情 報 保 護 學 會 誌
第18卷 第3號, 2008. 6

인터넷 멀웨어 분류 방법 및 탐지 메커니즘에 관한 고찰

전 용 희*, 오 진 태**, 김 익 균**, 장 종 수**

요 약

인터넷에서 발생하고 있는 심각한 문제의 대부분이 멀웨어(Malware)로 인하여 발생하고 있으며, 전 세계적으로 전파되고 그 영향은 점점 악화되고 있다. 이 악성소프트웨어는 점점 더 복잡하여 지고 있으며, 이에 따라 멀웨어에 대한 분석도 어렵게 되고 있다. 그러므로 멀웨어 탐지 기술 및 그 특징에 대한 분석이 절실히 요구된다. 본 논문에서는 효과적인 멀웨어에 대한 탐지 및 대응기법 수립을 위하여 인터넷 멀웨어를 분류하기 위한 방법과 탐지 기법에 대하여 분석 및 고찰하고자 한다. 또한 제로-데이 공격에 대응하고자 개발된 ZASMIN(Zero-day Attack Signature Manufacture Infrastructure) 시스템의 특징에 대하여도 간략히 기술한다.

Ⅰ. 서 론

멀웨어는 웹사이트 훼손에서 인간의 생명 손실까지 영향을 미칠 수 있으며, 현재 인터넷이 직면하고 있는 스팸, 피싱(phishing), 서비스 거부(DoS : Denial of Service) 공격, 봇넷(botnets), 웜과 같은 심각한 문제의 대부분은 이 악성 소프트웨어 때문에 발생한다[1,2]. 현대의 멀웨어는 점점 더 복잡하여 지고 있으며, 이에 따라 멀웨어에 대한 분석도 더욱 어려워지고 있다. 예를 들어, Barford[3]등이 수행한 봇넷 조사에 의하면 Agobot은 2002년에 처음 나온 이래로 580 개 이상의 변형이 존재하는 것으로 관측되었다. 근래의 Agobot 변형은 DoS 공격을 수행하고, 은행 패스워드와 계정 정보를 훔치고, 다양한 집합의 원격 익스플로잇을 이용하여 네트워크상으로 전파되며, 탐지와 분해(disassembly)를 회피하기 위하여 다형성(polymorphism)을 사용하고, 감염시스템으로부터 취약성을 패치하거나 경쟁 멀웨어를 제거할 수 있는 능력도 있다[2].

이와 같이 인터넷 멀웨어 수의 증가와 다양성이 문제를 더욱 어렵게 만든다. 마이크로소프트(MS)사의 조사에 의하면 2006년도 전반기동안 43,000 개 이상의 백도어 트로이목마와 봇의 새로운 변형이 발견되었다[4].

멀웨어에 대한 탐지는 보안 관련 연구의 중요 영역 중에 하나인 동시에, 최근 심각하게 발생한 개인정보 침해 문제의 해결을 위하여도 긴요하다. 본 논문에서는 효과적인 멀웨어에 대한 탐지 및 대응기법 수립을 위하여 인터넷 멀웨어를 분류하기 위한 방법과 탐지 기법에 대하여 분석 및 고찰하고자 한다.

Ⅱ. 멀웨어의 정의 및 관련연구

멀웨어는 악성 소프트웨어(malicious software), 악성코드(malicious code) 및 멀코드(malcode) 등과 같이 여러 가지 이름으로 불리어진다. [5]에 의하면, 멀웨어는 다음과 같이 정의 된다 : 멀웨어는 운영체제 커널이나 어떤 보안 민감 애플리케이션의 행위를 변화시키는 코드 조각이다. 이는 사용자의 동의 없이 이루어지며 운영체제 혹은 애플리케이션 (예, API)의 문서화된 특징을 사용하여 그러한 변화를 탐지할 수 없도록 행해진다. 한편 앤티-바이러스(A/V) 업계 및 대부분의 사람들에 의하여 사용되는 통상적인 정의는 단순한 봇넷(botnet) 에이전트와 같은 독립적인 애플리케이션으로 코드된 것이다. 이것은 OS 커널이나 다른 애플리케이션과 접속하지 않으며, 합법적으로 열린 TCP 포트 상의 명령어

* 대구가톨릭대학교 컴퓨터정보통신공학부 (yhjeon@cu.ac.kr)
** 한국전자통신연구원 정보보호연구본부 보안게이트웨이연구팀 (showme@etri.re.kr, ikkim21@etri.re.kr, jsjang@etri.re.kr)

를 듣는다. 이것은 [5]에 의한 위의 정의에 의하면 멀웨어로 분류될 수 없을 것이다.

Christodorescu와 Jha[6]는 멀웨어 인스턴스를 나쁜 목적을 가진 프로그램으로 묘사한다. McGraw와 Morrisett[7]는 "시스템의 의도되는 기능을 의도적으로 해치거나 파괴하기 위하여 소프트웨어 시스템에 추가되거나, 변경된 혹은 제거된 어떤 코드"로 악성코드를 정의한다. [1]에서는 Vasudevan과 Yerraballi[8]에 의한 정의를 채택하였다. 이들은 "바이러스, 트로이목마, 스파이 웨어 및 다른 침입 코드를 모두 망라하는 일반적인 용어"로 악성코드를 기술한다.

멀웨어 수집, 분석 및 시그니처 생성에 대한 관련 연구는 불변 콘텐츠(invariant content)에 초점을 맞추고 주로 정적인 바이트 수준의 시그니처에 초점을 맞추고 있다[9,10]. 폴리모피즘(polymorphism)과 메타모피즘(metamorphism)을 다루기 위한 한 가지 솔루션으로 행위적 분석이 제안되었다[2]. 이와 유사하게 앤티-스파이웨어 툴을 사용하여 스파이웨어 행위를 발견하기 위하여 멀웨어를 에뮬레이팅하는 방법도 제안되었다[11].

행위 프로파일은 여러 추상화 레벨에서 생성된다. 개별 시스템 호출이나, 명령어-기반 코드 템플릿, 멀웨어 감염상에 수행된 초기 코드(셀코드)와 네트워크 연결 및 세션 행위와 같은 하위 레벨에 초점을 맞춘 연구가 있다. 상위-레벨 행위의 여러 가지 관점이 행위 프로파일의 정의에 포함될 수 있다. 네트워크 행위는 멀웨어를 나타낼 수 있으며, 멀웨어 감염을 탐지하기 위하여 사용되고 있다. 예를 들어, Ellis 등[12]은 유사한 데이터가 한 기계에서 이웃으로 전송되는 것과 같은 네트워크 레벨 특징을 추출하였다. Kolter와 Maloof[13]은 바이트 코드의 n-그램을 사용하여 악성 실행물을 분류하기 위하여 기계학습(machine learning)의 적용을 연구하였다[2].

Ⅲ. 운영체제 상호작용-기반 분류 및 탐지방법

[5]에서는 운영체제와 어떻게 상호작용하는가에 따라서 멀웨어를 아래와 같이 분류한다.

3.1 Type 0 멀웨어

이 형태의 멀웨어는 어떤 비문서화 된 방법을 사용하여 운영체제의 어떤 부분이나 다른 프로세스들과 상호작용을 하지 않기 때문에 [5]의 정의에 의하면 멀웨어로 분류될 수 없다. 그러나 이 형태는 완전성을 위하여 분류에 포함되었다. 시스템 침해 탐지 관점에서 보면, 이런 형태 멀웨어의 행위는 단지 애플리케이션의 특징이며 운영체제는 침해하지 않는다. 이런 악성 행위의 예로는 사용자 디렉토리로부터 개인 파일 삭제, TCP 포트 개방 등이 있다. 이런 행위들은 시스템에서 운영중인 다른 애플리케이션들이나 프로세스들의 행위를 변화시키지 않는다.

3.2 Type 1 멀웨어

ROM과 같이 고정되도록 설계된 여러 가지 운영체제 자원들을 변경시키는 멀웨어가 유형 1 멀웨어로 분류된다. 고정 자원들의 예로는 실행 파일, 수행 프로세스 및 커널 내부 메모리 코드 섹션, BIOS 코드, PCI 장치 확장 EEPROM 같은 것이 있다. 어셈블리 언어에서와 같이 여러 가지 다른 레벨 및 다른 방법으로 코드를 수정함으로써 이 형태의 멀웨어를 생성하기 위한 방법은 사실상 무한하다. 이 형태 멀웨어의 예로는 Hack Defender, Apropos, Sony Rootkit, 리눅스용 Adore 가 있다.

