자동화된 도구에 의해 생성된 변종 악성코드의 공통 속성을 이용한 탐지 방법

박성빈*, 김민수**, 노봉남*

Detection Method Using Common Features of Malware Variants Generated by Automated Tools

Seong-Bin Park*, Min-Soo Kim**, and Bong-Nam Noh*

이 논문은 2012년도 전남대학교 연구년교수연구비 지원에 의하여 연구되었음.

요 약

최근 악성코드는 사회적으로 많은 이슈가 되고 있다. 또한 다양한 변종 악성코드를 생성할 수 있는 자동화된 도구의 등장은 안티 바이러스 업체들의 지속적인 노력에도 불구하고 감당하기 힘들 정도의 많은 변종 악성코드를 만들어 내고 있다. 이러한 어려움을 극복하기 위해 본 논문에서는 자동화된 도구에 의해 생성된 변종 악성코드의 탐지 및 분류를 위해 공통 속성을 이용한 탐지 방법을 제안한다. 정적 및 동적 분석을 통해 얻을 수 있는 7가지 종류의 공통속성을 선정하였으며, 총 11개의 샘플을 이용하여 실험하였다. 실험 결과는 자동화된 도구에 의해 생성된 변종 악성코드에 대해 효과적인 탐지 및 분류 결과를 보여준다.

Abstract

Recent malware has become an issue in the social. In addition, the automated tool that can be generate malware variants have emerged. Anti-virus vendors are constantly working for detect malware variants. However, automated tool is creating a large amount malware variants. To overcome these difficulties, this paper proposes a detection method using the common feature for detection and classification of malware variants generated by automated tools. The seven kind of common features that can be obtained through the static and dynamic analysis were select, and we experimented using a total of 11 samples. The experiment results showed the effective detection and classification for malware variants generated by automated tools.

Keywords

malware variants, automatic generation tools, common feature, detection and classification

* 전남대학교 시스템보안연구센터

** 목포대학교 정보보호학과

· 제1저자(First Author): 박성빈, 교신저자(Corresponding Author): 노봉남

· 접수일: 2012년 03월 20일, 수정일: 1차- 2012년 04월 19일, 2차- 2012년 05월 04일, 게재확정일: 2012년 09월 19일

Ⅰ. 서 론

악성코드란 컴퓨터 또는 네트워크에 어떤 악의적인 행위를 목표로 설계된 소프트웨어를 말하며[1], 바이러스, 웜, 스파이웨어 등이 이에 해당한다. 악성코드의 측면에서 변종은 새로운 계통을 의미하며, 원래의 코드에서 약간 수정된 버전을 말한다[2]. 이러한 변종 악성코드는 많은 변조 기술들을 통하여 기존의 시그너처 기반 탐지기법을 간단히 우회할 수 있다[3]. 또한 자동화된 도구를 이용하여 비전문가도 매우 간단히 변종 악성코드를 만들어 낼 수 있다. 특히 제우스나 스파이아이와 같은 도구들은 사이버 범죄 시장에서 불법적으로 거래되어 많은 피해를 입히고 있는 대표적인 사례라고 할 수 있다.

변종 악성코드를 쉽게 생성할 수 있는 자동화된 도구들의 유포로 인해 몇 년 사이에 발견된 악성코드의 수는 기하급수적으로 증가하고 있다. 독일의 민간 보안 연구 단체인 AV-Test에서 조사한 결과에 따르면 2011년 말까지 약 6천 4백만 개 이상의 악성코드가 발견되었으며, 이는 2010년도에 발생한 악성코드의 약 1.4배나 증가한 양이다[4]. 또한 2011년 11월 시만텍 보고서에서는 최근 악성 웹사이트에 의한 위협이 47.8%의 높은 증가율을 보이는 것으로 분석되었다. 악성 웹 사이트는 주로 공격자들이 피싱공격이나 악성코드를 배포하기 위해 이용되며, 공격 방법들 중 자동화된 도구의 사용이 54.6%에 이른다[5].

이러한 위협들로부터 변종 악성코드를 탐지하기 위한 방법들로는 Opcode 통계기능을 이용한 탐지방법[6], 행위 그래프를 이용한 탐지방법[7], Hybrid genetic 알고리즘을 이용한 종속성 그래프 기반의 탐지방법[8], 의심스러운 행위로부터 최적 명세서 통합을 이용한 탐지방법[9] 등 많은 연구들이 수행되어 왔다. 이러한 연구들은 탐지를 위한 척도로서 Opcode, 행위 그래프, 의존성 그래프 등을 이용하여 임계점에 대한 문제를 해결하거나 탐지 정확도를 향상시키고, 미탐(false negative)율을 줄이는 등의 장점을 가지고 있다. 하지만 코드 난독화(obfuscation) 기술에 대해서 제한된 범위에서만 작업이 가능하거나 기존의 시그너처 기반의 스캐너를 기반으로 하는 안티 바이러스 소프트웨어보다 많은 처리시간이 요구되는 등의 문제점들이 존재한다.

