情報保護學會論文誌
第 22 卷 第 6 號, 2012. 12

퍼지해시를 이용한 유사 악성코드 분류모델에 관한 연구*

박 창 욱,[1†] 정 현 지,[1] 서 광 석,[2] 이 상 진[1‡]
[1]고려대학교 정보보호대학원, [2]한국정보보호교육센터

Research on the Classification Model of Similarity Malware using Fuzzy Hash*

Changwook Park,[1†] Hyunji Chung,[1] Kwangseok Seo,[2] Sangjin Lee[1‡]
[1]Center for Information Security Technologies, Korea University,
[2]Korea Information Security Education Center

요 약

과거 일 평균 10종 내외로 발견되었던 악성코드가 최근 10년 동안 급격히 증가하여 오늘날에는 55,000종 이상의 악성코드가 발견되고 있다. 하지만 발견되는 다수의 악성코드는 새로운 형태의 신종 악성코드가 아니라 과거 악성코드에서 일부 기능이 추가되거나 백신탐지를 피하기 위해 인위적으로 조작된 변종 악성코드가 다수이다. 따라서 신종과 변종이 포함된 다수의 악성코드를 효과적으로 대응하기 위해서는 과거의 악성코드와 유사도를 비교하여 신종과 변종을 분류하는 과정이 필요하게 되었다. 기존의 악성코드를 대상으로 한 유사도 산출 기법은 악성코드가 사용하는 IP, URL, API, 문자열 등의 외형적 특징을 비교하거나 악성코드의 코드단계를 서로 비교하는 방식이 사용되었다. 하지만 악성코드의 유입량이 증가하고 비교대상이 많아지면서 유사도를 확인하기 위해 많은 계산이 필요하게 되자 계산량을 줄이기 위해 최근에는 퍼지해시가 사용되고 있다. 하지만 퍼지해시에 제한사항들이 제시되면서 기존의 퍼지해시를 이용한 유사도 비교방식의 문제점이 제시되고 있다. 이에 본 논문에서는 퍼지해시를 이용하여 유사도 성능을 높일 수 있는 새로운 악성코드간 유사도 비교기법을 제안하고 이를 활용한 악성코드 분류기법을 제시하고자 한다.

ABSTRACT

In the past about 10 different kinds of malicious code were found in one day on the average. However, the number of malicious codes that are found has rapidly increased reachingover 55,000 during the last 10 year. A large number of malicious codes, however, are not new kinds of malicious codes but most of them are new variants of the existing malicious codes as same functions are newly added into the existing malicious codes, or the existing malicious codes are modified to evade anti-virus detection. To deal with a lot of malicious codes including new malicious codes and variants of the existing malicious codes, we need to compare the malicious codes in the past and the similarity and classify the new malicious codes and the variants of the existing malicious codes. A former calculation method of the similarity on the existing malicious codes compare external factors of IPs, URLs, API, Strings, etc or source code levels. The former calculation method of the similarity takes time due to the number of malicious codes and comparable factors on the increase, and it leads to employing fuzzy hashing to reduce the amount of calculation. The existing fuzzy hashing, however, has some limitations, and it causes come problems to the former calculation of the similarity. Therefore, this research paper has suggested a new comparison method for malicious codes to improve performance of the calculation of the similarity using fuzzy hashing and also a classification method employing the new comparison method.

Keywords: Fuzzy Hash, Malware, Similarity

접수일(2012년 4월 16일), 수정일(2012년 9월 19일), 게재확정일(2012년 11월 7일)

* 본 논문은 지식경제부 산업융합원천기술개발사업으로 지원된 연구결과입니다. [10035157, 실시간 분석을 위한 디지털 포렌식 기술 개발]

† 주저자, sindoll@korea.ac.kr
‡ 교신저자, sangjin@korea.ac.kr

I. 서 론

독일의 보안연구소인 AV-TEST의 통계자료에 따르면, 악성코드의 발생률은 매년 급격히 증가하고 있으며, 일평균 55,000건 이상의 새로운 악성코드가 발견되고 있다[1]. 또한 국내 백신업체인 AhnLab의 통계자료에 따르면 2011년 발생한 악성코드는 총 177,473,697건으로 2010년 146,097,262건과 비교하여 21.5% 증가하였으며[2], 백신 업데이트 내역에 의하면 일평균 1,500건의 신종 악성코드가 추가 진단되고 있다[3]. 다른 국내 백신업체인 Hauri의 '02~'11년 백신 업데이트 내역에 의하면 악성코드 대응 초기인 2002년에서 2003년에는 신규 악성코드 대응 건수가 일평균 10종 이내였지만, 2007년 200건을 넘었고, 2008년 이후에는 1000건을 넘었으며, 2012년에는 하루에 3000건 이상의 신종 악성코드를 추가로 진단하고 있다[4].

그러나 일평균 3000건 이상 유입되는 악성코드가 모두 새로운 제작자가 제작한 신종 악성코드는 아니다. Hauri의 백신 업데이트 내역을 분석하면 RAT (Remote Administrator Tool) 유형의 악성코드인 Poison Ivy의 변종이 매일 10건 내외로 지속적으로 유입되는 것을 확인할 수 있다[4]. 이처럼 악성코드의 변종이 지속적으로 유입되는 원인은 악성코드도 지속적으로 버그수정 및 기능개선이 이뤄지고 있으며[23], 제작된 악성코드가 안티-바이러스에 의해 탐지되면 우회할 만큼만 조작한 후 악성코드를 유포하기 때문이다[5].

만약 유입되는 악성코드를 모두 일반적으로 처리한다면 각각 새롭게 기능을 분석하여야 하며 백신에 적용하기 위해 새로운 백신탐지패턴을 개발하여야 한다. 이는 백신의 성능저하 및 분석시간제한 등의 문제를 유발한다. 따라서 증가하는 다수의 악성코드를 효과적으로 대응하기 위해서는 우선 유입되는 악성코드를 신종과 변종으로 구별해야 한다. 악성코드가 신종인 경우에는 새롭게 상세히 분석하여 대응하고, 악성코드가 변종인 경우는 기존 악성코드와 차이점을 분석하여 기존에 대응이 이뤄진 부분과 그렇지 않은 부분을 확인하여 필요한 부분을 추가로 대응할 수 있어야 한다. 그리고 분석 및 대응 결과는 저장하여 향후 발생할 악성코드에 대비하여야 한다.

