Bootkit Threat Evolution in 2011

n0fate

Feedbeef.blogspot.com

개요

1. 소개

2. 주요 루트킷 의 파일 은닉 기술

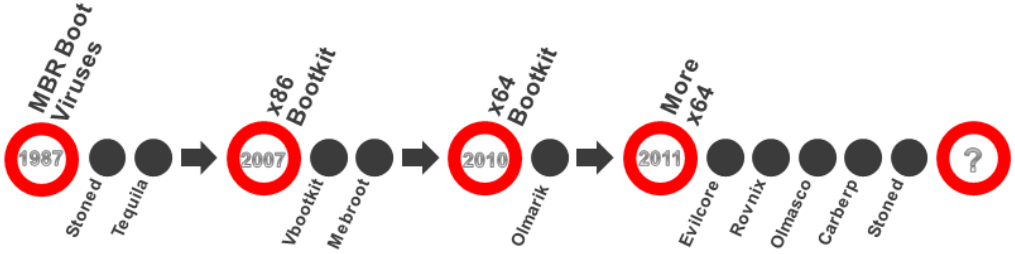
   - TDL4 bootkit
   - Olmasco bootkit
   - ZeroAccess rootkit
   - Rovnix bootkit

3. 결론

소개

- 부트킷 소개

- **부트킷**
  - Full Encryption system 공격 목적으로 사용됨.
    - ✓ Stone bootkit은 부트로더를 수정하여 암호화 키와 패스워드를 탈취
  - TPM을 이용하여 방어하고 있음



○ **Bootkit PoC evolution:**

- ✓ **eEye Bootroot (2005)**
- ✓ **Vbootkit (2007)**
- ✓ **Vbootkit v2 (2009)**
- ✓ **Stoned Bootkit (2009)**
- ✓ **Evilcore x64 (2011)**
- ✓ **Stoned x64 (2011)**

○ **Bootkit Threats evolution:**

- ✓ **Mebroot (2007)**
- ✓ **Mebratix (2008)**
- ✓ **Mebroot v2 (2009)**
- ✓ **Olmarik (2010/11)**
- ✓ **Olmasco (2011)**
- ✓ **Rovnix (2011)**
- ✓ **Carberp (2011)**

주요 루트킷의 파일 은닉 기술

- TDL4 bootkit
- Olmasco bootkit
- ZeroAccess rootkit
- Rovnix bootkit

소개

- **2010년 후반에 TDL4(Win32/Olmarik) 등장**
  - 최초로 64비트 시스템을 대상으로 광범위하게 전파된 부트킷
    - ✓ IsWow64Process() API를 이용하여 분기함
  - 봇넷 구성에 사용되었음
    - ✓ 악성코드 유포 희망자에게 URL을 제공
    - ✓ 감염된 사용자의 시스템은 자동으로 부여된 ID 값을 인자로하여 특정 사이트에 접속함.
  - HIPS를 우회하기 위해 신뢰있는 시스템 프로세스(spooler.exe)에 DLL을 인젝션 함.
    - ✓ AddPrintProvidor()는 내부적으로 LoadLibrary()함수를 호출
  - MS10-092 취약점을 이용함.