따라서 본 논문에서는 기존의 시그너처 기반의 탐지 기법을 간단히 우회할 수 있는 변종 악성코드의 수가 급증하고 있는 원인 중 하나로 자동화된 도구의 등장을 꼽았으며, 이러한 자동화된 도구에서 생성된 변종 악성코드를 탐지하고 분류하기 위한 척도로서 공통 속성을 이용한 방법을 제시하고자 한다. 공통 속성을 판별 척도로 선정한 이유는 자동화된 도구에 의해 생성된 수많은 변종 악성코드의 시그너처는 다를 수 있지만 논리적 구조의 핵심은 동일하기 때문이다. 따라서 이들이 공통적으로 가지고 있는 속성을 분석하여 비교함으로써 악성코드를 탐지하고, 분류할 수 있다.

2장에서는 기존의 변종 악성코드 탐지 방법에 대한 관련연구들을 살펴본다. 3장에서는 변종 악성코드를 분석하고, 판별을 위한 몇 가지 주요 공통 속성을 설명한다. 4장에서는 선정된 공통 속성을 기반으로 유사도 비교를 위한 방법을 제시하고, 이 방법을 통한 실험과 결과를 통한 성능 검증을 보여준다. 마지막으로 5장에서는 결론에 대하여 기술한다.

Ⅱ. 관련연구

변종 악성코드를 탐지하기 위해서는 먼저 해당 악성코드에 대한 분석이 필요하다. 악성코드를 분석하는 방법으로는 정적 분석과 동적 분석이 존재한다. 먼저 정적 분석은 악성코드를 실행시키지 않고, 분석 도구들을 이용하여 직접 분석하는 방법이다. 대표적인 정적 분석 방법으로는 바이너리 패턴 매칭, 데이터 플로우와 코드 플로우 분석 등이 있다. 이러한 정적 분석 방법은 악성코드를 직접 실행하지 않기 때문에 안전하고, 깊이 있는 분석이 가능하다는 장점이 있다. 하지만 실행압축(run-time packer)과 코드 난독화 기법을 사용한 변종 악성코드의 경우, 분석이 매우 어렵고, 궁극적으로 분석 자체가 불가능할 수 있다는 단점이 존재한다. 많은 악성코드들이 실제로 실행압축을 통해 코드를 보호하고 있으며, 정적 분석을 수행하기 전에 실행압축을 해

제하는 작업만으로도 많은 노력과 시간이 필요하다. 또한 실행 압축만으로도 다양한 변종을 만들어 낼 수도 있다. 코드 난독화 기법은 역공학(reverse engineering)을 통한 분석에서 코드를 읽기 어렵게 변조하는 기술이다. 이러한 난독화는 대상에 따라 크게 소스코드 난독화와 바이너리 난독화로 나눌 수 있으며, 이 또한 분석을 수행하는 데 큰 걸림돌이 된다[10]-[13].

이러한 정적 분석의 단점과 한계를 극복하기 위한 연구로는 P. Royal, M. Halpin 등이 발표[14]한 실행 압축 또는 난독화되어 숨겨진 주요 코드를 자동으로 추출하는 기법이 있다. Kevin Coogan 등의 연구[15]는 실행압축을 해제하기 위한 과정을 자동화하는 정적 분석 기법을 소개하고, M. D. Preda [16], V. Sathyanarayan[17] 등은 의미론적 분석을 통해 난독화된 악성코드 분석을 소개한다. 그리고 정구현 등의 연구[18]은 실행 압축 해제 과정을 진행하고 있을 때, 메모리 상태의 엔트로피 값의 변화를 관찰하여 오리지널 엔트리 포인트(OEP)를 찾아내는 기법을 소개하고 있다.

이러한 연구들은 특정한 API의 호출을 감시함으로써 악성코드의 의미론적 모델을 추출할 수 있었으며, 높은 탐지율을 보여주었다. 하지만 특정 API의 발생 빈도에 의존적이라는 문제점 때문에 의미 없는 행위를 삽입하는 형태의 난독화 기술에는 취약하다는 단점이 존재한다.

