韓國 컴퓨터情報學會 論文誌
第10卷 第6號, 2005. 12.

2005-10-6-1-14

시그너처 패턴기반의 악성코드 탐색도구의 개발

우 종 우*, 하 경 휘**

A Development of Malware Detection Tool based on Signature Patterns

Chong-Woo Woo*, Kyoung-Hui Ha**

요 약

최근 악성코드에 의한 피해는 상업용 백신의 지속적인 개발에도 불구하고 급격히 증가되고 있다. 일반적으로 백신은 이미 알려진 악성코드는 효과적으로 탐색이 가능하지만 아무런 정보가 없는 악성코드를 탐색하기는 어려우며, 또한 최근의 악성코드들은 백신의 갱신속도보다 훨씬 빨리 새로운 변종들을 만들어내고 있기 때문에 백신의 대응이 늦게 되는 경향이 있다. 본 논문에서는 이러한 악성코드들을 효과적으로 탐색할 수 있는 탐색도구의 설계 및 개발에 관하여 기술한다. 본 연구의 도구는 악성코드의 기능을 분석하여 특정한 시그너처를 추출함으로서 기존의 악성코드들 뿐 아니라 새로운 악성코드와 그 변종들에 대해서도 능동적으로 대처할 수 있다.

Abstract

Recently, the damages occurring from the malware are increasing rapidly, regardless of continuous development of commercial vaccines. Generally, the vaccine detects well-known malware effectively, but it becomes helpless without any information against the unknown ones. Also, the malware generates its variations fast enough, so that the vaccine always gets behind in its updates. In this paper, we are describing a design and development of malware detection tool, which can detect such malware effectively. We first analyze the general functionality of the malware, and then extracts specific signatures. Such that, we can actively cope with a malware, which may come in previous type, a new type, and any of its mutations also.

▸ Keyword : 악성코드(Malware), 악성코드 탐색도구(Malware Detection Tool), 시그너처 패턴(Signature Patterns)

• 제1저자 : 우종우
• 접수일 : 2005.10.24, 심사완료일 : 2005.11.16
* 국민대학교 컴퓨터학부 교수, ** 국민대학교 컴퓨터학부 대학원
※ 본 논문은 2005학년도 국민대학교 교내연구비 지원을 받아 수행되었음

템의 구조와 구성을 기술하고, 4장에서는 시스템의 구현을 설명하고 마지막 5장에서는 결론 및 향후 연구 과제를 논의한다.

Ⅰ. 서 론

반적으로 악성코드(malware)란 사용자의 의사와 이익에 반해 시스템을 파괴하거나 정보를 유출하는 등 악의적인 활동을 수행하도록 제작된 소프트웨어, 프로그램, 스크립트 등을 의미하며, 주로 바이러스, 웜, 또는 Trojans 등의 현태로 전자메일이나 컴퓨터 시스템의 취약점을 이용하여 공격한다. 이러한 악성코드에 대처하기 위한 다양한 방법들이 연구되고 있으며 대표적으로 에이전트를 이용한 침입탐지등이 연구되고 있다 [1][2]. 물론 다양한 기능의 상업용 백신의 개발이 지속되고 있지만, 완벽히 악성코드를 차단할 수는 없다 [3][4].이러한 백신은 악성코드내의 알려진 시그너처들 에만 반응하기 때문에 이미 알려진 악성코드는 효과적으로 탐색하지만, 새로운 악성코드나 그 변종에 대해서는 효과적이지 못한 단점이 있다. 따라서 이러한 악성코드로부터의 피해를 감소시키고 보다 능동적으로 대처하기 위해서는, 기존 백신의 사용이외에 또 다른 안전장치가 필요할 것이다.

본 논문에서는 이러한 악성코드의 가능성을 경고하여줄 수 있는 악성코드 탐색도구의 설계 및 개발에 관하여 기술한다. 본 연구의 도구는, 만약 사용자에게 주어진 특정 실행파일이 악성코드일 가능성에 대처하게 될 때, 이를 실행해 보지 않고도 파일의 안정성을 조언해줄 수 있기 때문에 백신이외의 또 다른 안전장치라고 할 수 있다. 본 연구의 탐색도구는 다음과 같이 수행된다. 우선 의심스러운 실행파일로부터 일반적으로 악성코드들이 주로 사용하는 윈도우즈 API함수들을 분석 및 추출한다. 그리고 이러한 함수들을, 사전에 잘 알려진 악성코드들로부터 추출하여 데이터베이스에 저장한 패턴과 비교한다. 최종적으로, 비교 분석된 결과가 악성코드인지 여부에 관한 결과를 사용자에게 제공한다. 이러한 접근은 기존의 악성코드뿐만 아니라 새로운 악성코드와 그 변종에 대해 능동적으로 대처할 수 있고, 사용자에게 그 분석 내용을 보고함으로써 사용자에게 보다 더 친화적인 악성코드 탐지 기능을 제공할 수 있게 된다.

