Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.1, Feb. 2016 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2016.26.1.117

악성코드 분석을 위한 Emulab 활용 방안 연구*

이 만 희,[1†] 석 우 진[2‡]
[1]한남대학교, [2]한국과학기술정보연구원

Research on Utilizing Emulab for Malware Analysis*

Man-hee Lee,[1†] Woo-jin Seok[2‡]
[1]Hannam University, [2]KISTI

요 약

빠르게 증가하는 악성코드를 효율적으로 분석하기 위해 가상화 환경이 많이 사용되고 있다. 하지만 이를 인지한 악성코드 제작자들은 가상화 환경 탐지 기술을 이용하여 악성코드가 가상화 환경에서 구동되는 것을 판단하면 악성 행위를 수행하지 않는 등의 분석 회피 기술을 적용하고 있다. 분석 회피 기술을 무력화하기 위한 연구도 계속되고 있지만 몇 가지 가상화 환경 탐지 기술로써 악성코드 분석은 상당히 저해를 받는다. 미국 Utah 대학에서 개발한 Emulab은 실제 시스템을 연구자가 원하는 대로 실시간으로 할당할 수 있다. 본 연구에서는 이 Emulab을 악성코드 분석에 어떻게 활용할 수 있는지 알아보고 그 방안을 제시한다.

ABSTRACT

Virtual environment is widely used for analyzing malware which is increasing very rapidly. However, knowing this trend, hackers are adopting virtual environment detection techniques for malware to kill itself or stop malicious behaviors when detecting virtual environments. Various research is going on in order to thwart any efforts to utilize anti-virtualization techniques, but until now several techniques can evade most of well known virtual environments, making malware analysis very difficult. Emulab developed by Utah University assigns real systems and networks as researchers want in realtime. This research seeks how to use Emulab for malware analysis.

Keywords: Cyber security, Emulab, Virtualization, Malware analysis

I. 서 론

악성코드의 피해가 날로 심각해지고 있다. IDC와 싱가포르국립대의 조사에 따르면 아시아지역 2014년에 악성코드에 감염된 PC를 치료 및 복구하기 위해 소요된 비용이 개인사용자 27조원, 기업 525조라고 한다[1]. 이는 전 세계 피해액의 40%이며, 악성코드들이 일으키는 사이버 공격에 의한 피해액을 포함하지 않은 것을 고려하면 악성코드로 인한 피해는 이보다 훨씬 클 것으로 사료된다. 더욱이 2014년에 전 세계 악성코드는 약 72% 증가하였고, 감염률은 38%이 증가하는 등 악성코드로 인한 문제는 더욱 심화될 것으로 예상된다[2][3].

빠르게 증가하는 악성코드에 대응하기 위해서는 자동화 분석 기술이 반드시 필요하다. 자동화 분석기술은 악성코드를 실행하지 않고 분석하는 정적 분석 방법과 실험환경 상에서 악성코드를 실행하고 그 행위를 관찰하는 동적분석 방법으로 나뉜다. 최근 악성코드의 분석을 방해하기 위한 수단으로 패킹, 암호화, 난독화 등의 기술이 적용되고 있어서 점차 정적 분석이 어려워지고 있다.

Received(10. 07. 2015), Modified(1st: 11. 17. 2015, 2nd: 12. 07. 2015), Accepted(12. 07. 2015)
* 본 논문은 2015년도 한남대학교 교비연구비에 의해 지원되었음.
† 주저자, manheelee@hnu.kr
‡ 교신저자, wjseok@kisti.re.kr(Corresponding author)

동적분석은 악성코드를 실행하며 그 행위를 모니터링하여 분석에 활용하므로 앞서 설명한 정적분석의 단점을 극복할 수 있다. 최근 동적분석의 동향은 가상환경을 이용한 분석이라고 할 수 있다. 과거 인스트럭션 레벨 에뮬레이션 방법의 분석 방법은 속도가 매우 느려 실제적인 가상화 환경을 구축하기 힘들었으나, Intel의 VT-x, AMD의 AMD-V로 알려있는 하드웨어 가상화 기술이 프로세서에 탑재되기 시작하면서 VMWare나 VirtualBox 등의 가상화 환경이 거의 실제 시스템 속도에 근접하고 있다[4][5][6][7]. 높은 성능의 가상환경과 더불어 가상환경을 사용할 경우, 악성코드 실행 정지·재실행, 가상환경 시스템 이미지 복사·복구, 네트워크 구성, 실행코드 메모리 접근 등 악성코드 분석에 유용한 기능들이 제공되므로 악성코드 분석에 많이 활용되고 있다.