신뢰할 수 있는 유사 악성코드 분류모델은 신종 및 변종 악성코드를 신속하게 식별하여 분석 및 대응하는 것 이외에도 7.7 DDoS 및 3.4 DDoS와 같이 공격 집단을 식별하고 공격의 위험성을 판단할 수 있으며 사고를 연관하여 책임추적을 위한 목적으로 사용할 수 있다.

기존의 악성코드의 유사도 산출기법은 대상 프로그램에서 비교인자를 추출한 후 비교하는 방식[14-18]과 프로그램 코드를 직접 비교하는 방식[19,20]이 있다. 유사도를 산출하는데 있어 가장 정확한 결과를 확인하는 방식은 코드레벨의 비교방식이다. 하지만 코드레벨의 비교방식은 결과를 확인하기 위해 많은 시간이 소요되는 단점이 있어 다수의 악성코드에 사용하기 부적합하다.

이러한 문제를 해결하기 위해 최근 주목받은 기법이 FuzzyHash중의 하나인 ssdeep이다. 악성코드의 ssdeep결과를 이용하면 유입되는 다수의 악성코드들의 유사성을 빠르기 식별할 수 있어 기존의 유사성을 비교하는 방법들에 비해 상대적으로 적은 계산으로 필요한 정보를 얻을 수 있다. 이러한 장점으로 바이러스토탈, FTK등 다양한 보안서비스에서도 ssdeep를 채택하여 사용하고 있다.

하지만 DigitalNinja[9], 카네기멜론CERT[22]는 이러한 ssdeep의 제한사항을 실험하여 ssdeep의 실효성에 의문을 제시하였다.

따라서 본 논문은 지금까지 제시된 ssdeep의 문제점을 분석하여 다수의 악성코드에 적용 가능하고 정확한 결과를 얻을 수 있는 비교인자 추출알고리즘을 제안하며 이를 활용할 수 있는 유사 악성코드 분류모델을 제안하고자 한다.

II. 배경지식 및 관련연구

2.1 Portable Executable 구조

악성코드의 대부분은 EXE, DLL등의 Portable Executable(PE) 구조로 구성된다. PE 구조는 헤더와 섹션으로 구분되며 헤더는 섹션들이 메모리에 저장될 위치를 저장하고 있으며, 섹션은 메모리에서 실제 사용되는 정보를 저장한다[6].

섹션은 공통으로 저장되는 정보는 .text, .data, .rsrc 섹션에 저장되며 컴파일러에 따라 .rdata, .edata 등의 추가 섹션을 사용하기도 한다. .text섹션은 프로그램이 실행되기 위한 명령어가 모여 있으며, .data 섹션은 명령어에서 초기에 사용하는 초기변수의 값이 저장된다. 그리고 .idata는 프로그램이 사용하는 API의 DLL목록을 저장하며, .rsrc는 프로그램이 사

용하는 리소스 정보들을 저장한다[6].

〔표 1〕 대표적인 PE섹션의 사용목적

종 류	이 름	내 용
코드	.text	프로그램을 실행하기 위한 코드
데이터	.data	초기화된 전역 변수
	.rdata	읽기 전용 데이터 섹션으로 문자열 및 가상함수 테이블
API	.idata	Import DLL과 해당 API의 정보
	.edata	Export DLL과 해당 API의 정보
리소스	.rsrc	다이얼로그, 아이콘 등 리소스 관련 정보

2.2 퍼지 해쉬(Fuzzy Hash)

일반적인 해시 알고리즘은 입력되는 파일을 대상으로 고정된 결과를 출력하며 출력된 결과는 고유한 값으로 두개의 파일이 해시결과가 같으면 같은 파일이라고 판단한다. 퍼지해시 알고리즘은 입력 값을 여러 조각으로 분리한 후 각각의 조각에 해시를 사용하여 하나의 결과를 만든다. 그리고 계산된 결과를 비교하여 얼마나 유사한지 확인한다[7].

초기 퍼지해시는 블록해시(Block-based Hash)로 입력 값을 고정된 크기의 블록으로 나누어 해시 결과를 비교하는 방식을 사용하였다. 하지만 입력 값이 조작(삽입, 삭제, 수정)되어 오프셋이 달라지면 블록의 내용이 달라지고 전체 결과에 영향을 주는 문제점이 있었다. 이러한 문제를 해결하기 위해 2006년 Kormblum은 CTPH(Context-Triggered Piece wise Hash)방식을 제안하였다. CTPH는 입력 값의 문맥을 구분할 수 있는 구분자를 식별하고 구분자로 조각난 조각에 대하여 해시를 사용하는 것이다[9]. 예를 들어 두권의 책을 비교하는 과정에서 기존의 방법은 책을 페이지 별로 비교하였으나 한 페이지가 수정되어 글자가 밀리면 모든 페이지의 글자가 수정되어 두책의 유사도를 확인하기 어렵다는 문제점이 있었다. 하지만 CTPH는 페이지가 아니라 문단별로 구분하고 있어 수정이 일어난 문단만 내용이 달라지고 나머지 문단은 변화가 없어 유사도를 확인하게 되는 것이다.

ssdeep는 2006년 CTPH를 제안한 Kornblum가 구현한 도구이다[10]. ssdeep는 입력 값의 퍼지해시 결과를 출력하는 기능과, 서로 다른 퍼지해시 결과를 비교하여 유사도를 계산하는 기능을 제공한다. ssdeep에서 CTPH 결과는 해당 Block의 Rolling Hash와 Traditional Hash를 적용하고 Base64를 이용하여 상위 6자리를 문자를 출력하는 The Spamsum Algorithm을 사용하여 모든 block의 처리결과를 출력한 결과이며 〔표 2〕와 같다. 결과에는 2가지 Signature가 기록되는데 출력된 Block의 크기로 비교한 결과가 Signature 1 이고, Block 크기의 2배로 계산을 한 결과가 Signature 2 이다[9].

```
b = compute.initial.block.size(input)

done =FALSE
while (done = FALSE) {
        initialize.rolling.hash(r)
        initialize.traditional.hash(h1)
        initialize.traditional.hash(h2)
        signature1 =""
        signature2 =""
        foreach byte d in input {
                update.rolling.hash(r,d)
                update.traditional.hash(h1,d)
                update.traditional.hash(h2,d)
                if (get.rolling.hash(r) mod b = b − 1)then {
                        signature1+= get.traditional.hash(h1) mod 64
                        initialize.traditional.hash(h1)
                }
                if (get.rolling.hash(r) mod (b ∗ 2) = b ∗ 2 − 1) then {
                        signature2+= get.traditional.hash(h2) mod 64
                        initialize.traditional.hash(h2)
                }
        }
        if length (signature1)< S/2 then
                b = b/2
        else
                done = TRUE
}
signature = b+ ":" +signature1+ ":" +signature2
```

