Cuckoo Sandbox를 회피하는 악성코드 탐지 방안 연구

정제성*, 김광조*

*한국과학기술원 전산학부

A Study on Detection of Evasive Malware in Cuckoo Sandbox

Je-Seong Jeong*, Kwangjo Kim*

*School of Computing, KAIST

요 약

악성코드는 정상적으로 시스템 사용을 못하게 하는 것으로서 컴퓨터 바이러스 부터 웜, 트로이 목마 등 그 종류도 다양하다. 이러한 악성코드를 분석하는 방법에는 정적분석과 동적분석이 있다. 이 둘은 각각의 장·단점이 있지만 요즘과 같이 악성코드의 수가 많은 시대에는 코드를 분석하는 정적분석 보다는 악성코드를 실행시켜 행위를 분석하는 동적분석이 시간적인 측면에서 봤을 시 좀 더 효율적이다. 그리고 요즘 동적분석으로 많이 사용하고 있는 것 중의 하나는 오픈소스 악성코드 자동 분석 시스템인 쿠쿠 샌드박스(Cuckoo Sandbox)이다. 하지만 요즘 악성코드는 분석 시스템을 우회하는 기능까지 탑재하여 분석을 어렵게 하고 있다. 따라서 본 논문에서는 이런 우회 기법을 분석 후 쿠쿠 샌드박스에 적용하여 탐지 기능을 향상시키는 방안을 제안하고자 한다.

I. 서론

개인 PC의 보급 증가와 인터넷 사용의 증가에 따라 악성코드에 감염되는 경로는 웹, 제휴 프로그램, Torrent 및 P2P, 불법 소프트웨어, 공유기 DNS 변조, 워드프로세서 취약점, 스팸메일, 저장매체 등으로 다양해 졌다. 또한, 제작 기술의 고도화에 따른 회피 기술 역시 향상 및 자동 생성 프로그램을 통해 예전보다 쉽게 악성코드를 생산하고 있다.

악성코드 피해 유형도 과거와 달리 단순 정보 유출 외에 DDoS 공격 등을 통한 시스템 파괴로 다양해지고 있다[1]. 따라서, 이런 악성코드를 사전에 탐지하여 피해를 막는 것은 매우 중요하다. 하지만 기존 악성코드 탐지기법인 시그니처(signature) 탐지기법은 공격 패턴을 변경 된 악성코드 또는 신종 악성코드를 만나면 대응 할 수가 없다. 또한, 악성코드 분석을 어렵게 하기 위해 코드 난독화 기법이나 암호화, 패킹 기법을 적용시킨 악성코드에 대해서는 정적분석을 하기가 어렵다.

이러한 문제점 극복을 위해 가상환경에서 악성코드를 동작시켜 악성행위를 분석하는 샌드박스(sandbox) 기반 악성코드 분석 도구인 쿠쿠 샌드박스(cuckoo sandbox)가 개발 되었다. 이는 가상환경에서 악성코드를 실행하기 때문에 시스템에 피해를 주지 않는다. 또한 악성코드를 실제 동작 시켜 악성행위를 분석하기 때문에 정적분석으로 힘든 악성코드도 분석이 가능하다.

하지만 분석 도구가 진보하는 것과 마찬가지로 악성코드 역시 진보하여, 쿠쿠 샌드박스를 우회하는 것들이 나타났다. 즉, 우회기법을 적용하여 쿠쿠 샌드박스를 무력화 시키는 것이다. 그래서 이를 보완 할 수 있는 방안 연구가 필요하겠다.

따라서, 본 논문에서는 기존논문[2]과는 다른 악성코드와 방법을 사용하여 쿠쿠 샌드박스의

악성코드 탐지 능력을 향상 시킬 수 있는 방안을 제시 하고자 한다.

II. 관련연구

본 장에서는 악성코드 최근 특성 및 분석 방법과 쿠쿠 샌드박스에 대해서 기술한다.