가상환경이 악성코드 분석에 일반적으로 사용됨에 따라 악성코드 제작자들은 악성코드가 가상환경에서 동작하는지 인지하면 프로그램을 종료하거나 악성행위를 숨기도록 악성코드를 제작하고 있다. 이로 인해 가상화 기술을 이용해 악성코드를 분석하는 시도는 큰 어려움에 봉착하였다.

이 문제를 해결하기 악성코드가 가상환경을 탐지하려고 할 때 이를 우회하는 방법에 관한 연구가 시도되었다[8][9][10]. 이 방법들은 가상환경 탐지 시도 시 응답 값에 실제 시스템이 리턴할 내용으로 바꿔서 응답하는 형식을 취한다. 하지만 한두 가지 방법을 간단히 사용하는 악성코드는 우회할 수 있으나 다양한 방법을 사용하거나 새롭게 나온 가상환경 탐지 기법에는 완전히 우회하기는 힘들다.

이 문제를 완전히 해결하기 위한 방편으로 실제 시스템을 보조적으로 사용하는 방안이 제시되고 있다. 본 연구에서는 이 실제 시스템으로 Utah 대학교에서 개발되어 전 세계 많은 연구자들이 활용하고 있는 Emulab을 사용하는 것을 제안한다[11]. 한국에서도 KREONet에 48개 노드의 Emulab을 구축하고 교육 및 연구 분야에 활용중이다. Emulab의 장점은 연구자가 필요한 개수의 노드, 네트워크 환경, 운영체제 등을 선택하면 on-demand로 실제 시스템이 할당되어 연구에 사용된다[12].

Emulab은 네트워크 분야에 아디이어 검증 등에 주로 사용되었지만, 보안 분야에서도 적극적으로 사용되고 있다[13]. 하지만 주로 악성코드 분석 자체보다는 악성코드가 생성하는 트래픽 및 네트워크 행위에 집중되어 있었다. 본 논문에서는 먼저 Emulab을 악성코드 분석 연구에 사용하기 적합한지를 검증한 후, Emulab을 악성코드 분석에 어떻게 활용하는지에 대한 방안을 제시한다.

본 연구의 주요 연구 결과로 Emulab은 악성코드들이 탐지하는 다양한 가상환경 탐지 방안으로 전혀 탐지 않는다는 것을 확인하였다. 또한 악성코드가 실행되어 시스템 파일 파괴, MBR 삭제 등으로 인해 정상적으로 해당 시스템의 하드디스크를 접근하기 어려운 상황에서도 하드디스크의 전체 이미지를 추출하여 원격지로 다운받아 추후 분석이 가능하였다. 본 연구는 네트워크 운용 및 악성코드의 네트워크 행위 등에 주로 사용되었던 Emulab을 악성코드 분석에 활용할 수 있는지를 검증하고 이 방안을 제시하는 연구로써 그 의미가 있다.

본 논문의 구성은 다음과 같다. 2장에서는 가상화 탐지 기술에 대해 소개하고 Emulab이 이 기술을 이용해 탐지되는지 확인한다. 3장에서는 Emulab을 이용한 악성코드 분석 방안에 대해서 논의한 후, 4장에서 결론을 맺는다.

II. Emulab 가상환경 탐지 기술 검증

2.1 가상환경 탐지 기술

Emulab이 가상환경에서 구동되지 않는 악성코드들을 구동시킬 수 있는 실제 환경이지만 아직까지 가상화 탐지 기술에 대해 검증된 바는 없었다. 본 장에서는 가상화 탐지 기술을 간단히 소개하고 해당 가상화 탐지 기술을 적용하여 Emulab이 악성코드 분석 환경에 적합한지 검증한다.

가상환경을 탐지하는 방법은 크게 3가지로 분류할 수 있다. 첫째, 가상환경은 실제 시스템보다 속도의 차이가 존재하므로 이 차이를 이용하여 가상환경을 탐지할 수 있다. 둘째, 가상환경 상에서 운영체제를 구동할 경우, 실제 시스템 상에서 구동되는 운영체제와 시스템 정보 등이 다르다. 이런 정보를 이용하여 실제 시스템에서 구동되는 일반적인 값들과 다를 때 가상환경을 탐지할 수 있다. 마지막으로는 특별한 가상환경을 사용할 경우에 특징적으로 나타나는 값들을 이용하여 특정 가상환경을 탐지할 수 있다.