소개

- 32비트 윈도우일 경우 수행 순서

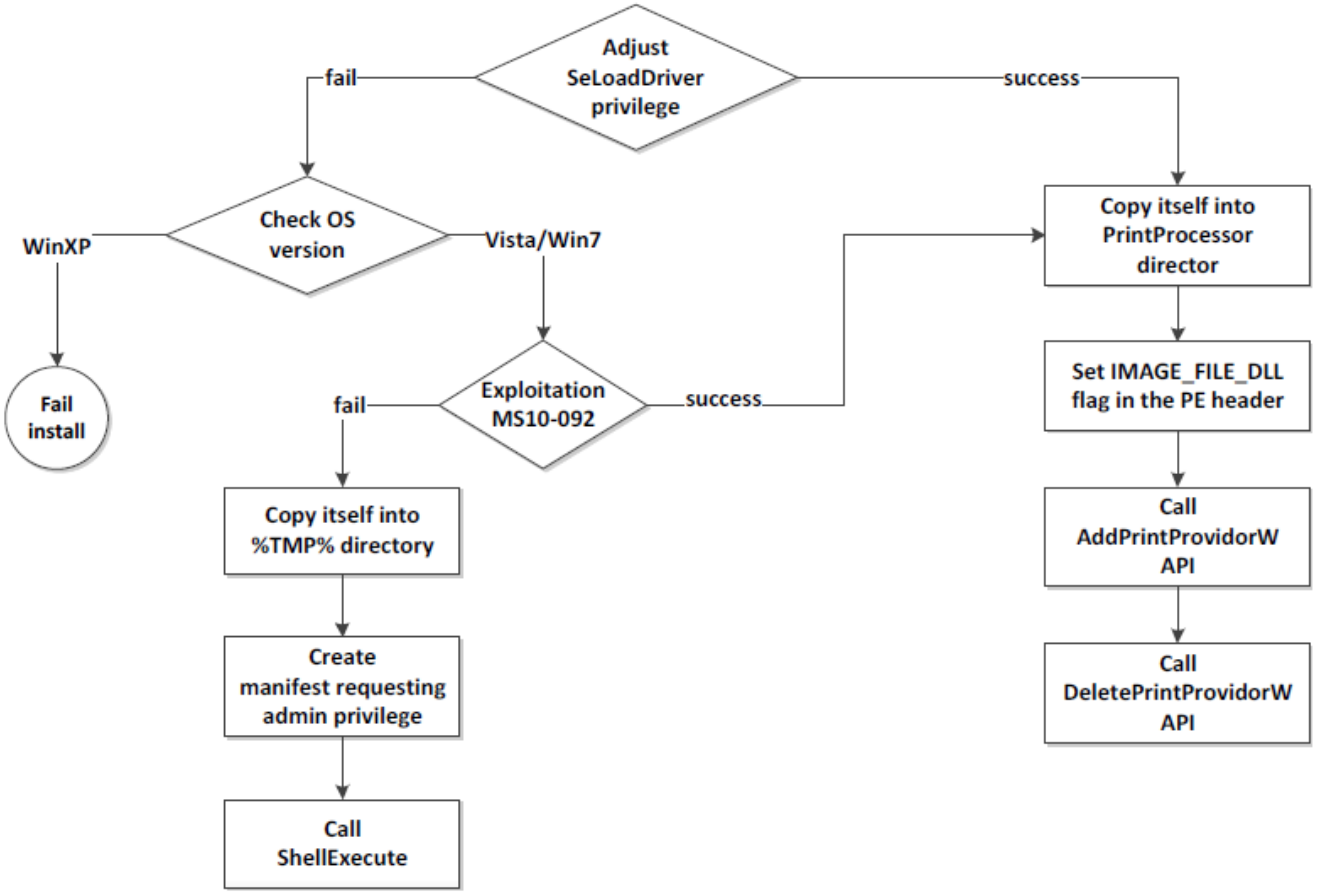
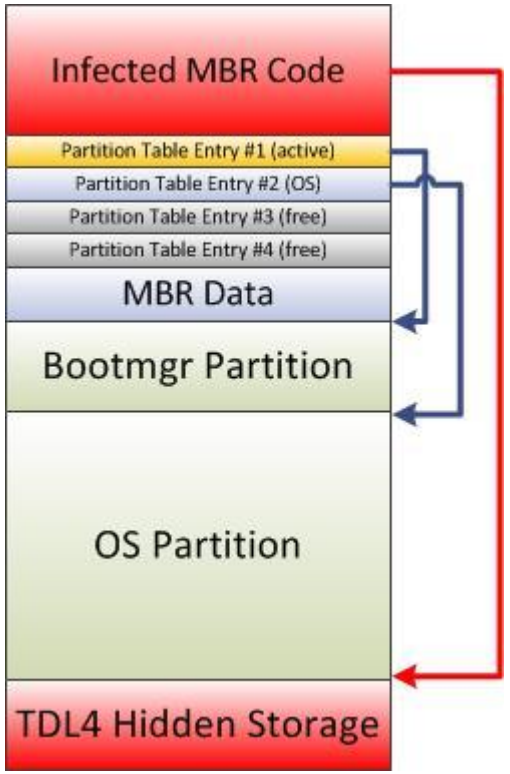