2.1 악성코드 특성

최근 악성코드는 제작 툴을 이용해 전문지식이 없는 인원도 단 시간에 만들 수 있는 특성으로 그 수가 급격히 증가 했다[3]. 이러한 특성으로 인해 피해 역시 급격히 증가하였으며 목표 역시 과거와는 달리 특정 국가기관 또는 금융기관을 대상으로 삼는 경우가 많아졌다[4]. 국내 사례로는 2011년 3.4 DDoS 공격, 2013년 3.20 및 6.25 사이버 공격 등이 있었다. 또한, 패킹, 코드 난독화, 안티 가상화 기법 등을 통해 백신 프로그램이 탐지를 어렵게 하고 있다.

2.2 악성코드 분석 방법

악성코드 분석 방법은 디버거(debugger)를 활용하여 분석하는 정적분석(static analysis)과 악성코드를 직접 실행시켜 결과를 분석하는 동적분석(dynamic analysis)으로 구분 할 수 있다. 특징으로는 정적분석은 모든 코드를 살펴볼 수 있기 때문에, 실제 실행되는 않는 루틴도 분석이 가능하다는 장점과 패킹 또는 암호화 되었을 때는 분석을 할수 없다는 단점이 있다. 동적분석은 패킹, 암호화 되었더라도 분석 할 수 있다는 장점과 악성코드가 탐지 회피 기법을 사용 시 파악 할 수 없다는 단점이 있다[2, 5, 6].

2.3 쿠쿠 샌드박스[7]

쿠쿠 샌드박스는 동적 기반의 악성코드 자동분석 시스템이다. 쿠쿠는 윈도우 운영체제 안에서 자동으로 악성코드를 실행시켜 파일을 분석하며, 분석된 동작 결과를 포괄적으로 수집하는데 사용한다. 쿠쿠는 윈도우 실행 파일, DLL 파일, PDF 문서 등을 분석하기 위해 설계 됐다. 그리고 악성코드에 의해 생성되는 프로세스로부터 수행되는 win32 API 호출 흔적, 악성코드 실행에 의한 파일 생성, 삭제, 다운로드, PCAP 포맷에서의 네트워크 트래픽 추적 등의 결과를 제공한다. 쿠쿠 샌드박스는 하나의 호스트 컴퓨터와 여러 게스트 컴퓨터로 구성되어 있다. 여기서 호스트 컴퓨터는 전체 분석 프로세스를 관리하는 샌드박스의 핵심 구성 요소를 실행하며, 게스트 컴퓨터는 실제 악성코드를 안전하게 실행시키고 분석하는 격리된 환경이다.

III. 샌드박스 우회기법[8]

악성코드를 분석하기 위해 쿠쿠 샌드 박스를 이용 할 수가 있다. 하지만 신종 악성코드들은 분석 시스템을 우회하는 기능을 추가하고 있다. 이런 악성코드는 쿠쿠 샌드박스를 이용해서 분석하기가 어렵다.

쿠쿠 샌드박스를 우회하는 방법에는 휴먼 인터랙션, 설정, 환경, 가상머신 회피 기법이 있다. 먼저 휴먼 인터랙션은 파일 기반의 샌드박스는 인간 사용자 없이 물리적 시스템을 동작하려 한다. 따라서 공격자들은 이러한 차이점을 악용하여, 마우스 클릭, 대화상자에 대한 사용자의 개입이 없으면 잠복을 하여 탐지를 회피한다. 둘째, 설정 회피기법은 파일을 수 분 동안 모니터 한 후에 의심스러운 행동이 나타나지 않으면 다음 파일로 넘어가는 특성을 이용한 잠복기, 정해진 날짜와 시간 후에만 악성코드를 실행함에 따라 이 시간 이전에는 탐지를 못하도록 하는 특정시간동작, 프로세스를 숨김으로써 탐지를 못하도록 하는 방법이 있겠다. 셋째, 환경 회피 기법은 특정한 운영체제와 어플리케이션의 조합이 없는 경우 실행되지 않도록 하여 탐지를 회피하는 방법이다. 마지막으로 넷째, 가상머신 회피 기법은 가상머신의 고유한 서비스 또는 파일을 확인하여 회피하는 방법이 되겠다.

