J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

Endpoint에 적용 가능한 정적 feature 기반 고속의 사이버 침투공격 분석기술 연구☆

Study on High-speed Cyber Penetration Attack Analysis Technology based on Static Feature Base Applicable to Endpoints

황 준 호[1] 황 선 빈[1] 김 수 정[1] 이 태 진[1*]
Jun-ho Hwang Seon-bin Hwang Su-jeong Kim Tae-jin Lee

요 약

사이버 침해공격은 사이버 공간에서만 피해를 입히는 것이 아니라 전기·가스·수도·원자력 등 인프라 시설 전체를 공격할 수 있기에 국민의 생활전반에 엄청난 피해를 줄 수 있다. 또한, 사이버공간은 이미 제5의 전장으로 규정되어 있는 등 전략적 대응이 매우 중요하다. 최근의 사이버 공격은 대부분 악성코드를 통해 발생하고 있으며, 그 숫자는 일평균 160만개를 넘어서고 있기 때문에 대량의 악성코드에 대응하기 위한 자동화된 분석기술은 매우 중요한 의미를 가지고 있다. 이에 자동으로 분석 가능한 기술이 다양하게 연구되어 왔으나 기존 악성코드 정적 분석기술은 악성코드 암호화와 난독화, 패킹 등에 대응하는데 어려움이 있고 동적 분석기술은 동적 분석의 성능요건 뿐 아니라 logic bomb 등을 포함한 가상환경 회피기술 등을 대응하는데 한계가 있다. 본 논문에서는 상용 환경의 Endpoint에 적용 가능한 수준의 가볍고 고속의 분석성능을 유지하면서 기존 분석기술의 탐지성능 단점을 개선한 머신러닝 기반 악성코드 분석기술을 제안한다. 본 연구 결과물은 상용 환경의 71,000개 정상파일과 악성코드를 대상으로 99.13%의 accuracy, 99.26%의 precision, 99.09%의 recall 분석 성능과, PC 환경에서의 분석시간도 초당 5개 이상 분석 가능한 것으로 측정 되었고 Endpoint 환경에서 독립적으로도 운영 가능하며 기존의 안티바이러스 기술 및 정적, 동적 분석 기술과 연계하여 동작 시에 상호 보완적인 형태로 동작할 것으로 판단된다. 또한, 악성코드 변종 분석 및 최근 화두 되고 있는 EDR 기술의 핵심요소로 활용 가능할 것으로 기대된다.

☞ 주제어 : 악성코드, 정적분석, 기계학습

ABSTRACT

Cyber penetration attacks can not only damage cyber space but can attack entire infrastructure such as electricity, gas, water, and nuclear power, which can cause enormous damage to the lives of the people. Also, cyber space has already been defined as the fifth battlefield, and strategic responses are very important. Most of recent cyber attacks are caused by malicious code, and since the number is more than 1.6 million per day, automated analysis technology to cope with a large amount of malicious code is very important. However, it is difficult to deal with malicious code encryption, obfuscation and packing, and the dynamic analysis technique is not limited to the performance requirements of dynamic analysis but also to the virtual There is a limit in coping with environment avoiding technology. In this paper, we propose a machine learning based malicious code analysis technique which improve the weakness of the detection performance of existing analysis technology while maintaining the light and high-speed analysis performance applicable to commercial endpoints. The results of this study show that 99.13% accuracy, 99.26% precision and 99.09% recall analysis performance of 71,000 normal file and malicious code in commercial environment and analysis time in PC environment can be analyzed more than 5 per second, and it can be operated independently in the endpoint environment and it is considered that it works in complementary form in operation in conjunction with existing antivirus technology and static and dynamic analysis technology. It is also expected to be used as a core element of EDR technology and malware variant analysis.

☞ keyword : Malware, Static analysis, Deep neural network

1 Department of Information Security, Hoseo University, Chungcheongnam-do Asan-si, 31499, Republic of Korea.
* Corresponding author (kinjecs0@gmail.com)
[Received 17 May 2018, Reviewed 22 May 2018, Accepted 2 August 2018]
☆ 이 성과는 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2018R1C1B5029849).