Figure 12 – The Algorithm of Infecting x86 System

부트킷 파일 은닉 기술

- MBR Overwriting
  - 운영체제가 로드되기 전에 접근하기 위해 파티션 테이블에는 별도의 수정 업이 MBR 코드를 덮어 씌움
- 스토리지에 페이로드를 저장
- 보호 기술
  - 저수준 후킹으로 자신을 보호함
  - RC4 스트림 사이퍼를 사용함
    - ✓ TDL3 - "tdl"을 키로 사용함
    - ✓ TDL4 – 암호화할 섹터 블럭의 32비트 LBA 값



부트킷 파일 은닉 기술

- 일반적인 구조는 TDL3와 유사함
  - 하드 드라이브 맨 끝의 잉여 영역에 부트킷을 위치함
- TDL3와 TDL4는 언파티션드 영역을 찾는 방식이 다름
  - TDL3 – MBR 정보를 활용
  - TDL4 – ZwDeviceIoControlFile()
    - ✓ IOCTL_CODE 0x4D014
    - ✓ IOCTL_SCSI_PASS_THROUGH_DIRECT

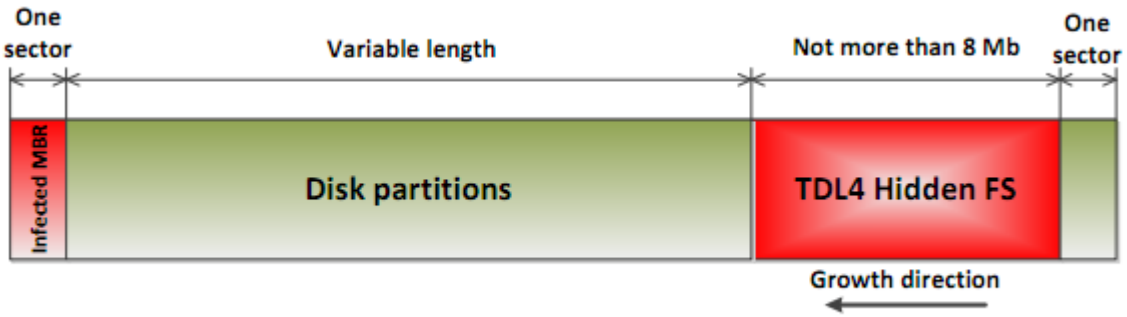


Figure 20 – Location of the Hidden File System on Disk

부트킷 파일 은닉 기술

```
typedef struct _TDL4_FS_BLOCK
{
	// Signature of the block
	// DC - root directory
	// FC - block with file data
	// NC - free bock
	WORD Signature;
	// Size of data in block
	WORD SizeofDataInBlock;
	// Offset of the next block relative to file system start
	WORD NextBlockOffset;
	// File table or file data
	BYTE Data[506];
}TDL4_FS_BLOCK, *PTDL4_FS_BLOCK;
```

Here is the format of the root directory:

```
typedef struct _TDL4_FS_ROOT_DIRECTORY
{
	// Signature of the block
	// DC - root directory
	WORD Signature;
	// Set to zero
	DWORD Reserved;
	// Array of entries corresponding to files in FS
	TDL4_FS_FILE_ENTRY FileTable[15];
}TDL4_FS_ROOT_DIRECTORY, *PTDL4_FS_ROOT_DIRECTORY;
```

```
typedef struct _TDL4_FS_FILE_ENTRY
{
	// File name - null terminated string
	char FileName[16];
	// Offset from beginning of the file system to file
	DWORD FileBlockOffset;
	// Reserved
	DWORD dwFileSize;
	// Time and Date of file creation
	FILETIME CreateTime;
}TDL4_FS_FILE_ENTRY, *PTDL4_FS_FILE_ENTRY;
```