3.3 Type 2 멀웨어

형태 2 멀웨어는 코드 섹션과 같이 고정 자원들의 어떤 것도 변경시키지 않는다. 이 형태의 멀웨어는 데이터 섹션과 같이 동적인 자원 상에서만 동작한다. 예를 들어, 어떤 커널 자료 구조에서 함수 포인터를 수정하는 것이다. 이 경우 공격자의 코드가 원래 시스템이나 애플리케이션 코드 대신에 실행된다. 형태 2 멀웨어에 의하여 변경되는 자원들은 운영체제나 애플리케이션 자신에 의하여 어떤 식으로든 수정되어야 하기 때문에, 멀웨어에 의하여 도입된 어떤 변경을 탐지하는 것은 매우 어렵다.

3.4 Type 3 멀웨어

형태 3 멀웨어는 시스템 메모리 내부와 소프트웨어가 볼 수 있는 하드웨어 레지스터의 단 한 바이트도 변경하지 않고 전체 운영체제를 통제할 수 있다. 정의에 의하면 이 형태는 어떤 식의 무결성 스캐닝으로는 탐지될 수 없다. 오히려 시스템에 도입하는 어떤 부작용을

조사함으로써 탐지될 수 있다.

3.5 형태별 탐지 방법

[5]에 의하여 정의된 형태 0 멀웨어를 탐지하기 위하여, A/V(anti-virus) 업계에서는 주어진 실행물(executable)이 악성인지 아닌지를 결정하기 위하여 여러 가지 메커니즘을 개발하였다. 이런 것들로는 행위 모니터링, sandboxing, 에뮬레이션, AI-기반 휴리스틱스와 모든 시그니처-기반 접근이 있다.

형태 1 멀웨어의 변형을 생성시키는 방법은 무수히 많기 때문에, 코드 파괴(code subversion)의 알려진 패턴을 기반으로 하는 탐지로는 충분하지 않다. 형태 1의 탐지는 모든 고정된 자원들의 무결성을 검증함으로써 행해져야 한다. 즉 메모리 영역에 있는 코드 섹션과 같은 주어진 자원이 수정되지 않았다는 것을 검증하는 것이다. 이를 위하여 무결성을 평가하기 위한 어떤 기준이 필요하다. 많은 경우에 이런 기준이 다행스럽게도 존재한다. 예로써 EXE, DLL, SYS 등과 같은 모든 윈도우 시스템 실행 파일들이 있다. 그러나 약간의 예외도 있다. 때때로 호스트 침입방지시스템(IPS : Intrusion Prevention System) 제품과 개인 방화벽에서와 같이 커널의 코드 부분에 수정을 도입하는 합법적인 프로그램들이 있다. 이런 경우 멀웨어같은 후킹(hooking)과 HIPS 같은 후킹을 구별할 수 없기 때문에 적절한 시스템 무결성 툴을 설계할 수 없게 된다. 그러므로 [5]에서는 애플리케이션 개발자들이 그들의 실행물(executable)을 디지털인증서를 첨부하여 서명하도록 해야 한다고 지적한다.

형태 2 멀웨어에 대하여는 해쉬나 디지털 서명을 사용하여 전체 데이터 부분의 무결성을 검증하는 것이 매우 어렵다. 일반적으로 형태 2 멀웨어 탐지를 위해서는 커널의 모든 데이터 부분과 그 모든 드라이버, 시스템에서 수행 중인 모든 보안 민감 애플리케이션들을 분석할 필요가 있다. 그러므로 시스템을 설계하고 작성할 때, 모든 민감 자료 구조들이 운영체제 제작자 자신에 의하여 표시될 것을 제안하고 있다. 이런 식으로 그들의 무결성을 자동으로 검증하는 것이 가능하게 될 것이다. 어떤 보안 중요 애플리케이션도 같은 전략을 채택할 수 있다.

하드웨어 가상화-기반 형태 3 멀웨어인 경우, 탐지보다는 예방에 의존하는 것이 제안되었다. 운영체제에 구축된 보호적 hypervisor에 의하여 이런 형태의 멀웨어가 로딩되는 것을 방지할 수 있다고 지적한다.

Ⅳ. 시스템 상태 변화-기반 분류

4장에서는 시스템 호출의 시퀀스나 패턴이 아닌 작성된 파일이나 생성된 프로세스 등과 같은 시스템 상태 변화에 의하여 악성 행위를 묘사하는 분류 기법을 소개한다[2].

현재까지 대부분의 자동 멀웨어 분류 및 분석은 콘텐츠-기반 시그니처에 초점을 맞추고 있다. 그러나 이 콘텐츠-기반 시그니처는 폴리모픽과 메타모픽(metamorphic) 기법 때문에 본질적으로 정확하지 않을 수 있다. 그 외에도 특정 익스플로잇 행위에 의한 시그니처-기반 접근은 멀티-벡터 공격의 출현으로 점점 더 복잡하게 된다. 결과적으로 멀웨어 방어를 위한 조직 사이의 정보 공유, 신규 위협 출현에 대한 탐지, 감염 격리와 제거에서의 위험 평가 능력을 제한하는 환경을 생성하게 된다.

본 장에서는 기존의 자동 분류 및 분석 툴의 제한을 다루기 위하여, 가상화 환경에서의 멀웨어 실행과 멀웨어 실행으로 인하여 생성된 운영체제 객체들의 추적을 기반으로 하는 동적 분석 접근을 기술 한다. 멀웨어 행위의 지문 생성을 위하여 사용자가 볼 수 있는 작성 파일이나 생성 프로세스와 같은 시스템 상태 변경의 축약된 집합이 사용된다. 이런 지문들이 프로그램적 행위를 나타내는 추상적 코드 시퀀스보다 더욱 불변적이고 직접적으로 사용가능하다는 분석이다. 따라서 유발된 잠재적인 손해를 평가하고, 새로운 위협을 탐지하고 분류하며, 이런 위협의 완화와 제거에서의 위험 평가에 도움을 주기 위하여 직접 사용할 수 있다는 것이다. 이를 위하여 이런 멀웨어 프로파일을 유사한 클래스의 행위를 나타내는 그룹들로 자동으로 묶는 방법을 제시하고 있다.

이 연구에서는 멀웨어 표본의 실제 실행과 지속적인 상태 변경 관측을 기반으로 하는 방법을 기술한다. 이런 상태 변경이 행위적 지문(behavioral fingerprint)을 만들고 유사한 상태 변경 행위를 나타내는 멀웨어 클래스와 서브클래스를 정의하기 위하여 다른 지문들과 묶여질 수 있다.

이 시스템에서는 시스템 상에서 멀웨어가 실제로 행하는 것을 포획하고자 한다. 이런 정보가 더욱 불변적이고 발생하는 잠재적인 손해를 평가하는데 직접 사용할 수 있다는 것이다. 멀웨어 행위를 멀웨어가 시스템 상에

〔표 1〕 10개의 고유한 멀웨어 표본[2]

표시(label)	MD5	P/F/R/N	McAfee	동향
A	71b99714cddd66181e54194c44ba59df	8/13/27/0	탐지되지 않음	W32/Backdoor.QWO
B	be5f889d12fe608e48be11e883379b7a	8/13/27/0	탐지되지 않음	W32/Backdoor.QWO
C	df1cda05aab2d366e626eb25b9cba229	1/1/6/1	W32/Mytob.gen@MM	W32/IRCBot-based!Maximus
D	5bf169aba400f20cbe1b237741eff090	1/1/6/2	W32/Mytob.gen@MM	탐지되지 않음
E	eef804714ab4f89ac847357f3174aa1d	1/2/8/3	PWS.Banker.gen.i	W32/Bancos.IQK
F	80f64d342fddcc980ae81d7f8456641e	2/11/28/1	IRC/Flood.gen.b	W32/Backdoor.AHJJ
G	12586ef09abc1520c1ba3e998baec457	1/4/3/1	W32/Pate.b	W32/Parite.B
H	ff0f3c170ea69ed266b8690e13daf1a6	1/2/8/1	탐지되지 않음	W32/Bancos.IJG
I	36f6008760bd8de057ddb1ef99c0b4d7	3/22/29/3	IRC/Generic Flooder	IRC/Zapchast.AK@bd
J	c13f3448119220d006e93608c5ba3e58	5/32/28/1	Generic BackDoor.f	W32/VB-Backdoor!Maximus

끼치는 비-과도(non-transient) 상태 변경으로 정의한다. 상태 변경은 개별적인 시스템 호출보다는 상위 레벨 추상화이며, 암호화된 바이너리와 비결정적 이벤트 순서화와 같은 하위-레벨 시그니처와 마찬가지로 정적 분석을 따돌리는 많은 통상적인 회피(obfuscation) 기법을 없앤다. 이 기법에서는 멀웨어 실행으로부터 얻어진 raw event 로그로 부터의 상태 변경에 대한 단순한 묘사를 추출한다. 생성(spawn) 프로세스 이름, 수정 레지스터리 키, 수정 파일 이름, 네트워크 연결 시도가 로그로부터 추출되고 그러한 상태 변경 목록이 멀웨어 표본의 행위 프로파일이 된다. 멀웨어 행위의 관측은 바이너리를 실제로 실행하는 것이 필요한데, 여기서는 윈도우 XP를 설치하여 가상 머신 안에 개별 바이너리를 독립적으로 실행한다.