본 논문의 구성은 다음과 같다. 2장 관련 연구에서는 악성코드의 전파 방법 및 윈도우즈 실행파일인 PE 파일에 대하여 서술한다. 3장에서는 시스템의 특성을 중심으로 시스

Ⅱ. 관련연구

본 장에서는 악성코드의 탐지 기법에 관한 관련연구와, 이러한 악성코드를 탐지할 수 있는 탐색도구의 개발에 필수적요소인 악성코드의 전파방법 및 윈도우즈 PE파일 포맷에 관하여 기술한다.

2.1 악성코드 탐지 기법

최근의 악성코드들은 비주얼 베이직, mIRC스크립트, 자바 스크립트 등과 같은 스크립트를 이용하여 생성되고 있으며, 이렇게 작성된 악성 스크립트들은 주로 전자 우편을 통하여 전파되고 있다. 이러한 스크립트를 미리 차단하기 위해서는 주로 시그너처 스캐닝에 의한 방법, 어플리케이션 변환 기법, 정적 분석에 의한 감지 등이 연구 되고 있는데, 그중 시그너처 기반의 스캐닝이 보편화 되어 사용되고 있다.

시그너처 기반의 스캐닝은 사전에 시그너처를 추출한 악성코드만을 감지할 수 있으므로 알려지지 않은 대부분의 악성코드들은 또 다른 방법으로 탐지하여야 하는데 주로 다음과 같이 연구되고 있다.

2.1.1 기존의 시그너처 기반 탐색법

기존의 탐색방법은 주로 악성코드에 존재하는 특별한 문자열을 데이터 베이스화 하고 이 문자열의 존재를 탐색함으로서 해당 스크립트의 악성여부를 진단 하였다. 이 방법은 속도는 빠르지만, 스크립트 추출방법이 사람에 의존하기 때문에 새로운 안티 바이러스의 까지는 악성코드에 무방비 상태가 되는 문제점이 있다 [5][6]

2.1.2 휴리스틱 스캐닝

알려지지 않은 악성 스크립트를 탐색하기 위하여 기존의 악성코드에서 악성행위를 위한 메소드(method)나 내장함수(intrinsic function) 호출들을 미리 데이터 베이스화 하여 두고, 대상 스크립트를 스캔하여 일정 수 이상의 위험한 호출이 나타나면 이것을 악성 스크립트로 간주하는 방식이다

[7]. 이것은 속도가 빠르고 높은 감지율을 나타내지만, 선의의 스크립트를 악성으로 판단하는 긍정오류의 가능성이 있다.

2.1.3 행위 감시기법

행위 감시 기법은 프로그램 수행에 필요한 시스템 호출들을 가로채어 감시하다가 악성행위로 판단되는 시스템 콜의 시퀀스가 나타나면, 해당 프로그램을 악성코드로 간주하는 감지방식이다 [8]. 이 방법은 실시간 감지가 가능하나, 구현이 어렵고 부하가 큰 것으로 알려져 있다.

2.2 악성코드의 전파 방법

악성코드는 일반적으로 대상파일이나 시스템으로의 감염 및 전파를 목적으로 동작하며, 이를 위한 많은 기법들이 존재한다. 악성코드를 이해하기 위해서는 이러한 기법들에 대한 이해는 필수적이며, 이러한 전파 기법에는 '전자 메일을 이용한 전파', '네트워크를 이용한 전파', '재부팅시 악성코드 실행기법', 'DLL injection', 'Polymorphism', 등 다양한 방법들로 분류된다[9][10][11]. 본 논문에서는 탐색도구의 개발과 관련 있는 '전자메일을 이용한 전파기법과 네트워크를 이용한 전파기법'에 대해서만 기술한다.

2.3 전자메일을 이용한 전파기법

대부분의 악성코드들은 전파를 목적으로 전자메일의 첨부파일을 이용하여 악성코드 자신을 첨부 한다 [12][13]. 보안에 관한 인식이 부족한 사용자라면, 첨부된 악성코드가 실행되어 쉽게 시스템이 감염된다. 전자메일 관련 전파기법은 다음과 같이 요약된다.

2.3.1 SMTP 릴레이 서버를 이용한 메일전송 기법

SMTP 릴레이 서버를 이용한 전파 기법은 SMTP 릴레이 서버의 자체적인 문제보다는 메일 어플리케이션이 내포하고 있는 보안상 취약점을 이용한다. 이 방법은 악성코드 스스로 메일을 발송하지 못하는 단점이 있다.