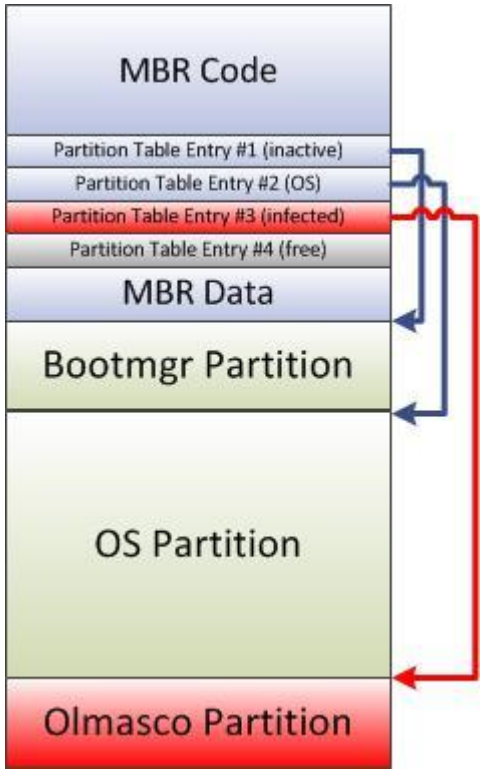
소개

- **2011년 초에 등장한 새로운 부트킷**
  - Win32/Olmasco or MaxSS로 불림
  - TDL4를 기반으로 더 발전된 기술을 적용하였음.
  - 두 루트킷이 유사한 이유는 두가지를 꼽고 있음
    - ✓ TDL4 개발한 팀이 새롭게 개발하였음
    - ✓ TDL4 개발자가 소스를 판매했거나 부트킷 빌더를 다른 범죄 조직에 판매
  - TDL4와 동일하게 대상 시스템을 점령하여 C & C 서버를 구축함
  - 프로세스 에러 발생 시 악성코드는 에러 리포트를 발송함.
    - ✓ 피드백을 받아 패치하기 위해 사용함

부트킷 파일 은닉 기술

- **디스크의 파티션 테이블 조작**
  - MBR 코드에 별도의 패치를 수행하지 않음.
  - 비어있는 파티션 테이블 엔트리를 찾아서 새로운 파티션을 생성함.
    - ✓ 하드 디스크의 마지막 잉여 공간에 파티션을 생성
  - 페이로드와 설정정보를 저장함
  - VBR에 NTFS 파티션인 것처럼 명시하여 탐지 우회를 수행함.

```
0000  EB 52 90 4E 54 46 53 20  20 20 20 00 02 08 00 00  ьRPNTFS    .....
0010  00 00 00 00 00 F8 00 00  3F 00 FF 00 53 AC FF 00  .....°..?. .Sм .
0020  00 00 00 00 80 00 80 00  9C 53 00 00 00 00 00 00  ....A.A.bS......
0030  39 05 00 00 00 00 00 00  00 02 00 00 00 00 00 00  9...............
0040  F6 00 00 00 01 00 00 00  8B 62 C8 E9 B8 4B 28 D5  Ў.......ЛbLш¬K(-
0050  00 00 00 00 FA 31 C0 8E  D0 BC 00 7C FB 0E 1F 0E  ....·1LO¦-.|v...
0060  07 66 60 88 16 00 7E C6  06 04 7E 1E B4 48 BE 04  .f`И..~¦..~.+H-.
0070  7E CD 13 B0 50 0F 82 71  01 83 2E 13 04 14 A1 13  ~=.-P.Bq.Г....б.
```



