Journal of The Korea Institute of Information Security & Cryptology(JKIISC) ISSN 1598-3986(Print)
VOL.23, NO.4, August 2013 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2013.23.4.679

악성코드 DNA 생성을 통한 유사 악성코드 분류기법

한 병 진,†‡ 최 영 한, 배 병 철
한국전자통신연구원 부설연구소

Generating Malware DNA to Classify the Similar Malwares

Byoung-Jin Han,†‡ Young-Han Choi, Byung-Chul Bae
The Attached Institute of ETRI

요 약

2013 국가정보보호백서에 따르면, 2012년 민간분야 침해사고 접수·처리 현황 중 해킹사고는 19,570 건으로 2011년에 비해 67.4%가 증가한 수치이며, 해가 갈수록 증가하고 있다. 이러한 증가의 원인으로는 특히 금전적인 이윤 추구와 감염기법의 다양화 등이 꼽히고 있다. 하지만, 악성코드를 분석 하고 대응하기 위한 전문가 수의 증가 속도보다 악성코드의 발전 속도가 빠르기 때문에, 악성코드로 인한 보안위협에 대응하는데 어려움이 있다. 이에 따라, 악성코드 자동분석 도구에 대한 관심이 높아지고 있다. 본 방법은 악성코드 자동분석의 일환으로 악성코드 DNA 생성을 통한 유사 악성코드 분류방법을 제안한다. 제안하는 기법은 기존 자동 분석도구와는 달리, 악성코드의 특성인자를 추출하여 '악성코드 DNA'를 생성하고, 이를 통한 유사도 계산을 통해 악성코드를 분류한다. 본 기법을 사용함으로써, 전문가의 악성코드 분석 시간 절약 및 정확성을 향상 시켜 줄 수 있고, 신뢰성 있는 사전 지식을 전달할 수 있다.

ABSTRACT

According to the national information security white paper 2013, the number of hacking attempt in 2012 is 17,570 which is increased by 67.4% than in 2011, and it has been increasing year after year. The cause of this increase is considered as pursuit of monetary profit and diversification techniques of infection. However, because the development of malicious code faster than the increase in the number of experts to analyze and respond the malware, it is difficult to respond to security threats due to malicious code. So, the interest on automatic analysis tools is increasing. In this paper, we proposed the method of malware classification by similarity using malware DNA. It helps the experts to reduce the analysis time, to increase the correctness. The proposed method generates 'Malware DNA' from extracted features, and then calculates similarity to classify the malwares.

Keywords: Malware, Classification, Similarity, Static Analysis

I. 서 론

지난 3.20 방송·금융사 전산장비 파괴 사태를 비롯해 DDoS 공격, 주요 포털의 개인정보 유출 등 다양한 사건·사고가 발생하고 있다. 2013 국가정보보호백서에 의하면, 한국인터넷진흥원에서 접수·처리한 민간분야 침해사고 접수 처리현황 중 해킹사고는 2012년 19,570건으로 2011년 11,690건 대비 67.4% 증가한 것으로 나타났다[1].

이러한 증가의 원인은 금전적인 이윤 추구와 감염기법의 다양화로 인해 악성코드가 증가하였기 때문으로 분석되고 있으며, 정치목적의 사이버 공격 등 다양한 형태의 공격도 발견되고 있다. 또한, 특정 지역에서만 발견되는 지역화 악성코드가 등장하고 있다.

접수일(2013년 4월 4일), 수정일(2013년 6월7일), 게재확정일(2013년 6월17일)
† 주저자, bjhan@ensec.re.kr
‡ 교신저자, bjhan@ensec.re.kr(Corresponding author)

하지만, 이러한 악성코드 증가추세에 비하면, 국내 기업의 정보보호 투자 비율은 여전히 낮은 수준이다. 방송통신위원회와 한국인터넷진흥원(KISA)가 발표한 2011 정보보호 실태조사에 따르면 현재 국내 기업의 62.6%는 정보보호에 투자를 하지 않고 있으며, 정보화 투자 대비 정보보호 투자비율이 1% 미만인 기업은 무려 82.9%에 달한다[2].

늘어나는 악성코드에 비해 정보보호에 대한 투자가 적은 현상은 악성코드 분석에도 영향을 미친다. 즉, 악성코드 분석에 있어 적은 인원 대비 높은 분석 효율을 내야 하는 것이다. 이에 따라, 악성코드 자동분석 도구들이 주목 받고 있다.

본 논문에서는 이러한 자동 분석 도구에 대한 필요성에 부응하고자, 자동화된 악성코드 분류 도구를 제안한다. 제안하는 '악성코드 DNA 생성을 통한 유사 악성코드 분류기법'은 정적분석을 통해 다양한 특성인자를 추출하여 Malware-DNA(MAL-DNA)를 생성하고, 다른 악성코드와의 유사도를 분석하여 유사 악성코드 끼리 분류한다.

기존 자동 분석도구와는 달리, 단순 시그니처 기반의 분류가 아니라, 악성코드의 특성인자를 추출하고 MAL-DNA를 생성하고, 이를 이용한 유사도 계산을 수행하여 가장 유사한 악성코드 그룹을 찾아 분류한다. 제안하는 기법을 이용하면, 전문가의 악성코드 분석 시간을 절약할 수 있고, 분석의 정확성을 향상 시킬 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 악성코드 유사도 분류 관련 연구를 살펴본다. 3장에서는 제안하는 악성코드 분류기법 및 특성인자를 정의하고, 제안하는 악성코드 분류기법의 적용 방법을 제시한다. 4장에서는 제안한 기법을 검증하고. 마지막으로 5장에서는 결론을 맺는다.

II. 관련 연구

2.1 유사도 판별 기술 연구

악성코드를 유사도 기반으로 분류하려는 다양한 연구가 진행 되고 있으며 크게 바이너리 자체의 유사도를 이용하는 Locality Sensitive Hashing (LSH) 기법과, 악성코드에 포함된 문자열의 유사도를 이용하는 String Match 기법, 악성코드의 통계적 특징을 찾아내는 방법, 악성코드의 메타데이터를 추출하여 그룹화 하는 방법 등으로 나뉠 수 있다.

2.1.1 Locality Sensitive Hashing (LSH)

Locality Sensitive Hashing 즉 부분정보가 반영되는 해싱기법은 일반적으로 알려진 암호학적 해싱기법과는 개념이 다르다. 일반적인 암호학적 해싱기법은 원문에서 1비트라도 차이가 발생하면 전체 해시 값이 바뀌는 성질을 가진다. 하지만, 단지 '1 비트'만 차이나는 두 파일도 전혀 다른 해시 값을 나타내기 때문에 유사도를 판단하는 데는 사용할 수 없다. 이에, 여러 파일 중에서 유사한 파일들을 찾아내기 위한 목적으로 Fuzzy Hashing 이라는 기법을 이용한 ssdeep 이라는 도구가 개발되었다[3][4].

대표적인 LSH 중에 하나인 Fuzzy Hashing 기법은 파일을 일정한 크기의 구역으로 나눈 다음 각각의 구역에 대한 해시 값을 계산하여 하나의 문자열로 합친다. 따라서 동일한 구역에 대한 해시값은 같고, 일부분만 변형된 형태로 나타난다. 이를 이용하여 두 파일의 유사도를 계산하는 방식이다.

ssdeep 도구는 두 가지 해시 알고리즘을 이용하여 각각 계산된 두 개의 해시 값과 구역의 크기정보를 해당 파일에 대한 전체 해시 값으로 생성한다. [그림1]은 샘플파일의 Fuzzy Hashing 값을 텍스트 파일로 저장하는 명령으로, Fuzzy Hashing 결과 값은 "blocksize:hash:hash,filename"의 구조이다. 하단의 명령은 상단에서 구한 Fuzzy Hashing 값을 이용하여 샘플 디렉토리에 포함된 파일 중에 유사도가 60% 이상인 파일을 선택하는 명령이다. 이를 통해 유사한 파일을 얻을 수 있다.

```
C:\ssdeep>ssdeep -b c:\sample\mpgdex.dll > mpgdex_dll_ssdeep.txt
ssdeep,1.1--blocksize:hash:hash,filename
192:a8Eg/uGal4IjQfat+Jkjfat+30sSWKFQmHFGKsYxVLo3WuW89W38G
WcxUWXx:a8EgAl10sSlQWxsYxVQv9qWcxUWXx,"mpgdex.dll"

C:\ssdeep>ssdeep -m mpgdex_dll_ssdeep.txt -brt 60 c:\sample
mpgdex.dll matches mpgdex_dll_ssdeep.txt:m (100)
mpgdex2.dll matches mpgdex_dll_ssdeep.txt:m (99)
```

[그림 1] ssdeep을 이용한 유사도 분석

또 다른 LSH 도구로 simhash 라는 도구가 있다. 이는 Similarity Hashing 이라는 기법을 이용하며 n차 기약 다항식을 이용한 방식으로 해시 값을 생성한다. ssdeep과는 다르게 바이너리 형태의 파일을 해시 결과로 생성하지만 역시 다른 파일과의 유사도 계산에 활용된다[5][6].

2.1.2 String Match (FingerPrint)

FingerPrint[7]는 HBGary社에서 제작한 String을 이용한 유사도 분석 도구다. 입력되는 파일(디렉토리를 입력하면, 포함된 모든 파일)에서 String (ASCII/Unicode) 패턴을 추출하여 XML 형태의 결과로 저장한다. 이 XML 결과 파일은 정보를 누적하여 저장하며, 기존에 검사했던 다른 파일과 유사도를 비교해 준다. 이를 통해 1차적으로 비교할 수 있는 도구로 활용 가능하다.

