

Bootkit Threat Evolution in 2011

n0fate

Feedbeef.blogspot.com





1. 소개

2. 주요 루트킷 의 파일 은닉 기술

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

3. 결론

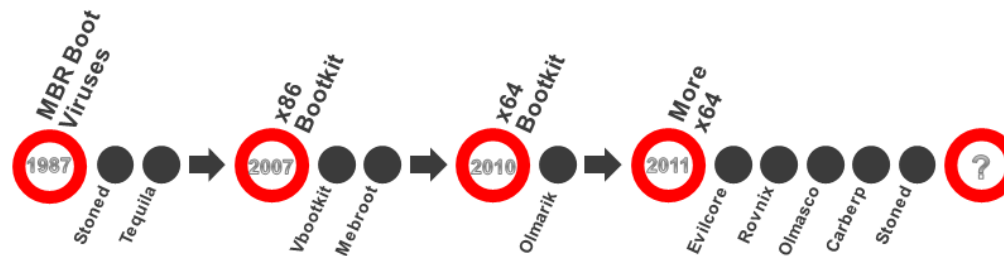
소개

- 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.
 - ✓ AddPrintProvider()는 내부적으로 LoadLibrary()함수를 호출
- MS10-092 취약점을 이용함.



소개

- 32비트 윈도우일 경우 수행 순서

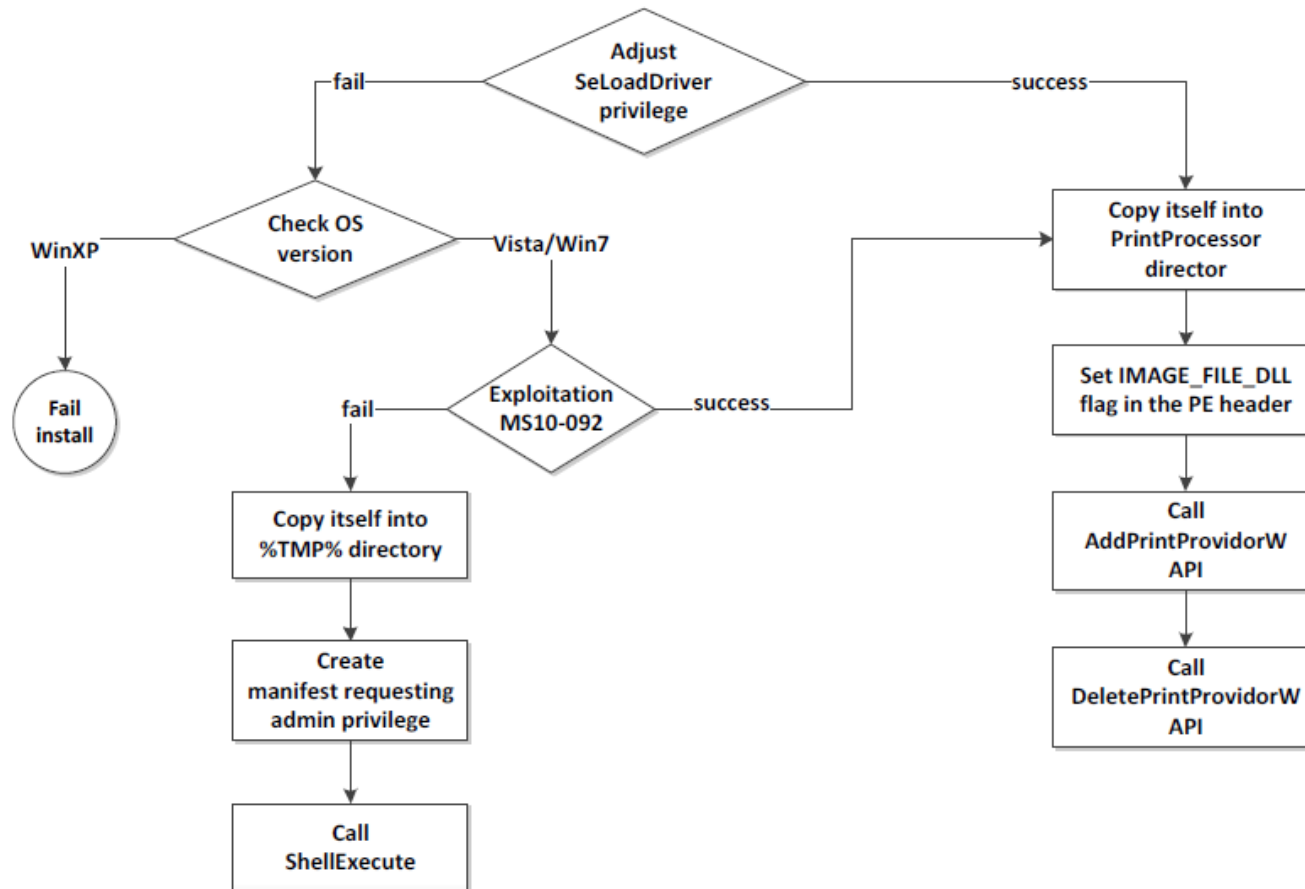
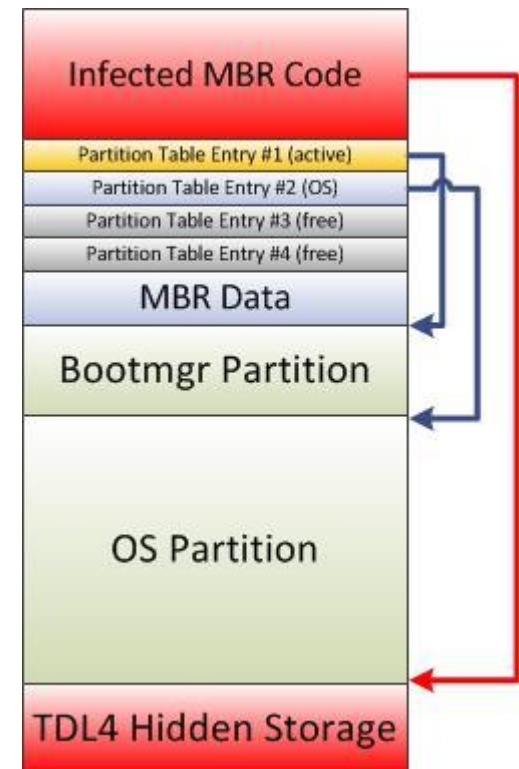