부트킷 파일 은닉 기술

- **스토리지 구조**
  - TDL4의 스키마를 사용하면서 기능을 몇가지 추가함
    - ✓ 파일과 폴더를 가지는 트리 구조
    - ✓ 컴포넌트의 손상 여부를 체크하여 무결성을 확인함
    - ✓ 내부 파일 시스템 구조의 관리 능력을 높임
  - 루트 디렉터리는 '₩' 심볼
  - 최초 VBR에선 '₩boot'를 로드함.

```
seg000:01EA halt:                                       ; CODE XREF: seg000:0075↑j
seg000:01EA                                             ; sub_B0+35↑j ...
seg000:01EA                 hlt
seg000:01EA sub_191         endp
seg000:01EA
seg000:01EB ; ---------------------------------------------------------------------------
seg000:01EB                 jmp     short halt
seg000:01EB ; ---------------------------------------------------------------------------
seg000:01ED aBoot           db '\boot',0                ; DATA XREF: seg000:008E↑r
seg000:01F3                 db    0
```

  - 무결성 체크는 CRC32 체크섬을 활용함.
    - ✓ 체크섬이 틀리면 파일을 제거함

부트킷 파일 은닉 기술

- **스토리지 구조**
  - 효율적인 파일 시스템 관리 및 무결성을 돕기 위해 다음 기능을 포함함
    - ✓ $bad
      - 손상된 데이터가 위치한 섹터 정보를 가짐
    - ✓ $bitmap
      - 파일과 디렉터리의 저장된 위치 정보를 가짐
  - NTFS와 상당히 유사한 구조로 되어 있어서 보안 소프트웨어가 잘못 판단하게 함

소개

- **2011년 초에 64비트 ZeroAccess(Win32/Sirefef) 수정 버전이 등장함.**
  - TDL4처럼 부트킷의 기능을 수행하진 않음.
  - 64비트 시스템을 대상으로 하였지만 커널 모드 드라이버는 없음.
    - ✓ x86 시스템을 대상으로 하는 드라이버만 포함
    - ✓ 이에 32비트와 64비트의 운영체제 버전에 따라 감염 방식이 다름

소개

- **루트킷 로드 방식**
  - x86 시스템의 ZeroAccess는 TDL3 루트킷와 유사함.
    - ✓ 커널모드 부팅 시점에 로드되는 특정 드라이버를 완벽하게 자신의 코드로 덮어씌움.
      - 부팅 시점에 악성 드라이버가 로드됨
    - ✓ 백신으로부터 자신을 보호하기 위해 Storage device driver stack을 후킹함
  - 64비트는 커널 모드 드라이버가 없기 때문에 다른 방법을 사용함
    - ✓ consrv.dll을 "systemroot₩system32"디렉터리에 드롭(drop)
    - ✓ Windows Subsystem으로 등록하여 세션 관리 서브시스템(smss.exe)에서 시스템 구동 시점에 로드함
      - "HKLM₩SYSTEM₩CurrentControlSet₩Control₩Session Manager₩SubSystems"
      - 레지스트리 키가 변경되지 않으면 자시자신을 제거함.

부트킷 파일 은닉 기술

- **사용자 정의 파일 시스템을 사용함.**
  - 페이로드와 설정 정보를 가지고 있음
  - 히든 스토리지의 컨텐츠는 운영체제 파일 시스템 내의 특정 파일로 저장
    - ✓ "WindowsDir₩$NtUninstallKB_BotId$₩BotId" 디렉터리를 생성
  - 악성코드는 파일 시스템 필터 드라이버 처럼 동작함
    - ✓ 페이로드에 읽기/쓰기 명령을 후킹함
  - 접근 시 상황에 따라 다르게 행동함
    - ✓ 루트킷이 접근 시 – 루트킷 코드를 수행함.
    - ✓ 시스템이 접근 시 – 백업한 원본 코드로 변경함