아래 [그림 2]를 보면, 상단에서 FingerPrint를 이용하여 샘플파일에 포함된 문자열을 기반으로 특성인자를 추출한다. 이후, -c 옵션을 사용하여, 기존에 분석한 샘플 중에서 입력한 샘플파일과 유사한 파일의 정보를 추출한다.

```
C:\FingerPrint\bin>FP c:\SAMPLE\mpgdex.dll
Fingerprint v1.0, Copyright ⓒ 2010 HBGary, Inc. All Rights Reserved.
Scanning 1 file(s)...

Name: mpgdex.dll
Hash: E1DFBEB63DA1C6DEE2F317CA8B3C1792
PE Timestamp                2001-10-25 오전 12:54:07
Linker version              v6.0
DllCharacteristics          00000000
PE Sections                 .text | .rdata | .data | .usdata | .rsrc
File IO                     Win32

Events                      yes
Thread Creation             Generic
CreateProcess               Generic
Compiler                    Microsoft Visual C++ 4.2
Windows socket library      yes
Services                    main
Command shell               Generic
Winsock                     Generic
FPO count                   2
PE Headers                  1

C:\FingerPrint\bin>FP -c c:\SAMPLE\mpgdex.dll
Fingerprint v1.0, Copyright ⓒ 2010 HBGary, Inc. All Rights Reserved.
Scanning 1 file(s)...

mpgdex2.dll, 6F090D25188433E7E12039BC0B41F29E, 100.00
mpgdex.dll, E1DFBEB63DA1C6DEE2F317CA8B3C1792, 100.00
```

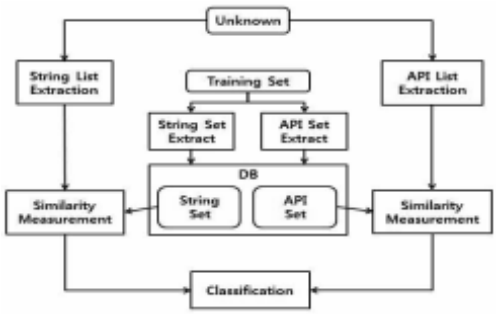
[그림 2] FingerPrint를 이용한 유사도 분석

2.2 기존 악성코드 자동분류 연구

2.2.1 문자열과 API를 이용한 악성코드 자동분류 시스템

문자열과 API를 이용한 악성코드 자동분류 시스템[8]은 정적분석을 통해 바이너리 실행파일에 포함된 문자열을 추출하고, 동적 분석을 통해 실행되는 모든 API와 호출된 횟수를 정리한 후, 기존 악성코드와의 유사도를 측정하여 자동 분류하는 방식이다.

우선, 무작위로 선정된 정상프로그램에서 추출한 공통 문자열을 화이트 리스트로 정의하고, 악성코드 샘플로부터 추출한 Training Set을 악성코드 집합별로 만들어 둔다. 다음으로, 입력된 악성코드의 바이너리 실행파일에 포함된 문자열을 추출한 후, 화이트리스트에 포함된 문자열을 제거한 다음 악성코드 집합별 Training Set과의 유사도를 비교한다.



[그림 3] 문자열과 API를 이용한 자동분류 시스템

API 유사도는 동적 분석을 이용한다. 우선, 악성코드 분류별로 악성코드 실행 시 API 호출 리스트를 호출빈도 순으로 정렬하여 Training Set으로 만든다. 이후, 입력된 악성코드를 실행시켜 발생하는 API를 호출 빈도로 정렬 후, Training Set과 유사도를 비교한다.

마지막으로, 스트링 유사도와 API 유사도의 가중치 작업을 통해 가장 높은 유사도를 가지는 집합으로 분류한다.

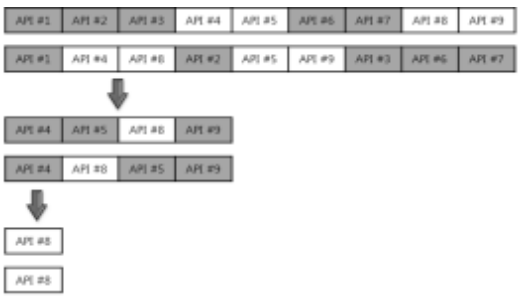
이 방식은, 같은 집합에 포함된 악성코드는 포함된 문자열과 API가 비슷할 것이라는 가정에서 출발한다. 하지만, 악성코드에 포함된 문자열과 API를 빈도순으로 정렬하여 우선적으로 통합하는 과정에서 오류가 발생한다. 악성코드에 포함된 문자열이나 API는 바이너리 안에서 호출된 횟수 혹은 발견된 횟수가 적다고 중요하지 않은 정보가 아니다. 오히려, 중요한 역할을 수행하는 악성 API 및 이에 관련된 문자열은 적은 횟수 호출되고, 일반적인 API와 일반적인 문자열의 횟수가 높을 수 있기 때문이다.

또한, Digital Ninja의 연구결과[9]에서 볼 수 있듯이, 유사도를 판단하는 방식에는 대표 값과 비교하는 Top-Down 방식이 아닌, 전수 비교를 수행하는 Bottom-up 방식이 적합한데, 이 기법은, 미리 악성코드 집합을 만들고, 집합에 대한 Training set을 만드는 과정을 통해 극단적인 Top-down 방식을 사용하여, 정확도 높은 결과를 기대할 수 없는 구조를 가지고 있다.

2.2.2 API 순차적 특징을 이용한 악성코드 변종 분류 기법

API 순차적 특징을 이용한 악성코드 변종 분류기법[10][11]은 악성코드의 IAT 테이블에서 API 리스트를 추출하고, API의 종류 및 호출되는 순차적 특징을 이용하여 악성코드를 분류하는 방법이다.

우선, 악성코드가 아닌 정상적인 프로그램으로부터 API리스트를 DLL별로 저장하여 화이트 리스트를 생성한다. 이후, 입력받은 악성코드에서 추출한 API리스트에서 화이트 리스트를 제거한다. 그 다음, 비교대상이 되는 악성코드에서 추출한 API리스트에서 화이트 리스트를 제거한 후, 입력받은 악성코드의 API 리스트와 동일하게 포함된 API를 추출한 다음 API의 순서를 고려하여 유사도를 계산한다.



[그림 4] API의 순차적 특징을 이용한 악성코드 분류 기법

본 논문은 API 호출의 순차적인 특징을 살펴봄으로써, 악성코드의 변종을 찾아내는 결과를 보였다. 하지만, 유사한 기능을 하는 다른 API 조합이 가능한 경우도 있기 때문에 예상과 달리 유사도가 낮게 계산되는 경우도 존재하였다.

또한, 악성코드 두 개를 입력하여 두 개의 악성코드 간의 유사도만 비교하는 방법이기 때문에, 새로 입력된 악성코드를 분석하여 분류를 찾는데 활용하기 힘든 구조적인 특징을 가지고 있다.

또, API만을 이용하여 분석함으로써, PE 파일 포맷과 String 등에서 제공되는 각종 정보를 활용하지 못하는 단점을 가지고 있다.

유사한 연구로, 악성코드의 통계적 특징 13가지를 추출하여, 데이터마이닝을 이용하는 기법이 있다[12]. 이 기법 또한, 바이너리 유사도만을 고려하기 때문에, 유사한 변종 악성코드는 잘 분류하지만, 전체적인 분류에 활용하기는 힘든 특징을 가지고 있다.

III. 악성코드 DNA를 이용한 유사 악성코드 분류 기법

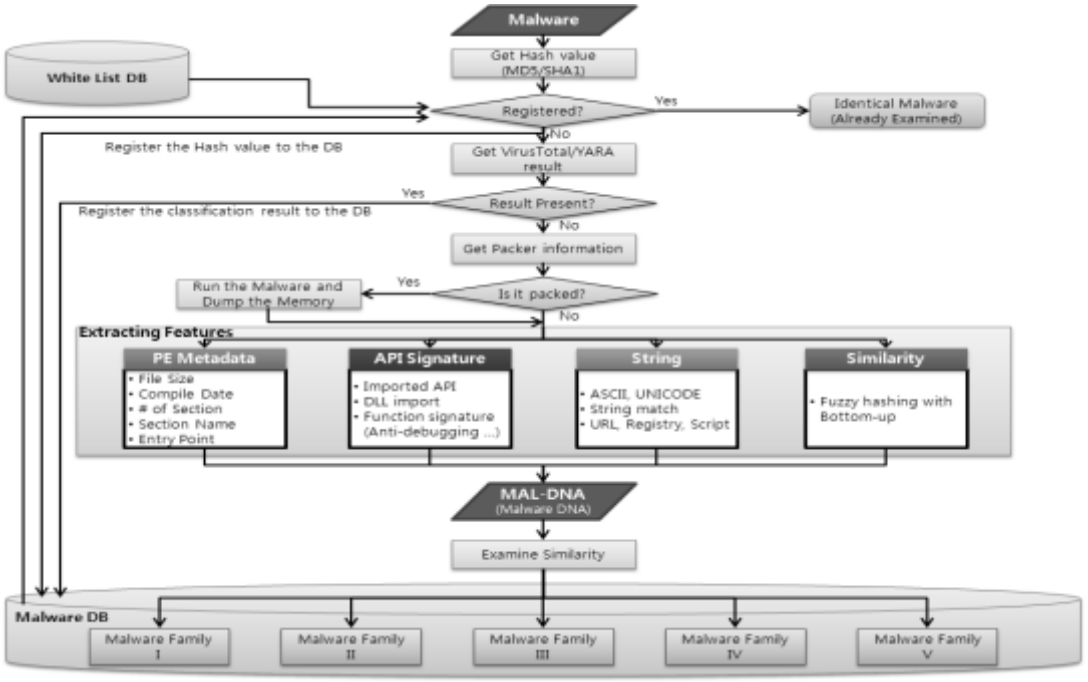
앞서 소개된 관련 연구들은 악성코드 유사도 분석에 다양한 방법론을 제시해 주고 있다. 하지만, 악성코드의 유사도를 분석하는데 있어 다양한 한계점을 가지고 있는데, "악성코드 DNA를 이용한 유사 악성코드 분류 방법"은 기존 기법의 단점을 극복하기 위해 다음과 같은 다양한 요소를 적용하였다.