첫 번째 성능차이를 이용한 가상환경 탐지 기술에서 대표적인 방법이 rdtsc 명령어를 이용하는 것이다[14]. rdtsc 명령어는 time stamp counter (TSC)라는 64-bit 레지스터의 값을 읽어와

EDX:EAX 레지스터에 값을 저장한다. TSC는 컴퓨터가 리셋된 후 누적 클락 수를 저장하며 Pentium 이후의 모든 x86 프로세서에서 지원한다. rdtsc 명령어를 연속적으로 수행하고 그 차이를 비교하면 최근 프로세서들은 대부분 100 미만의 값이 나온다. 하지만 가상환경에서 구동할 경우, 1000을 넘는 등 비정상적으로 큰 값을 보인다.

두 번째 가상환경의 운영체제와 실제 운영체제의 차이점을 이용한 대표적인 방법이 Interrupt Descriptor Table (IDT)의 주소를 이용하는 것이다[15][16]. IDT는 인터럽트가 발생할 때 이를 처리할 인터럽트 처리 루틴의 주소를 저장한다. 가상환경에서 구동되는 Guest 운영체제와 실제 시스템에서 구동되는 Host 운영체제는 각각 다른 IDT를 가져야 하므로 IDT의 저장 주소가 서로 다르게 된다. IDT의 시작 주소를 저장하는 IDTR 레지스터의 값이 일반적인 값과 다르면 가상환경이라고 할 수 있다.

세 번째 가상환경의 특징적인 환경 변수 등을 이용한 탐지법의 대표적인 예는 레지스트리 정보를 이용하는 것이다[17]. 악성코드 분석에 사용될 만한 가상환경의 숫자가 한정적이므로 이런 가상환경의 특징을 테스트해보면 악성코드가 구동되는 환경이 Guest 운영체제인지 Host 운영체제인지 알 수 있다. 예를 들면 VirtualBox에서 구동되는 윈도우 Guest 운영체제의 레지스트리 정보 중에서 현재 하드디스크 콘트롤러의 정보를 담고 있는 레지스트리인 /HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Disk\Enum 은 VirtualBox 상에서 구동되는지 알 수 있는 DiskVBoX_HARDDISK 라는 정보가 저장되어 있다.

2.2 Emulab 가상환경 탐지 기술 검증

앞서 설명한 방법 외에도 많은 방법들로 가상환경을 탐지하고 있으며 새로운 방법들이 계속해서 고안되고 있으므로 이를 모두 나열하는 것은 본 연구의 범위를 벗어나는 것이다. 하지만 가능하면 많은 가상화 탐지 기술을 Emulab에 적용해야 하므로 본 연구에서 선택한 도구는 Pafish (Paranoid fish) 이다[18]. Pafish는 악성코드들이 주로 사용하는 가상환경 또는 디버깅 환경을 탐지 및 우회 기술 테스트를 제공하는 도구이다. Pafish가 이제까지 알려진 모든 가상환경 탐지 방법을 구현한 것이 아니므로 Pafish를 이용한 검증이 완벽하다고 할 수는 없다. 하지만 Pafish는 악성코드에서 주로 사용되는 다양한 기술들을 적용하고 있으므로 대부분의 가상환경 탐지 기술이 적용된 악성코드의 행동과 유사하다고 볼 수 있다.

Table 1은 Pafish를 가상환경 내에서 수행한 결과와 Emulab에서 수행한 결과를 비교한다. 검사 종류 별로 1개~17개의 세부 검사 항목이 존재한다. 가상환경에서 실행한 Pafish의 경우 CPU 성능에서 3개 항목이 탐지 되었고, 일반 가상환경 검사에 5개, VMWare 항목에서는 7개 모두 탐지된 것을 알 수 있다. 이 결과로 구동 환경이 VMWare인 것을 알 수 있다. 이와 반대로 Emulab은 Pafish의 어떤 검사에도 검출되지 않았다. 즉, 가상화 탐지 기능이 탑재된 악성코드도 Emulab을 실제 시스템으로 인지할 것으로 판단된다.