멀웨어를 분류하고 의미있는 라벨을 주기위하여 멀웨어 샘플들이 그룹화 되어야 한다. 프로파일을 그룹화하는 한 가지 방법은 어떤 두 개의 프로파일 사이의 차이를 측정하는 거리 척도(distance metric)를 생성하여 집단화를 위하여 사용하는 것이다. 첫 번째 방법은 유사성을 정의하기 위하여 편집거리(edit distance) 개념을 기반으로 한다. 그러나 이 방법이 가지고 있는 두 가지의 주요한 단점으로 NCD(Normalized Compression Distance)를 사용한다.

NCD는 다음과 같이 정의되며, 정보 내용의 근사화를 제공하기 위한 방법이다.

$$NCD(x,y) = \frac{C(x+y) - \min(C(x), C(y))}{\max(C(x), C(y))}$$

여기서, $x+y$는 x와 y의 접속(concatenation)이며, C(x)는 x의 zlib-압축 길이를 나타낸다. 직관적으로 NCD는 두개의 표본사이에 정보 중복을 보여준다. 결과

〔표 2〕 10 멀웨어 표본 사이의 NCD 행렬[2]

	A	B	C	D	E	F	G	H	I	J
A	0.06	0.07	0.84	0.84	0.82	0.73	0.80	0.82	0.68	0.77
B	0.07	0.06	0.84	0.85	0.82	0.73	0.80	0.82	0.68	0.77
C	0.84	0.84	0.04	0.22	0.45	0.77	0.64	0.45	0.84	0.86
D	0.85	0.85	0.23	0.05	0.45	0.76	0.62	0.43	0.83	0.86
E	0.83	0.83	0.48	0.47	0.03	0.72	0.38	0.09	0.80	0.85
F	0.71	0.71	0.77	0.76	0.72	0.05	0.77	0.72	0.37	0.54
G	0.80	0.80	0.65	0.62	0.38	0.78	0.04	0.35	0.78	0.86
H	0.83	0.83	0.48	0.46	0.09	0.73	0.36	0.04	0.80	0.85
I	0.67	0.67	0.83	0.82	0.79	0.38	0.77	0.79	0.05	0.53
J	0.75	0.75	0.86	0.85	0.83	0.52	0.85	0.83	0.52	0.08

적으로 동일하지는 않으나 유사한 행위는 가까운 것으로 보여 진다. 예를 들어, 다른 값을 가지는 두 개의 레지스터리 항목, 같은 위치 안에서의 임의 파일 이름 등이 있다. 정보 내용 차이 문제는 정규화(normalization)로 다루어진다. [표 2]는 [표 1]에서 기술된 멀웨어에 대한 정규화된 압축 거리 행렬을 보여준다.

행위지문의 두 집합 사이에 공유된 정보 내용을 알면, 유사성을 기반으로 여러 조각의 멀웨어를 결합한다. 본 연구에서는 잘 알려진 계위적 클러스터링 알고리즘[14]을 이용하여 트리 구조를 구축한다. 그런 후 트리로부터 의미 있는 그룹을 추출하기 위한 메커니즘을 사용한다. 클러스터의 거리 사이에 크기 차이를 계산하기 위하여 "불일치성(inconsistency)" 측도를 채택한다. 그리하여 트리는 별개의 클러스터들로 자를 수 있게 된다.

[표 3]은 [표 2]에 열거된 멀웨어에 대하여 이 기법을 사용하여 생성된 클러스터를 보여준다.

10 개의 고유한 멀웨어 조각들이 4 개의 다른 클러스

〔표 3〕 표 2에 대한 클러스터 생성 결과[2]

클러스터	요소	중복	예제
C1	C, D	67.86%	scans 25
C2	A, B	97.96%	cygwin 루트킷 설치
C3	E, G, H	56.60%	AV 무력화
C4	F, I, J	53.59%	IRC

터를 생성하였다. C1 클러스터에서 C는 mytob의 행위를, D는 대량 메일링 스캐닝 웜의 행위를 나타낸다. C와 D에 대하여 관측된 행위 중에서 25번 포트 스캔과 같이 5개는 동일하였으며, 여러 개의 다른 것은 행위 폴리모피즘을 보여주는 것으로 나타났다.

이 기법의 유효성을 증명하기 위하여 6개월 동안 수집된 3,700 개의 멀웨어 표본의 자동 분류 및 분석에 적용하였다. 그리고 분류 기법의 완전성, 간결성 및 일관성을 보여주었다. 본 연구는 멀웨어 사이의 관련성을 이해하고, 멀웨어를 분류하기 위한 새로운 방법으로 여겨진다.

V. 정보 수집-기반 멀웨어 탐지 기법 분류[1]

악성코드를 분류하는 것은 새로운 버전들이 기존 범주에 속하는 것들과 결합하는 형태를 가지기 때문에점점 더 복잡해지고 탐지 또한 어려운 문제이다. 멀웨어 탐지 메커니즘을 구현한 멀웨어 탐지기는 보호하고자 하는 시스템 상에 상주할 수도 있고, 그렇지 않을 수도 있다. 멀웨어 탐지 기술은 다음 세 가지의 다른 방법 중에서 하나를 택할 수 있다 : 정적(static), 동적(dynamic), 혼합(hybrid). 이것은 다음과 같이 멀웨어를 탐지하기 위하여 어떻게 정보를 수집 하느냐에 의하여 결정된다 :

- 정적 : 프로그램/프로세스의 신택스(syntax)나 구조적 성질을 이용한다. 이 방법은 악의성을 결정하기 위하여 바이트 시퀀스와 같은 구조적 정보만 단지 사용한다. 일반적으로 검사 중인 프로그램이 실행되기 전에 멀웨어 탐지를 시도한다.
- 동적 : 수행시간(runtime) 스택 상에서 본 시스템과 같이 프로세스의 수행시간 정보를 사용한다. 프로그램 실행 중이나 후에 악성 행위 탐지를 시도한다.
- 혼합 : 두 가지 접근을 결합한 방법이다. 정적이고 동적인 정보 모두가 멀웨어 탐지를 위하여 사용된다.

명세서(specification)-기반 탐지는 검사 중인 프로그램의 악의성을 결정하기 위하여 정당한 행위에 대한 어떤 명세서 혹은 룰셋(ruleset)을 사용하다는 점에서 비정상(anomaly)-기반 탐지의 특별한 형태이다. 그러므로 명세서를 위반하는 프로그램은 비정상으로, 보통 악의적으로 간주된다. 5.2절에서는 어노멀리-기반 탐지와는 별도로 명세서-기반 탐지를 기술한다.

5.1 비정상-기반 탐지

무엇이 정상인지에 대한 지식을 기반으로, 비정상-기반 탐지는 무엇이 비정상 행위인지 평가한다. 이러한 탐지는 보통 두 단계로 이루어진다 : 학습 단계와 탐지 단계. 탐지기는 학습단계 동안 정상 행위에 대한 학습을 시도한다. 이 탐지 기법의 장점은 멀웨어 탐지기에게 사전에 알려지지 않은 제로-데이 공격을 탐지할 능력이 있다는 것이다. 단점으로는 높은 오탐율과 정상 행위를 결정하기 위한 복잡성이 있다.