① API 뿐만 아니라, 다양한 특성인자를 추출하여 유사도 분석에 활용
② 동일한 목적으로 호출되는 API에 대해 같은 특성인자로 묶어서 추출
③ 각 특성인자마다, 고유 값을 설정하여, 입력되는 악성코드 분석 결과를 특성인자의 고유 값을 조합하여 MAL-DNA(Malware-DNA) 산출
④ 유사도 분석 시, MAL-DNA만을 이용한 간편한 유사도 비교
⑤ 계층적 Bottom-Up 방식 도입으로 Top-Down 방식의 성능과, Bottom-Up의 정확성 확보

3.1 제안 기법의 구조 및 절차

본 연구에서 제안하는 유사 악성코드 분류 방법은 악성코드를 입력 하면 다양한 특성인자를 활용하여 악성코드 DNA를 생성하고, 유사도를 계산하여 자동으로 분류를 해주는 시스템이다. 이 도구는 단순 변종 탐지에 그치지 않고, 유사한 악성코드를 제작하는 악성코드 개발자 그룹으로 확장 할 수 있다.

새로 도입한 요소들은 기존 기법대비 성능향상에 기여를 하였는데, 1) API 외에 다양한 특성인자를 추출함으로써 서로 다른 악성코드 간의 연관성을 찾을 수 있기 때문에 악성코드 개발자 그룹으로의 확장이 가능하고, 2) 같은 역할을 수행할 수 있는 API를 같은 인자로 묶어 처리함으로써 세부 API보다는 목적에 의의를 두어 유사도 분석에 유연성을 향상시켰다. 예를 들어 Debugger를 탐지하기 위한 기법은 매우 여러 가지가 있지만, 이를 같은 특성으로 분류하여 유연성을 높인다. 3) MAL-DNA를 생성함으로써 악성코드를 분석 하지 않고도 악성코드의 특성 값을 바로 확인 할 수 있다. 4) MAL-DNA를 통한 유사도 비교를 통해 빠른 속도로 분류할 수 있으며, 5) 계층적 Bottom-Up방식을 사용하여 잘못된 대푯값 오류를



〔그림 5〕 악성코드 자동 분류 절차

해결하여 정확도를 높일 수 있다.

위 〔그림 5〕는 악성코드 입력 후 분류를 수행하는 절차이다. 이 시스템은 악성코드가 입력되면 MD5, SHA1으로 해시 값을 계산하여 화이트 리스트 및 기존 분석 결과 DB에 존재여부를 검사한다. 화이트 리스트에 포함되어 있거나, 이미 DB에 등록된 악성코드는 다시 분류할 필요가 없으므로 바로 종료한다.

다음으로, VirusTotal과 YARA를 이용한 기존 Anti-Virus 제품의 결과를 참조한다. 이미 다른 Anti-Virus 제품에서 분석이 된 경우, 상세 분석을 통한 결과 값이고, 본 연구의 목적은 상세 분석 이전에 악성코드에 대한 각종 정보를 추출하고, 이를 통한 자동 분류를 제공하는데 목적이 있으므로, Anti-Virus 제품의 결과를 반영하여, 악성코드 DB에 분류하여 넣고 프로세스를 종료한다.

위에 해당하지 않고 진행되는 경우는, 아직 DB에 포함되지도 않았고, 다른 Anti-Virus 결과도 존재하지 않는 악성코드이다. 이런 경우가 바로 본 연구에서 제안하는 시스템의 타깃이다. 악성코드 분석가의 상세 분석 이전에 악성코드의 특성인자를 추출하여 MAL-DNA를 생성하고, 자동분류 결과를 산출하여 분석가에게 제공한다.

분석 대상으로 판단되면 우선, Packing 여부를 검사 한다. Packing 된 경우, 각종 정보를 추출하는데 제약이 있기 때문에, Hybrid 분석기법을 이용한다. 즉 Dynamic과 Static분석 기법을 혼합하여 사용하는데, Packing 여부가 알려지면, 해당 악성코드를 메모리에 탑재 후 메모리를 덤프 한다. 이 덤프파일을 일반 Packing 안된 파일과 함께 특성인자 추출 루틴에 포함시켜 특성인자를 추출한다. 이 특성인자를 분석하여, MAL-DNA 를 생성하고, 이를 이용하여 유사도 계산 결과를 통해 악성코드를 분류한다.

3.2 계층적 Bottom-Up 방식

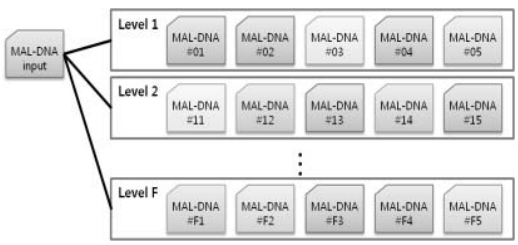
앞선 연구에서 살펴보았듯이, 악성코드 유사도를 이용한 분류기법에는 대표 값과 비교를 수행하는 Top-Down 방식은 대표 값 선정 시 모든 샘플의 특징을 대표하는 값을 선정할 수 없기 때문에 실제 해당 그룹에 속함에도 이를 찾아 낼 수 없는 상황이 발생한다. 예를 들어 50%이상 유사하면 하나의 그룹을 형성한다고 하였을 때 A1을 A그룹을 대표하는 대푯값으로 세웠다. 이때, A2는 A1과 60%의 유사도를 가진다. N3는 새로 들어온 파일인데 A1과는 30%의 유사도 밖에 없어 A그룹에 속하지 못하게 되었다. 하지만, N3와 A2는 60%의 유사도를 가지기 때문에 A그룹으

로 판단할 수도 있다. 이처럼, 대푯값이 가지는 한계 때문에 유사도 비교 방식에 있어 Top-Down 방식은 부적절하고, 전수비교를 수행하는 Bottom-Up 방식이 적절하다.

하지만, Bottom-Up 방식은 치명적인 단점이 있는데, 1:N의 계산이 필요하게 된다는 점이다. 입력되는 악성코드가 증가할수록 유사도 계산에 사용되는 시간이 증가하게 된다.

이러한 단점을 극복하기 위해, 계층적으로 구성된 Bottom-Up 방식을 제안한다. 임계 값을 가지고 계층적으로 구성한 샘플과 계층별로 비교를 통해 유사도를 계산하고 그룹을 분류하는 방식이다. 계층적 Bottom-up 방식은 다음과 〔그림 6〕과같이 구성된다.

등록된 샘플 MAL-DNA는 계층적으로 구성된다. 가능한 모든 부류의 MAL-DNA가 골고루 섞이고, 비율도 적절히 배분된 MAL-DNA 샘플 계층이 구성되면, 이 계층적으로 구성된 샘플을 이용하여 Bottom-up 비교를 수행한다.



〔그림 6〕 계층적 Bottom-Up 비교 방식

이 때, 임계값이 세 개가 사용되는데, 한 계층에 포함되는 MAL-DNA의 개수, 총 계층의 수, 그리고 유사도 값이다. 새로운 악성코드의 MAL-DNA가 들어오면, 한 계층에 포함된 모든 MAL-DNA와 유사도를 계산한다. 이때, 유사도가, 임계값을 넘기는 값이 존재한다면, 그 중에 가장 높은 유사도를 보이는 샘플 MAL-DNA를 선정하여 해당 샘플의 분류정보를 따라 새로운 악성코드를 분류한다.

만약, 임계값을 넘기는 유사도가 존재하지 않으면, 다음 계층의 모든 MAL-DNA와 비교하여 임계값을 넘기는 값이 존재하면 그 분류정보를 따르는 식으로 반복한다. 최종 계층의 MAL-DNA와 비교하여도 임계값을 넘기는 유사도가 나오지 않으면, 새로운 분류로 할당한다.

3.3 특성인자 선정

본 논문에서 제안하는 기법은, 악성코드로부터 다양한 특성인자를 추출하는 것으로 시작한다. 그 중, 악성코드 분석의 가장 기초가 되는 특성인자(Features)를 대분류로 선정하였다. 〔표 1〕는 대분류에 속하는 특성인자의 필요성과 추출방법 그리고, 필요성을 나타낸다.

3.3.1 Hash Value (MD5, SHA1)

정적분석에 있어 해시 값은 해당 파일에 대한 일종의 인덱스(index) 역할을 수행한다. 즉, 해당 파일을 유일하게 식별할 수 있는 값이다.