Figure 12 – The Algorithm of Infecting x86 System



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 블록의 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

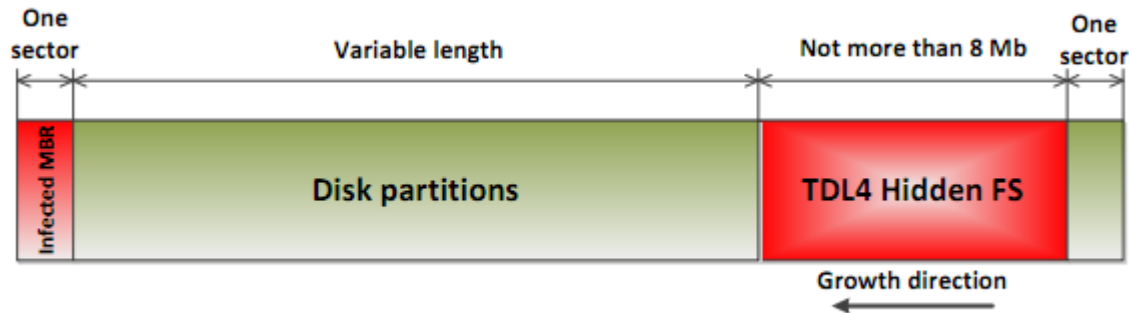


Figure 20 – Location of the Hidden File System on Disk



부트킷 파일 은닉 기술

```
typedef struct _TDL4_FS_BLOCK
{
    // Signature of the block
    // DC - root directory
    // FC - block with file data
    // NC - free block
    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



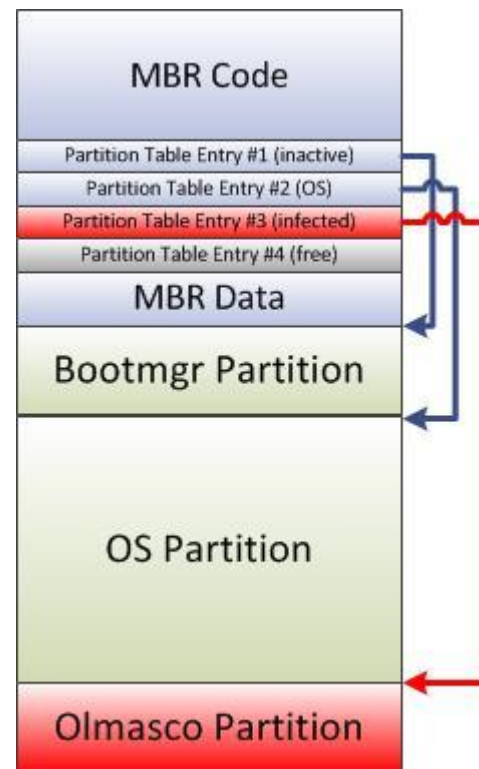
부트킷 파일 은닉 기술

▪ 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 00 F8 00 00 3F 00 FF 00 53 AC FF 00  .....°...?. .SM .
0020 00 00 00 00 00 00 00 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  ѳ.....ЛбLщ-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 'W' 심볼
- 최초 VBR에선 'Wboot'를 로드함.
- 무결성 체크는 CRC32 체크섬을 활용함.
 - ✓ 체크섬이 틀리면 파일을 제거함

```
seg000:01EA halt: ; CODE XREF: seg000:0075↑j
seg000:01EA ; sub_00+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
```



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 루트킷과 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 콘텐츠는 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

3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 10

```

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

U	48 B	09/04/2011	21:38:25	09/04/2011
L	152 B	09/04/2011	21:38:25	09/04/2011
(Root directory)	4.1 KB	09/04/2011	21:38:25	09/04/2011
\$Extend	344 B	09/04/2011	21:38:25	09/04/2011
8000000c0.sys	44.5 KB	08/08/2021	13:57:08	09/04/2011
800000002.sys	10.0 KB	29/05/2031	06:48:24	09/04/2011
800000001.sys	21.5 KB	14/06/2031	10:52:17	09/04/2011
800000000.sys	21.5 KB	22/05/2031	18:19:57	09/04/2011
0000000c0.sym	1.0 KB	08/08/2021	13:57:08	09/04/2011
000000011.sym	38 B	08/08/2021	13:57:08	09/04/2011
000000002.sym	6.8 KB	29/05/2031	06:48:24	09/04/2011
000000001.sym	43.0 KB	29/05/2031	10:11:24	09/04/2011

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



- 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



- 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



- 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