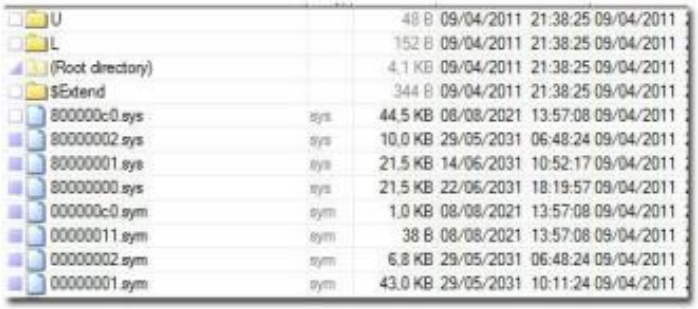
부트킷 파일 은닉 기술

- 히든 볼륨은 암호화되어 있음
  - 루트킷이 읽기/쓰기 명령을 수행할 때마다 암호화/복호화를 수행
  - 가상 볼륨은 systemroot₩system32₩config₩<random file name>에 파일을 위치하고 내부에 저장됨
    - ✓ 파일은 언제나 암호화 되어 있음
  - RC4 스트림 사이퍼를 사용하며, 128비트 키를 이용함
    - ✓ 섹터 단위 암호화 사용

부트킷 파일 은닉 기술

```
crypt_sector:                   ;
push    offset RC4_key  ; RC4 key
lea     esi, [ebp+Sbox] ; RC4 S-Box base address
call    Generate_RC4_Table ; Generate RC4 S-Box
push    SectorSize
xor     eax, eax
push    edi
call    CryptBuffer     ; Encrypt/Decrypt sector
mov     eax, SectorSize
add     edi, eax
sub     ebx, eax
jnz     short crypt_sector ;
```

Rootkit driver I/O encryption/decryption



Rootkit file system decrypted

소개

- **2011년에 새로운 트렌드**
  - VBR과 부트스트랩 코드를 수정함.
    - ✓ Win32/Rovnix, Win32/Carberp
  - 보안 기술과 안티바이러스 프로그램을 회피하기 위해 여러 기술을 사용함.
  - Rovnix는 VBR bootkits 빌더로 판매되고 있음.
  - Carberp는 지속적으로 부트킷의 기능을 추가하며 여름이 끝날 때쯤부터 판매를 시작함
    - ✓ 가장 위험한 뱅킹 트로이 목마 중 하나

결론

- 결론
- 해야 할 일

결론

- 루트킷은 커널 정보 조작 또는 설정 조작을 수행함
  - 구동을 위해서는 운영체제 커널이 로드된 후 드라이버를 로드함
  - 보안 소프트웨어 로드 후 실행되는게 대부분이기 때문에 탐지될 확률이 상당히 높음
- 부트킷은 부트로더를 조작함
  - 커널이 로드되기 전에 루트킷의 행위가 실시될 수 있음
  - 보안 소프트웨어보다 선행되서 로드되기 때문에 탐지 기법을 우회할 수 있음
    - ✓ 드라이버 읽기/쓰기 명령에 대해 오리지널 코드를 제공하여 은닉하는 기법 등
- 자체 파일 시스템을 가지거나 파일을 은닉함
  - 시스템 비사용 영역에 파티션을 생성하여 관리
    - ✓ 파티션 암호화
    - ✓ 사용자 정의 파티션
  - 시스템 파티션 영역에 파일을 위치하고 자신을 은닉
    - ✓ 디스크 읽기/쓰기 함수 후킹

부트킷 대응 방법?

- 디스크 포렌식
  - 파티션 되지 않은 영역의 분석이 필요함.
    - ✓ 데이터 존재 여부 판단
    - ✓ 데이터 암호화 여부 판단
  - 라이브 분석 가능 시 라이브 상태의 파일 해시와 디스크 이미지의 파일 해시 비교
- 메모리 포렌식
  - 메모리 상에 존재하는 드라이버의 무결성 검사
  - 바이오스 영역 분석

참고 문헌

- Bootkit Threat Evolution in 2011
  - ESET
  - http://blog.eset.com/2012/01/03/bootkit-threat-evolution-in-2011-2?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+eset%2Fblog+%28ESET+ThreatBlog%29
- The Evolution of TDL: Conquering x64
  - ESET
  - http://go.eset.com/us/resources/white-papers/The_Evolution_of_TDL.pdf
- ZeroAccess – an advanced kernel mode rootkit
  - Prevx
  - http://www.prevxresearch.com/zeroaccess_analysis.pdf

Q & A