〔그림 1〕 CTPH 의사 알고리즘

서로 다른 ssdeep 결과를 비교하여 유사도를 측정하는 과정은 결과에서 Signature 1과 Signature 2를 추출하고 해당 block의 크기를 확인 한 후 Signature에서 추가된 수, 삭제된 수, 변경된 수, 뒤바뀐 수를 이용하여 계산한다[9].

i is the number of insertions
d is the number of deletions
c is the number of changes
w is the number of swaps.

$$e = i + d + 3c + 5w$$

$$c + w \leq \min(l1, l2)$$

$$i + d = |l1 - l2|$$

$$M = 100 - \left(\frac{100.Se(s1, s2)}{64(l1 + l2)} \right)$$

〔그림 2〕 CTPH 유사도 산출 공식

그리고 서기민 외., Vassil Roussev 외.는 ssdeep를 사용하여 문서나 이미지 등 다양한 종류의

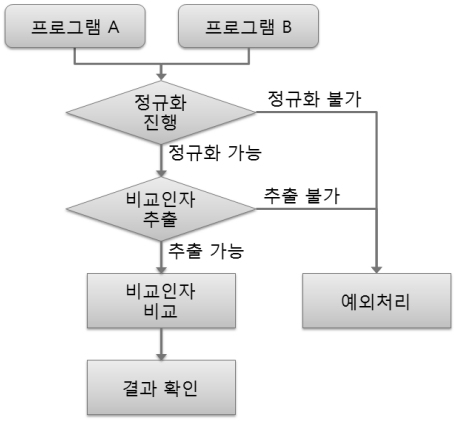
일반파일 및 악성코드의 유사성을 비교하여 효율성을 확인하였다[7, 8]. 그리고 2008년에 Virus Total은 악성코드 식별을 위해 ssdeep결과를 추가하였고[11], 2009년 포렌식 도구인 FTK도 ssdeep 기능을 추가하여 포렌식 조사에 활용하고 있다[12].

[표 2] ssdeep 결과의 구성

이 름	내 용
blocksize	6144
Signature 1	Jv7Wc4dyC7dXNBzn68YoC+6VoQSkgrpZHqk61peBN1L+I8pfezYeWHMzyy14pL1k
Signature 2	JvSbJxPRC+XQSxb6Dc7RwIWHeGL7GOK
filename	"calc.exe"

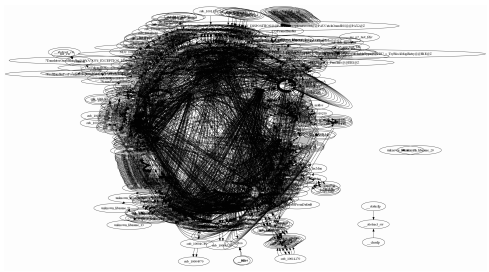
2.3 일반적인 악성코드 유사도 산출 기법

유사도 산출기법의 일반적인 절차는 정규화, 비교인자 추출, 비교인자 비교, 분석 순서로 진행된다[5].



[그림 3] 일반적인 유사도 산출 프로세스

절차 중 비교인자 추출은 동적인 추출방식과 정적인 추출방식으로 구분된다[21]. 동적인 비교인자 추출방식은 악성코드를 에뮬레이터로 동작시켜 나타나는 외형적인 부분(파일, URL 등)을 유사도 산출을 위한 비교인자로 사용하는 방식이다[14]. Q. Miao 등[15]은 악성코드를 에뮬레이터로 동작시켜 발생하는 악성코드 행위인 API를 기록하여 생성된 결과를 비교인자로 사용하였다, 정적인 비교인자 추출방식으로 한경수 등[16]은 악성코드의 IAT에 있는 API 목록을 추출하여 비교인자로 사용하였고, 박재우 등[17]은 API 목록에 추가로 악성코드에 있는 문자열을 추출하여 비교인자로 활용하였다. 또한 Halvar Flake [18]는 악성코드의 함수 간 CFG(Control Flow Graph) 관계를 추출하여 비교인자로 활용하였다. [그림 4]는 CFG기반의 악성코드 비교의 예로 함수 및 API간의 상호 호출관계를 시각화 하여 표현한 것으로 IDA를 활용하여 함수간의 목록을 추출하고 DOT언어를 이용하여 관계를 표현한다. CFG기반의 유사도 비교는 함수 및 API간의 호출관계도가 유사하면 유사한 악성코드로 판단한다.



[그림 4] CFG 기반의 악성코드 비교

Z.Wang[19]는 BMAT을 이용하여 악성코드를 함수별로 비교하여 동일한 부분을 식별하였다. 이러한 방법을 이용한 도구는 "IDA Compare(2005)", "eEye Binary Diffing Suites(2006)", "Patch diff(2008)" 등이 있다[20]. 하지만 악성코드 전체를 비교하는 방

[표 3] 7.7DDoS와 3.4DDoS의 비교

기준 \ 사건	'09. 7.7 DDoS	'11. 3.4 DDoS
공격대상 사이트	국내 21개 사이트 해외 14개 사이트	국내 40개 사이트
유포경로	파일 공유 사이트	파일 공유 사이트
유포수법	자동업데이트 파일 바꿔치기	자동업데이트 파일 바꿔치기
DDoS 공격수법	세션공격 중 CC공격	세션공격 중 CC공격
공격특징	특정일시에 맞춰 공격대상 재지정	특정일시에 맞춰 공격대상 재지정
자료삭제 기능	특정일시 좀비PC 삭제	특정일시 좀비PC 삭제
C&C 서버	61개국 435대 4단계	70개국 746대 3단계
악성코드 기능	디도스 공격 / 정보유출	디도스 공격

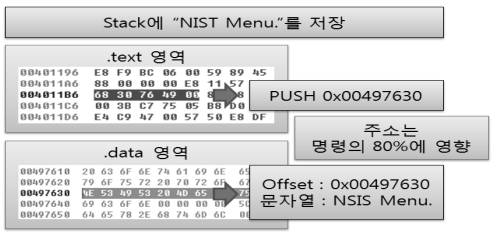
식은 결과를 확인하는데 많은 시간이 소요되기 때문에 악성코드 대상이 아닌 제로데이 취약점 식별을 위해 주로 사용된다.

악성코드 유사도 기법을 사용한 예로 2011년 국내에서 발생한 3.4 DDoS와 7.7 DDoS가 있다.