이와 같이 다양한 연구에도 불구하고 변종 악성코드에 대한 정적 분석 방법은 계속해서 진화하는 코드 난독화 기술로 인해 악성코드가 실제로 수행하는 행위를 판별하는 데 많은 어려움이 존재하고 있다. 이러한 어려움을 극복하기 위한 방법으로 동적 분석을 사용할 수 있다. 동적 분석은 가상 머신과 같은 제어 가능한 환경에서 악성코드를 실제로 동작시켜 그 행위를 분석하는 방법이다. 그렇기 때문에 실행 압축과 코드 난독화 기법 등에 상관없이 실제 악성코드가 하는 행위를 분석할 수 있다는 장점이 존재한다. 하지만 이러한 동적 분석 은 실제로 악성코드를 실행하기 때문에 실험 환경의 감염이 우려되며, 행위 관찰과 분석 결과를 활용하는 데 많은 시간이 소요된다는 단점이 존재한다[7].

동적 분석 방법을 이용한 연구로는 C. Williems 팀의 CWSandbox로 잘 알려진 GFISandbox[19]와 U. Bayer 팀의 Anubis로 잘 알려진 TTAnalyze[20]이 대표적인 동적 분석 방법을 이용한 연구이다. CWSandbox는 매우 잘 알려진 악성코드 행위 분석 도구로서 악성코드 바이너리를 분석하기 위해 프로세스 이미지를 생성하고, 대상 바이너리에 DLL을 삽입하여 API 후킹과 모니터링 된 행위를 전송함으로써 분석을 수행한다. 분석된 결과는 보고서의 형태로 제공된다[19]. TTAnalyze는 윈도우 실행파일 형태의 악성코드에 대한 행위 분석에 초점을 맞춘 도구이다. 악성코드 바이너리는 에뮬레이터 환경에서 분석되며, 마찬가지로 분석된 결과는 보고서의 형태로 제공한다[20]. 이 연구가 중단된 이후 Anubis [21]라는 이름으로 웹 기반 악성코드 분석 유틸리티를 제공하고 있으며. 이는 현재까지도 많은 연구자들이 활용하고 있다.

따라서 본 논문에서는 정적 및 동적 분석을 모두 활용한다. 자동화된 도구에 의해 생성된 변종 악성코드 바이너리에 대해 정적 분석을 통해 문자열, DLL, API 정보들을 공통 속성으로 추출하고, 동적 분석을 통해 파일, 레지스트리, 프로세스, 네트워크와 같은 정보들을 공통 속성으로 추출한다. 추출된 각각의 공통 속성들을 통합하여 테스트베드를 구축하고, 유사도 계산을 통한 탐지 및 분류를 수행한 결과로 성능을 검증한다.

III. 변종 악성코드 공통 속성 분석

3.1 변종 악성코드 공통 속성 개요

변종 악성코드 탐지 및 분류를 위한 공통 속성은 정적 분석과 동적 분석을 통해 추출된다. 자동화된 도구에서 생성된 수많은 변종 악성코드는 행위에 있어서 약간의 차이가 있을 뿐 논리적 구조의 핵심은 동일하기 때문에 대부분 유사한 속성을 가지고 있다. 각 속성은 표 1에 정리된 것과 같이 크게 7가지의 속성으로 분류되며, 동적 분석에 대한 결과는 세부적으로 11가지의 속성을 갖는다.

표 1. 변종 악성코드 공통 속성 개요
Table 1. Overview of the common feature for malware variants

구분	속 성		설 명
정적 분석	String		문자열 정보
	DLLs		DLL 리스트
	APIs		API 리스트
동적 분석	File	Path	파일생성 경로
		Name	파일생성 이름
	Registry	Key	레지스트리 키
		Key value	레지스트리 키 값
		Path	레지스트리 경로
	Process	Path	프로세스 생성 경로
		Name	생성된 프로세스 이름
	Network	Protocol	통신 프로토콜
		DNS	DNS 질의 리스트
		IP	통신한 IP주소
		Port	통신한 목적지 포트

3.2 정적 분석을 이용한 공통 속성 추출

정적 분석을 이용하여 각각의 변종 악성코드에서 String, DLL, API에 대한 속성을 추출한다. 여기서 각 속성을 선정한 이유와 정적 분석을 이용한 추출 방법에 대해 간단히 설명한다.