IV. 가상화 우회 방지 기법

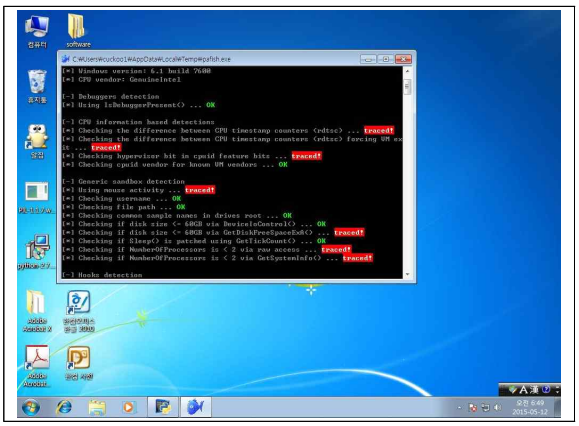
본 장에서는 악성코드가 가상화를 우회하지

못하도록 하는 방안에 대해서 정리한다. 먼저 실험 환경은 [표 1]과 같이 구성하였다.

[표 1] 실험 환경

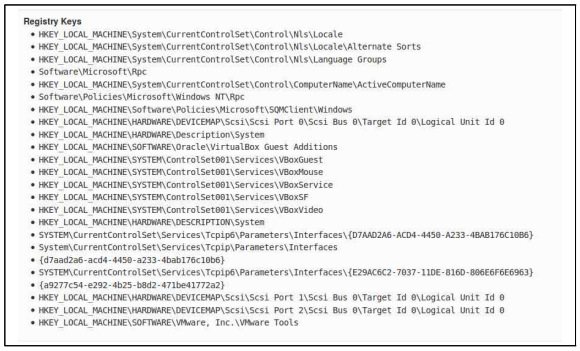
구 분	내 용	
O S	Host	Ubuntu 14.04 64bit
	Guest	Windows7 32bit
RAM	Host	8GB
	Guest	2GB
HardDisk	Host	100 GB
	GUEST	50GB
Virtual Machine	virtualbox 4.3.26 for LINUX, 64bit	
Malware	W32.Ramnit.B	

악성코드가 가상머신을 우회하는 것을 방지하기 위해서는 먼저, 악성코드가 가상머신의 어떤 부분을 인지하고 우회를 하는지를 알아야 한다. 이를 위해서는 악성코드가 분석을 피하기 위해 자주 사용하는 Pafish를 사용해서 가상머신을 분석해야한다[9]. Pafish는 안티디버거나 가상머신, 샌드박스의 실행 여부를 확인 할 수 있는 툴로서 가상화 환경 Windows7에서 실행한 결과는 [그림 1]과 같다.



[그림 1] Pafish 실행 화면

여기서 'traced' 라고 표기 된 부분은 가상머신으로 탐지가 되었다는 부분으로서 마우스 활동 등을 분석하고 있다. 또한 [그림 2]에서 볼 수 있듯이 Pafish는 Scsi 포트, "SystemBios-Version" 레지스트리 키, "VideoBiosVersion" 레지스트리 키, VBoxMouse.sys 드라이버 파일을 토대로 버추얼 박스 환경을 탐지했다[10].



[그림 2] Pafish 레지스트리

따라서, 악성코드에 의한 쿠쿠 샌드박스의 탐지를 어렵게 하기 위해서는 플러그인 또는 모듈을 수정하는 것이 필요하다.

따라서, 다음에 해야 할 일은 센서가 이것을 읽지 못하도록 CuckooMon의 폴더의 hook_reg.c 파일의 RegOpenExA와 RegQueryValueExA를 수정해야겠다[10].