경찰청의 3.4 DDoS사건 보도자료에 따르면 악성코드의 유포방법, 해외 공격근원지, 악성코드 설계방식, 악성코드의 통신방식 등이 일치하기에 3.4 DDoS의 공격범이 7.7 DDoS와 동일범으로 결론지었다[13]. 이는 외부와 통신하는 명령구조, 사용하는 IP/URL 등의 외형적인 요소와 악성코드 기능 및 API, 구조적 형태 등 내부적인 요소를 종합적으로 비교하여 식별한 것이다.

2.4 기존 퍼지해시 기반 악성코드 분류 방법의 문제점

2010년 카네기멜론대학 CERT가 지적한 것처럼 파일 전체를 대상으로 유사도를 비교하는 것은 효율성이 낮다는 문제점이 있다[22]. 이에 본 논문에서는 해당 문제점은 PE구조에서 주소참조방식에 의한 명령어구조가 원인임을 증명하고자 한다. 대부분의 악성코드는 PE구조로 제작되며 PE 구조는 명령을 수행하기 위해 실제 명령코드는 .text섹션에 저장하고 참고하기 위한 문자열, 변수 등의 정보는 .data 섹션에 저장하여 주소참조방식으로 각각의 명령을 수행한다. [그림 5]에 따르면 변수 및 함수의 위치가 달라질 경우 주소를 참조하는 push, call, jmp등의 전체 명령어 5Byte중 참조주소가 차지하는 4Byte에 영향을 줘서 전체 80%에 변화를 주는 것을 확인할 수 있다[24].



[그림 5] push명령어의 주소참조 예

악성코드의 기능은 같아도 함수나 변수의 위치가 달라지면 push 등의 참조주소기반의 명령어가 PE구조 전체에 영향을 주어 기존의 악성코드를 대상으로 한 ssdeep기반의 유사도 식별방법은 효율성이 낮아진다.

이에 본 논문에서는 기존의 ssdeep를 악성코드에 바로 사용하는 방식에서 악성코드를 PE구조의 섹션별로 구분하고 그중 .data섹션에 ssdeep를 적용하는 방안을 제안한다.

III. 악성코드 비교인자 추출 알고리즘 제안

3.1 악성코드 유사성 비교인자

악성코드의 유사성 확인을 위해 비교인자의 필수조건은 다음과 같다.

- ⊙ 악성코드에서 추출 가능해야 함
- ⊙ 비교인자로서 단시간에 비교 가능해야 함
- ⊙ 유사도 결과에 충분한 신뢰도를 제공해야 함

본 논문에서는 비교인자의 필수조건을 따르는 .data 섹션을 비교하는 방식을 제안한다. .data섹션은 PE구조의 악성코드에 대부분 존재하고 IMAGE_OPTIONAL_HEADER의 SectionAlignment 값에 따라 0x1000 (4KB)의 배수의 크기를 가지며 악성코드가 사용하는 문자열 및 변수의 초기값을 저장하여 비교인자의 첫 번째 조건을 만족한다. 또한 비교인자로 .data 섹션의 위치는 PE헤더에 명시되어 가상의 에뮬레이터나 파일스캔을 통한 탐색과정이 필요 없어 단시간에 추출이 가능하여 두 번째 조건도 만족한다. 또한 .rsrc 섹션 등은 악의적으로 조작 시 가장 먼저 변조되는 섹션으로 .data 섹션은 조작이나 변조가 어려워 효율적인 유사도 비교인자로 적합하다. 마지막으로 결과의 신뢰도는 실험 1과 실험2를 통해 기존 악성코드 ssdeep 방식보다 .data 섹션 ssdeep 방식이 유사도 식별에 더 신뢰할 수 있는 것을 확인할 수 있다.

3.2 유사성 비교인자 추출 알고리즘

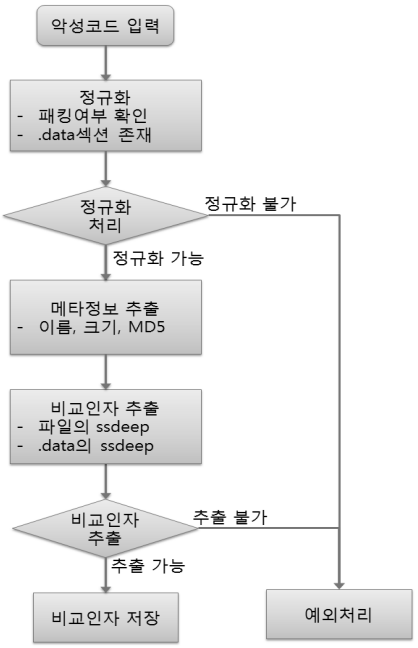
악성코드는 대응기법을 우회하기 위해 패킹과 같은 다양한 조작이 이뤄진다[5]. 그리고 패킹과 같이 조작된 악성코드는 일반적인 방법으로 대응할 수 없다. DigitalNinja[27]는 실험을 통하여 이를 증명하였다. 따라서 조작된 악성코드를 대응하기 위해 [그림 6]과 같이 정규화 과정을 통해 조작된 악성코드를 식별하여 조작되기 전으로 되돌리거나 예외처리 등의 방식으로 대응한다. 그리고 정규화 과정을 통과한 악성코드를

대상으로 비교인자 추출 및 비교인자 비교의 절차를 수행한다.

.data 섹션을 비교인자로 사용하기 위해서는 정규화 과정에서 다음의 조건을 만족하여야 한다.

⊙ PE 구조가 패킹(Packing)되지 않아야함
⊙ 비교인자인 악성코드의 .data 섹션이 존재함

정규화 과정의 패킹 유무는 복잡도(Entropy) 기반의 탐지 방법을 사용하여 식별하며[25], 정규화 과정을 통과한 악성코드는 메타정보를 추출한 후 유사도 비교 인자를 추출한다. 비교인자 추출모듈은 Python 기반의 PeFile과 Pyssdeep를 이용하여 필요한 정보를 추출하고 추출된 비교인자는 XML 형식으로 저장한다.