2.3.2 SMTP 메일 엔진을 이용한 메일 전송 기법

SMTP 자체 메일 엔진을 사용한 전파 기법은 악성코드 내부에서 SMTP를 사용할 수 있는 메일 엔진을 탑재하고 있어 자체적으로 메일을 생성하고 악성코드 자신을 첨부하여 전송하는 기법이다. 악성코드는 감염시킨 시스템 내부를 검색하여 전송 대상의 메일 주소를 수집한다. 메일 서버는 메일의 발신자에 대한 인증 절차를 수행하지 않는 취약점이 존재하기 때문에 이러한 취약점을 이용해 발신자의 주소를 임의로 조작할 수 있다.

2.4 네트워크를 이용한 전파 기법

대부분의 컴퓨터들은 크고 작은 네트워크에 연결되어 통신 및 파일 전송 등을 하지만, 다음과 같은 심각한 보안상의 위협을 초래할 수 도 있다.

2.4.1 공유 폴더를 이용한 전파 기법

공유 폴더에 접속하는 방법은 사용자가 설정한 공유 폴더의 접속과 관리목적 공유 폴더의 접속으로 분류된다. 첫째, 사용자 설정 공유 폴더의 접속은 공유 폴더의 읽기와 쓰기에 대한 권한 설정에서 야기되는 취약점을 이용하여, 악성코드는 자신의 복제 본을 공유 폴더 취약성을 가진 시스템으로 복사할 수 있다. 둘째, 관리 목적 공유 폴더의 접속은 패스워드 취약점을 이용, 시스템의 관리자 권한을 획득할 수 있으며 시스템을 장악할 수도 있다. 공유 폴더의 접속이 허용되면 악성코드는 WIN32 API를 이용하여 파일을 전송하거나, 대상 시스템이 특정 서버에 접속하도록 하여 악성코드를 다운로드 받음으로서 대상 시스템을 원격으로 조정할 수 있다.

2.4.2 P2P 프로그램을 이용한 전파 기법

P2P 프로그램이 설정한 공유 폴더에 악성코드 자신을 복사하여 P2P 프로그램 사용자에 의해 전파되는 특징을 가진다. 악성코드는 P2P 프로그램의 다운로드 폴더의 경로명을 획득하여 다운로드 폴더 안에 P2P 프로그램 사용자가 흥미를 가질 수 있는 파일 이름으로 자신의 복사본을 생성한다. 이 경우 악성코드는 능동적으로 전파할 수 없고 사용자가 악성코드를 다운로드 받는 형태의 수동적 전파만 이루어진다.

2.5 PE File Format

PE(Portable Executable) 파일은 이식 가능한 실행 프로그램을 뜻하며, Win32의 기본적인 파일형식으로, UNIX 시스템에서 사용하는 파일포맷인 COFF (Common Object File Format)를 기본으로 설계되었다. PE 파일포맷의 필드는 RVA(Relative Virtual Address)로 정의되는데, RVA는 파일이 메모리에 맵핑된 위치로부터의 상대적인 가상 주소로, 다음과 같은 공식으로 얻어 진다 [14].

```
RVA(Relative Virtural Address) =
Virtual Address - Base Address
```

PE 파일은 PE 로더에 의해 실행되는데 (그림 1), 파일의 각 부분에 대한 설명은 다음과 같다.

2.5.1 MS-DOS MZ Header and Stub Program

PE 파일포맷의 첫 번째 부분은 MS-DOS Header이다. MS-DOS 헤더와 스텁 프로그램은 PE 파일을 지원하지 않는 운영체제에서 파일을 로드할 때, PE 파일이 실행되지 않는다는 정보를 사용자에게 제공하기 위해 존재한다. 또한 PE 파일포맷을 지원하는 운영체제는 MS-DOS 헤더 안에 기록된 PE 파일포맷의 위치를 사용한다.

2.5.2 PE Header

MS-DOS 헤더는 128byte의 PE 헤더에 저장하는데, 여기에는 PE 파일포맷을 나타내는 PE 시그너처 및 PE 파일의 정보와 PE 부가 헤더에 대한 정보가 저장되어 있다.

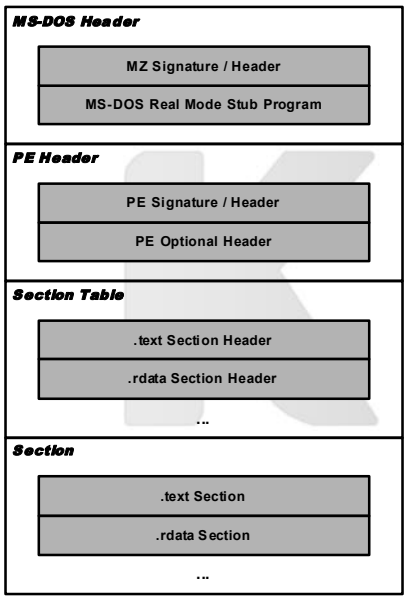

그림 1. PE File Format 구조
Fig 1. PE File Format Structure

2.5.3 PE Optional Header

PE 헤더 다음의 224byte는 PE 부가 헤더이다. PE 부가 헤더는 윈도우즈가 파일을 로드할 때, 로더가 필요로 하는 정보가 저장되어 있다. 이곳에는 실행파일의 이미지, 초기화된 스택의 크기, 프로그램 엔트리 포인트(Entry Point) 주소, 운영체제 버전 등이 저장되어 있다.