Table 1. Pafish analysis for Emulab

Test category	No of Tests	VMWare Detections	Emulab Detections
Debugger	1	0	0
CPU performance	4	3	0
Generic Sandbox	10	5	0
Hook	1	0	0
Sandboxie	1	0	0
Wine	2	0	0
VirtualBox	17	0	0
VMWare	7	7	0
QEMU	2	0	0
Cuckoo	1	0	0

III. Emulab을 활용한 악성코드 분석

3.1 Emulab을 활용한 악성코드 분석 절차

만약 분석 대상 악성코드에 가상화 환경 우회 기능이 탑재되어 있는지 알 수 있다면 그 결과에 따라 Emulab 또는 가상화 환경에서 분석을 진행하면 될 것이다. 현재까지 악성코드에 가상환경 탐지 기능이 있는지 확인하는 방법은 분석가의 수작업에 의존할 수밖에 없으며 그 작업은 많은 시간을 요구한다. 최근 이를 자동화하기 위한 연구들이 진행되고 있으며[19][20][21][22], 2014년과 2015년에 발표된 BareCloud와 MalGene이 최신 연구 동향을 대표

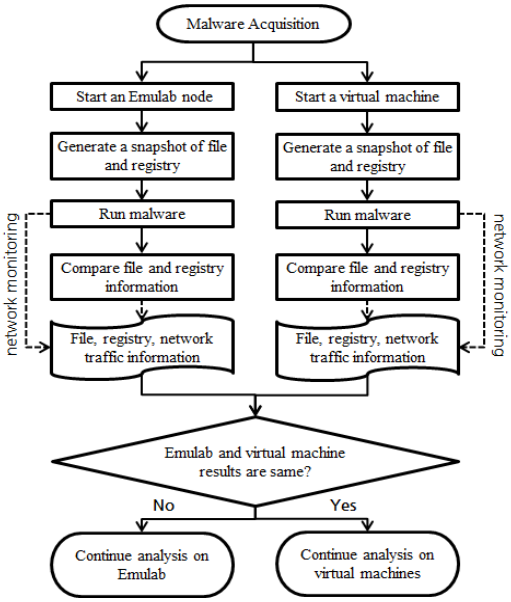


Fig. 1. Decision on malware analysis platform

한다고 할 수 있다. BareCloud는 가상화 시스템과 Bare-metal이라고 불리는 일반 시스템에서의 시스템 호출 순서를 분석하여 그 차이를 이용해 탐지하는 반면, MalGene은 가상화 우회를 위한 시스템 호출 순서에 대한 시그너쳐를 미리 생성한 후 분석대상의 시스템 호출 순서에서 해당 시그너쳐를 찾는 방법으로 탐지한다[21][22]. 하지만 아직 도구 수준으로 공개된 바는 없다.

Emulab을 활용한 악성코드 분석은 Emulab 시스템을 Bare-metal처럼 사용하므로 BareCloud와 접근법이 비슷하다. 악성코드 분석절차는 먼저 수집한 분석대상 악성코드를 가상화 시스템과 Emulab 시스템 모두에서 분석한다(Fig. 1). 가상화 시스템의 분석 결과와 Emulab에서의 분석 결과를 비교하여 가상화 시스템에서는 발현되지 않았던 행위가 Emulab에서 발현되었는지 확인한다. 두 시스템에서의 행위가 완전히 같다면 분석이 편한 가상환경을 이용해서 계속 분석을 진행하고 같지 않다면 Emulab을 이용한 분석을 진행한다. Emulab과 가상화 환경에서의 결과가 같은지 판단하기 위해 본 연구에서 제안하는 비교 방법은 실제 시스템에서 비교적 쉽게 얻을 수 있는 다음의 정보를 이용한다.

- 파일 정보: 악성코드를 실행시키기 전과 후의 파일 시스템을 비교하여 어떤 파일이 추가, 변경, 삭제되었는지 확인. 악성코드들은 악성행위를 위한 새로운 파일을 생성하는 경우가 많음.
- 레지스트리 정보: 악성코드 실행 전후의 레지스트리 정보 비교. 많은 악성코드들은 감염 후 재부팅 시에도 동작하기 위한 레지스트리 정보 수정.
- 네트워크 정보: 악성코드 실행 중 생성되는 네트워크 트래픽 정보. 봇, 웜 등의 악성코드들은 추가 시스템 감염 및 공격을 위해 네트워크 행위가 필수.

많은 악성코드가 위 세 가지 행위를 단독 혹은 복합적으로 사용하므로 가상환경 우회 기능이 탑재된 악성코드는 상기 행위들이 발현되지 않지만 Emulab에서는 발현될 것이다. 이전 연구들에서 비교 대상으로 주로 사용되었던 시스템 호출 정보를 사용하지 않는 이유는 [21]에서 밝혀진 바와 마찬가지로 매우 복잡하고 긴 시스템 호출 정보를 통해 두 분석 결과가 가상화로 인해 달라졌다는 것을 알아내기 쉽지 않기 때문이다.