[그림 6] 비교인자 추출 알고리즘

IV. 악성코드 유사도 비교 인자 평가

4.1 실험 1 - 일반 실행파일 간 유사도 비교

실험 1에서는 두 실행파일의 유사도를 비교하기 위해 새롭게 제안한 .data섹션의 실효성을 증명하고자 한다. 실험의 목적은 NIST 2.45와 NIST2.46를 섹션 단위로 구분하여 ssdeep로 비교하여 유사성을 보이는지 확인하고자 한다. 카네기멜론CERT에서 사용한 공개소프트웨어인 NIST 2.45와 NIST 2.46을 대상으로 실험한다. 실험을 위해 각 파일의 섹션을 분리한 후 md5를 이용하여 동일하지 않음을 확인하고, ssdeep를 이용하여 각 섹션이 유사도를 비교하였다. [표 4]는 두 개의 NIST 파일을 섹션별로 분리하여 MD5를 비교한 결과이다. 대부분의 섹션은 일치하지 않아 일부 기능개선이나 변조가 일어난 것을 확인할 수 있으며, .rsrc 섹션은 동일하여 기능적인 부분만 개선이 이뤄진 것을 알 수 있다.

[표 4] NIST 2.45와 NIST 2.46 섹션의 MD5 해쉬값

대 상	NIST 2.45	NIST 2.46
.data 섹션	589ac23548625ebe8 d270c36a99bc4e0	80e1811904d791242 55fca6ffa8bd272
.text 섹션	a7ec8f654f732e67d3 acb753e4577256	e4f8b3e1c61acbaa83 4943fae9b892a3
.rsrc 섹션	6d99bbadb8a472f82 ea2917c6c307054	6d99bbadb8a472f82 ea2917c6c307054

[표 5]를 보면 NIST파일 전체를 대상으로 ssdeep 유사도 비교결과는 0점이었다. 하지만 각각의 섹션별로 유사도를 비교한 결과 .text섹션은 유사도가 없으며 .rsrc섹션은 MD5가 동일하여 ssdeep 비교결과가 100점으로 나왔다. 또한 .data 섹션은 MD5가 다름에도 85점의 유사도가 계산되었다. 즉, 기존 ssdeep기반의 유사도 비교는 .text영역에 상대적으로 많은 영향을 받으며 .data영역이나 .rsrc영역은 상대적으로 적은 영향력을 미치는 것을 알 수 있다. 그리고 .text영역은 참조주소방식으로 명령어가 구성되어 NIST 2.45의 .data섹션과 NIST 2.46의 .data섹션처럼 일부 내용이 달라진 경우에는 .text영역이 많이 달라진다는 것을 알 수 있다. .rsrc영역의 경우 동일한 제조사이기에 동일한 리소스를 사용하여 유사도가 100임을 확인하였다.

[표 5] NIST 2.45와 NIST 2.46 유사도 비교

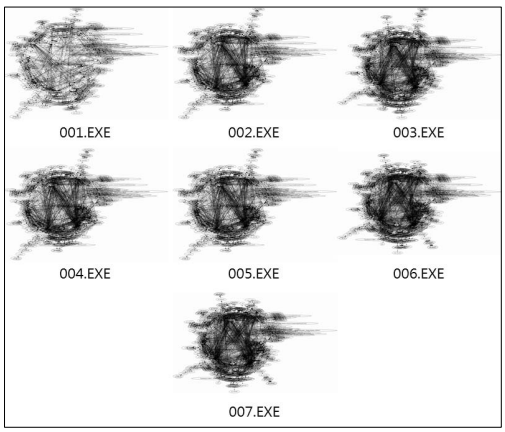
대 상	ssdeep 유사도
파일 전체	0
.data 섹션	85
.text섹션	0
.rsrc섹션	100

실험 1을 통하여 NIST 2.45와 NIST 2.46의 유사도 비교는 파일이 아닌 .data 섹션을 사용하여 유사도 비교가 가능함을 알 수 있다.

4.2 실험 2 - 악성코드 간 유사도 비교

실험 2에서는 기존의 파일대상 ssdeep 비교방식으로 유사도 식별이 불가능한 악성코드 7종의 .data 섹션을 비교인자로 사용하여 유사도 식별이 가능한 지를 확인한다. 이를 통해 본 논문에서 제안한 방법이 악성코드에도 동일하게 적용될 수 있다는 것을 증명한다.

실험대상으로 Halvar Flake가 사용했던 CFG 방식의 악성코드 유사도 산출기법을 통해 유사하다고 판단한 악성코드 7종을 선별하였다. 실험대상이 되는 7종의 악성코드 001~007은 동적분석과정을 통하여 유사성을 사전 인지하고, 2.3절의 CFG 방식의 비교기법으로 악성코드가 사용하는 함수간의 관계를 비교하여 〔그림 7〕과 같이 유사한 구조를 가진 악성코드임을 확인하였다.



〔그림 7〕 CFG 비교결과 유사 악성코드 7종

실험방법으로 기능이 유사한 001~007까지의 악성코드를 3장 2절의 유사성 비교인자 추출 알고리즘을 사용하여 추출한 후 각각의 유사도를 비교하여 결과를 확인하였다. 이를 통해 기존의 ssdeep방식의 유사도 비교결과 유사도 식별이 어려운 악성코드임을 알 수 있다.

실험결과 .data 섹션을 대상으로 ssdeep를 적용한 경우 〔표 7〕과 같이 실험대상 7종의 유사도는 서로 높은 연관성을 보이고 있음을 확인할 수 있다. 실험 2을 통하여 악성코드를 대상으로 유사도를 비교하기 위해서 악성코드 전체를 비교하는 것보다 .data 섹션만을 이용하는 것이 더 효율적임을 알 수 있다.

〔표 6〕 파일 기반 ssdeep 유사도 비교결과

대 상	001	002	003	004	005	006	007
001		0	0	0	0	0	0
002	0		0	0	0	0	0
003	0	0		0	0	0	0
004	0	0	0		0	0	0
005	0	0	0	0		0	99
006	0	0	0	0	0		0
007	0	0	0	0	99	0	

〔표 7〕 .data 섹션 기반 ssdeep 유사도 비교결과

대 상	001	002	003	004	005	006	007
001		74	82	75	72	69	72
002	74		71	72	82	71	82
003	82	71		82	69	69	69
004	75	72	82		77	66	77
005	72	82	69	77		69	100
006	69	71	69	66	69		69
007	72	82	69	77	100	69	

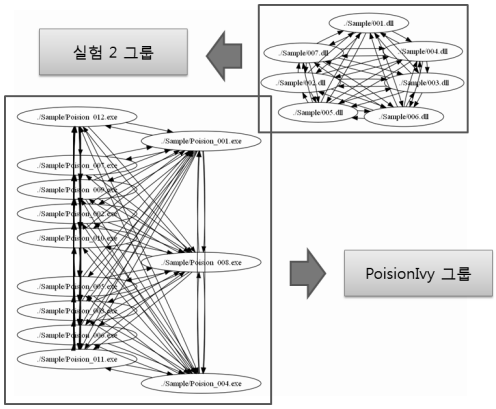
4.3 실험 3 - 다수의 파일에서 유사도 비교

실험 3에서는 여러 종류의 실행파일을 대상으로 하였을 때, 유사성 비교가 가능한 지 확인한다. 이 실험을 위해 실험 2에서 사용한 악성코드 7종, PoisionIvy 유형의 악성코드 12 종, 그리고 윈도우 시스템 파일 11 종을 모아서 동시에 유사도를 비교한다.