본 연구에서는 MD5와 SHA1을 사용하여 해시 값을 계산하며, MD5는 특성인자 추출결과를 저장하기 위한 디렉토리 이름으로 사용하여 index로의 역할을 수행한다. 또한, 이 값은 향후에 기존 백신을 이용한 진단명을 알아내기 위한 값으로도 사용된다.

두 가지 해시 알고리즘을 사용하는 이유는 만약의 경우 발생할 수 있는 충돌 가능성에 대비하여, 두 가지의 서로 다른 해시 알고리즘을 사용함으로써, 충돌에 대한 대비를 수행하기 위함이다.

〔표 1〕 특성인자 대분류

특성인자	추출 내용	필요성
Hash Value	바이너리의 해시 값	악성코드의 Index로 활용
AV Result	AV의 악성코드 진단 결과 (VirusTotal, YARA, ClamAV)	이미 진단된 AV의 결과를 분류에 활용
Packer	패킹, 난독화 여부 및 사용된 기술	사용된 Packing 유사도 분류
statistic	Binary의 N-gram에 대해 통계정보 계산(entropy 등)	악성코드 구조의 통계적 특징
PE Analysis	파일크기, Entry point, 섹션정보, Rich Signature 등	각종 부가적인 정보
API	악성행위와 연관된 특정 DLL 및 API 포함여부	악성 의심 행위 가능성 체크
String	바이너리내의 특정 단어 포함여부	가장 기본적인 특징
Similarity	Locally Sensitive Hash 값	기존 유사도 판별기술 활용

3.3.2 Anti-Virus Result

시그니처를 기반으로 악성코드를 탐지하는 방법은 거의 모든 Anti-Virus 제품군에서 적용하고 있는 방식이다. 이중에서 잘 알려진 공개 툴인 YARA와 ClamAV를 사용한다.

YARA는 바이너리파일에서 특정 시그니처 포함여부를 알려주는 시그니처 기반 악성코드 탐지 도구이다. 룰 작성 시에 헥사값 또는 텍스트형식의 값을 지정할 수 있으며, 탐지 시작 지점을 지정할 수 있고, 정규표현식을 사용할 수 있어 다양한 패턴을 생성할 수 있는 툴이다[15].

ClamAV 역시 시그니처 기반으로 악성코드를 탐지하는 공개 Anti-Virus 도구로써, 룰 업데이트 및 사용자의 개별 시그니처 제작이 용이하다는 장점이 있다. 또한, ClamAV와 YARA는 서로의 룰을 상호 호환성 있게 변경 할 수 있는 도구가 많다[16].

본 연구에서는 ClamAV의 룰을 YARA룰로 변경하여 YARA를 이용한 악성코드 진단을 통해 악성코드 유사도 구분의 특성인자로 활용 하고자 한다. 해당 악성코드에 대한 분석이 완료되었기 때문에, 유사도 분류 시 유력한 단서가 될 수 있기 때문이다.

```
rule Trojan_Agent_64822
{
    strings:
        $a0 = { 474554202f }
        $a1 = { 687474703a2f2f }
        $a2 = { 2e7265706c6163652822687474703a2f2f252e2a732229 }
        $a3 = { 7777772e676f6f676c65 }

    condition:
        $a0 and $a1 and $a2 and $a3
}
```

[그림 7] YARA Rule로 표현된 악성코드 시그니처

VirusTotal은 악성코드에 대한 43개 Anti-virus 제품의 진단명을 DB로 유지하며, 사용자의 질의에 응답해주는 시스템으로, 웹기반 서비스로 제공되고 있다[17]. 개발자를 위하여 각종 언어에서 사용할 수 있는 API를 제공하고 있으며, 사용자의 KEY를 제공받아 이를 기반으로 1분에 4차례 질의를 요청할 수 있도록 하고 있다.

VirusTotal은 실제 악성코드를 입력으로 받을 수도 있고, 악성코드의 Hash 값(MD5, SHA1)을 입력으로 받을 수 도 있다. 입력에 대한 결과는 해당 결과 보고서의 링크와, 최초 보고일, 그리고 43개 Anti-Virus의 진단명을 보여준다.

VirusTotal의 결과 역시 이미 보고된 악성코드의 경우에만 결과가 나오기 때문에, 이미 알려진 악성코드에 대한 1차 분류에 활용 될 수 있으며, 알려진 악성코드의 유사도 분류의 기본 비교 값으로 적용 할 수 있다.

3.3.3 Packer

악성코드 제작자는 리버스 엔지니어들에 의해 악성코드가 분석되는 것을 최대한 방해하고자 각종 난독화(Obfuscation) 기법을 사용한다[18]. 널리 사용되고 있는 난독화 기법 중에 하나로 Packing 기법이 있다. 악성코드 제작자는 원래의 실행 코드와 Import 테이블 등을 숨기기 위해 Packing 기법을 활용한다. Packing된 악성코드에는 난독화된 실제 악성코드와 악성코드를 복원하기 위한 unpacking 루틴이 들어있다. 악성코드 실행 시 unpacking 루틴이 동작하여 원래 악성코드를 복원하고, 시작지점을 재설정 하여 실행시키게 된다[19].

PEiD 같은 프로그램은 이러한 Packing 기법 적용여부를 시그니처를 통해 탐지하는 기능을 가지고 있다[20]. 물론, 이 프로그램에서 탐지 할 수 있는 Packing 기법은 이미 알려진 것에 한정되고, 직접 개발한 Packing 기법은 탐지할 수 없다. PEiD의 user_db.txt 파일에는 알려진 패킹도구에 대한 시그니처가 담겨있다. 또한, 해당 파일에는 잘 알려진 컴파일러의 버전 정보도 시그니처로 담겨 있어 악성코드 제작자에 대한 정보를 추출할 수 있다. 다음 [표 2]는 알려진 패킹 툴 목록이다.

또 다른 방법으로 Packing 여부를 알 수 있는 방법은 Debugger, PEView 등을 이용하여, 실제 악성코드를 분석하는 방법이다. 파일 크기에 비해 Import Table이 너무 작거나, Code Body가 너무 작거나, 섹션 이름이 특이한 경우 의심해 볼 수 있고, 여기에, unpacking 루틴이 존재하고, 실제 Import Table을 복원하고, 원래 악성코드를 실행시키기 위해 레지스터를 초기화 하고, 실행지점으로 점프(JMP) 하는 등의 코드가 존재한다면 Packing 여부를 확인 할 수 있다[21]. 하지만, 이 방법은 Packer 별로 적용 방법이 다르기 때문에 범용으로 동작하는 도구를 개발하기는 힘들다.

〔표 2〕 잘 알려진 패킹 툴 목록 (일부)

0	Armadillo	1	ASPack
2	ASprotect	3	BatExe
4	Bat2ex.BDTmp	5	Batlite
6	BitArts.Fusion	7	CryptFF
8	CryptFF.b	9	Crypter
10	CryptZ	11	DebugScript
12	DBPE	13	DoomPack
14	Exeshield	15	Eagle
16	Embedded_CAB	17	Exe2Dll
18	ExeStealth	19	FlySFX
20	JDPack	21	MEW
22	Mmpo	23	NDrop
24	PEncrypt	25	PE_Patch.AvSpoof
26	PECRC	27	PCPEC
28	Pex	29	PE-Crypt.Negn
30	PECrc32	31	PEBundle
32	PE-Crypt.Moo	33	PE-Crypt.UC
34	PE-Crypt.Wonk	35	PE-Pack
36	PE_Patch.Aklay	37	PE_Patch.Ardurik
38	PE_Patch.Elka	39	Pingvin
40	PE_Patch.ZiPack	41	PE_Patch.Upolyx
42	Polyene	43	Stxe
44	tElock	45	Teso
46	Yoda_Crypter	47	ZiPack

3.3.4 Statistics

악성코드의 특징을 찾아내는 접근 방법으로 바이너리의 통계적 특징을 이용하는 방식이 있다. 바로 Entropy를 이용하는 방식이다〔22〕.

PE 파일은 구조상 0x00 바이트가 가장 많이 존재하도록 되어있고, 중간에 코드가 존재하기 때문에 적당한 수준의 Entropy를 가진다. 하지만, Packing 기법을 사용하면, Packing된 섹션의 경우 원래코드를 Packing 하여 다른 코드로 대체되었기 때문에, 0x00의 빈도가 극도로 줄어들거나, 특정 섹션의 경우 내용이 모두 0x00으로만 채워져 있는 경우가 발생하게 되고, 이에 따라서 통계적 특성이 변하게 된다.

M개의 기호를 표현할 수 있는 정보원 X에 대한 Entropy H(X)를 구하는 식은 (1)과 같다〔22〕.

$$H(X)=\sum_{i=0}^{M-1}P(x_i)\log_2[\frac{1}{P(x_i)}] \quad (1)$$

이때, 정보원 X를 1바이트로 한다면, 1바이트는 256개의 다른 기호를 표현할 수 있고, 엔트로피의 최대값은 모든 기호마다 출현 비율이 같을 경우 이므로, 식 (1)에 의해, 1바이트로 표현할 수 있는 가장 큰 Entropy는 8이다. 이를 이용여 일반적인 Packer 판별에 사용할 수 있다. 섹션별 Entropy를 계산하여 그 값이 극한 값인 8이나 0과 같거나, 극한에 가까운 섹션이 있으면 Packer에 의한 PE 파일이 변조를 의심 할 수 있으며, 이는 악성코드 유사도를 구분하기 위한 특성인자로 활용될 수 있다.