2.5.4 Section Table (Section Header)

PE Optional Header 다음에는 섹션 테이블이 존재한다. 섹션 테이블은 파일 내에 존재하는 섹션 헤더가 모여 있는 집합을 나타낸다. 섹션 테이블 안의 섹션 헤더의 개수는 PE File Header의NumberOfSection에 저장되어 있다.

2.5.5 Section

섹션은 코드, 데이터, 리소스등 파일을 실행하기 위한 정보들을 가지고 있으며, 성격에 따라 Code섹션 데이터 섹션 등으로 나눈다 코드 섹션은 기존의 파일포맷에서 사용하는 모든 코드 세그먼트를 하나로 합친 것으로 프로그램 개발자나 운영체제는 통합된 코드 섹션으로 가상 메모리를 쉽게 관리할 수 있다. 데이터 섹션은 그 기능에 따라 “.bss”, “.rdata”, “.data”로 분류한다. “.bss” 섹션은 프로그램에서 초기화되지 않은 데이터를 나타내며, 함수와 모듈의 모든 변수 선언도 포함한다. “.rdata” 섹션은 읽기 전용 데이터 즉 문자열이나 상수 그리고 디버그 디렉터리 정보 등을 나타낸다. 그 외 다른 변수는 “.data” 섹션에서 저장한다. 또한 모듈이나 프로그램에서 사용하는 전역 변수는 “.data” 섹션에 포함된다.

윈도우즈는 실행 파일을 지원하기 위해 코드 섹션과 데이터 섹션 외에도 추가로 몇 개의 섹션을 지원하는데, “.rsrc”섹션은 모듈에서의 리소스 정보를 담당하고 있다. “.idata”와 “.edata” 섹션은 파일의 Import와 Export 정보를 담당하며, “.debug” 섹션은 파일의 디버깅 정보를 가지고 있다.

III. 시스템 설계

본 장에서는 악성코드 분석을 위한 도구의 전체적인 구조와 각각의 구성에 대해 설명하고, 이들의 상호연관성에 대하여 설명한다.

3.1 시스템 구조

악성코드 탐색 도구는 크게 두 가지 측면에서 악성코드 탐색 작업을 수행한다. 첫째는 대상 파일에서 악성코드에 대한 시그너처를 추출하는 시그너처 추출부(Signature Extracting Phase)이이고, 둘째는 추출된 시그너처의 패

턴을 분석하여 악성코드의 가능성을 분석하는 분석부(Analysis Phase)이다.

시그너처 추출부는 바이너리 코드를 로드하여 섹션 형태로 구분하기 위한 "Binary Structure"와 PE 파일포맷 구조를 분석하는 "PE Structure", 분석된 바이너리 코드에서 악성코드에 대한 시그너처를 추출하기 위한 "API Signature List" 등으로 구성되어 있다. 분석부는 시그너처의 패턴을 분석하기 위한 모듈과 함께 분석된 데이터를 사용하여 악성코드의 기능에 대한 분석 결과를 표현하기 위한 "Functionality List", 악성코드일 가능성을 표현하기 위한 "Analysis Result"로 구성되어 있다.

전체적인 시스템의 흐름은 바이너리 데이터를 로드하는 것으로부터 시작되며 (그림 2), 전반적인 진행 과정을 요약하면 다음과 같다.

로드된 바이너리 코드는 PE Parsing 모듈에 의해 PE 헤더와 섹션들로 분류되고, 분류된 각각의 정보들은 PE Signature에 저장된다.

파싱 과정이 종료된 후 시그너처 추출부에 의해 "Signature" 데이터베이스의 정보와 일치하는 API 시그너처들을 추출한다. "Signature" 데이터베이스는 악성코드의 기능들을 구현할 수 있는 API들이 저장되어 있고, 바이너리 코드 내에 이들 API와 일치하는 데이터를 검사하여 시그너처를 생성한다.

시그너처 추출 과정이 종료된 후 분석 작업이 수행된다. 분석 작업은 추출된 시그너처의 패턴과 일치하는 "Rule" 데이터베이스에 저장된 규칙을 검색하고, 이에 해당되는 결과를 돌려준다.

이 결과를 기반으로 악성코드의 기능에 대한 가능성과 관련된 정보를 "Functionality List"에 표현하고 악성코드에 대한 가능성을 Result에 저장한다. 마지막으로 저장된 각각의 정보를 Report 모듈을 통하여 사용자에게 보고하는 기능을 수행한다.