5.1.1 동적 접근

Wang과 Stolfo[15]에 의하여 제안된 PAYL, Lee와 Stolfo[16]에 의하여 제안된 데이터 마이닝 접근, Boldt와 Carlson[17]에 의하여 제시된 PIS(privacy-invasive software)의 개념, Sekar 등[18]에 의하여 제시된 유한상태 오토마타(FSA : Finite State Automata) 기반 접근, NATE(Network Analysis of Anomalous Traffic Events) 등이 이 범주에 속한다.

PAYL은 시스템 상의 각 서비스 포트에 대하여 기대되는 페이로드를 계산하는 툴이다. 바이트 빈도 분포를 이용하여 각 호스트의 서비스에 대한 centroid 모델이 개발된다. 이 centroid 모델이 학습 단계 동안 계산된다. 탐지기는 입력 페이로드와 centroid 모델을 비교하고, 둘 사이의 Mahalanobis 거리(distance)를 측정한다. Mahalanobis 거리는 특징 벡터(feature vector)의 평균값뿐만 아니라 더욱 더 강한 유사성의 통계적 척도를 생성하는 분산(variance)값과 공분산(covariance) 값도 고려한다. 만약 입력 페이로드가 centroid 모델로부터 너무 떨어져 있으면, 즉 큰 Mahalanobis 거리 값인 경우, 그 페이로드는 악성으로 간주된다. 1999 MIT Lincoln Labs 자료를 사용하여, 포착할 수 있었든 97개의 공격 중에서 57개를 탐지할 수 있었다. 전반적으로 이 기법에 대한 탐지율은, 오탐율이 1 퍼센트 이하일 때, 대략 60 퍼센트였다. PAYL은 CUCS(Columbia University Computer Science) 데이터셋에서 코드레드

II 멀웨어에 대하여 버퍼 오버플로 공격을 탐지할 수 있었다. MIT 데이터는 시뮬레이트된 자료인 반면, CUCS 데이터는 실제 네트워크로부터 획득한 실제 데이터이다.

Lee와 Stolfo[16]는 침입탐지시스템을 위한 데이터마이닝 기법의 사용을 제안하였다. 이 기법은 타깃 호스트의 여러 가지 보안 중요 관점에 대한 룰셋으로 언급될 수 있다. 이러한 룰셋이 호스트의 서비스에 대하여 무엇이 정상인지에 대한 기초로 사용된다. 입력 연결에 대하여 무엇이 정상인지를 학습하기 위하여 tcpdump 데이터를 사용하였다. 정상이거나 비정상인 데이터의 수동 조사를 통하여, 그들의 방법이 비정상 행위의 가능성에 대한 신호를 보내기 위하여 사용될 수 있다는 것을 추정할 수 있었다.

Hofmeyr 등[19]은 악의성을 탐지하기 위하여 시스템 호(call) 시퀀스를 감시하는 기법을 제안하였다. 시스템 서비스의 정상적인 행위를 나타내기 위하여 프로파일이 개발되어야 한다. 어떤 시스템 호 시퀀스가 다른 것과 얼마나 유사한지를 결정하는데 해밍거리(Hamming distance)가 사용된다. 대표적으로 큰 해밍거리 값을 보여주는 프로세스는 비정상이다. 이 방법은 sendmail, lpr, ftpd와 같은 여러 가지 유닉스 프로그램을 이용하려고 시도하는 침입을 발견할 수 있었다.

유한상태오토마타(FSA) 접근에서는, 각 노드는 알고리즘이 정상적인 데이터를 더 빠르게 학습하고 보다 나은 탐지를 수행하기 위하여 사용하는 상태를 나타낸다. FSA에서의 천이는 시스템 호에 의하여 주어진다. FSA를 구축하기 위하여 프로그램은 여러 번 실행된다. 시스템 호가 호출되면 새로운 천이가 오토마톤에 추가된다. 복수 실행으로부터 생기는 오토마톤이 알고리즘이 정상으로 간주하는 것이 된다. 수행시간 동안 시스템 호는 차단되고 프로그램의 상태는 기록된다. 이 수행과정에서 오류가 발생하면 비정상이 발생한다. 다음에 FSA의 현재 상태에서 새로이 호출된 시스템 호로 유효한 천이에 대하여 알고리즘은 조사한다. 그러한 천이가 없으면 비정상이 존재한다. 이전의 두 단계가 성공적이면, FSA는 다음 상태로 천이된다.

Sekar 등은 Hofmeyr 등이 제안한 n-그램 접근과 그들의 접근을 비교하였다. httpd, ftpd 및 nsfd에 대하여 평가한 결과 n-그램에 비하여 낮은 오탐율을 가지는 것이 발견되었다. 그러나 n-그램 접근에 대한 최적의 파라미터가 선택되었는지의 여부에 대하여는 분명한 근거가 없다.

NATE 기법은 네트워크 프로토콜 취약성을 이용하는 공격에 초점을 맞춘다. 이 접근은 악성 패킷들이 작은 수의 ack 패킷을 가지는 반면에, 많은 수의 syn, fin 및 reset 패킷들을 가지는 경향이 있다는 가정을 기반으로 한다. 이 접근은 각 패킷에 대하여 전달되는 바이트 속에서 나타나는 비정상성에 의존한다.

5.1.2 정적 접근

정적 비정상-기반 탐지에서는 검사 중인 프로그램의 파일 구조에 대한 특성이 악성 코드를 탐지하기 위하여 사용된다. 이 방법의 장점은 멀웨어를 가지고 있는 프로그램을 호스트 시스템 상에서 수행하지 않고도 멀웨어 탐지를 가능하게 하는 것이다.

Li 등[20]은 멀웨어 탐지를 위한 수단으로 파일지문(Fileprint, n-gram) 분석을 기술하고 있다. 학습단계동안 구조적(바이트) 구성을 기반으로 하는 시스템 상의 여러 가지 파일 형태를 특성화하기 위한 시도로 모델이나 모델집합이 유도된다. 이러한 모델은 시스템이 다루고자 하는 파일 형태를 학습하여 유도된다. 양성적인 파일은 각각의 형태에 대하여 예측 가능한 정규적 형태 구성을 가진다는 가정에서 출발한다. 예를 들어, 양성적인 pdf 파일은 exe나 doc 파일과는 다른 유일한 바이트 분포를 가진다. 검사 중인 파일이 주어진 모델이나 모델집합과 너무 다르다고 생각되면, 의심스러운 것으로 표시된다. 이런 의심스러운 파일들은 다른 메커니즘에 의한 추가적인 검사를 위하여 혹은 그것이 실제로 악성인지를 결정하기 위하여 표시된다.

5.1.3 혼합 접근

Wang 등[21]은 “ghostware"로 알려진 멀웨어 유형을 탐지하기 위한 방법을 제안하였다. 이것은 운영체제의 질의(query) 유틸리티로부터 자신의 존재를 숨기려고 하는 멀웨어이다. 예를 들어, 파일을 숨기는 고스트웨어로는 ProBot SE와 Aphex, registry를 숨기는 것으로는 Hacker Defender 1.0과 Vanquish, 프로세스나 모듈을 숨기는 것으로는 Berbew와 FU가 있다. 이런 유형의 멀웨어 탐지를 위하여 ”cross-view diff-based" 접근을 제안하였다. 시스템 호출이 행해질 때, 고스트웨어가 함수 호출(function call)을 가로채기 할 수 있는데, 이런 취약성에 대비하여 소위 cross-view 기반으로 비교를 한

다는 것이다. 즉 "dir"과 같은 상위 레벨 시스템 호출의 결과를 시스템 호출을 사용하지 않고 동일한 데이터의 하위 레벨 액세스까지를 비교하는 것이다. 이외에도 이 접근은 멀웨어 탐지를 위한 두 가지 방법을 제안하고 있다 : 상자 내부(inside-the-box) 접근과 상자 외부(outside-the-box) 접근.