3.3.5 PE Analysis

PE 파일 포맷은 PE파일에 대한 다양한 정보를 포함하고 있다. 이러한 PE 포맷에서 가능한 많은 메타데이터를 수집하여 악성코드의 유사도를 구분하기 위한 특성인자로 활용할 수 있다. 우선 파일의 크기를 대, 중, 소로 구분하여 특성인자로 활용한다. 악성코드가 수행하려는 행위의 복잡도와, 위장 여부 등에 따라 크기가 달라지며, 유사성을 구분 할 수 있는 특성인자가 될 수 있다.

본 연구에서 다루는 악성코드는 PE 파일이다. 하지만, PE는 다양한 서브시스템 형태를 가질 수 있다. PE의 형태에 따라 적합한 Malware 유형이 결정 될 수 있다. 따라서 이 정보는 악성코드의 유사도를 구분하기 위한 특성인자로 활용할 수 있다.

PE 파일은 여러 개의 섹션으로 나누어져 있는데. 파일마다 섹션의 이름, 개수, 구성, 크기가 모두 다르다. 하지만, 특정 Packer 등이 사용된 경우나, 일반적인 사용의 경우 섹션의 이름 및 개수 등이 유사하게 결정되는 특징을 가지고 있다. 따라서 섹션의 특징을 중요한 특성인자로 활용 할 수 있다.

PE 파일의 Entry Point는 일반적으로 실행코드가 들어있는 섹션(주로 .text)의 처음이 되지만, Anti-debugging, Packing 등의 기법을 사용한 경우는, Entry Point가 일반적인 곳과 다른 경우가 발생한다. 이 또한, 악성코드의 유사도를 계산하기 위한 중요한 특성인자로 활용 할 수 있다.

PE 파일 포맷에는 파일이 생성된 시간(컴파일시간)이 포함되게 되는데, 특정 개발도구를 사용하거나, 일부 악성코드의 경우 해당 부분을 0으로 바꾸어 놓는 경우가 있다. 이 또한 추가적인 정보를 제공 할 수 있기 때문에 특성인자로 활용할 수 있다.

또한, PE 파일 포맷에는 DOS Stub과 PE Header시작지점 사이에 Rich Signature라고 불리는 정보가 암호화 되어 저장되어 있다〔23〕〔24〕. MS의 Linker에 의해 생성되는 이 정보는 Undocu-

mented 상태이며, 라이브러리 파일 생성 시 사용된 컴파일러의 버전(링커의 버전)을 나타내고 있다. Rich Signature에는 compid 라고 불리는 컴파일러 버전 정보 값이 여러 개 포함되어있는데, 각각은 해당 PE파일에 정적으로 링크된 라이브러리들(예를 들면 kernel32.lib)의 컴파일러 버전을 나타내고, 가장 마지막의 컴파일러 버전은 해당 PE가 개발된 환경 정보를 나타낸다. 따라서 이 Rich Signature의 값을 사용하면, 해당 악성코드를 개발한 환경과 포함된 라이브러리의 컴파일러 버전을 알아낼 수 있다. 이는 유사 악성코드를 식별하기 위한 하나의 특성인자로 활용할 수 있다.

Rich Signature는 다음과 같이 구성되어 있다. 위치는 앞서 언급한 대로 DOS Stub과 PE Header 사이이며, Rich라는 ASCII 코드가 포함되어 있고, 암호화 되어있다.

```
00000080 5462EF9B 100381C8 100381C8 100381C8 Tbï  ...È...È...È
00000090 37C5EFC8 110381C8 37C5FCC8 120381C8 7ÅïÈ...È7ÅüÈ...È
000000A0 37C5FAC8 0B0381C8 100380C8 C90381C8 7ÅúÈ...È..€ÈÉ..È
000000B0 37C5ECC8 330381C8 37C5FDC8 110381C8 7ÅìÈ3..È7ÅýÈ...È
000000C0 37C5F9C8 110381C8 52696368 100381C8 7ÅùÈ...ÈRich...È
```

〔그림 8〕 PE파일 내의 Rich Signature

Rich라는 ASCII 코드 뒤에 DWORD(4byte)가 암호 키이다. 암호 키를 이용하여 0x80부터 Rich 문자열 앞부분인 0xC7까지 XOR(^)을 이용해 암호를 해제 하면, 다음과 같다.

```
00000080 44616e53 00000000 00000000 00000000 DanS............
00000090 27c66e00 01000000 27c67d00 02000000 'Æn.....'Æ}.....
000000A0 27c67b00 1b000000 00000100 d9000000 'Æ{.........Ù...
000000B0 27c66d00 23000000 27c67c00 01000000 'Æm.#...'Æ|.....
000000C0 27c67800 01000000 52696368 100381c8 'Æx.....Rich...È
```

〔그림 9〕 해독된 Rich Signature

해독된 부분을 살펴보면 시작부분에 DanS 라는 ASCII 코드가 보이는데, 이것이 Rich Signature의 시작을 알리는 ASCII 코드이다. DanS를 포함한 3개의 0x0000로 된 DWORD를 지나면, 2개의 DWORD를 사용하여 compid와 counter가 담겨있다. 즉, Rich Signature의 구조는 다음과 같다.

'DanS' ^ key	key	key	key
compid1 ^ key	count1 ^ key	compid2 ^ key	count2 ^ key
...	...	...	...
compidn ^ key	countn ^ key	'Rich'	key

〔그림 10〕 Rich Signature의 구조

즉, 이 경우 해당 PE에 사용된 컴파일러 버전 compid는 0x0078C627로써, 이는 "Version 8.00.50727, Microsoft Visual Studio 2005"를 가리킨다. 특히, compid의 하위 2byte는 0xC627로써, 50727과 일치함을 볼 수 있다.

3.3.6 API Signature Tracing

PE 파일의 IAT 테이블을 조회하여, 로드된 DLL과 사용하게 될 API의 목록을 수집한다. 로드된 DLL목록 및 API 목록의 양과 종류는 주요 특성인자로 활용할 수 있다. 만일, 파일의 크기에 비해 로드되는 DLL의 수가 적고, 사용되는 API가 적다면, Packing 등을 통해 실제 실행코드를 숨겨 놓았을 수 있다. 또한, 유사한 분류의 악성코드는 유사한 행위를 하기 위해 거의 같은 DLL 및 API를 사용해야 하므로, 중요한 특성인자로 활용할 수 있다.

악성코드의 행위들은 몇몇 API들의 연속된 호출로 특정지어질 수 있다. 이를 Tracing 하면서 이러한 행위들을 탐지하면, 특정 유형의 악성코드 여부나 제작자의 성향, 제작방법의 유사도 등을 측정할 수 있는 특성인자로 활용 할 수 있다. 본 연구에서는 Malware Analyzer〔25〕를 참조하여 특성인자를 추출하였으며, 이러한 특성인자는 〔표 3〕과 같다.

〔표 3〕 API Signature Tracing 항목

Tracing 항목	의 미
Anti Debugging	Anti-Debugging 탐지
File System Activity	파일조작 탐지
System Hook Calls	시스템 후킹여부 탐지
Keyboard Hook alls	키보드 후킹여부 탐지
Rootkit	루트킷 특성 형태 탐지
DEP (Data Execution Prevention) Setting Change	데이터 실행방지 메모리 보호 설정을 변경하거나 우회하는 악성코드 탐지
DLL injection	DLL Injection 시도 탐지
Network	네트워크 사용여부 탐지
Privilege Escalation	권한상승 시도 탐지
Internet	인터넷 접근 탐지
Anti Process Dump	프로세스 덤프를 차단 확인
Service Register	윈도우 서비스 접근 탐지
Process Creation	프로세스 생성 또는 쉘 실행 등을 탐지
TLS aware	스레드 저장공간 접근 탐지
Clipboard aware	클립보드의 내용 접근 탐지
Process Enumeration	프로세스 목록에 접근 여부 탐지

3.3.7 SSDEEP

악성코드를 분류하는데 파일 자체의 유사도를 활용할 수 있다. 파일 자체의 유사도를 비교하기 위한 기술로 가장 유명한 것이 Fuzzy hashing 기술이다.

ssdeep을 이용한 Fuzzy Hash 값을 사용할 때는 유의할 점이 있는데 Top-Down 방식이 아닌 Bottom-up 방식을 사용해야 한다는 점이다.

Top-down 방식은 특정 악성코드 X와 유사한 악성코드를 찾는 과정에 있어, 대표적인 악성코드의 샘플과 비교하여 가장 유사한 샘플과 같은 유형으로 진단을 내리는 방식이고, Bottom-up 방식은 모든 악성코드의 ssdeep 파일을 비교하여 가장 높은 유사도를 보이는 악성코드를 찾고 그 악성코드의 분류 유형에 따르는 방식이다. 두 방식의 차이점은, 분류 유형 선택 방식이다. 같은 유형으로 분류된 악성코드들이라고 해서 파일의 바이너리 자체가 유사한 형태를 띄는 것은 아니기 때문에, 특정 분류의 대표적인 악성코드로 지목된 샘플과의 유사도 비교는 좋은 결과를 낼 수 없다.

Digital Ninja[9]의 연구 결과는, Bottom-Up 방식의 당위성을 재차 확인할 수 있다. ssdeep을 이용한 유사도가 80% 이상 나온 파일들은 모두 같은 유형의 악성코드지만(Bottom-up 방식), 같은 유형의 악성코드들에 대한 ssdeep 유사도 분석 결과는 천차만별이다(Top-down 방식).