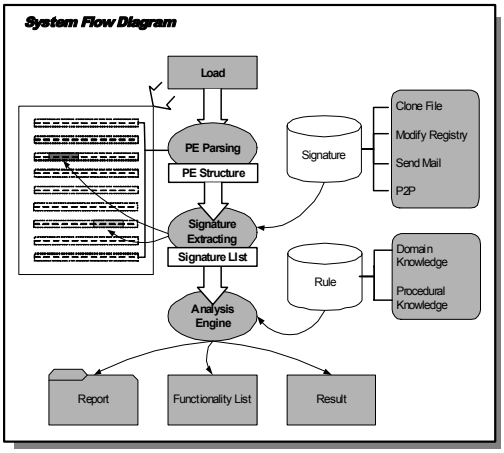

그림 2. 시스템 흐름도
Fig 2. System Control Flow

3.2 시그너처 추출부 (Signature Extracting Phase)

주 기능은 입력 받은 바이너리 코드를 PE 구조에 맞추어 분리하고, API 시그너처를 추출한다.

3.3 PE 구조(PE Structure)

바이너리 코드에서 API에 대한 시그너처를 추출하기 위해서 선행되어야 할 작업은 바이너리 코드의 PE 구조를 분류하는 것이다. 그 이유는 바이너리 코드 내의 API 시그너처는 주소의 형태로 존재하기 때문에 직접적으로 시그너처를 추출할 수 없기 때문이다. 아래 (그림 3)은 시그너처 추출부에서 바이너리 코드를 통해 PE 구조를 생성하는 과정을 보여준다. 생성된 PE 구조는 구조화된 트리형태로 표현된다.

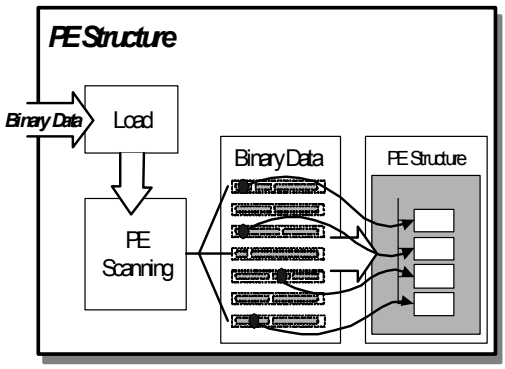

그림 3. PE 구조 분류과정
Fig 3. A Process of PE Structure Classification

3.4 API 시그너처

API 시그너처는 악성코드의 기능에 대한 API 함수들을 의미한다. 예를 들면, (그림 4)는 악성코드의 기능중 자기 복제 기능을 구현한 예이다. 자기복제를 위해서는 GetModuleFileName, CreateFile, CopyFile 등의 API를 사용하여 그 기능을 수행하게 된다.

```
strcpy(fname, "malware.exe");
GetModuleFileName(NULL, selfpath, MAXPATH);
SetFileAttributes(fname, FILE_ATTRIBUTE_ARCHIVE);
hFile = CreateFile(fname, GENERIC_WRITE,
FILE_SHARE_READ|FILE_SHARE_WRITE, NULL,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL,
NULL);
CopyFile(selfpath, fname, FALSE);
ExitProcess(0)
```

그림 4. 자기복제 기능에 대한 예
Fig 4. An Example of Self-duplication

(그림 5)에서는 API 시그너처를 획득하는 과정을 보여준다. 전반적인 흐름은 바이너리 코드와 분석된 PE 구조를 사용하여 API에 대한 시그너처를 획득하고, 데이터베이스에서 악성코드의 기능을 구성하는 특정 API에 대한 정보를 받아와 특정 API를 추출하게 된다.

그림5. 바이너리 코드에 존재하는 API 시그너처
Fig 5. API signature residing in binary code

(그림 6)은 시그너처 추출부에서 API 시그너처를 획득하는 과정을 보여준다. 바이너리 코드와 분석된 PE 구조를 사용하여 API에 대한 시그너처를 획득하고, 데이터베이스에서 악성코드의 기능을 구성하는 특정 API에 대한 정보를 받아 특정 API를 추출한다.

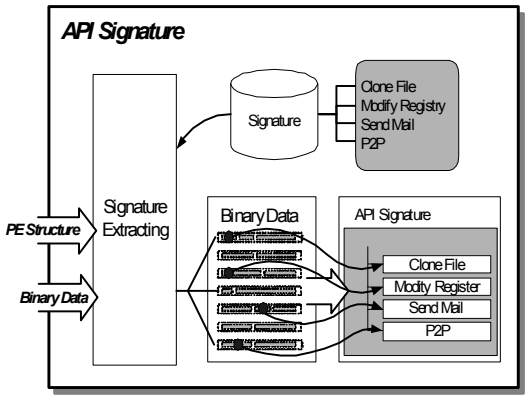

그림 6. 시그너처 추출부에서 시그너처 획득 과정
Fig 6. Signature Extraction in Signature Extracting Phase

본 도구에서는 총 15 개의 API에 대한 시그너처를 통하여 각각의 기능들을 판별할 수 있도록 구현하였다. 또한 API 시그너처 이외에 ".rdata" 섹션에서 획득한 데이터들을 부가적으로 첨부하여 API Signature를 생성하게 된다.