- String 속성

악성코드의 바이너리에는 특정한 문자열을 포함하고 있는 경우가 많기 때문에 이러한 문자열 정보는 악성코드를 탐지하는 데 중요한 속성이 될 수 있다. 윈도우에서 사용 가능한 Sysinternals사의 Strings[22]를 통해 바이너리의 문자열을 추출하였고, 해당 변종 악성코드의 특징에 따라 필터링하여 선정된 문자열 정보는 공통 속성으로 사용하였다.

- DLL 및 API 속성

일반적으로 PE구조를 가진 실행파일의 내부에는 IAT(Import Address Table)를 포함하여 호출할 DLL과 API리스트를 관리한다. Stud_PE[23]과 PEView[24]를 이용하여 간단하게 악성코드 바이너리에서 DLL 및 API리스트를 추출하였다. API리스트의 경우 그 양이 매우 많기 때문에 정상 프로그램으로부터 자주 포함되는 API리스트를 화이트리스트 list)로 작성하여 이를 제외한 API리스트를 변종 악성코드의 공통 속성으로 사용하였고, 화이트리스트는 국·내외 자주 사용되는 응용프로그램 10개를 선정하여 도출하였다. 표 2와 같이 정상 응용프로그램과 변종 악성코드의 중복된 API리스트를 제거함으로써 수많은 API리스트의 비교를 피하고, 보다 효율적인 유사도 계산을 도와준다.

3.3 동적 분석을 이용한 공통 속성 추출

동적 분석을 이용하여 각각의 변종 악성코드에서 File, Registry, Process, Network에 대한 속성을 추출한다. 여기서 각 속성을 선정한 이유와 동적 분석을 이용한 추출 방법에 대해 간단히 설명한다.

표 2. 화이트리스트를 적용한 예
Table 2. An example of applying the white list

<table>
<tr><th></th><th>정상 응용프로그램</th><th>변종 악성코드 1</th><th>변종 악성코드 2</th></tr>
<tr><td rowspan="3">APIs</td><td colspan="2">RaiseException, TlsSetValue, TlsGetValue, LocalAlloc, TerminateProcess, CreateRemoteThread, PostQuitMessage, (...중략...), EnableWindow, ControlService, CloseServiceHandle</td><td rowspan="2">RegEnumKeyA, CreateFileMappingA, MapViewOfFile, GetFileAttributesA, SetFileAtterburesA, GetTempPathA, GetWindoesDirectoryA, (...중략...), wvsprintfA, wsprintfA, PostMessageA</td></tr>
<tr><td>CryptMsgGetParam, CertOpenStore, GetFileVersionInfoW, VerQueryValueW, IsDebuggerPresent, (...중략,,,), InternetCrackUrlW, InternetOpenW, SetupIterateCabinetW,</td><td>LoadLibraryExA, WriteProcessMemory, WinExe, (...중략...), OpenServiceA, OpenSCManagerA</td></tr>
<tr><td colspan="2">FindFirstFileA, FindClose, GetFileSize, DeleteFileA, CreateFileA, Sleep, FindWindowA</td></tr>
<tr><td colspan="3">FindClose, GetFileSize, Sleep</td></tr>
</table>

• File 속성

악성코드가 최초에 PC에 감염되어 실행되면, 악의적인 행위를 하기 위해 악성코드의 실체인 자신을 시스템의 어딘가에 안착시킨다. 주로 임시폴더에 최초로 생성되긴 하지만 이후에 임의적인 위치에 복사하는 경우가 많다. 뿐만 아니라 인터넷을 통해 추가적인 악성코드를 다운로드하거나 정상적인 파일에 자신을 인젝션(injection)하기도 한다. 따라서 변종 악성코드 탐지를 위해 파일 행위에 대한 속성을 추출한다.

• Registry 속성

최초 감염 이후에도 악성코드는 부팅 시 자동으로 시작되도록 하거나 자신의 활동을 위해 레지스트리를 수정하는 작업을 수행한다. 이는 지속적으로 시스템에 피해를 입히는 행위이다.

• Process 속성

이렇게 감염된 악성코드는 주로 독립적인 프로세스의 형태로 동작하거나 다른 정상 프로세스에 인젝션되어 쓰레드 상태로 동작한다. 또한 DLL 인젝션을 통해 보안 프로그램을 종료시키는 등 다양한 프로세스 관련 행위들을 수행한다. 이는 악성코드를 탐지하는 데 중요한 속성 정보이다.