3.3.8 String

Strings는 PE 파일 및 Object 파일 내에서 ASCII 문자열 뿐 아니라, 일반적으로 쉽게 찾아내기 힘든 Unicode 문자열을 추출하는 도구이다[26]. ASCII로 추출한 문자열에는 Import Table에 존재하는 DLL 파일 정보와 호출되는 API 목록, 각 섹션의 이름 등이 포함되어 있다. Unicode 문자열에는 ASCII에서는 볼 수 없었던 각종 메시지, URL, 레지스트리 주소, 파일 경로, 스크립트 등이 포함되어 있는 경우가 종종 발견된다.

String을 통한 분석은 매우 광범위하기 때문에 다양한 특성인자를 추출할 수 있다. 예를 들어 정상적인 PE파일에는 대부분 존재하게 되는 "!This program cannot be run in DOS mode."문자열의 경우 없는 경우, 제작자가 PE파일 포맷에 무엇인가 조작을 가했을 가능성이 높고 이는 제작자의 성향이나, 제작방법의 유사도를 측정할 수 있는 특성인자로 사용될 수 있다.

또한, 일반적인 경우 URL 주소, 레지스트리 주소, 파일 경로, 스크립트 등이 포함되는 경우는 많지 않기 때문에, 이를 각각 특성인자로 선택하여 악성코드의 유사도를 측정할 수 있다.

3.4 악성코드 DNA (MAL-DNA, Malware DNA) 부여 방법

악성코드 DNA를 생성하기 위해 각각의 특성인자별로 4바이트씩 할당한다. 4바이트는 0x00000000부터 0xFFFFFFFF의 4,294,967,295가지 특성인자를 표현 할 수 있는 32개 비트로 이루어져 있다. 이중 최상위 8비트는 특성인자 대분류에 활용하고, 그 다음 8비트는 특성인자 중분류에 활용하고, 그 다음 16비트는 소분류를 표현한다. 중분류와 소분류를 표기하는 방법은 대분류별로 다르다. 아래 [표 4]는 특성인자 대분류별 소분류 설정방법이다.

예를 들어 Packer 종류 특성인자는 0x02부터 시작한다. 만약 Packer의 종류를 알아낸 방법이 PEiD DB를 이용한 것이라면 중분류로 0x0200가 할당된다. 여기에 만약 Packing 기법으로 "ASPack"이 사용 되었다면, [표 2]에 의해 소분류 0x0001에 해당되어, MAL-DNA 조각은 0x02000001이 된다. 만일 "ZiPack"이 사용되었다면, 소분류 0x002F에 해당하여, MAL-DNA 조각은 0x0200002F가 된다.

또한, PE 메타데이터 정보 중에, 파일 생성시간 정보의 경우를 살펴보자, 상위 소분류에 할당된 8비트는 수치의 정상여부이고, 하위 소분류에 할당된 8 비트로 표현할 수 있는 수의 범위는 0~255 이다. 32비트로 표현된 시간정보를 8비트로 표현하기 위해 24 비트를 Right shift 수행한다. 이를 통해 194일 범위내의 시간은 같은 값을 갖게 된다. 이 단위는 악성코드의 변종 악성코드가 생성되는 범위를 표현하기 적합하고, 시간을 1byte로 표현 가능하여 선정하였다. 예를 들어 2009년 7월 7일경 생성된 파일의 시간 정보는 1,248,652,800 이다. 이는 0x4A6CEE00이고, 이를 24비트 Right Shift를 수행하면 0x4A(74)가 되고, 이는 정상적인 시간이다. 따라서 MAL-DNA는 0x0305004A로 설정된다. 만일 악성코드 제작자가 시간정보를 0으로 설정했다면, MAL-DNA는 0x03050100으로 설정된다.

〔표 4〕 MAL-DNA 부여 기준

대분류 (8-bit)	중분류 (8-bit)	소분류 (16-bit)	
AV Result (0x01)	악성코드유형 (유형별 할당, 0x00~0F)	진단결과가 있는 AV수	
Packer (0x02)	Packer진단방법 (진단방법별 할당, 0x00~0F)	리스트에 정리된 Packer ID	
PE Metadata (0x03)	파일 크기 (0x00)	대, 중, 소	
	MAGIC 넘버 (0x01)	WinGUI, WinConsole, WinDriver, EXE, DLL	
	섹션 개수 (0x02)	섹션 개수	
	섹션 이름 (0x03)	특이 섹션이름 존재여부	
	EntryPoint (0x04)	EntryPoint 이상여부	
	파일생성시간 (0x05)	이상여부 (8bit)	시간 (8bit,192일)
	Rich Signature (0x06)	컴파일러버전 (compid)	
API Signature (0x04)	Anti Debugging (0x00)	Anti Debugging API별 비트체크	
	File 처리 (0x01)	File 처리 API별 비트체크	
	System 후킹 (0x02)	System 후킹 API별 비트체크	
	Keyboard 후킹 (0x03)	키보드 후킹 API별 비트체크	
	RootKit (0x04)	RootKit API별 비트체크	
	DEP 설정변경 (0x05)	DEP 설정변경 API별 비트체크	
	DLL Injection (0x06)	DLL Injection API별 비트체크	
	Network 사용 (0x07)	Network 사용 API별 비트체크	
	권한 상승 (0x08)	권한 상승 API별 비트체크	
	Internet 접근 (0x09)	Internet 접근 API별 비트체크	
	메모리 Dump 방지 (0x0A)	메모리 Dump 방지 API별 비트체크	
	서비스 등록 (0x0B)	서비스 등록 API별 비트체크	
	프로세스 생성 (0x0C)	프로세스 생성 API별 비트체크	
	쓰레드 저장소 (0x0D)	쓰레드 저장소 접근 API별 비트체크	
	클립보드 접근 (0x0E)	클립보드 접근 API별 비트체크	
	프로세스 목록 (0x0F)	프로세스 목록 API별 비트체크	
String Match (0x05)	This Program (0x00)	"!This Program..." 여부	
	URL 주소 (0x01)	URL 개수	
	레지스트리 (0x02)	레지스트리 개수	
	파일경로 (0x03)	파일경로 개수	
	스크립트 (0x04)	스크립트 라인 수	
Similarity (0x06)	악성코드 유형 (유형별 할당, 0x00~0F)	가장 높은 유사도	

한편, Anti-Debugging API Signature Tracing 특성인자는 0x04부터 시작한다. Anti-Debugging에 사용되는 API는 여러 가지가 있으며, 이들은 중복하여 사용할 수도 있고, 단독으로만 사용 될 수도 있다. 각 API를 하위부터 비트하나로 표현한다. 즉, 만일 IsDebuggerPresent() API가 발견된 경우 최하위 비트를 1로 세팅하고, GetTickCount()가 발견된 경우 두 번째 비트를 1로 세팅한다.

이에 따라 구분하면, 0x040000D5 이라는 DNA 조각은 Anti-Debugging 시도가 발견되었으며, API는 IsDebuggerPresent(), DebugActiveProcess(), CheckRemoteDebuggerPresent(), GetTickCount(), NtYieldExecution()가 사용되었고, 0x04000041 이라는 DNA 조각은 An-

ti-Debugging 시도가 발견되었으며, API는 IsDebuggerPresent(), GetTickCount()가 사용되었음을 알려준다.

3.5 악성코드 DNA 유사도 계산방법

MAL-DNA의 근본적인 사용 목적은 유사도를 계산하기 위함이다. 악성코드 분류는 유사도 계산을 근간으로 하기 때문이다. 유사도를 계산하는 방법은 유사도 가중치와 특성인자별 유사도 계산식을 바탕으로 계산된다. 각 특성인자에 대한 유사도 가중치는 다음 〔표 5〕과 같다.

AV 결과가 있는 경우, 해당 결과를 바로 유사도 분류에 활용하기 때문에, 100% 가중치를 반영하고, AV 결과가 없는 경우에는 두 MAL-DNA의 조각이 포함된 합집합의 가중치를 모두 합한 다음, 공통 MAL-DNA 조각에 대한 가중치에 대한 비율로 나타낸다. 유사도 계산식은 (2)과 같다.