상자내부 접근은 상위레벨과 하위레벨 결과의 비교를 동일 기계 상으로 제한한다. 그러나 고스트웨어가 전체 운영체제를 침해할 수 있고, 그 경우 하위레벨 스캔이 신뢰될 수 없을 수 있다. 외부상자 접근에서는, 다른 침해되지 않은 호스트가 타깃 호스트가 알지 못한 체 하위레벨 접근을 수행한다. 타깃 호스트의 상위레벨 스캔이 침해되지 않은 호스트로부터의 하위레벨 스캔과 비교된다. 내부상자나 외부상자 접근에서 하위레벨 혹은 상위레벨 스캔 사이에 차이가 존재하면, 타깃 호스트 내에 고스트웨어가 존재한다고 판단하게 된다. Forrest 등[22]이 제안한 Self-Nonself 기법도 이 범주에 속한다.

5.2 명세서-기반 탐지

대부분의 비정상-기반 탐지 기법과 관련된 대표적인 높은 오탐율을 다루기 위한 방법이다. 비정상-기반 탐지의 한 부류로써, 애플리케이션이나 시스템 구현에 대한 근사(approximate)를 시도하는 대신에, 명세서-기반 탐지는 애플리케이션이나 시스템을 위한 요구사항에 대하여 근사화를 시도한다. 학습단계에서 어떤 룰셋을 획득하고, 이것은 보호되는 시스템이나 검사 중인 프로그램에 대하여 어떤 프로그램이 보여주는 모든 정당한 행위를 명세 한다. 시스템이 보여줘야 할 전체적인 정당 행위의 집합을 완전하고 정확하게 명세 하는 것이 어려운 것이 제한점이다.

5.2.1 동적 접근

실행물의 악의성을 결정하기 위하여 수행시간에 관측되는 행위를 이용하는 것이 이 부류에 속한다.

Masri 등[23]은 DIFA(Dynamic Information Flow Analysis)라는 자바 애플리케이션을 위하여 특별히 설계된 툴을 기술하고 있다. 이 툴은 애플리케이션의 바이트코드 클래스들을 계측함으로써 수행시간의 메쏘드 호출(method call)을 감시할 수 있는 능력이 있다.

Xiong[24]은 네트워크 내의 악성 호스트를 탐지하기 위한 방법을 제안하였다. 목표는 전자메일 부착물을 통하여 확산되는 멀웨어를 방지하는 것이다. ACT (Attachment Chain Tracing) 방법은 전통적인 역학(epidemiology) 개념을 따라 모델 하였다. 두 호스트 사이의 역학 링크와 계층(layer) 관계 개념이 이용된다. 의심스러운 노드를 식별하기 위하여, 한 호스트가 전송하는 전자메일의 수에 대한 경계치를 먼저 설정한다. 탐지 간격은 서버로부터 다운로드 될 전자메일의 부착물에 대한 정상적인 기간을 기초로 한다. 그런 후 무슨 호스트가 감염되었는지 결정하기 전에 어떤 호스트 I로 부터 식별되어야 할 계층 수를 시스템 관리자는 결정하여야 한다.

이외에 [1]에서는 이 부류에 속하는 탐지방법으로 Ko 등[25]이 제안한 우선권 프로그램에서의 취약성의 자동화 탐지, Sekar 등[26]이 제안한 프로세스 행위 감시, Lee 등[27]이 기술한 악성 코드 주입에서의 하드웨어 열거 방법, Kirda 등[28] 등이 제안한 클라이언트 쪽으로부터 Cross-site scripting(XSS) 공격을 방지하는 방법, Suh 등[29]이 제안한 동적 정보 플로 추적, Milenkovic[30] 등이 제안한 명령어 블록 시그니처 사용 기법, Schechter[31] 등이 제안한 스캐닝 웜의 고속 탐지 기법, Linn 등[32]이 제안한 예기치 못한 시스템 호출에 대한 보호기법 등도 이 부류에 포함시키고 있다.

5.2.2 정적 접근

정적 기반 접근은 탐지기간 동안 악의성 결정을 위하여 프로세스나 프로그램의 구조적 특성을 이용한다. Bergeron 등[33]은 실행물(executable)이 실행되기 전에 악의적 의향 분석 시도를 위한 방법을 제안하였다. 먼저 실행물이 무엇을 수행하는지를 알기 위하여 이진 실행물(binary executable)이 어셈블리 코드의 추상적인 중간 표현형태로 분해된다. 그런 후 통제 플로 그래프(CFG : Control Flow Graph)가 생성되는 구문 트리(syntax tree)를 발생하기 위하여 어셈블리 코드를 파스(parse) 한다. 이 통제 그래프로부터 API-그래프가 생성되며, 단지 API 호출만 유지되고 모든 다른 계산은 묘사되지 않는다. 이 API 그래프로부터 중요-API 그래프가 생성되며, 여기서 사용자는 보안 오토마톤(security automaton)이라는 오토마톤으로 표현되는 보안정책을 사용하여 무슨 API 호출이 중요한지 결정한다. 보안 오토마톤에서의 천이는 호스트 시스템이 검사 중인 프로세스의 행위를 수행할 때 발생한다. 보안 오토마톤의 상

태 중에서 적어도 한개는 나쁜 상태로 식별되어야 한다. 만약 프로그램의 중요-API 그래프에 기초하여 시스템이 나쁜 상태로 들어가면, 그 때 시스템은 오토마톤에 의하여 기술된 보안 정책을 위반하게 되고 해당 실행물은 악성으로 여겨진다. 그러므로 악성 행위의 존재를 결정하기 위하여 중요-API 그래프가 보안 오토마톤에 대하여 조사된다. 이 방법을 이용하여 성공적으로 시험한 표본 악성 프로그램으로 WINIPX.EXE가 있다.

Bergeron의 또 다른 연구[34]에서는 상위레벨 명령법 표현 유도와 프로그램 슬라이싱 사용에서 주로 차이가 나는 다른 방법을 제안하였다. Debbabi 등[35]은 멀코드 탐지를 위하여 컴파일러 접근을 제안하였다. Adelstein 등[36]은 펌웨어에서의 멀코드 탐지 방법을 제안한다.

5.2.3 혼합 접근

Rabek 등[37]은 DOME(Detection of Malicious Executables) 기법을 제시하였다. DOME은 주입된, 동적으로 생성된, 애매한(obfuscated) 코드를 탐지하기 위하여 설계되었다. 두 단계로 특성화되는데, 처음 단계에서 프로세스를 정적으로 전처리한다. 전처리과정에서는 1) 시스템 호출 주소, 2) 이름, 3) 각 시스템 호출 바로 뒤의 명령어 주소를 저장하는 것으로 이루어진다. 3)은 실행물에서의 시스템 호출에 대한 반환 주소(return address)이다. 2)에서 DOME은 수행시간에 실행물을 감시함으로써 수행시간에 행해진 모든 시스템 호출이 1) 단계에서 수행된 정적 분석 보고서와 일치되는 것을 보장한다. 저자에 의하여 수행된 개념연구 증명에서 DOME은 악성코드에 의한 모든 시스템 호출을 탐지할 수 있는 것으로 밝혀졌다.

Wagner와 Dean[38]은 프로그램의 소스 코드를 분석하고 시스템 호출 트레이스를 나타내는 호 그래프 모델을 유도하는 기법을 제안하였다. Giffin 등[39]은 원격에서 실행되는 프로세스를 가지고, 악성 리모트 스트림이라는, 시스템 호출의 악성 시퀀스의 실행을 위하여 다시 호스트로 요구를 전송하는 호스트 머신을 보호하기 위한 기법을 제안한다. SQL 주입 공격을 방지하기 위하여 Halfond와 Orso[40]에 의하여 설계된 AMNESIA (Analysis and Monitoring for NEutralizing SQL-Injection Attacks) 구현도 이 범주에 속한다. Stack-Guard[41]는 "스택 스매싱"이라는 일종의 버퍼 오버 플로를 방지한다. SPiKE[42]는 사용자가 악성 행위를 발견할 목적으로 애플리케이션의 행위를 감시하는데 도움을 주도록 설계된 프레임워크이다.

5.3 시그니처-기반 탐지

검사 중인 프로그램의 악의성을 결정하기 위하여, 시그니처-기반 탐지는 시그니처(signature)라고 하는 이미 알려진 지식에 대한 특성화를 채택한다. 즉 멀웨어의 악성 행위를 모델하고 이 모델을 사용하여 탐지를 한다. 모든 이런 모델의 집합이 시그니처-기반 탐지 지식을 나타내며, 이 모델을 시그니처라 한다. 시그니처는 저장소(repository)에 보관되며, 검사 중인 프로그램이 알려진 시그니처를 포함하고 있는지를 평가하기 위하여 저장소를 조사한다. 현재는 프로그램에 의하여 표현되는 악성 행위를 나타내는 시그니처 생성을 위하여 대부분 전문가에 의존한다. 일단 시그니처가 생성되면 시그니처 저장소에 추가된다. 이 방법의 한 가지 단점은 저장소에 해당 시그니처가 저장되어 있지 않는 공격인 제로-데이 공격을 탐지할 수 없다는 것이다.