상기 방법론의 효용성 검증을 위해 본 연구에서는 ZeuS Builder를 이용하여 제우스 악성코드를 만들고 이 코드를 수정하여 가상화 탐지 기능, 레지스트리 정보 수정, 폴더 생성 기능 추가하였다. 또한, 수정된 제우스 봇은 사용자를 속이기 위해 실행파일명을 notepad.exe로 하고 실제로 마지막에 노트패드를 구동시킨다. 이 코드를 실행하여 가상환경에서 구동중인 것으로 판단되면 노트패드만 구동시키고 종료를 하고, 가상환경이 아니라면 레지스트리 정보 변경, 폴더 생성, 제우스 서버와 통신을 수행하면서 노트패드를 구동한다. 이 코드의 수행결과가 Table 2에 나타나있다. 레지스트리와 파일 변경 정보를 얻기 위해서 Regshot과 Process Monitor를 사용하였고, Wireshark을 통해 악성코드 실행 중에 접속한 네트워크 정보를 확인하였다. VMWare의 결과와 달리 Emulab에서의 HKLM\SOFTWARE\CBSTEST\ CBSTestW DSLogFile은 일반적으로 프로그램 실행시 접근되지 않는 레지스트리 정보라는 것을 알 수 있다. 파일 행위의 경우도 Emulab에서만 testfoler가 생성되는 것을 확인할 수 있다. 네트워크 정보도 VMWare에는 발현되지 않았던 제우스 봇의 행동인 config.bin파일을 내려받은 후 gate.php를 접속하는 트래픽이 캡처 되었다.

Table 2. Registry, file, network operation difference between VMWare and Emulab

Test category	VMWare	Emulab
Registry	▪ *prefix*+HRZR_PGYFRFFVBA: 00-···-01 ▪ *prefix*+{S38OS404-1Q43-42S2-9305-67QR0O28SP23}\rkcybere.rkr: 00-···-00 *prefix*: HKU\S-1-5-21-895307998-2392027894-2155765514-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\	▪ **HKLM\SOFTWARE\CBSTEST\CBSTestWDSLogFile: "C:\BVTBin\Tests\installpackage\cbstest\x86\CBSTestWDS.log"** **--> "Hello_bot"** ▪ *prefix*+HRZR_PGYFRFFVBA: 00-···-02 ▪ *prefix*+{S38OS404-1Q43-42S2-9305-67QR0O28SP23}\rkcybere.rkr: 00-···-00 prefix: HKU\S-1-5-21-2741640254-59048613-1012305304-1013\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\
File	… C:\Users\Ssong\Desktop C:\Windows\System32\sechost.dll C:\Windows\System32\sechost.dll C:\Windows\System32\imm32.dll C:\Windows\System32\imm32.dll …	… C:\Users\mhlee\Desktop C:\Windows\System32\sechost.dll C:\Windows\System32\sechost.dll **C:\Users\mhlee\Desktop\testfolder** C:\Windows\System32\cmd.exe C:\Windows\System32\apphelp.dll …
Network	GET / HTTP/1.1 HTTP/1.1 200 OK (text/html) GET /css/main_v20151110.css GET /search/css/2015/api_atcmp_0319.css HTTP/1.1 GET /loginv4/www.css?1 HTTP/1.1 …	GET /xampp/1/config.bin HTTP/1.1 HTTP/1.1 200 OK POST /xampp/1/gate.php HTTP/1.1 200 OK …

3.2 Emulab을 이용한 악성코드 피해 분석

Emulab은 가상화 시스템에서 정상 작동하지 않는 악성코드를 작동시킬 수 있는 실제 시스템을 대체하는 효과 외에도, 악성코드의 피해 분석 플랫폼으로써 실제 시스템보다 활용도가 훨씬 높다. 가상화 시스템에서 활성화 되지 않았던 악성행위가 운영체제 시스템 파일 변경 혹은 MBR (Master Boot Record) 삭제라면 이 악성코드를 실제 시스템에서 분석하는 것은 매우 어려워진다. 왜냐하면 악성코드 수행 즉시 파일 시스템 파괴 등이 일어나면 악성코드 행위를 파악하기 위해 수집한 정보 자체가 파괴되기 때문이다. 재부팅이 안 되는 경우가 많기 때문에 분석은 더욱 어려워진다. 실제 시스템에서는 해당 하드디스크를 물리적으로 추출하거나 부팅 가능한 디스크를 이용하여 부팅한 후 포렌식 도구를 동작시켜 악성코드가 입힌 피해를 분석할 수 있지만 그 절차가 길고 물리적으로 작업이 필요하므로 다량의 악성코드를 분석하기에는 어려운 접근법이라고 할 수 있다.