• Network 속성

마지막으로 악성코드는 추가적인 악성코드를 다운받거나 개인정보 유출, 스팸메일 발송 등의 악성 행위를 위한 네트워크 행위를 수행하기도 한다. 목적지 IP는 근원지 IP에서 특정 서버 혹은 호스트로의 연결 관계를 파악하기 위해 사용되며, 포트 또한 악성코드를 판별하기 위한 중요한 속성이 될 수 있다. 특히 악성 봇넷의 경우 C&C 서버와의 통신에서 주로 사용되는 80번, 443번, 8080번 포트와 스팸메일 발송을 위한 25번 포트 등이 하나의 속성 정보로 사용되었다. 그리고 DNS와 프로토콜에 대한 정보 또한 악성코드를 탐지하는 데 중요한 속성이다.

이러한 동적 분석을 이용하여 추출할 수 있는 주요 속성들은 Anubis[21], SysAnalyzer[25], CWSandbox[26]을 이용하였고, 각 분석도구들은 결과로서 리포트를 제공한다.

3.4 변종 악성코드 공통 속성 추출 알고리즘

3.2장과 3.3장에서 설명한 공통 속성을 추출하기 위한 알고리즘은 표 3과 같다.

표 3. 변종 악성코드 공통 속성 추출 알고리즘
Table 3. The algorithm of common feature extraction for malware variants

```
1:  function feature_extractor(malware_binary)
2:    function dynamic_analysis
3:      analyzer(anubis, malware_binary, report[0]);
4:      analyzer(cwsandbox, malware_binary, report[1]);
5:      analyzer(sysanalyzer, malware_binary, report[2]);
6:    end function
7:    function static_analysis
8:      strings(malware_binary, malware_strings);
9:      pe_parser(malware_binary, malware_iat);
10:   end function
11:   function report_parser
12:     for i < 3 do
13:       res[i] = parser(report[i]);
14:     end for
15:   end function
16:   function extraction
17:     string_filter(malware_strings, res_str);
18:     iat_filter(malware_iat, white_list, res_iat);
19:     report_combine(res, res_report);
20:   end function
21:   feature_string = res_str;
22:   feature_dll = res_iat.dll;
23:   feature_api = res_iat.api;
24:   feature_file_activity = res_report.file_activity;
25:   feature_reg_activity = res_report.reg_activity;
26:   feature_proc_activity = res_report.proc_activity;
27:   feature_net_activity = res_report.net_activity;
28: end function
```

변종 악성코드의 공통 속성을 추출하기 위한 알고리즘은 크게 5단계로 이루어진다. 1단계는 악성코드의 바이너리를 동적으로 분석하고, 그 결과를 동일한 포맷으로 저장한다. 2단계는 정적 분석 단계로 문자열과 IAT를 파싱(parsing)하여 DLL 및 API리스트를 추출하여 저장한다. 3단계는 동적 분석 결과로 얻은 리포트를 분석하는 단계로 각 분석 도구의 리포트 결과를 파싱하여 저장한다. 4단계는 동적 분석 결과와 정적 분석 결과를 통합 및 정렬하는 단계이다. 마지막 5단계는 추출된 속성들을 각각 저장하는 단계이다.

Ⅳ. 유사도 비교

4.1 유사도 비교 방법

본 논문에서 제안하는 유사도 비교 방법은 공통 속성을 이용하는 것이다. 3장에서 선정하여 추출한 공통 속성들을 데이터베이스에 저장하여 데이터셋(dataset)을 구축한다. 공통 속성들 간의 중복 저장을 피하고, 빠른 검색을 위해 인덱스를 이용하여 테이블을 구성하였다. 대부분의 악성코드는 유사한 목적을 가지고 있으며, 결국 유사한 공통 속성을 가지고 있다. 또한 파일이나 레지스트리 경로와 같은 경우 검색해야 할 문자열의 길이가 상당히 길어 성능 효율이 떨어질 수 있다. 따라서 모든 공통 속성이 중복되어 저장되는 것을 방지하기 위해 통합하여 관리하며, 각 샘플에 대한 공통 속성 정보는 인덱스 번호만을 저장함으로써 보다 빠르고 효율적인 검색을 수행할 수 있다. 그리고 구축한 데이터베이스를 기반으로 그림 1과 같은 과정을 통해 변종 악성코드에 대한 유사도 비교 평가를 수행한다.