3.5 분석부(Analysis Phase)

분석부는 바이너리 코드를 파싱하여 획득한 API 시그너처를 사용하여 악성코드의 가능성 여부를 분석하는 기능을 수행한다. 분석부의 구성은 Rule Engine, 규칙데이터베이스, 그리고 Functionality List와 Result모듈들로 구성된다 (그림 7). 데이터베이스에는 "Domain Knowledge" 와 "Procedural Knowledge"가 저장되어 있는데, "Domain Knowledge"는 시그너처와 매칭되는 규칙들이며, "Procedural Knowledge"는 수행 흐름에 관한 규칙들이다.

분석부의 주요 기능은 API 시그너처를 통하여 악성코드의 가능성을 타진하는 것이지만 이러한 분석 결과를 사용자에게 효과적으로 제시하여 사용자의 이해를 돕는 시각적인 부분도 중요한 역할을 담당한다. 분석 모듈에 의하여 제시된 Functionality List와Result는 사용자에게 대상 바이너리 코드가 악성코드로 판단된 원인과 결과를 명확히 파악할 수 있도록 하는 도구로써 사용될 수 있다.

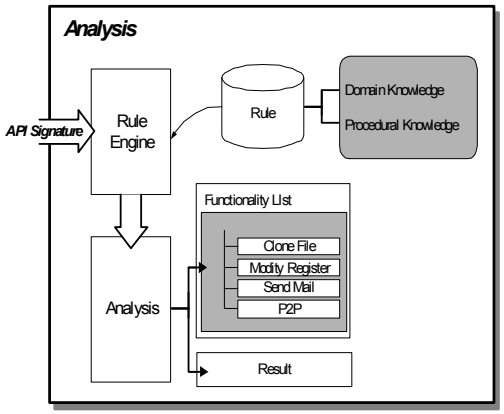

그림 7. 분석부의 분석 과정
Fig 7. Process of Analysis in Analysis Phase

분석부의 핵심적인 모듈은 Rule Engine이다. 먼저 API 시그너처를 입력으로 받아 규칙에 적용할 Flag를 생성한다. Flag는 위에서 언급한 기능에 대해 각각 존재하며, Flag의 값에 따라 분석을 수행하게 된다. 분석이 완료되면 Analysis 모듈에 의해 선택된 규칙들에 의해 추가된 특정 값을 분석하여 악성코드의 기능에 대한 가능성 여부를 판단한다. 또한 각 기능에 대한 값을 분석하여 악성코드의 가능성 여부를 판단하게 된다.

3.6 보고 모듈

보고 모듈은 분석 모듈에서 분석된 데이터를 사용하여 사용자에게 시각적으로 결과를 제시한다.

3.6.1 악성코드 기능별 리스트(Functionality List)

악성코드 기능별 리스트는 API 시그너처에 대한 분석 결과를 처리하는 기능을 수행한다. 악성코드 기능별 리스트의 주요한 역할은 악성코드 기능에 대한 정보와 해당 기능에 대한 가능성 정도를 나타내는 것이다. 또한 악성코드 기능에 대한 설명을 포함하고 있어 사용자가 악성코드에 대해 쉽게 이해할 수 있도록 한다. 그리고 이들 정보를 기반으로 악성코드의 가능성을 판단하는 역할을 수행한다.

악성코드 기능별 리스트는 각각의 기능별 이름과 가능성 여부에 대한 정보를 포함하고 있다. 이 정보는 도구에서 별도의 다이얼로그 형태를 통해서 표현되며, 각 기능별 가능성 정도와 그에 따른 색상 정보를 보여주게 된다. 악성코드 기능에 대한 가능성과 관련된 색상은 그 정도에 따라 각각 “High 적색”, “Medium-주황색”, “Low-노란색”을 나타내어 시각적으로 사용자가 손쉽게 가능성을 판단할 수 있도록 해준다.

3.6.2 악성코드 판단 결과(Result)

악성코드 판단 결과는 악성코드 기능별 리스트의 정보를 사용하여 산출한다. 악성코드 판단 결과는 악성코드 기능별 리스트와 마찬가지로 악성코드 가능성 정도와 그에 따른 색상 정보를 저장하고 있다. 이 정보는 악성코드 기능별 리스트와 함께 별도의 다이얼로그 형태로 표현된다.