본 연구에서는 악성코드 피해 분석을 위해 손상된 하드디스크를 원격에서 접근하여 이미지 파일로 생성하는 절차를 제안하고 검증한다. 이 내용은 과거 Emulab 관련 연구에서는 사용되지 않았던 방법으로써 본 연구에서 최초로 시도하였고, Utah 대학 Emulab 운영자와 KREONet Emulab 관리자의 협조를 통해 검증에 성공하여 그 의미가 크다고 할 수 있다.

Emulab에서 손상된 하드디스크의 이미지를 추출하는 절차는 악성코드 분석 시스템 이름 확인, Emulab 원격 접속, 관리자 모드 활성화를 통한 해당 시스템 재부팅, 손상된 이미지 추출 단계를 거친다. 우선 악성코드를 분석하던 시스템은 악성코드의 공격으로 파일 시스템이 파괴되었고 시스템이 종료되었거나 재부팅 단계에서 부팅 실패 상태이며 이로 인해 해당 시스템으로의 원격 접속은 불가능한 상태인 것으로 가정한다. 피해 분석 첫 단계인 악성코드 분

석 시스템 이름 확인은 Emulab 웹페이지의 detail 탭에서 확인할 수 있으며 pcXX 형태로 되어 있다. 둘째, 해당 시스템이 아닌 일반 사용자 접속용 시스템으로 로그인한다. KREONet Emulab은 users.emulab.kreonet.net 으로 접속할 수 있다.

셋째, 관리자 모드 활성화를 통해 첫 단계에서 확인한 시스템을 재부팅한다. 관리자 모드 활성화는 "node_admin on pcXX" 명령을 수행하면 pcXX이 즉각 FreeBSD로 리부팅되며 해당 시스템으로 원격 접속이 가능하게 된다. Emulab은 수십~수백대의 시스템을 원격에서 자동으로 시스템을 켜고 끄며 다양한 운영체제 이미지를 동적으로 다운 받아 부팅할 수 있는 기능을 위한 관리자 모드를 제공한다. 관리자 모드는 특정 시스템을 네트워크 부팅시키면서 해당 시스템의 하드디스크를 사용하지 않고 램디스크를 사용해서 부팅할 수 있는 매우 가벼운 FreeBSD 운영체제로 부팅한다. 본 연구에서 이 모드를 사용하려고 시도한 이유는 악성코드가 해당 시스템의 하드디스크를 파괴하여 운영체제가 종료되지만 Emulab 관리 시스템 입장에서는 idle 제한 시간에 이르지 않았기 때문에 해당 시스템을 swap-out 하지 않은 상태이다. 즉, 악성코드가 변경한 하드디스크 내용이 아직 해당 시스템에 남아 있으므로 이를 추출할 가능성이 있을 것으로 예상했기 때문이다.

넷째, 해당 시스템에 ssh 접속하여 dd 명령어를 이용하여 하드디스크의 내용을 추출한다. 부팅된 운영체제에서 확인한 결과, 이전 윈도우 운영체제의 C 드라이브는 /dev/mfid0 디바이스 파일에 매핑되어 있지만 마운트 되어 있지 않다. 디스크의 이미지를 생성하기 위해서 dd 명령어를 사용한다. dd 명령어는 유닉스 계열 운영체제에서 디바이스 파일 형태로 매핑되어 있는 디스크 드라이브를 일반 파일처럼 읽고 쓸 수 있는 기능을 제공하여, 하드디스크의 부트 레코드 백업 및 전체 백업 등에 사용된다. 본 연구에서 사용한 명령은 "dd if=/dev/mfid0 of=/proj/DStesting12/images/pcXXnode.img bs=1M count=5200" 이다. 그 의미는 읽을 디바이스 파일은 /dev/mfid0로 사용하고 출력 파일을 pcXXnode.img로 사용하며, 블록 사이즈를 1M 바이트로 5200개를 읽으라는 것이다. 즉, /dev/mfid0에 매핑되어 있는 하드디스크에서 5.2GByte를 읽어서 이미지 파일을 생성하라는 것이다. KREONet Emulab에서는 성공적으로 이 명령이 수행되어 이미지 파일이 생성되며 이 파일을

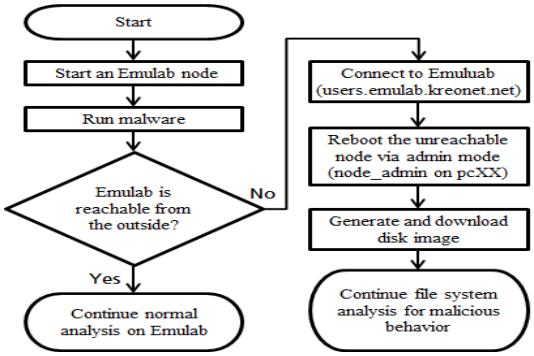


Fig. 2. Malware analysis process using Emulab

외부로 가져와서 img 타입의 파일을 읽는 소프트웨어에서 확인할 수 있었다. Fig. 2.는 상기 분석 절차를 도식화하였다.

