저 모드 어플리케이션 영역을 제외
    - ✓ Small 메모리 덤프 : 64KB 의 시스템 메타 데이터

- **크래쉬 덤프 파일 생성**
  - 제어판 -> 고급 -> 설치 & 복구 -> 덤프 파일 생성 및 위치 설정
  - HKLM₩System₩CurrentControlSet₩Services₩i8042prt₩Parameters₩CrashOnCtrlScroll (DWORD) 값 생성후 0x01 로 세팅
  - 부팅 시 우측 Ctrl 키를 누른 상태에서 스크롤 락키를 2번 누르면 MANUALLY_INITIATED_CRASH 버그 체크를 STOP(0x000000E2) 시킴
    - ✓ STOP 코드는 STOP 이라는 단어와 함께 블루 스크린에 함께 나옴
  - Kd.exe 에서 ".crash" 를 입력하면 MANUALLY_INITIATED_CRASH 버그 체크를 STOP 시킴
    - ✓ 덤프 파일이 타겟 머신에 생성됨

- **크래쉬 덤프 분석**
  - Kd.exe 에서 –z 옵션

```
KD.exe %DBG_LOGFILE% %DBG_SYMBOLS% %DBG_CONNECT% -z C:\windows\MEMORY.DMP
```

    - ✓ 파일이 로드되면 ".bugcheck" 명령을 통하여 검증
  - 호스트 타겟 설치보다는 편하지만 정적 스냅샷 이기 때문에, 브레이크 포인트나 프로그램 흐름 제어 관련 명령어를 사용할 수 없음
    - ✓ Windows Debugging Tools Online Help 에서 가능한 Command 확인 가능

Debugger

- **로컬 커널 디버깅 (비추)**
  - 메모리 읽기 쓰기가 되더라도, 커널 디버거 명령어 들이 거의 먹히지 않음
- **CPU 성능이 좋다면**
  - 가상 머신을 이용한 커널 디버깅을 추천
  - 이는 파이프를 이용하여 통신하며, 유연하게 2 머신을 하나의 머신에서 구현가능

질문 및 답변