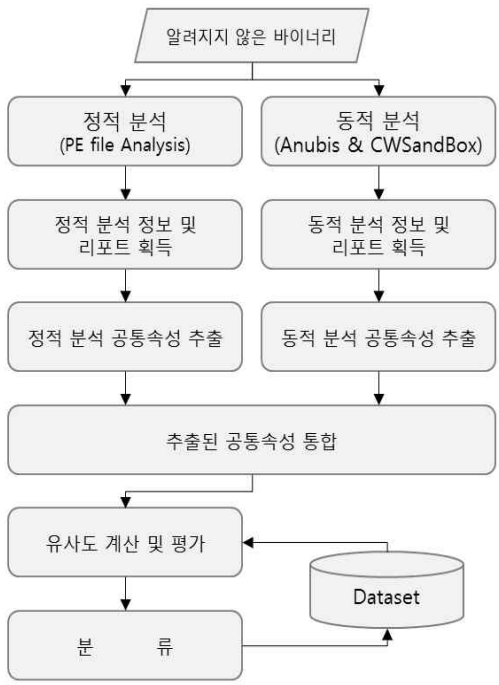

그림 1. 유사도 비교 흐름도
Fig. 1. Flow chart of similarity comparison

각각의 속성에 대한 유사도 비교를 위한 계산은 정적 분석 된 속성과 동적 분석 된 속성이 각기 다른 공식을 갖는다. 정적 분석된 속성의 경우 식 (1)을 이용하여 각 속성에 대한 유사도를 계산한다.

$$s_f = \frac{M_f \cap U_f}{M_f} \tag{1}$$

공식에서 f는 속성을 의미하며, M_f는 데이터베이스에 저장된 악성코드의 속성 카운트를 의미한다. U_f는 알려지지 않은 바이너리의 분석된 속성 카운트를 의미하고, 동적 분석된 속성의 경우 식 (2)를 이용하여 각 속성에 대한 유사도를 계산한다. 알려지지 않은 바이너리의 분석된 속성 값이 비교할 악성코드의 속성에 포함되면 유사도를 1로 간주한다.

$$s_f = \begin{cases} 1, & U_f \in M_f \\ 0, & U_f \notin M_f \end{cases} \tag{2}$$

식 (1)과 (2)에서 계산된 각 속성의 유사도는 최종적으로 자동화된 도구에 의해 생성된 변종 악성코드를 탐지하고 분류하기 위한 전체 유사도 계산에 사용된다. 하지만 변종 악성코드마다 특징이 다르기 때문에 각 악성코드를 분석한 결과를 가중치로 부여하여 좀 더 높은 정확도를 제공하였다. 최종적으로 식 (3)을 이용해 최종 악성코드에 대한 유사도 S_m을 계산한다. 여기서 w_f는 속성 가중치를 의미한다.

$$S_m = Average(w_f \cdot s_f) \tag{3}$$

4.2 실험 설정

자동화된 도구에 의해 생성된 변종 악성코드의 탐지 및 분류 실험을 위해 9종의 샘플을 수집하였다. 그 중에서 실행 가능한 바이너리를 생성할 수 있었던 도구 8종에 대한 샘플을 각각 100개씩 생성하였으며, 웹에서 1종의 샘플을 23개 수집하였다. 수집된 결과는 표 4에서 볼 수 있다. 데이터셋의 모든 바이너리들은 윈도우 파일 포맷을 가졌으며, 대부분 트로이목마 형태의 악성코드이다.

표 4. 변종 악성코드 데이터셋
Table 4. Dataset of malware variants

Family Name	Malware Type	Dataset
JPS Virus Maker	Backdoor-Trojan	100
Necro Virus Maker	Dropper-Trojan	100
NecronomikonsWindoze	Worm	100
NX Virus Creation	Agent-Trojan	100
SAT TCK	PWS-Trojan	100
Toxic CV	VB-Trojan	100
Pinch3 Builder	PWS-Trojan	100
Zeus Bilder	Zbot-Trojan	100
Mydoom	Email-Worm	23
Total	N/A	823

4.3 실험 결과

추출된 공통 속성과 제안하는 유사도 비교 방법을 이용하여 대상 샘플에 대한 유사도 비교를 수행한 결과는 표 5와 같다. 자동화된 도구에 의해 생성된 변종 악성코드의 정적 분석과 동적 분석을 통하여 공통 속성을 추출한 후, 추출된 공통 속성을 통합하여 데이터셋을 구축하고, 새로운 샘플에서 추출된 공통 속성을 4.1장에서 제안한 유사도 비교 방법을 통해 계산한 결과 최소 81.3%에서 최대 99%까지의 높은 탐지율을 보여준다. 특별한 예로 Necronomilons의 경우는 하나의 생성도구에서 mailworm, myworm, p2pworm과 같이 세 가지 형태의 악성코드를 생성하기 때문에 이들 간의 유사도는 다른 샘플들에 대한 유사도에 비해 비교적 더 높은 탐지율을 나타낸다.