현재 ETRI에서는 시그니처를 실시간으로 자동 생성하여 제로-데이 공격에 대응할 수 있는 ZASMIN 시스템을 개발하고 있다[43].

5.3.1 동적 접근

악의성을 결정하기 위하여 프로그램의 실행동안 수집되는 정보만을 사용하여 특성화한다. 이 방법은 프로그램의 진정한 악성 의도를 보여주는 행위 패턴을 조사하여 탐지한다.

Ilgun 등[44]이 제안한 규칙-기반 IDS 접근에서는 공격을 상태 천이 다이어그램으로 모델 한다. 데이터 감사(data audit) 메커니즘이 존재한다는 가정 하에, 데이터를 전처리기에 전달하고, 여기에서 상태 천이 다이어그램으로 분석될 수 있는 방법으로 데이터를 포맷한다. 그런 후 이 데이터는 상태 천이 다이어그램의 형태인 알려진 침투에 대하여 비교된다.

Ellis 등[45]에서는 네 가지의 다른 행위 시그니처를 제시하고 있다. 기본 시그니처(base signature)는 단일 노드에서 입출력되는 데이터 플로를 감시함으로써 식별되는 것을 말한다. 이 기본 시그니처는 서버가 클라이언트로 변경할 때 식별되기 때문에, server-to-client 시그니처라고 한다. 웜은 서버를 침해한 후 스스로 전파되어

야 하기 때문에 다른 호스트에서 보통 같은 취약성을 통하여 더 많은 머신을 감염시킬 희망으로 다시 클라이언트로 행동해야 한다. 이 방법은 피어-대-피어 환경에서는 효과적이지 못하다. 다른 기본 시그니처로는 alpha-in/alpha-out, fanout, inductive 시그니처가 있다.

5.3.2 정적 접근

프로그램의 악의적 의향을 나타내는 코드 시퀀스를 조사하여 특성화함으로써 탐지한다. 즉 프로그램의 행위를 나타내는 코드를 평가한다. 코드의 정적 분석은 검사 중인 실행물의 수행 시간 행위를 개략적으로 제공한다. 이 방법의 장점은 실행물을 수행하지 않고도 프로그램이 분석되고 악의성이 정확히 결정될 수 있는 것이다.

[1]에서는 이 범주에 속하는 것으로, Sung 등[46]이 제안한 SAVE(Static Analysis for Vicious Executables), Christodorescu 등[47]이 제시한 의미론 인지(semantic aware) 탐지 방법, Kumar와 Spafford[48]가 제안한 정규 표현 매칭 기반 일반 바이러스 스캐너, Christodorescu와 Jha[49]가 제안한 SAFE(Static Analyzer for Executables), Kreibich와 Crowcrott[50]가 제안한 Honeycomb 등에 대하여 기술하고 있다.

5.3.2 혼합 접근

이 방법은 위의 방법의 특성을 모두 이용하는 것이다. Mori 등[51]은 모바일 자기-암호화(self-encrypting) 및 폴리모픽 바이러스를 탐지하는 툴을 제안하였다. 이런 유형의 바이러스는 패턴 매칭 기법을 우회하도록 설계되었기 때문에 전통적인 패턴 매칭 기법으로는 탐지가 불가능하다. 이 기법에서는 코드가 운영체제의 에뮬레이터 안에서 스스로 복호화되도록 한다. 복호화된 페이로드에 대하여 행해진 시스템 호출을 식별하기 위하여 정적 분석이 수행된다. 탐지 정책은 악성 행위를 나타내는 상태 기계로 모델 된다.

Castaneda 등[52]은 악성 프로세스를 포획하기 위한 하니팟 IDS를 사용하는 방법을 제안하였다. 일단 악성 프로세스가 포획되면 악성 페이로드 발견을 시도한다. 실행물의 사본을 만들고, 앤티-웜 페이로드로 수정된 실행물을 가상머신으로 전송하여, 성공적으로 수행함으로써 악성 페이로드를 발견하도록 한다.

Lo 등[53]은 실행물의 분석을 위한 툴로 MCF (Malicious Code Filter)을 제안하였다. 실행물을 택하여, 그 실행물의 CFG인 중간 표현물을 생성한다. 여러 가지 멀웨어 분석을 통하여, 멀웨어에 대한 "tell-tale" 부호(sign)를 개발하였다. 이 부호는 여러 가지 클래스의 멀웨어를 묘사하며, 멀웨어가 나타내는 유일한 pervasive 성질이다. 어떤 프로그램이 악성인지에 대한 최종 결정은 전문가에 의하여 행해진다. 이때 tell-tale 부호가 지침으로 사용되며, 이런 tell-tale 부호를 기반으로 하여 프로그래머가 필터를 생성한다.

Filiol[54]은 블랙 박스 시그니처 추출 기법에 대하여 강건한 멀웨어 패턴 탐지 기법을 제시하였다. 이것은 멀웨어를 독립적으로 신뢰성있게 탐지하기 위하여 시그니처를 다수의 서브-시그니처로 분할할 수 있다는 생각에서 출발한다. 실제 서브-시그니처는 프로세서의 일련번호, MAC 주소, 소유주의 사용자 이름과 전자메일 주소, 어떤 패스워드와 같은 인자에 의하여 결정된다.

Ⅵ. ZASMIN 시스템

현재의 정보보호시스템은 시그니처가 알려지지 않은 공격에 대하여 효과적으로 방어할 수 있는 수단을 제공하지 못하고 있다. 이로 인하여 고속으로 전파되는 슈퍼웜 형태의 공격이 발생하는 경우, 그 피해가 매우 크다. 따라서 비록 알려지지 않은 공격이라도 이를 탐지하고 차단할 수 있는 시그니처를 실시간으로 생성하여 해당 공격에 대한 피해를 최소화할 수 있는 시스템이 요구된다. 이를 위하여 본 연구원에서는 ZASMIN(Zero-day Attack Signature Manufacture Infrastructure) 시스템을 개발하였다[43].

ZASMIN 시스템은 자동적으로 트래픽 분석과 다양한 악성 코드 탐지 기법을 적용한 공격패턴 자동 생성 엔진을 탑재하여, 기존 공격패턴 생성 과정에서 변종 웜/바이러스를 구분해 주지 못하는 단점을 극복하고자 하였다. 또한 본 시스템에서는 선로 속도의 트래픽 분석과 다양한 악성 코드 탐지 기법을 적용한 공격패턴 자동 생성 엔진을 탑재하고 있다.

기존에는 보안전문가들이 수작업으로 공격패턴을 생성하기 때문에 비효율적이만, 본 시스템은 저비용의 자동화된 공격패턴을 생성하기 때문에 획기적인 침입방지 기술로 여겨진다. ZASMIN 시스템 기술은 국제 표준화 기구의 국제전기통신연합(ITU-T)에 의하여 지난 해 10월 스위스 제네바에서 개최한 국제표준회의(Q.6/WP2

/SG17)에서 국제 표준화 과제로 신규 채택된 바 있다.

Ⅶ. 맺음말

본 논문에서는 인터넷 멀에워의 분류 방법 및 탐지 방법에 대하여 기술하였다. 먼저 멀웨어에 대한 여러 가지 정의에 대하여 알아보고, 관련 연구에 대하여 살펴보았다. [5]의 분류에서는 멀웨어가 어떻게 운영체제와 상호작용하는지에 따라 4가지 유형으로 멀웨어를 분류하였다. 그리고 각 유형에 대하여 탐지 기법에 대하여도 소개 하였다.

시스템 상태 변화 기반 분류에서는 기존의 앤티-바이러스와 같은 호스트-기반 기법들이 멀웨어에 대하여 유용한 라벨을 제공하지 못함을 증명하고, 가상화 환경에서 멀웨어를 실행하여 멀웨어 동작의 행위 지문을 생성하는 동적 방법에 대하여 기술하였다[2]. 웜의 자동 분류 및 해석을 위하여 NCD를 거리 측도로 사용하여 지문들의 계위적 클러스터링을 적용하는 방법에 대하여 기술하였다.