실험 방법은 실험 2와 동일한 방식으로 진행되었으며, 추가로 관계도 형성을 위한 시각화 과정을 추가하였다. 실험대상 30종의 파일이 서로 얼마나 유사한지 보다 파일 간 유사도가 있는 경우를 분류하여 유사한 파일 간의 관계를 형성하여 유사도 식별인자로 사용 가능한 지 확인한다. 관계의 기준은 두 파일의 .data 섹션의 ssdeep비교결과가 1 이상인 경우로 한다. 실험결과를 표현하기 위해 DOT언어를 처리하는 Graphviz〔26〕을 이용하여 파일간의 관계를 시각화한다.

실험 결과 30종의 실험대상은 〔그림 8〕과 같이 2개의 그룹으로 분리되었으며 2개의 그룹에 포함되지 못한 11 개의 파일이 존재하였다. 11개의 파일은 정상적인 윈도우 시스템 파일로 11개 간 서로 관계가 없어

서 그룹화 하지 못하였으며 2개의 그룹은 각각 실험 2의 악성코드 7종과 PoisionIvy유형의 악성코드 12종이다.

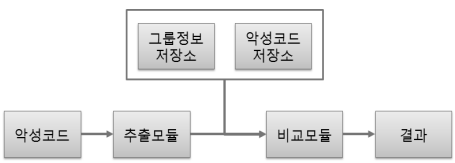


〔그림 8〕 악성코드 유사도 기반 그룹화 결과

실험 3을 통하여 .data 섹션을 이용한 유사도 비교는 다양한 종류의 파일도 구분할 수 있음을 입증하였다.

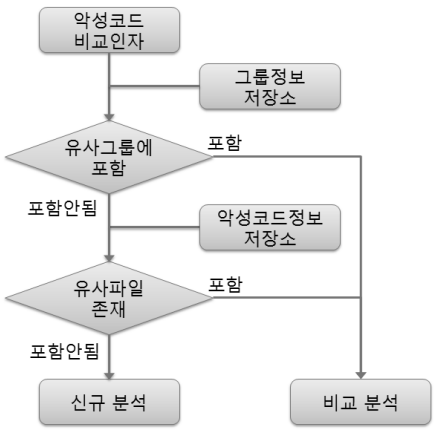
V. 유사 악성코드 분류모델 제안

본 장에서 제안하는 유사 악성코드 분류모델을 통해 사전에 기존의 악성코드를 구분하여 새로 유입되는 악성코드가 기존 악성코드 그룹에 포함되는지 여부를 확인하여 신종과 변종을 신속하게 분류할 수 있다. 유사 악성코드 분류모델은 〔그림 9〕와 같이 유입되는 악성코드에서 비교할 정보를 추출하여 저장되어 있는 정보와 비교하는 모델로 비교인자 추출모듈, 기존 악성코드에서 추출한 정보들의 저장소, 저장소의 정보와 신규 악성코드를 비교하는 비교모듈로 구성된다.



〔그림 9〕 신종 악성코드 분류모델 개념도

추출모듈은 3.2에서 설명한 비교인자 추출 알고리즘으로 악성코드에서 ssdeep결과 획득하기 위해 정규화, 메타정보 추출, 비교인자 추출로 구성된다. 정규화는 신규악성코드가 처리가능한지 여부를 판단하며 메타정보 추출은 향후 식별에 활용할 이름이나 MD5결과를 추출하는 과정이고 비교인자 추출은 악성코드간 유사도 비교를 위한 ssdeep 결과를 계산하는 과정이다. 저장소는 과거 악성코드에서 추출한 정보를 보관하고 있는 데이터베이스로 유사한 악성코드를 그룹으로 묶고 그룹을 대표하는 악성코드 1종을 선택하여 관련된 정보를 저장한다. 비교모듈은 신규 악성코드가 기존의 저장소에 저장된 악성코드 중 어떠한 그룹에 속하는지 비교하는 모듈로 사전에 수집된 악성코드 그룹의 대표 ssdeep 결과를 신규 악성코드와 비교하는 모듈이다.



〔그림 10〕 비교인자 비교 알고리즘

각 그룹의 대표는 그룹 내 객체들 사이에서 가중치가 가장 높은 객체로 선정하며, 공식(1)은 그룹의 대표를 선택하는 방법으로 그룹 내 객체들 사이의 유사도의 합으로 객체들 사이에 관계가 없으면 0점이고 관계가 있으면 해당 유사도를 사용하여 가중치를 계산한다. 가중치가 높은 그룹 대표는 그룹정보 저장소에 추가로 저장하여 유입되는 악성코드의 유사도 비교 시 비교대상으로 이용한다.

$G = \{a_1, a_2, ..., a_n\} =$ 유사 악성코드 집합
$GW = \{A_1, A_2, ..., A_n\} =$ 가중치 집합

$\forall a_1, a_2$는 $a_1, a_2 \in G$, a_1, a_2의 유사도 $= R(a_1, a_2)$

$$a_1\text{의 가중치} = A_1 = \sum_{i=1}^{n} R(a_1, a_i) \quad (1)$$

이때 $A_t = Max(GW)$ 이면
G의 대표는 a_t이다.

(단, 중복일 경우 t가 작은 경우를 대표로 선정)

VI. 유사 악성코드 분류모델 평가

6.1 성능평가방법 정리

유사 악성코드 분류모델의 성능 평가를 위해서 최악의 분류모델과 제안한 분류모델을 비교하겠다. 최악의 분류모델은 신규 악성코드 하나가 신종인지 변종인지 확인하기 위해 과거의 악성코드 전체와 1대1로 비교하는 방식으로 N개의 대상과 비교하기 위해 N번의 비교가 필요하다.

$$\begin{aligned} &N = \text{악성코드 개수} \\ &N' = \text{신규 악성코드 개수} \\ &\text{1대1 기반 비교량} = N' \times N = N'N \end{aligned} \quad (2)$$

예를 들어, 기존 100종의 악성코드에 신규 10종의 악성코드를 1대1로 비교하면 1000번의 비교가 필요하다.