〔표 5〕 유사도 가중치

유사도 계산 방법	대분류 혹은 중분류	가중치
1	AV-Result (0x01)〔대〕	100
	합 계	100
2	Packing (0x02)〔대〕	5
	파일 크기 (0x0300)〔중〕	2
	MAGIC 넘버 (0x0301)〔중〕	3
	섹션 개수 (0x0302)〔중〕	3
	섹션 이름 (0x0303)〔중〕	3
	EntryPoint (0x0304)〔중〕	3
	파일생성시간 (0x0305)〔중〕	2
	Rich Signature (0x0306)〔중〕	3
	Anti Debugging (0x0400)〔중〕	5
	File 처리 (0x0401)〔중〕	4
	System 후킹 (0x0402)〔중〕	4
	Keyboard 후킹 (0x0403)〔중〕	3
	RootKit (0x0404)〔중〕	4
	DEP 설정변경 (0x0405)〔중〕	2
	DLL Injection (0x0406)〔중〕	4
	Network 사용 (0x0407)〔중〕	2
	권한 상승 (0x0408)〔중〕	4
	Internet 접근 (0x0409)〔중〕	4
	메모리 Dump 방지 (0x040A)〔중〕	2
	서비스 등록 (0x040B)〔중〕	2
	프로세스 생성 (0x040C)〔중〕	3
	쓰레드 저장소 (0x040D)〔중〕	2
	클립보드 접근 (0x040E)〔중〕	2
	프로세스 목록 (0x040F)〔중〕	3
	This Program 문구 (0x0500)〔중〕	2
	URL 주소 (0x0501)〔중〕	5
	레지스트리 (0x0502)〔중〕	5
	파일경로 (0x0503)〔중〕	2
	스크립트 (0x0504)〔중〕	2
	Binary 유사도 (0x06)〔대〕	10
	합 계	100

$$S_{total} = \frac{\sum_{f \in (A \cap B)} S_f}{\sum_{f \in (A \cup B)} W_f} \times 100 \quad (2)$$

유사도는 수치를 이용하여 〔표 6〕을 참조하여 유사 여부를 판단한다. 〔표 6〕의 값은, 실험에 의해 얻어진 값으로 False Positive와 정탐률을 조정하는 과정에서 작성되었으며, 사용하는 목적과 해석에 따라 변경하여 사용할 수 있다.

〔표 6〕 유사도 판단 기준

Min(S_{total})	Max(S_{total})	유사 여부
0	30	관계없음
30	50	일부 유사
50	100	유사

악성코드의 유사도를 계산하는 예를 보자, 예를 들어 두 개의 악성코드에 대한 MAL-DNA의 일부분이 각각 〔0x02000001 0x0305004A 0x040000D5 0x05000000〕와 〔0x0200002F 0x0305004B 0x04000041 0x05000000 0x05010005〕 일 때 유사도를 측정해 보자.

유사도를 비교할 때에도 대분류와 소분류가 각각 사용되며, 대분류 별로 유사도 계산법이 다르다. 이 예시에서 제시된 가중치의 총합은 19 이다.

우선, Packer 유사도는 다음과 같이 계산한다. Packer 유사도의 가중치는 5가 할당되어 있고, 이 가중치에 대한 분배는, 대분류에 50%, 소분류에 50%의 가중치를 분배한다. 두 악성코드에 대한 MAL-DNA의 조각 중에 Packing 항목은 0x02000001 와 0x020002F이다. 이 둘은 상위 8비트에 의한 첫 번째 바이트 값이 동일하기 때문에 동일한 대분류 특성인자를 가짐을 확인 할 수 있고, 두 번째 바이트가 동일하므로 Packer 여부를 알아낸 방법도 동일하다. 나머지 하위 2 바이트 Packer의 종류에 대한 일련번호이다. 따라서 전체 유사도를 구하는 식은 다음과 같다.

$$S_{packer}(A,B) = W_{packer}(0.5 \times isSame(A_{bc}, B_{bc}) + 0.5 \times isSame(A_{sc}, B_{sc})) \quad (3)$$

위 (3)을 이용하면, 두 악성코드 MAL-DNA 조각은, Packer에 대한 대분류는 같지만, 소분류가 다르기 때문에. 가중치인 5의 절반 값인 2.5를 획득 할 수 있다.

다음으로, 파일 생성 시간의 유사도는 다음과 같이 계산 될 수 있다. 가중치는 2가 할당되어 있고, 가중치에 대한 분배는 중분류 없이 소분류에만 가중치가 분배된다. 하나의 악성코드가 발생하고, 거기서 파생된 다른 악성코드가 생성되는 기간은 대략적으로 1년 범위 내에 들어간다고 볼 수 있다. 194일로 표현되는 시간 값에 ±194일의 오차를 적용하면 대략 1년 범위가 나올 수 있다. 따라서 0x0305004A의 경우, 0x0305004A와는 100% 일치하여 가중치 2가 계산되고, 0x03050049 또는 0x0305004B의 경우는 50% 일치하여 가중치 1이 계산된다. 따라서 두 악성코드간의 파일생성시간 유사도는 1이다.

여기서, 한 가지 고려해야 할 사항은, 같은 제작자라고 해서 비슷한 시기에 비슷한 악성코드를 작성한다는 보장은 없다. 하지만, 컴파일 시간 정보는 제작자가 임의로 수정할 수 있기 때문에, 컴파일 시간을 알 수 없게 하기 위하여 값을 0으로 변경하거나, 특정일자로 변경하는 행위가 있을 수 있다. 이 경우, 일반적인 경우에 비하여 동일한 값이 나올 수 있는 확률이 높아지기 때문에, 이 비교는 의미가 있다.

Anti-debugging 기법 유사도는 다음과 같이 계산한다. Anti-debugging의 가중치는 5가 할당되어 있고, 이 가중치에 대한 분배는 중분류에 25%, 소분류에 75%의 가중치를 분배한다. 두 개의 악성코드에 대한 MAL-DNA의 조각 중에 Anti-Debugging 항목은 0x040000D5 와 0x04000041이 있다. 이 둘은 상위 2바이트에 의한 네 개의 헥사값이 동일하기 때문에 동일한 대분류 및 중분류 특성인자를 가짐을 알 수 있고, 나머지 하위 16비트를 Jaccard Index기법을 이용하여 비교한다[27]. 전체 유사도를 구하는 식과, Jaccard Index를 구하는 식은 다음과 같다.

$$S_{debug}(A,B) = W_{mc-debug}(0.25 \times isSame(A_{mc}, B_{mc}) + 0.75 \times Jaccard(A_{sc}, B_{sc})) \quad (4)$$

$$Jaccard = \frac{N(A_{sc} \cap B_{sc})}{N(A_{sc} \cup B_{sc})} \quad (5)$$

위 (4)과 (5)을 이용하면 Jaccard Index는 전체 인자 5개중에 2개가 같으므로 40%가 같음을 알 수 있다. 이 40%에 0.75를 곱하면 30%가 되고, 여기에 중분류 일치로 인한 25%를 더하면 55%가 나온다. 여기에 Anti-debugging의 가중치는 [표 5]에 의해 5이기 때문에 Anti-debugging으로 인한 유사도는 2.75점을 획득할 수 있다.

또한, 두 악성코드 모드 "!This program cannot be run in DOS mode." 문자열을 포함하기 때문에. 유사도 점수 2점을 획득 할 수 있다.

이제, 전체 유사도를 계산하는 (2)에 대입해 보면, 분모인 전체 가중치는 19, 분자인 획득한 유사도 점수는 8.25 = 2.5 + 1 + 2.75 + 2 이다. 따라서 43.42 = 8.25 / 19 × 100을 얻을 수 있다. 유사도 점수 판단기준을 [표 6]에서 찾아보면, 2단계에 해당하는 값으로, 일부 유사한 악성코드임을 알 수 있다. 즉, MAL-DNA에 의해, 두 악성코드는 비슷한 행위(Anti-Debugging)를 수행하고, Packing 기법을 사용하였으며, 비슷한 시간대(194일 전후)에 제작되었기에 '일부 유사'하다고 판단 할 수 있다.

예시의 MAL-DNA는 일부정보만을 가지고 만들어 짧고 단순하게 해석되지만, 실제 악성코드를 분석하면 보다 복잡한 MAL-DNA가 생성될 것이다. 하지만, MAL-DNA를 한번 생성하면, 유사도 계산은 단순하게 계산할 수 있으므로, 새로운 악성코드가 입력되었을 때 유사도 계산을 통한 악성코드 분류는 짧은 시간에 가능하다.

이처럼, 특정 악성코드에 대한 MAL-DNA를 생성하면, 계층적 Bottom-UP 방식을 이용하여 이미 생성된 MAL-DNA와 비교를 수행한다. 이후, 3.2절의 내용처럼 임계값 이상의 유사도를 가지는 유사도 집합으로 분류한다.

IV. 검 증

본 연구에서 제안한 악성코드 DNA 생성을 통한 유사 악성코드 분류기법의 성능을 검증하기 위하여 4개 분류 각 100개의 악성코드를 이용한 분류 실험을 하였다. 실험은 사전 단계와 분류 단계로 나누었으며, 사용된 악성코드 그룹은 다음과 같다.

우선, asprox, torpig, mydoom, dirtjump 의 네 가지 그룹을 VirusTotal에서 검색하여 358개의 악성코드를 수집한다.

사전 단계는 각 그룹의 악성코드의 MAL-DNA를 생성한 후 해당 그룹 폴더에 모아 놓는다. 따로 학습

등의 단계가 필요하지 않고, 분석을 통한 MAL-DNA를 그룹별로 분류해 놓는다.

분류 단계는 MAL-DNA를 이용하여 어느 그룹과 유사한지를 판단한다. 계층적 Bottom-Up 기법을 활용하기 위하여 각 그룹의 MAL-DNA를 5계층에 고르게 분배한다, 입력된 악성코드의 MAL-DNA를 제일 아래 계층에 속한 MAL-DNA와 유사도를 비교한 후, 유사(점수 50이상)한 그룹을 선정한다. 이때, 유사한 그룹이 선정되지 않으면, 다음 계층의 악성코드와 비교하게 된다. 최종적으로 유사한 그룹은 복수개가 선택될 수 있다.

실험 결과는 아래 〔표 7〕과 같다. 표를 보는 법은 다음과 같다. 입력으로 asprox로 알려진 악성코드 98개를 입력했을 때, 결과로 asprox 악성코드와 유사(50점 이상)하다고 나온 결과가 81개이다.