마지막으로, 45 개의 멀웨어 탐지 기법에 대한 조사를 통하여, 멀웨어를 크게 비정상-기반, 시그니처-기반 및 명세서-기반으로 나눈 멀웨어 탐지 분류 기법에 대하여 살펴보았다[1]. 이 분석에서 비정상-기반 방법에서는 정적 접근에 대한 연구가 별로 없고, 다수의 연구 방법들이 비정상-기반과 명세서-기반에 집중되고 있음을 볼 수 있었다. 기존 연구에서는 상용(COTS : Commercial Off-The-Shelf) 멀웨어 탐지기들이 쉽게 난독화(obfuscated)[1] 됨을 지적한다. [1]에서 소개된 모든 연구의 멀웨어 탐지기 프로토타입들이 COTS에 비하여 우수한 성능을 나타낸다고 기술하고 있다.

현재 인터넷이 직면하고 있는 심각한 대부분의 문제가 본 논문에서 기술한 악성 소프트웨어 때문에 발생하기 때문에, 멀웨어의 효과적인 탐지 및 대응 기술에 대한 연구가 지속적으로 수행되어야 한다고 판단된다.

참고문헌

[1] Nwokedi Idika and Aditya P. Mathur, A Survey of Malware Detection Techniques, Department of Computer Science, Purdue University, Feb. 2007.

[2] Michael Bailey et al., "Automated Classification and Analysis of Internet Malware", RAID 2007, pp.178-197, 2007.

[3] Barford P., Yagneswaran, V., "An inside look at botnets", Advances in Information Security, Springer, Heidelberg, 2006.

[4] Microsoft, Microsoft security intelligence report, Oct. 2006. http://www.microsoft.com/technet/security/default. mspx

[5] Joanna Rutkowska, Introducing Stealth Malware Taxonomy, COSEINC Advanced Malware Labs, Nov. 2006, Ver. 1.01.

[6] M. Christodorescu and S. Jha, Testing malware detectors, In Proceedings of International Symposium on Software Testing and Analysis, July 2004.

[7] G. McGraw and G. Morrisett, Attacking malicious code : A report to the infosec research council, *IEEE Software*, 17(5) : 33-44, 2000.

[8] A. Vasudevan and R. Yerraballi, Spike : Engineering malware analysis tools using unobtrusive binary-instrumentation. In Proceedings of the 29th Australasian Computer Science Coference, pp.311-320, 2006.

[9] Newsome, J., Karp, B., Song, D., "Polygraph : Automatically generating signatures for polymorphic worms", In Proceedings of IEEE Symposium on Security and Privacy, Oakland, CA, USA, May, 2005.

[10] Li, Z. et al., "Hasma : Fast Signature Generation for Zero-day Polymorphic Worms with Provable Attack Resilience", In Proceedings of IEEE Symposium on Security and Privacy, 2006.

[11] Moshchuk, A. et al., "A Crawler-based Study of Spyware in the Web", In Proceedings of the Network and Distributed System Security Symposium(NDSS), San Diego, CA, 2006.

[12] Ellis D., et al, "A Behavioral Approach to Worm Detection", In Proceedings of the ACM Workshop on Rapid Malcode (WORM04), October 2004, ACM Press, New York, 2004.

[13] Kolter, J.Z. and Maloof, M.A., "Learning to Detect and Classify Malicious Executables in the

1) 보안 제품까지 무력화하는 해킹을 '난독화(難讀化. obfuscated) 해킹'이라고 하며, 2007년 상반기부터 급증하고 있는 것으로 알려져 있다. 또 최근에는 거의 모든 해킹이 전문적인 난독화 해킹 도구를 사용하고 있다.

Wild", Journal of Machine Learning Research, 2007.

[14] Hastie, T. et al., "The Elements of Statistical Learning", Data Mining, Inference, and Prediction, Springer, Heidelberg, 2001.

[15] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection", In Proceedings of the 7th International Symposium on (RAID), pp. 201-222, Sep. 2004.

[16] W. Lee and S. Stolfo, "Data mining approaches for intrusion detection", In Proceedings of the 7th USENIX Security Symposium, 1998.

[17] M. Boldt and B. Carlsson, "Analyzing privacy-invasive software using computer forensic methods", http://www.e-evidence.info/b.html, Jan. 2006.

[18] R. Sekar, M. Bendre, P. Bollineni, and D. Dhurjati, "A fast automation-based approach for detecting anomalous program behaviors", In IEEE Symposium on Security and Privacy, 2001.

[19] S. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls", Journal of Computer Security, pp. 151-180, 1998.

[20] W. Li, K. Wang, S. Stolfo, and B. Herzog, "Fileprints : Identifying file types by n-gram analysis", 6th IEEE Information Assurance Workshop, June 2005.

[21] Y. M. Wang, D. Beck, B. Vo, R. Roussev, and C. Verbowski, "Detecting stealth software with strider ghostbuster", In Proceedings of the 2005 International Conference on Defendable Systems and Networks, pp. 368-377, 2005.

[22] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination", In Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, May 1994.

[23] W. Masri and A. Podgurski, "Using dynamic information flow analysis to detect attacks against applications", In Proceedings of the 2005 Workshop on Software Engineering for secure systems-Building Trustworthy Applications, May 2005.

[24] J. Xiong, "Act : Attachment chain tracing scheme for email virus detection and control", In Proceedings of the ACM Workshop on Rapid Malcode(WORM)", 2004.

[25] C. Ko, G. Fink, and K. Levitt, "Automated detection of vulnerabilities in privileged programs by execution monitoring", In Proceedings of the 10th Annual Computer Security Applications Conference, pp.134-144, Dec. 1994.

[26] R. Sekar, T. Bowen, and M. Segal, "On preventing intrusions by process behavior monitoring", USENIX Intrusion Detection Workshop, 1999.

[27] R. B. Lee, D. K. Karig, P. McGregor, and Z. Shi, "Enlisting hardware architecture to thwart malicious code injection", International Conference on Security in Pervasive Computing (SPC), 2003.

[28] E. Kirda, C. Kruegel, G. Vigna, and N. Jovanovic, "Noxes : A client-side solution for mitigating cross-site scripting attacks", In the 21st ACM Symposium on Applied Computing (SAC), 2006.

[29] G. E. Suh, J. Lee, and S. Devadas, "Secure program execution via dynamic information flow tracking", International Conf. Architectural Support for Programming Languages and Operating Systems, 2004.

[30] M. Milenkovic, A. Milenkovic, and E. Jovanov, "Using instruction block signatures to counter code injection attacks", ACM SIGARCH Computer Architecture News, 33;108-117, March 2005.

[31] S. E. Schechter, J. Jung, and Berger A. W., "Fast detection of scanning worms infections", In Proceedings of 7th International Symposium on RAID, 2004.

[32] C. M. Linn et al., "Protecting against unexpected system calls", Usenix Security Symposium, 2005.

[33] J. Bergeron, M, Debbabi, J, Desharnis, M.M. Erhioui, and N. Tawbi, "Static detection of malicious code in executable programs", International Journal of Req. Eng., 2001.

[34] J. Bergeron, M, Debbabi, M.M. Erhioui, and B. Ktari, "Static analysis of binary code to isolate malicious behavior", In 8th Workshop on

Enabling Technologies : Infrastructure for Collaborative Enterprises, 1999.

[35] M. Debbabi et al., "Secure self-certified cots", In Proceedings of the 9th IEEE International Workshop on Enabling Technologies : Infrastructure for Collaborative Enterprises, pp.183-188, 2000.

[36] F. Adelstein, M. Stillerman, and D. Kozen, "Malicious code detection for open firmware", In Proceedings of the 18th Annual Computer Security Applications Conference, 2002.

[37] J. Rabek, R. Khazan, S. Lewandowski, and R. Cunningham, "Detection of injected, dynamically generated, and obfuscated malicious code", In Proceedings of the 2003 ACM Workshop on Rapid Malcode, pp.76-82, 2003.

[38] D. Wagner and D. Dean, "Intrusion detection via static analysis", IEEE Symposium on Security and Privacy, 2001.