먼저 RegOpenExA는 [그림 3]에서 [그림 4]과 같이 lpsubKey 탐지는 "!=NULL" 설정하여 VirtualBox를 찾을 때마다 가짜 응답을 주어 악성코드가 실행되게 해야겠다.

```
static IS_SUCCESS_LONGREG();

HOOKDEF(LONG, WINAPI, RegOpenKeyExA,
    __in        HKEY hKey,
    __in_opt    LPCTSTR lpSubKey,
    __reserved  DWORD ulOptions,
    __in        REGSAM samDesired,
    __out       PHKEY phkResult
) {
    LONG ret = Old_RegOpenKeyExA(hKey, lpSubKey, ulOptions, samDesired,
        phkResult);
    LOG("psP", "Registry", hKey, "SubKey", lpSubKey, "Handle", phkResult);
    return ret;
}
```

[그림 3] RegOpenExA 수정 전

```
HOOKDEF(LONG, WINAPI, RegOpenKeyExA,
    __in        HKEY hKey,
    __in_opt    LPCTSTR lpSubKey,
    __reserved  DWORD ulOptions,
    __in        REGSAM samDesired,
    __out       PHKEY phkResult
) {
    LONG ret;
    if (strstr(lpSubKey, "VirtualBox") != NULL) {
        ret = 1;
        LOG("s", "Hardening", "Faked RegOpenKeyExA return");
    }

    else if (strstr(lpSubKey, "VBOX") != NULL) {
        ret = 1;
        LOG("s", "Hardening", "Faked RegOpenKeyExA return");
    }
    else if (strstr(lpSubKey, "vbox") != NULL) {
        ret = 1;
        LOG("s", "Hardening", "Faked RegOpenKeyExA return");
    }
    else if (strstr(lpSubKey, "oracle") != NULL) {
        ret = 1;
        LOG("s", "Hardening", "Faked RegOpenKeyExA return");
    }
    else if (strstr(lpSubKey, "virtualbox") != NULL) {
        ret = 1;
        LOG("s", "Hardening", "Faked RegOpenKeyExA return");
    }
    else if (strstr(lpSubKey, "ControlSet") != NULL) {
        ret = 1;
        LOG("s", "Hardening", "Faked RegOpenKeyExA return");
    }
    else {
        ret = Old_RegOpenKeyExA(hKey, lpSubKey, ulOptions, samDesired, phkResult);
    }
    LOG("psP", "Registry", hKey, "SubKey", lpSubKey, "Handle", phkResult);
    return ret;
}
```

[그림 4] RegOpenExA 수정 후

다음으로 RegQueryValueExA도 [그림 5]에서 [그림 6]과 같이 수정해야겠다.

```
HOOKDEF(LONG, WINAPI, RegQueryValueExA,
    __in        HKEY hKey,
    __in_opt    LPCTSTR lpValueName,
    __reserved  LPDWORD lpReserved,
    __out_opt   LPDWORD lpType,
    __out_opt   LPBYTE lpData,
    __inout_opt LPDWORD lpcbData
) {
    ENSURE_DWORD(lpType);
    LONG ret = Old_RegQueryValueExA(hKey, lpValueName, lpReserved, lpType,
        lpData, lpcbData);
    if(ret == ERROR_SUCCESS && lpType != NULL && lpData != NULL &&
            lpcbData != NULL) {
        LOG("psr", "Handle", hKey, "ValueName", lpValueName,
            "Data", *lpType, *lpcbData, lpData);
    }
    else {
        LOG2("psLL", "Handle", hKey, "ValueName", lpValueName,
            "Type", lpType, "DataLength", lpcbData);
    }
    return ret;
}
```