제안한 분류모델은 N개의 악성코드가 M개의 그룹으로 분류될 경우 새로운 인자 1개의 결과를 얻기 위해서는 최대 $M+N$번의 비교가 필요하다.

$$\begin{aligned} &N = \text{악성코드 개수}, M = \text{유사그룹 개수} \\ &N' = \text{신규 악성코드 개수} \\ &N^* = \text{기존 그룹의 변종인 악성코드 개수} \\ &\text{그룹 기반 비교량} = N'(M+N) - N^*N \end{aligned} \quad (3)$$

예로 기존 10개의 그룹으로 분류되는 악성코드 100종에 신규 10종의 악성코드를 그룹으로 비교하면 최소인 경우는 10종이 모두 그룹에서 찾을 경우로 100번의 비교가, 최대인 경우는 모두 다 악성 코드에서 찾을 경우로 1100 번의 비교가 필요하다

6.2 유사 악성코드 분류모델 성능평가

유사 악성코드 분류모델의 성능을 실제 데이터를 이용하여 평가한다. 성능 평가에 사용한 데이터는 OO기관의 1일간 수집된 악성코드 200종이다. 성능 평가 실험을 위해 사전에 실험 3과 동일한 방식으로 악성코드 그룹화를 진행하였으며 200종의 악성코드 중 98종의 악성코드가 유사성을 보여 8개의 그룹으로 분류하였다. 8개의 그룹에 속한 악성코드는 온라인 백신서비스인 바이러스-토탈의 대표 백신엔진의 탐지명을 비교한 결과 〔표 8〕과 같이 각 동일한 탐지명으로 식별되었으며 각 대표의 악성코드 중 관련도가 가장 높은 악성코드를 선택하여 해당 그룹의 대표 비교인자로 사용하였다.

〔표 8〕 OO기관 1일 악성코드 그룹화 결과

대상	백신	탐지명
그룹01	V3	Win-Trojan/Starman.Gen
	Virobot	Worm.Win32.Allaple.Gen
	Kaspersky	Net-Worm.Win32.Allaple.e
그룹02	V3	Win-Trojan/Xema.variant
	Virobot	Backdoor.Win32.RBot.50176
	Kaspersky	Backdoor.Win32.Rbot.adqd
그룹03	V3	Win32/Allaple.worm.B
	Virobot	Worm.Win32.Net-Allaple.11730
	Kaspersky	Net-Worm.Win32.Allaple.e
그룹04	V3	Win-Trojan/Starman.Gen
	Virobot	- 미탐
	Kaspersky	Net-Worm.Win32.Allaple.d
그룹05	V3	Win-Trojan/Starman.Gen
	Virobot	Backdoor.Win32.RBot.50176.C
	Kaspersky	Backdoor.Win32.Rbot.bni
그룹06	V3	WORM/Allaple.Gen
	Virobot	Net-Worm.Win32.Allaple.e
	Kaspersky	Backdoor.Win32.RBot.11335
그룹07	V3	Win-Trojan/Starman.Gen
	Virobot	Worm.Win32.Allaple.Gen
	Kaspersky	Net-Worm.Win32.Allaple.e
그룹08	V3	Win-Trojan/Starman.Gen
	Virobot	Worm.Win32.Net-Allaple.11730
	Kaspersky	Net-Worm.Win32.Allaple.e

유사 악성코드 분류모델의 성능 평가를 위해 1일간 유입된 악성코드 200종을 대상으로 1대1로 유사도를 비교한 결과와 사전 8개의 그룹으로 분리하고 8개 그룹의 대표 악성코드와 200종의 악성코드를 비교한 결과를 비교하여 모델의 성능을 평가하였다.

최악의 모델의 경우 공식(2)에 따라 기존 악성코드 200종과 신규 악성코드 200종으로 200^2인 최대 40000건의 비교가 발생한다. 제안한 모델의 경우 공식(3)에서 악성코드 200종과 유사그룹 8개, 신규 악성코드 200종 유사그룹에 포함되는 신규 악성코드 98종을 공식에 적용하여 최대 22000건의 비교가 발생한다.

실제로 실험 결과는 제안한 분류모델은 23689번의 비교가 발생 하였으며, 최악의 분류모델은 39800번의

비교로 분류하였다. 분류 결과는 1대1 비교하였을 경우 98건의 악성코드가 변종 악성코드로 분류되었지만 그룹만을 이용하여 비교한 방식의 경우 89건의 악성코드가 그룹으로 분류되어 9건의 누수가 발생하였다. 누수가 발생한 원인은 그룹을 대표하는 파일과의 관계가 없었고 대표와 관계가 있는 다른 즉, 200건의 악성코드를 제안한 악성코드 분류기법으로 비교할 경우 최악의 분류방식으로 분류할 때보다 약60%에 해당하는 계산으로 분류 가능하였다.

VII. 결 론

일반적으로 악성코드에 대응하는 과정은 악성코드의 동작을 분석하여 악의적인 행위를 판단하고 그에 맞는 적절한 조치를 취해야 한다. 하지만 국내외 악성코드 관련 통계자료를 확인하면 신종 및 변종 악성코드의 발생률은 지속적으로 증가하면서 모든 악성코드를 상세히 분석할 수 없게 되었다. 따라서 악성코드에 효율적으로 대응하기 위해서는 유입되는 악성코드가 신종인지 변종인지 구분하여 신종은 상세히 분석하고 대응하고 변종은 과거 분석결과를 활용하여 신속하게 대응하는 과정이 중요하다. 하지만 기존의 API, CFG 등의 분류방식은 악성코드를 분류하기 위해 많은 시간이 필요하기 때문에 다수의 악성코드에는 부적합하다.

본 논문에서는 기존의 ssdeep을 활용하여 단시간에 신뢰도를 높일 수 있는 .data 섹션을 비교인자로 제안하고 신뢰성을 입증하였다. 그리고 제안한 비교인자를 효과적으로 적용하기 위한 유사도 산출모델을 제시하였다. 또한 제안한 유사 악성코드 분류 모델의 성능을 평가함으로써 해당 분류 모델이 대량의 악성코드를 효과적으로 분류할 수 있음을 확인하였다. 이를 이용하여 실제 악성코드 관련 침해사고 조사 과정에서 유용하게 사용할 수 있을 것이다.