[39] J. T. Giffin, S. Jha, and B. Miller, "Detecting manipulated remote call streams", 11th USENIX Security Symposium, 2002.

[40] W. Halfond and A. orso, "Amnesia : Analysis and monitoring for neutralizing sql-injection attacks", In Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, pp.174-183, 2005.

[41] C. Cowan et al., "Stackguard : Automatic adaptive detection and prevention of buffer-overflow attacks", In Proceedings of the 7th USENIX Security Conference, Jan. 1998.

[42] A. Vasudevan and R. Yerraballi, "Spike : Engineering malware analysis tools using unobtrusive binary-instrumentation", In Proceedings of the 29th Australasian Computer Science Conf., pp.311-320, 2006.

[43] 오진태, 김익균, 장종수, 전용희, "제로-데이 웜 공격 대응을 위한 ZASMIN 시스템 구조", 한국정보보호학회지, 제18권, 제 1호, 81-87쪽, 2008. 2월.

[44] K. Ilgun, R. A. Kemmerer, and Porras P. A., "State transition analysis : A rule-based intrusion detection approach", IEEE Transactions on Software Engineering, 1995.

[45] D. Ellis et al., "A behavioral approach to worm detection", Proceedings of the 2004 ACM Workshop on Rapid Malcode, pp.43-53, 2004.

[46] A. Sung, J. Xu, P. Chavez, and S. Mukkamala, "Static analyzer of vicious executables (save)", In Proceedings of the 20th Annual Computer Security Applications Conf. (ACSAC '04), 00 : 326-334, 2004.

[47] M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant, "Semantics-aware malware detection", In Proceedings of the 2005 IEEE Symposium on Security and Privacy, pp.32-46, 2005.

[48] S. Kumar and Spafford E. H., "A generic virus scanner in c++", In Proceedings of the 8th Computer Security Applications Conference, pp.210-219, 1992.

[49] M. Christodorescu and S. Jha, "Static analysis of executables to detect malicious patterns", USENIX Security Symposium, 2003.

[50] C. Krebich and J. Crowcroft, "Honeycomb-creating intrusion detection signatures using honeypots", In 2nd Workshop on Hot Topics in Network, 2003.

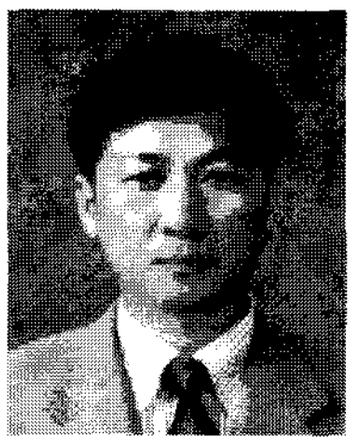
[51] A. Mori, T. Izumida, T. Sawada, and T. Inoue, "A tool for analyzing and detecting malicious mobile code", In Proceedings of the 28th International Conf. on Software Eng. pp.831-834, 2006.

[52] F. Castaneda, E. C. Sezer, and J. Xu, "Worm vs. worm : preliminary study of an active counter-attack mechanism", Proceedings of the 2004 ACM Workshop on Rapid Malcode, 2004.

[53] R. W. Lo, K. N. Levitt, and R. A. Olsson, "Mcf : Malicious code filter", Computers and Society, pp.541-566, 1995.

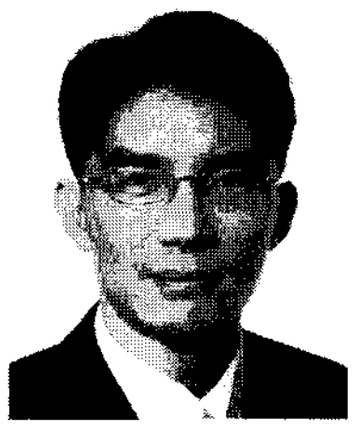
[54] E. Filiol, "Malware pattern scanning schemes secure against black-box analysis", Journal of Computer Virol., 2006.

〈著者紹介〉



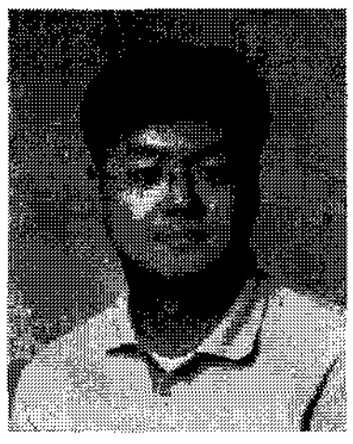
전 용 희 (Yong-Hee Jeon)
종신회원

1971년 3월~1978년 2월 : 고려대학교 전기전자전파공학부
1985년 8월~1987년 8월 : 미국 플로리다 공대 대학원 컴퓨터공학과
1987년 8월~1992년 12월 : 미국 노스캐롤라이나주립 대학원 Elec. and Comp. Eng. 석사, 박사
1978년 1월~1978년 11월 : 삼성중공업(주)
1978년 11월~1985년 7월 : 한국전력기술(주)
1979년 6월~1980년 6월 : 벨기에 벨가톰사 연수
1989년 1월~1989년 6월 : 미국 노스캐롤라이나주립대 Dept of Elec. and Comp. Eng. TA
1989년 7월~1992년 9월 : 미국 노스캐롤라이나주립대 부설 CCSP (Center For Comm. & Signal Processing) RA
1992년 10월~1994년 2월 : 한국전자통신연구원 광대역통신망연구부 선임연구원
1994년 3월~현재 : 대구가톨릭대학교 컴퓨터 · 정보통신공학부 교수
2001년 3월~2003년 2월 : 대구가톨릭대학교 공과대학장 역임
2004년 2월~2005년 2월 : 한국전자통신연구원 정보보호연구단 초빙연구원
2007년 1월~2007년 12월 : 한국정보보호학회 학회지 편집위원장
2008년 1월~현재 : 한국정보보호학회 부회장
<관심분야> 네트워크 보안, 웜 모델링 및 대응 기술, 통신망 성능분석



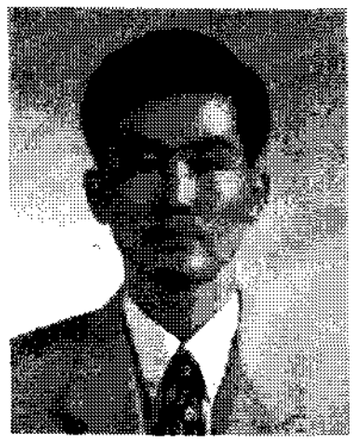
오 진 태(Jintae Oh)
정회원

1990년 2월 : 북대학교 전자공학과 공학사
1992년 2월 : 북대학교 전자공학과 석사
1992년 2월~1998년 2월 : 한국전자통신연구원 선임연구원
1998년 3월~1999년 1월 : 미국 MinMax Tech. 연구원
1999년 2월~2001년 10월 : 미국 Engedi Networks. Director
2001년 10월~2003년 1월 : 미국 Winnow Tech. Co-founder, CTO 부사장
2003년 3월~현재 : 한국전자통신연구원 선임연구원, 정보보호연구본부 보안게이트웨이연구팀 팀장
2008년 1월~현재 : 한국정보보호학회 학회지 편집위원(간사)
<관심분야> 네트워크보안, 비정상 행위탐지기술, 공격 시그니처 자동생성기술, 보안하드웨어기술



김 익 균(Ikkyun Kim)
정회원

1994년 : 경북대학교 컴퓨터공학과 공학사 .
1996년 : 경북대학교 컴퓨터공학과 석사
1996년~1999년 : 한국전자통신연구원
2000년~2001년 : (주) 팍스콤 선임연구원
2004년~2005년 : 미국 Purdue University 객원연구원
2001년~현재 : 한국전자통신연구원 정보보호연구본부 보안게이트웨이연구팀 선임연구원
<관심분야> 네트워크보안, 컴퓨터 네트워크



장 종 수(Jong-Soo Jang)
종신회원

1984년 : 경북대학교 전자공학과 학사
1986년 : 경북대학교 대학원 전자공학과 석사
2000년 : 충북대학교 대학원 컴퓨터공학과 박사
1989년 7월~현재 : 한국전자통신연구원 정보보호연구본부 책임연구원
2008년 1월~현재 : 한국정보보호학회 부회장
<관심분야> Network Security, 정책기반보안관리, 비정상트래픽탐지, 유해정보차단