[그림 5] RegQueryValueExA 수정 전

```
HOOKDEF(LONG, WINAPI, RegOpenKeyExA,
    __in        HKEY hKey,
    __in_opt    LPCTSTR lpSubKey,
    __reserved  DWORD ulOptions,
    __in        REGSAM samDesired,
    __out       PHKEY phkResult
) {
    LONG ret;
    if (strstr(lpSubKey, "VirtualBox") != NULL) {
        ret = 1;
        LOG("s", "Hardening", "Faked RegOpenKeyExA return");
    }

    else if (strstr(lpSubKey, "VBOX") != NULL) {
        ret = 1;
        LOG("s", "Hardening", "Faked RegOpenKeyExA return");
    }
    else if (strstr(lpSubKey, "vbox") != NULL) {
        ret = 1;
        LOG("s", "Hardening", "Faked RegOpenKeyExA return");
    }
    else if (strstr(lpSubKey, "oracle") != NULL) {
        ret = 1;
        LOG("s", "Hardening", "Faked RegOpenKeyExA return");
    }
    else if (strstr(lpSubKey, "virtualbox") != NULL) {
        ret = 1;
        LOG("s", "Hardening", "Faked RegOpenKeyExA return");
    }
    else if (strstr(lpSubKey, "ControlSet") != NULL) {
        ret = 1;
        LOG("s", "Hardening", "Faked RegOpenKeyExA return");
    }
    else {
        ret = Old_RegOpenKeyExA(hKey, lpSubKey, ulOptions, samDesired, phkResult);
    }
    LOG("psP", "Registry", hKey, "SubKey", lpSubKey, "Handle", phkResult);
    return ret;
}
```

[그림 6] RegQueryValueExA 수정 후

여기서 주목 할 점은 SystemBiosVersion, Identifier, ProductId, VideoBiosVersion을 "!=NULL"로 설정하여 악성코드가 이 레지스트리를 읽지 못하도록 하여 악성코드가 실행 되도록 하는 것이다[10]. 위와 같이 수정을 마친 후에 DLL 파일을 컴파일 후 dll 폴더로 이동 후 실행시키면 악성코드가 가상머신을 인식 못하고 실행될 것이다. 이는 기존 논문[2]이 Upclicker.exe 파일을 이용한 것과는 차이가 있다.

V. 결론

본 논문에서는 악성코드가 샌드박스를 우회하는 다양한 기법들 중에서 가상화를 인지 하고 우회하는 기법을 사용하지 못하도록 하는 방안을 정리해 보았다. 이 방안을 통해 좀 더 안전하게 악성코드를 동적 분석을 할 수가 있을 것이다. 하지만 가상화 내에서의 악성코드는 네트워크 구성의 제한에 따라 부분적인 활동만을 할 것이다. 따라서, 악성코드를 정확히 분석하는 것은 다소 제한 적일 것이다. 그리고 현재는 위와 같은 방법을 통해 악성코드가 가상화를 인지 하지 못하도록 할 수가 있겠으나 이후 악성코드가 좀 더 발전이 되었을 시에는 이와 같은 방법도 의미가 없어질 것이다. 따라서, 악성코드가 탐지할 수 없는 실제 호스트에서와 같이 가상 네트워크를 구성하여 악성코드를 분석 할 수 있도록 시스템을 만드는 것과 더불어 악성코드 탐지 기법을 비교 분석하는 것이 추후 필요하다.

[참고문헌]

[1] 정기보고서 및 통계, https://www.krcert.or.kr /kor/data/reportList.jsp#none

[2] 주정욱, 회피기법이 있는 악성코드 분석을 위한 샌드박스용 사용자 이벤트 발생기 설계, 목표대학교 대학원 정보보호기술학협동과정, 석사학위논문, 2015년 2월

[3] Cisco2014AnnualSecurityRepor

[4] http://www.hackmageddon.com

[5] Andreas Moser, Christopher Kruegel, and Engin Kirda, Limits of Static Analysis for Malware Detection, ACSAC. 2007

[6] Ulrich Bayer, Andreas Moser, Christopher Kruegel, Engin Kirda, Dynamic analysis of malicious code, Springer-Verlag France 2006

[7] CuckooSandbox. http://www.cuckoosandbox.org

[8] FireEye, 파일 기반의 샌드박스를 쉽게 회피하는 악성코드, 2013

[9] https://github.com/a0rtega/pafish

[10] 디지트 오크타비안토 외 지음, Cuckoo 샌드박스를 활용한 악성코드 분석, acorn+PACKT Technical Book