IV. 시스템 구현

본 도구는 C++ 언어를 사용하였고, GUI 환경을 위해서 MFC를 사용하였다. 윈도우즈 플랫폼을 이용한 이유는 대부분의 악성코드들이 윈도우즈 계열을 목표로 제작되고 있기 때문이다.

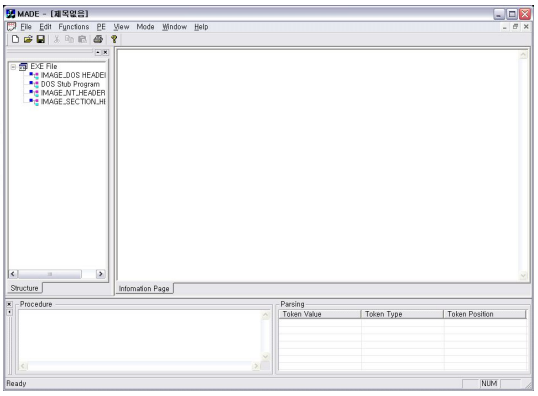

그림 8. 시스템 구성화면
Fig 8. Display of System Construction

(그림 8)은 악성코드 탐지 도구의 기본 화면 구성을 보여주는 그림이다. 좌상단의 창에는 PE 구조를 확인할 수 있게 트리 형태로 PE 구조를 생성함으로서 PE 구조의 각 부분들을 확인할 수 있다. 사용자가 선택한 노드는 그 노드의 악성코드 탐지의 수행한 결과를 우상단의 화면에 출력한다. 하단의 창에는 악성코드 탐지 수행과정에 대한 로그를 출력하고, API 시그너처들이 파싱 되는 과정을 출력한다.

메뉴의 구성은 기본 메뉴에서 “Function”, “PE” “View”의 3 가지를 추가하였다. 첫째, Function 메뉴는 “File Scan”, “Analysis PE”, “Get Signature”, “Match MW”

4 가지로 구성되어 있으며, 각각 악성코드 탐지를 위한 파싱과 분석 과정을 시작하기 위한 메뉴이다. 두 번째, PE메뉴는 분석된PE 구조를 출력하는 기능을 수행한다. 세부 메뉴는 PE 구조의 세부 구조로 PE 파일을 확인하는데 용이하다. 세 번째, View 메뉴는 악성코드 탐지를 위한 각각의 수행 과정에 대한 결과 화면을 출력하는 기능을 수행한다.

4.1 PE 파일 분석

먼저 시작화면에서 대상 파일을 선택한 후 “Scan File”을 수행하고, 파일을 로드한 후 “Analysis PE”을 수행하게 되면, 파싱 모듈에 의해 바이너리 코드가 분석되고, 그 결과는 PE Structure에 저장된다. 파일의 분석이 완료되면 (그림 9)와 같이 결과 화면과 수행창이 수정된다. 결과 화면의 정보는 PE 파일 포맷의 전체적인 내용이 아니라 API 시그너처를 추출하기 위해 필요한 정보들만 출력된다.

바이너리 코드의 파싱이 완료된 후, “Get Signature” 메뉴를 선택하면 추출된 API 시그너처와 데이터베이스에 저장된 API와의 비교를 통해 특정 API 시그너처를 선택하게 된다. 시그너처 추출 모듈을 통해 복제 기능, 레지스트리 수정 기능, 메일 전파 기능, P2P 전파 기능에 대한 API와 PE 구조를 통해 추출된 API를 비교하여 각 기능에 대한 시그너처를 획득하게 된다. (그림 10)은 데이터베이스와 비교한 후 특정 API 시그너처와 “.rdata” 섹션에서 추출한 특정 데이터를 획득한 후 출력한 화면이다.