본 연구에서 주안점을 가지고 확인한 내용은 악성코드가 MBR을 파괴하여 부팅뿐 아니라 정상적으로 파일시스템을 읽을 수 없는 상태에서 원래 하드디스크에 저장되었던 정보를 얻을 수 있을까 하는 것이다. 이 실험을 위해 사용한 악성 코드는 MBR 영역을 임의의 데이터로 덮어 쓴 후 시스템을 리부트 시킨다. 이 코드를 실행시킨 후 앞서 설명한 절차대로 하드디스크 이미지를 추출하고 Hex Editor를 이용하여 MBR 부분을 읽은 내용이 Fig. 3이다. MBR 부분이 파티션 정보와 부팅 코드가 아닌 아스키 값으로 "PRINCPES"가 반복되며 덮어 쓰인 것을 알 수 있다.

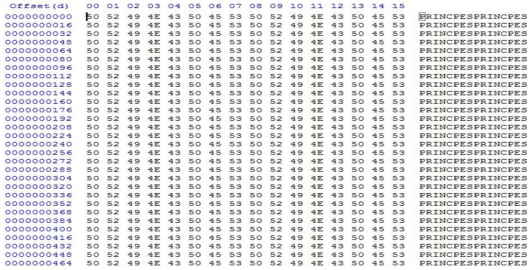


Fig. 3. MBR contents after malware activation

IV. 결 론

본 연구는 Emulab이 악성코드 분석 연구에 적합한지 그 가능성을 확인하고, 그 장단점을 파악하고 연구 가능할 시에 어떻게 연구를 수행하는지 방법을 제시하는데 그 목적이 있다.

악성코드 연구를 위한 Emulab의 가장 큰 장점은

가상환경이 아니므로 다양한 가상환경 탐지 기법을 탑재한 악성코드를 구동시킬 수 있다. 연구의 중요 결과로써 Emulab과 가상화 시스템을 활용하여 악성코드를 분석하는 절차와 방법을 제안하였다. 악성코드의 주요 행위 내용인 파일, 레지스트리, 네트워크 상태 정보를 비교하면 분석 대상 악성코드가 가상화 우회 기능이 탑재되었는지 판단할 수 있었다.

또 다른 주요 결과로써 파괴된 하드디스크의 이미지를 원격에서 추출하는 절차를 제안하고 검증하였다. 이 실험의 중요성은 악성코드 분석가가 운영하는 실제 시스템에서 악성코드를 수행하는 것보다 Emulab에서 수행하는 것이 훨씬 더 간단히 악성코드의 피해 정도를 파악할 수 있었다. 결과적으로 실제 시스템과 같은 구동환경을 제공하면서도 원격 접속 및 파괴된 하드디스크 복구가 가능한 Emulab을 가상화 시스템 기반 악성코드 분석에 보조적으로 사용한다면 훨씬 더 정확한 분석이 가능할 것으로 사료된다.

KREONet Emulab은 미국의 보안에 특화된 Emulab인 DETER Lab과 같이 악성코드 트래픽이 외부로 나가는 것을 막는 containment 기능, 이전 실험의 내용이 다음 실험에서 확인할 수 없도록 하는 zeroization 기능 등이 없다. 이로 인해 worm 같은 형태의 악성코드를 실험할 경우 Emulab이 악성코드를 전파하거나 외부 시스템을 공격하는 소스가 될 가능성이 있다. 이를 막기 위해 공격 트래픽을 막고 정상 트래픽만 접근을 허용하는 등의 기술 적용이 필요하다. 현재 Emulab을 활용한 악성코드 분석 방안의 고도화로 본 연구에서 제안한 가상화 시스템과 Emulab 시스템의 악성코드 행위 결과 자동비교 기능과 MBR 파괴 여부 자동 탐지 기능을 개발 중이다.

References

〔1〕 2014 Malware Damages, ITWORLD, http://www.itworld.co.kr/news/86687

〔2〕 Malware creation increasing, Trojans most popular attack, TrendMicro, http://blog.trendmicro.com/malware-creation-increasing-trojans-popular-attack/

〔3〕 New malware numbers jump sharply in 2014, DIGITAL TRENDS, http://www.digitaltrends.com/computing/pc-malware-rise-warn-security-firms/

〔4〕 R.L. Sites, A. Chernoff, M.B. Kirk, M.P. Marks, and S.G. Robinson, "Binary translation," Communications of the ACM, vol. 36, no. 2, pp. 69-81, Feb. 1993.