참 고 문 헌

〔1〕 AV-TEST, "AV-TEST Statistic," http://www.av-test.org/en/statistics/, 2012년 2월.

〔2〕 AhnLab, "2011 악성코드 통계 분석 보고서," ASEC Report, Vol.24, 2012년 1월.

〔3〕 AhnLab, "V3 최신엔진 업데이트," http://www.ahnlab.com/kr/site/securitycenter/statistics/, 2012년 2월.

〔4〕 Hauri, "Virobot 최신엔진 업데이트 목록," http://www.hauri.co.kr/customer/security/update.html, 2012년 2월.

〔5〕 Andrew Walenstein, Arun Lakhotia. "The Software Similarity Problem in Malware Analysis," In Proceedings Dagstuhl Seminar 06301: Duplication, Redundancy, and Similarity in Software, Dagstuhl, Germany, pp. 10 July. 2006.

〔6〕 이호동, "Windows 시스템 실행파일의 구조와 원리," 한빛미디어, 서울 마포구, 2005년 5월.

〔7〕 서기민, 임경수, 이상진, "디지털 포렌식 수사를 위한 유사 파일 탐지," 한국정보기술학회논문지 7(2), pp. 182-190, 2009년 4월.

〔8〕 Vassil Roussev, Golden G. Ricahard III, Lodovico Marziale, "Multire solution similarity hashing," Proceedings of the 7th annual Digital Forensics Research Workshop, pp. 105-113, August. 2007.

〔9〕 J. Kornblum, "Identifying almost identical files using context triggered piecewise hashing," Digital Investigation, vol. 3(S1), pp. 91-97, March. 2006.

〔10〕 J. Kornblum, "Fuzzy Hashing and ssdeep," http://ssdeep.sourceforge.net, 2006.

〔11〕 jcanto, "Extra metadata field : ssdeep," Blog VirusTotal, November 2008.

〔12〕 Dustin Hurlbut, "Fuzzy Hashing for Digital Forensic Investigators," AccessData, January. 2009.

〔13〕 경찰청, "3.4 DDoS 사건의 공격자는 7.7 DDoS와 동일범," 경찰청 보도자료, 2011년 4월.

〔14〕 Joel Yonts, "Building a Malware Zoo," SANS Institute InfoSec ReadingRoom, Decemver. 2010

〔15〕 Q.Miao, Y.Wang, Y.Cao, X.Zhang, Z.Liu, "APICapture - a Tool for Monitoring the Behavior of Malware," Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering, pp. 390-394, August. 2010.

〔16〕 한경수, 김인경, 임을규, "순차적 특징을 이용한 악성코드 변종 분류 기법 API," 보안공학연구논문지 8(2), pp. 319-335, 2011년 4월.

〔17〕 박재우, 문성태, 손기욱, 김인경, 한경수, 임을규, 김일곤, "문자열과 API를 이용한 악성코드 자동 분류 시스템," 보안공학연구논문지 8(5), pp. 611-626,2011년 10월.

〔18〕 Halvar Flake, "Graph-based binary analysis," In Blackhat Briefings, July. 2002

〔19〕 Z. Wang, K. Pierce, and S. McFarling, "BMAT: A Binary Matching Tool for Stale Profile Propagation," The Journal of .nstruction-Level Parallelism, vol. 2, May. 2000.

〔20〕 Oh, J. "Fight against 1-day exploits: Diffing binaries vs anti-diffing binaries," In Blackhat technical Security Conference, July. 2009.

〔21〕 정용욱, 노봉남, "공격용 툴킷 및 변형 코드의 유사성 기준 선정," 보안공학연구논문지 9(1), pp. 31-44, 2012년 2월

〔22〕 David French, "Fuzzy Hashing Against Different Types of Malware," http://blog.sei.cmu.edu/post.cfm/fuzzy-hashing-against-differnt-types-of-malware, October. 2010.

〔23〕 PosionIvy, "Poisonivy-rat Development," http://www.poisonivy-rat.com/index.php?link=dev, 2008.

〔24〕 Intel, "Intel Architecture Software Developer's Manual Volume 2," Intel Corporation, May. 2010.

〔25〕 한승원, 이상진, "악성코드 포렌식을 위한 패킹 파일 탐지에 관한 연구," 정보처리학회 논문지 16(5), pp. 555-562, 2009년 10월.

〔26〕 Graphviz, "Graphviz - Graph Visualization Software," http://www.graphviz.org/, 1998

〔27〕 DigitalNinja, "Fuzzy Clarity: Using Fuzzy Hashing Techniques to Identify. Malicious Code," http://digitalninjitsu.com/ taxonomy/term/8, April. 2007

〈著者紹介〉

박 창 욱 (ChangWook Park) 학생회원
2007년 2월: 고려대학교 정보수학과 이학사
2007년 3월~2012년 8월: 고려대학교 정보보호대학원 수료
〈관심분야〉 정보보호, 악성코드, 디지털 포렌식

정 현 지 (Hyunji Chung) 학생회원
2010년 2월: 고려대학교 컴퓨터정보공학, 산업시스템공학 공학사
2010년 3월~2012년 2월: 고려대학교 정보보호대학원 공학석사
2012년 3월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 소셜 네트워크 포렌식, 클라우드 포렌식

서 광 석 (Kwangseok Seo) 정회원
1982년 2월: 고려대학교 수학과 학사
1984년 2월: 고려대학교 수학과 석사
1989년 8월: 고려대학교 수학과 박사
1991년 3월~2001 6월: 서남대학교 교수
2000년 9월~2008년 2월: 고려대학교 정보보호대학원 겸임/객원교수
2002년 3월~2007년 2월: 경기대학교 대학원 객원교수
2002년 9월~2006년12월: 국민대학교 대학원 겸임교수
2002년 3월~현재: 사이버 테러 정보전학회 이사
2001년 6월~현재: 한국정보보호교육센터 대표이사/원장
〈관심분야〉 정보보호, 암호, 침해대응, 디지털 포렌식

이 상 진 (Sangjin Lee) 종신회원
1987년 2월: 고려대학교 수학과 학사
1989년 2월: 고려대학교 수학과 석사
1994년 8월: 고려대학교 수학과 박사
1989년 10월~1999년 2월: ETRI 선임 연구원
1999년 3월~2001년 8월: 고려대학교 자연과학대학 조교수
2001년 9월~현재: 고려대학교 정보보호대학원 교수
2008년 3월~현재: 고려대학교 디지털포렌식연구센터 센터장
〈관심분야〉 디지털 포렌식, 심층 암호, 해쉬 함수