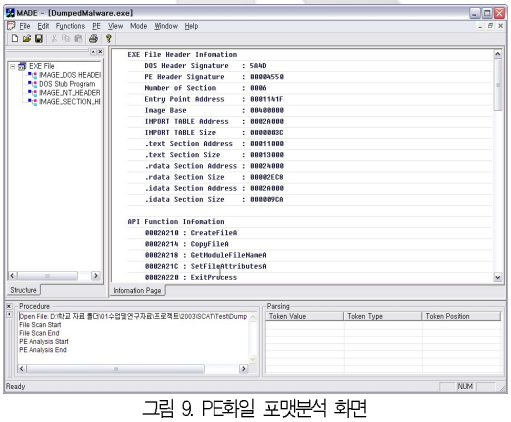

그림 9. PE화일 포맷분석 화면
Fig 9. Display of PE File Format Analysis

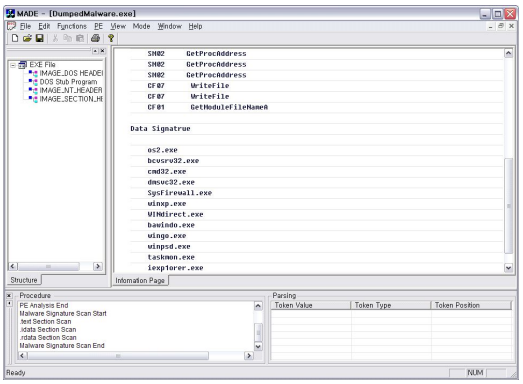

그림 10. 시그너처 추출화면
Fig 10. Display of Signature Extraction

VI. 결 론

악성코드의 피해는 최근 사회 전반적으로 치명적인 피해를 주고 있다. 그러나 현실적으로는 악성코드에 대한 미흡한 대응책으로 인해 보안 체계의 허점으로 작용하고 있다. 본 연구에서는 악성코드에 대한 기본 개념과 기능들에 대해 분석하고, 이를 기반으로 하여 API 시그너처로 분석하는 악성코드 탐지 도구를 설계 및 구현하였다. 기존의 백신이나 기타 악성코드 탐지 방안들은 단순 시그너처 비교를 통한 악성코드 탐지만 가능하기 때문에 신종 악성코드나 변종에 대한 대처가 미흡하지만, 본 논문에서 제안한 도구는 신종 악성코드나 변종에 대해 능동적이고 신속하게 대처할 수 있는 장점이 있다. 또한, 탐지된 악성코드의 기능에 대해 각각 분석하고, 분석된 내용을 사용자에게 보고할 수 있다.

본 논문은 기존 휴리스틱 기반 탐지도구의 성능을 보다 향상 시킬수 있는 방안을 제안 하였다. 그러나 최근의 악성코드의 작성 기법들은 급속도로 발전하고 있으며, 또한 그 기능도 광범위하게 증가하고 있는 추세이다. 특히 실행압축 기법이나 암호화 등이 그 대표적인 예라 할 수 있다. 따라서 악성코드의 시그너처를 탐지하기 이전에 실행압축 해제나 암호화 해제 작업 등이 선행되어야 할 것 이다. 한다. 향후 연구 과제에서는 이들에 대한 연구가 요구된다.

참고문헌

[1] 김효남, "침입기법을 응용한 침입자 역추적 시스템 설계에 관한 연구" 한국 컴퓨터정보학회 논문지 제8권 제3호 (2003) p34-39

[2] 황인선, 박 경우, "이동에이전트를 이용한 침입탐지 모델의 제안" 한국 컴퓨터정보학회 논문지 제9권제1호 (2003) p55-62

[3] ASEC: ASEC Annual Report 2003. Available at http//b2b.ahnlab.com/securityinfo/annual_report /2003/2003_01.html (2003)

[4] Wells, J. "Virus Timeline", Available at http://www.research.ibm.com/antivirus/ timeline.htm (1996)

[5] Bontchev, V. "Marco Virus Identification Problems," Proceedings of the 7th international Virus Bulletin Conference, p175-196, (1997)

[6] Kennedy, M. "Script-Based Mobile Threats", Symantec Antivirus Research Center (2000)

[7] Charlier, B., Swimmer M., and Mounji A. "Dynamic detection and classification of computer viruses using general behavior patterns", 5th International Virus Bulletin Conference, (1995)

[8] Fernandez, F. "Heuristic Engines", 11th Internation Virus Bulletin Conference, p407-44, (2001)

[9] Weaver, N., Paxson, V., and Cunningham, R. "A Taxonomy of Computer Worms", ACM CCS Workshop on Rapid Malcode (2003)

[10] Spafford, E. "The Internet Worm Program: An Analysis". Tech. Report CSD-TR-823 (1988)

[11] West, C. "Advanced communication techniques of remote access Trojan horses on windows operation systems". SANS GSEC Practical Vol. 1.4, Available at http://www.giac.org/practical/ GSEC/ Candid_Wueest_GSEC.pdf (2004)

[12] Skoudis, E., and Zeltser, L. "Malware: Fighting Malicious Cod", Prentice Hall PTR (2003)

[13] Grimes, R "Malicious Mobile Code: Virus Protection for Windows", O'Reilly (2001)

[14] Visual C++ Business Unit. Microsoft Portable Executable and Common Object File Format Specification. Available at http://download.microsoft.com/download/e/b/a/eba1050f-a31d-436b-9281-92cdfeae4b45/pecoff.doc (1999)

저 자 소 개

우 종 우

1978년 서울대학교 농생물학사
1983년 미네소타 주립대 (전산학 석사)
1991년 일리노이공대 (전산학 박사)
1994년~현재 국민대학교 컴퓨터학부 교수
<관심분야> 인공지능, 에이전트, 정보보호

하 경 휘

2003년 국민대학교 컴퓨터학부 졸업
2005년 국민대학교 일반대학원 전산과학과(이학석사)