〔5〕 Joel Auslander, Matthai Philipose, Craig Chambers, Susan J. Eggers, Brian N. Bershad, "Fast, effective dynamic compilation," Proceedings of the ACM SIGPLAN 1996 conference on Programming language design and implementation, pp. 149-159, May 1996.

〔6〕 Intel Virtualization Technology, Intel Corp., http://www.intel.com/content /www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html

〔7〕 AMD Virtualization, AMD, http://www.amd.com/en-us/solutions/servers/virtualization

〔8〕 P. Ferrie, "Attacks on virtual machine emulators," Symantec Security Response, Dec. 2006

〔9〕 On the Cutting Edge: Thwarting Virtual Machine Detection, http:// handlers.sans.org/tliston/ThwartingVMDetection_Liston_Skoudis.pdf, 2006

〔10〕 Blue Pill Project, http://web.archive. org /web/20080418123748/http://www.bluepillproject.org/

〔11〕 Utah Emulab: Network Emulation Testbed Home, http://www.emulab. net/

〔12〕 KISTI Emulab: Network Emulation Test bed Home, http://www.emulab. kreonet. net/

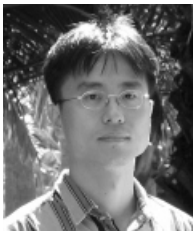
〔13〕 M. Lee and W. Seok, "Research on the Trend of Utilizing Emulab as Cyber Security Research Framework," Journal of the Korea Institute of Information Security and Cryptology, 23(6), pp. 1169-1180, Dec. 2013.

〔14〕 Using the RDTSC Instruction for Performance Monitoring, Intel Corp., https://www.ccsl.carleton.ca/~jamuir/rdtscpm1.pdf

〔15〕 Thompson, Christopher, Maria Huntley, and Chad Link, "Virtualization detection: New strategies and their effectiveness," http://www-users. cs. umn. edu/ cthomp/papers/vmm-detect-201.
〔16〕 D. Quist and V. Smith, "Detecting the Presence of Virtual Machines Using the Local Data Table," http://www.offensivecomputing.net/files/active/0/vm.pdf
〔17〕 X. Chen, J. Andersen, Z. Mao, M. Bailey, and J. Nazario, "Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware," Proceedings of Dependable Systems and Networks (DSN), pp. 177-186, June 2008.
〔18〕 Pafish, https://github.com/a0rtega/pafish
〔19〕 M. Lindorfer, C. Kolbitsch, and P.M. Comparetti. "Detecting Environment - Sensitive Malware," Proceedings of Symposium on Recent Advances in Intrusion Detection (RAID), pp. 338 - 357, Sep. 2011.
〔20〕 N. M. Johnson, J. Caballero, K. Z. Chen, S. McCamant, P. Poosankam, D. Reynaud, and D. Song, "Differential slicing: Identifying causal execution differences for security applications," Proceedings of IEEE Symposium on Security and Privacy, pp. 347-362, May 2011.
〔21〕 D. Kirat, G. Vigna, and C. Kruegel, "Barecloud: bare-metal analysis-based evasive malware detection," Proceedings of the 23rd USENIX conference on Security Symposium, pp. 287-301, Aug. 2014.
〔22〕 D. Kirat and G. Vigna, "MalGene: Automatic Extraction of Malware Analysis Evasion Signature," Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 769-780, Oct. 2015.

〈저자소개〉

이 만 희 (Manhee Lee) 종신회원
1995년 경북대학교 컴퓨터공학과 공학사
1997년 경북대학교 공학석사
2008년 Texas A&M 대학교 컴퓨터공학과 공학박사
1997년~2003년 한국과학기술정보연구원 연구원
2008년~2009년 Cisco Systems, San Jose
2010년~2011년 국가보안기술연구소 선임연구원
2012년~현재 한남대학교 조교수
〈관심분야〉 네트워크/시스템/스마트폰 보안, 고성능 시스템, 컴퓨터교육



석 우 진 (Woojin Seok) 정회원
1996년 경북대학교 컴퓨터공학과 학사
2003년 Univ. North Carolina, Computer Science 석사
2008년 충남대학교 컴퓨터공학과 박사
2003년~현재 한국과학기술정보연구원 선임연구원
〈관심분야〉 무선/이동 QoS, TCP 성능 분석