결국 새로운 변종 악성코드가 입력으로 들어왔을 때 기존 악성코드에 대해 변조가 이루어졌다 할지라도 공통 속성을 추출하여 유사도를 비교하기 때문에 이 실험 결과와 같이 새로운 변종 악성코드를 탐지하고 분류하는 작업을 충분히 수행할 수 있을 것으로 예상된다.

V. 결 론

본 논문에서는 자동화된 도구에 의해 생성된 다양한 변종 악성코드를 탐지하고 분류하기 위해 공통 속성을 이용한 유사도 비교를 제안하였다. 기존에 많은 연구가 이루어진 프로젝트와 분석 도구들을 활용하여 정적 및 동적 분석한 결과로 얻은 데이터들로 공통 속성을 추출하였다.

표 5. 변종 악성코드 유사도 비교 결과
Table 5. The result of similarity comparison for malware variants

Family Name	JPS Virus Maker	Mydoom	Necro-mailworm	Necro-myworm	Necro-p2pworm	Necro Virus Maker	NX Virus Creation	Pinch	Sat binary	Toxic-CV	Zeus Builder 1.2.4.2
JPS Virus Maker	**0.990**	0.053	0.050	0.050	0.050	0.000	0.166	0.280	0.116	0.122	0.068
Mydoom	0.137	**0.888**	0.040	0.039	0.039	0.000	0.203	0.191	0.081	0.022	0.107
Necro-mailworm	0.298	0.190	**0.919**	0.188	0.688	0.000	0.369	0.146	0.140	0.054	0.108
Necro-myworm	0.542	0.238	0.265	**0.850**	0.250	0.260	0.525	0.233	0.173	0.287	0.373
Necro-p2pworm	0.298	0.190	0.544	0.188	**0.813**	0.000	0.369	0.146	0.140	0.054	0.108
Necro Virus Maker	0.175	0.000	0.000	0.250	0.000	**0.900**	0.250	0.000	0.000	0.325	0.250
NX Virus Creation	0.185	0.324	0.064	0.063	0.063	0.000	**0.856**	0.159	0.202	0.026	0.080
Pinch	0.068	0.239	0.031	0.030	0.030	0.000	0.106	**0.978**	0.081	0.108	0.070
Sat binary	0.244	0.127	0.086	0.086	0.086	0.000	0.314	0.186	**0.951**	0.054	0.164
Toxic-CV	0.130	0.130	0.100	0.100	0.100	0.130	0.130	0.300	0.130	**0.860**	0.130
Zeus Builder 1.2.4.2	0.108	0.072	0.056	0.167	0.056	0.144	0.219	0.111	0.108	0.036	**0.876**

이를 기반으로 공통 속성 추출을 위한 알고리즘을 제안하였으며, 유사도 비교를 위해 총 11개의 샘플을 가지고 데이터베이스를 구축하였다. 데이터베이스는 효율성을 위해 인덱스를 이용한 통합관리 방식을 사용하였고, 정적 및 동적 분석 결과로 얻어진 속성에 대한 유사도 공식을 각각 제안하였다. 변종 악성코드마다 특징을 분석하여 가중치를 부여해 정확도를 높였으며, 실험을 통해 공통 속성을 이용한 유사도 비교로 탐지 및 분류의 가능성을 보였다.

차후 연구에서는 더욱 많은 변종 악성코드 생성 도구에서 공통 속성을 추출하는 데 있어서 주관적이고, 모호한 부분을 통계를 이용하여 보완할 필요가 있다.

참 고 문 헌

[1] Wikipedia, Malware, http://en.wikipedia.org/wiki/Malware/.

[2] Webopedia, Variant, http://www.webopedia.com/TERM/V/variant.html/.

[3] 박남열, 김용민, 노봉남, "우회기법을 이용하는 악성코드 행위기반 탐지 방법", 정보보호학회 논문지, 제 16권, 제 3호, pp. 17-28, 2006년 6월.

[4] AV-Test, http://www.av-test.org/en/statistics/malware/.

[5] Symantec.cloud MessageLabs, Symantec Intelligence Report, Nov. 2011.

[6] Babak Bashari Rad and Maslin Masrom, "Metamorphic Virus Detection in Portable Executables Using Opcodes Statistical Feature", ICASEIT, Jan. 2011.

[7] 권종훈, 이제현, 정현철, 이희조, "행위 그래프 기반의 변종 악성코드 탐지", 정보보호학회 논문지, 제 21권, 제 2호, pp. 37-47, 2011년 4월.