〔표 7〕 실험결과

실험결과		결 과			
		asprox	torpig	mydoom	dirtjump
입력	asprox(98)	81	0	0	0
	torpig(99)	0	72	6	0
	mydoom(90)	0	1	76	0
	dirtjump(61)	0	0	14	59

정탐률과 신뢰도를 측정하기 위해서는 조건부 확률을 사용하는데, 입력으로 특정 그룹의 악성코드(예를 들어 asprox)를 넣은 사건을 A라고 하고, 결과로 입력된 그룹이 나오는 사건을 B(위의 예에서 asprox)라고 하면, 정탐률은 P(B|A), 즉 입력이 A였을 때 결과가 B일 확률로 나타낼 수 있고, 신뢰도는 P(A|B), 즉 결과가 B 일 때, 입력이 A였을 확률을 나타낸다.

〔표 8〕 정탐률, 오탐률, 신뢰도 (%)

실험결과	결 과			
	asprox	torpig	mydoom	dirtjump
정탐률 P(B|A)	82.7	72.7	84.4	96.7
오탐률 1-P(B|A)	17.3	27.3	15.6	3.3
신뢰도 P(A|B)	100	98.6	79.2	100

검증 결과 정탐률, 오탐률, 신뢰도는 〔표 8〕과 같이 계산되었다. asprox와 같은 경우 제안한 기법으로 높은 신뢰도로 식별이 가능한 MAL-DNA가 생성된 것으로 분석되며, mydoom의 경우 다른 그룹의 속한 악성코드의 MAL-DNA와 비슷한 결과를 보임으로써 신뢰도가 낮게 계산되었다.

V. 결 론

본 연구에서는 악성코드 자동분석 도구의 일환으로써, 악성코드 분석 및 대응 인력의 부족으로 인한 보안위협 대응의 어려움을 극복하기 위하여 악성코드 DNA 생성을 통한 유사 악성코드를 분류 기법을 제안하였다.

본 연구에서 제안하는 기법은 기존 자동 분석도구와는 달리, 단순 시그니처 기반의 분류가 아니라, 악성코드의 특성인자를 추출하여 악성코드의 DNA (MAL-DNA)를 생성하고, 이를 통한 유사도 계산을 통해 악성코드를 분류한다.

이를 위해, HASH 값, AV 결과, Packer정보, PE Metadata, API Signature, String, Similarity 등 20여 가지 대/중/소분류의 특성인자를 추출하고, 각각의 특성인자를 MAL-DNA 조각으로 생성한다.

악성코드 분류는 악성코드 유사도 비교를 이용하는데, 악성코드 유사도 비교에는 MAL-DNA가 사용된다. 각각의 특성인자에 대한 가중치와 악성코드 유사도 계산식을 이용하여 유사도를 비교하게 된다.

전체적인 시스템은 Bottom-Up 구조로 구성되어 있다. Bottom-Up 방식의 단점인 시간 소모를 최소화하기 위하여 계층적 Bottom-Up 구조를 제안하였으며, 계층별 MAL-DNA의 개수, 총 계층 수, 유사도 임계값 등을 최적화 하여 사용 할 수 있다.

본 연구는, 새로운 악성코드 분류방법을 제안한 연구로서의 활용성뿐만 아니라, 특정 악성코드의 MAL-DNA를 추출하고, 이와 유사한 악성코드들의 DB를 구축하는 등 다양한 방법으로 사용할 수 있다. 향후, 한 그룹에 속한 MAL-DNA가 보다 명확하게 악성코드의 특징을 대표할 수 있도록 실험과 연구를 통해 개선할 계획이다.

참고문헌

〔1〕 2013 국가정보보호백서, 국가정보원, 미래창조과학부, 방송통신위원회, 안전행정부, 2013년 4월.

〔2〕 2011년 정보보호 실태조사 결과, 한국인터넷진흥원, 2012년 3월.

〔3〕 Jesse Kornblum, "Identifying almost identical files using context triggered

piecewise hashing," Digital Investigation vol 3, pp. 91 - 97, Sep. 2006.
〔4〕 ssdeep, ssdeep Project Homepage, Avainable from: http://ssdeep. sourceforge.net, Accessed on Mar. 2012.
〔5〕 Mark Manasse, "Finding similar things quickly in large collections," Available from: http://research.microsoft.com/en-us/projects/pageturner/similarity.aspx, Accessed on Mar. 2012.
〔6〕 simhash, Ubuntu Manpage of "simhash," Available from: http://manpages.ubuntu.com/manpages/natty/man1/simhash.1.html, Accessed on Mar. 2012.
〔7〕 FingerPrint, FingerPrint homepage, Available from:http://www.hbgary.com/free-tools#fingerprint, Accessed on Mar. 2012.
〔8〕 박재우, 문성태, 손기욱 등, "문자열과 API를 이용한 악성코드 자동 분류 시스템," 보안공학연구논문지 (Journal of Security Engineering), 8(5), pp.611-626, 2011년 10월.
〔9〕 DigitalNinja, "Fuzzy Clarity-Using Fuzzy Hashing Techniques to Identify Malicious Code," Available from: http://www.shadowserver.org/wiki/uploads/Information/FuzzyHashing.pdf, Accessed on Apr. 2012.
〔10〕 한경수, 김인경, 임을규, "API 순차적 특징을 이용한 악성코드 변종 분류 기법," 보안공학연구논문지 (Journal of Security Engineering), 8(2), pp.319-335, 2011년 4월.
〔11〕 Qinghua Zhang, "MetaAware: Identifying Metamorphic Malware," Computer Security Applications Conference 2007, pp.411-420, Dec. 2007.
〔12〕 S. Momina Tabish, M. Zubair Shafiq, and Muddassar Farooq. "Malware Detection using Statistical Analysis of Byte-Level File Content," CSI-KDD '09 Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics, ACM Press, pp.23-31, Jul. 2009.
〔13〕 "HBGary General Dynamics DARPA Cyber Genome Program Proposal," Available from: http://info.publicintelligence.net/DARPA-CyberGenome.pdf, Accessed on Apr. 2012.
〔14〕 "Digital DNA Datasheet," HBGary, Available from: http://www.hbgary.com/digital-dna, Accessed on Apr. 2012.
〔15〕 YARA, YARA-project homepage, Available from: http://code.google.com/p/yara-project/, Accessed on Apr. 2012.
〔16〕 ClamAV, ClamAV homepage, Available from: http://www.clamav.net/lang/en/, Accessed on Apr. 2012.
〔17〕 VirusTotal, VirusTotal homepage, Available from: https://www.virustotal.com/, Accessed on Apr. 2012.
〔18〕 Ilsun You and Kangbin Yim, "Malware Obfuscation Techniques: A Brief Survey," BWCCA 2010, pp.297-300, Nov. 2006.
〔19〕 Dennis Elser, "Unpacking malicious software using IDA Pro extensions," Available from: http://old.idapalace.net/papers/unpacking_malware_using_ida_pro_extensions.pdf, Accessed on Apr. 2012.
〔20〕 PEiD, PEiD homepage, Available from: http://www.peid.info/, Accessed on Mar. 2012.
〔21〕 Paul Craig, "Unpacking Malware, Trojans and Worms - PE Packers Used in Malicious Software," RuxCon 2006, Oct. 2006.
〔22〕 GuHyeon Jeong et. al., "Generic unpacking using entropy analysis," 2010 5th International Conference on Malicious and Unwanted Software, pp.98-108, Oct. 2010.
〔23〕 Daniel Pistelli, "Microsoft's Rich Signature (undocumented)," Available from: http://ntcore.com/files/richsign.htm, Accessed on May. 2012.
〔24〕 Peter Kleissner, "Microsofts Rich Header," Available from: http://web-17.webbpro.de/index.php?page=micros

ofts-rich-header, Accessed on May. 2012.
〔25〕 Malware Analyzer, Malware Analyzer homepage, Available from: http://malwareanalyser.blogspot.com/, Accessed on Mar. 2012.
〔26〕 Strings, Windows Sysinternals Strings homepage, Available from: http://technet.microsoft.com/en-us/sysinternals/bb897439.aspx,Accessed on Mar. 2012.
〔27〕 Jaccard index, Wikipedia - Jaccard index, Available from: http://en.wikipedia.org/wiki/Jaccard_index, Accessed on Mar. 2012.

〈저자소개〉

사 진

한 병 진 (Byoung-Jin Han) 정회원
2007년 2월: 성균관대학교 컴퓨터공학과 학사
2009년 2월: 성균관대학교 컴퓨터공학과 석사
2009년 1월~2011년 9월: 한국인터넷진흥원 선임연구원
2011년 10월~현재: ETRI 부설연구소 연구원

사 진

최 영 한 (Young-Han Choi) 정회원
2002년 2월: 한양대학교 전자·전기공학과(학사)
2004년 2월: 한국과학기술원(KAIST) 전자공학과(석사)
2009년 9월~현재: 고려대학교 정보보호대학원 박사과정
2004년 2월~현재: ETRI 부설 연구소 선임연구원

사 진

배 병 철 (Byung-Chul Bae) 정회원
1994년 2월: 홍익대학교 컴퓨터공학과 학사
1996년 2월: 홍익대학교 전자계산학과 석사
2005년~현재: 충남대학교 컴퓨터공학과 박사과정
1996년~1999년: 국방정보체계연구소 연구원
1999년~2000년: 국방과학연구소 연구원
2000년~현재: ETRI 부설연구소 책임연구원/부장