[8] Keehyung Kim and Byung-Ro Moon, "Malware Detection based on Dependency Graph using Hybrid Genetic Algorithm", GECCO'10, pp. 1211-1218, July 2010.

[9] Matt Fredrikson, Somesh Jha, Mihai Christodorescu, Reiner Sailer, and Xifeng Yan, "Synthesizing Near-Optimal Malware Specifications from Suspicious Behaviors", 2010 IEEE Symposium on Security and Privacy, pp. 45-60, May 2010.

[10] F. Cohen, "Computer viruses: Theory and experiments", In DOD/NBS Com. and Sec. Conf., Vol. 6, pp. 22-35, Sep. 1987.

[11] D. Chess and S. White, "An undetectable computer virus", In Virus Bulletin Conf., Sep. 2000.

[12] A. Moser, C. Kruegel, and E. Kirda, "Limits of static analysis for malware detection", In ACSAC, pp. 421-430, Dec. 2007.

[13] A. Moser, C. Krügel, and E. Kirda, "Exploring multiple execution paths for malware analysis", *IEEE Security and Privacy*, pp. 231-245, May 2007.

[14] Paul Royal, Mitch Halpin, David Dagon, Robert Edmonds, and Wenke Lee, "PolyUnpack: Automating the Hidden-Code Extraction of Unpack-Executing Malware", *IEEE, ACSAC'06*, pp. 289-300, Dec. 2006.

[15] Kevin Coogan, Saumya Debray, Tasneem Kaochar, and Gregg Townsend, "Automatic Static Unpacking of Malware Binaries", *IEEE, WCRE'09*, pp. 167-176, Oct. 2009.

[16] M. D. Preda, M. Christodorescu, S. Jha, and S. Debray, "A Semantics-Based Approach to Malware Detection", *ACM Trans. Program. Lang. Syst.*, Vol. 30, No. 5, Aug. 2008.

[17] V. S. Sathyanarayan, P. Kohli, and B. Bruhadeshwar, "Signature Generation and Detection of Malware Families", ACISP, Vol. 5107, pp. 336-349, 2008

[18] 정구현, 추의진, 이주석, 이희조, "엔트로피를 이용한 실행 압축 해제 기법 연구", 한국정보기술학회 논문지, 제 7권, 제 1호, pp. 232-238, 2009년 2월.

[19] C. Willems, T. Holz, and F.C. Freiling, "Toward Automated Dynamic Malware Analysis Using CWSandbox", *IEEE Security & Privacy,* Vol. 5, No. 2, pp. 32-39, March 2007.

[20] C. K. Ulrich Bayer and E. Kirda, "Ttanalyze: A Tool for Analyzing Malware", In 15th Ann. Conf. of European Inst. for Computer Antivirus Research (EICAR), pp. 180-192, 2006.
[21] Anubis, http://anubis.iseclab.org/.
[22] Windows Sysinternals, Strings, http://technet.microsoft.com/en-us/sysinternals/bb897439/.
[23] Stud_PE, http://www.cgsoftlabs.ro/studpe.html/.
[24] PEView, http://www.magma.ca/~wjr/.
[25] SysAnalyzer, http://securitytnt.com/sysanalyzer/.
[26] CWSandbox, http://mwanalysis.org/.

저자소개

박 성 빈 (Seong-Bin Park)

2011년 2월 : 전남대학교 전자컴퓨터공학부(공학사)
2011년 3월 ~ 현재 : 전남대학교 정보보안협동과정 석사과정
관심분야 : 악성코드, 시스템 보안

김 민 수 (Min-Soo Kim)

1993년 : 전남대학교 전산통계학과 (이학사)
1995년 : 전남대학교 전산통계학과 (이학석사)
2000년 : 전남대학교 전산통계학과 (이학박사)
2000년 ~ 2001년 : 한국인터넷진흥원 선임연구원
2000년 ~ 2004년 : 전남대학교 연구교수
2005년 ~ 현재 : 목포대학교 정보보호학과 부교수
관심분야 : 시스템 보안, 네트워크 보안, 정보보안, 신경망 등

노 봉 남 (Bong-Nam Noh)

1987년 : 전남대학교 수학교육과 (이학사)
1982년 : KAIST 전산학과 (이학석사)
1994년 : 전북대학교 전산과 (이학박사)
1983년 ~ 현재 : 전남대학교 전자컴퓨터공학부 교수
2000년 ~ 현재 : 시스템보안연구센터 소장
관심분야 : 디지털 포렌식, 시스템 및 네트워크 보안, 정보사회와 사이버 윤리