1. 서 론

사이버 공격은 매년 큰 폭으로 증가할 뿐 아니라 전기, 가스 및 수도 등 사회 기반시설이 모두 연결되어 가면서 사이버 상의 피해를 넘어서 우리의 삶 전반에 큰 위협이 되고 있다. 이러한 사이버 공격은 대부분 악성코드를 통해 발생하고 있으며, 그 숫자는 일평균 160만개를 넘어서고 있기 때문에 이렇게 끊임없이 쏟아지는 대량의 악성코드를 대응하기 위해서는 자동화된 악성코드 분석기술이 매우 중요한 의미를 가지고 있다. 이에 자동으로 분석 가능한 기술이 다양하게 연구되어 왔는데 기존의 악성코드 정적 분석기술은 암호화나 난독화 및 패킹 등을 대응하는데 어려움이 있고, 동적 분석기술은 동적 분석의 성능요건 뿐만 아니라 logic bomb 등을 포함한 가상환경 회피기술 등을 대응하는데 한계가 있다. 또한, 탐지되지 않고 내부 유입된 악성코드는 악성코드 감염PC를 만들고 공격자에게 내부망 구성, 추가 악성코드 설치 등의 기회를 제공하여 사이버공격이 발생하게 된다. 따라서, 본 논문에서는 상용 환경의 endpoint에 적용 가능한 수준의 가볍고 고속의 분석성능을 보유하면서 동시에 기존 분석기술의 탐지성능 단점을 개선한 머신러닝 기반 악성코드 분석기술을 제안한다. 본 논문에서 제안하는 악성코드 탐지정책에서는 현재 유포되고 있는 악성코드의 대다수를 차지하는 PE(Portable Executable) 파일의 정적 정보의 의미와 동시에 악성코드와 정상파일의 통계적 분포를 통해 그 특징을 구분 짓고 분포를 가공하여 악성코드를 탐지하는 feature로서 활용하게 되는데 특히, 정적 분석기술에서 기존에 사용되던 주요한 feature 이외에 큰 의미를 부여하지 않았던 PE 파일 내의 정적 정보도 가공하여 feature로 사용함으로써 본 정책은 anti-virus 시스템의 분석을 어렵게 하기 위해 anti-debugging 기법을 적용하는 악성코드들의 탐지 회피 정책들을 무효화 하는 동시에 정적 분석기술의 장점을 가질 수 있었다. 또, 본 연구 결과물은 복잡한 가상 환경을 운용해야하는 sandbox 형태의 분석 기법이 아닌 머신러닝을 이용한 logic을 사용함으로써 endpoint 환경에서 독립적으로 운영 가능하며 기존의 anti-virus 기술 및 기존의 정적, 동적 분석기술과 연계하여 동작 시 상호 보완적인 형태로 동작할 것으로 판단된다.

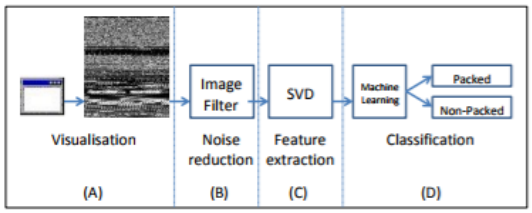
본 논문에서 제안하는 메커니즘은 악성코드 변종분석 및 최근 트렌드인 EDR(Endpoint Detection and Response) 기술의 핵심요소로 활용 가능할 것으로 예상된다. 다음으로 2장에서는 기존의 정적, 동적 악성코드 분석기술의 특징 및 장단점, network 단의 사이버 공격에 대한 연구를 기술하고, 3장에서는 본 연구에서 제안하는 악성코드 탐지 자동화 정책 및 기술을 제시한다. 4장에서는 제안 정책을 적용한 악성코드 탐지기술의 성능 분석 결과를 제시하고 5장에서는 본 논문에서 제안하는 정책과 메커니즘의 의미와 결론을 제시한다.

2. 관련연구

악성코드는 다양한 변종 형태를 쉽게 제작할 수 있어서 anti-virus 시스템의 분석을 어렵게 한다. 악성코드 제작자들은 고도화된 anti-debugging 기법들을 적용하여 기존의 자동화된 분석시스템을 무효화 시키는데 이러한 지능적인 대량의 악성코드들에 대응하기 위하여 악성코드들을 자동으로 분석하기 위한 연구들은 꾸준히 이루어지고 있다. sandbox와 같은 가상 환경을 탐지하여 회피하는 기법을 분석하고 탐지기술을 제안하는[1], N-gram 분석기법과 opcode의 pre-filtering을 이용한 보다 빠른 SVM(Support Vector Machine) 탐지 기법[2], 변종 악성코드들에 대해서 악성코드 그룹 별로 categorized를 통한 정적 분석 기법을 제안하는[3], opcode, byte sequence 등을 이용한 heuristic 탐지 기법 연구[4], API(Application Programming Interface)의 기능적 특성 관점으로 sequence를 분석하여 HMM(Hidden Markov Model)과 연계한 SVM 기법 으로 분류하는 모델[5], N-gram 기법을 이용하여 악성코드와 정상파일의 API call 패턴을 분석하고 SVM 기법을 이용하여 binary 파일을 분류하는[6], 악성코드와 정상파일의 API와 DLL(Dynamic Link Library)의 빈도를 분석하여 악성코드를 탐지하는 정적 분석기술을 제안하는[7] 등이 있다. 이러한 연구 동향에서 동적 분석기술을 사용하는 기법들은 대체적으로 시스템이 가볍지 않아 운용환경에 제약이 따르고 기존에 활발하게 연구된 opcode, API, DLL 기반의 분석기술의 경우에 그 성능이 악성코드에 적용된 anti-debugging 등의 기술에 의존성을 띄는 경향이 있다. Table 1.은 악성코드의 특징을 고려한 정상파일과의 opcode 빈도 분석표[8]인데 대다수의 악성코드는 악의적인 목적으로 패킹을 해두기 때문에 현재는 이러한 악성코드들을 탐지하기 위해서는 기존의 정적 분석기술에서 크게 진보하여 Figure 1과 같이 PE 파일을 image화 후, 파일의 noise를 줄이고 정보를 추출하는 것과 같은 고도화된 정책[9]을 적용하여야 의미 있는 결과를 도출해 낼 수 있다. 하지만 이러한 고도화된 정책의 경우에는 기존 정적 분석기술의 장점을 모호하게 하므로 현재까지는

(Table 1) Opcodes Frequency

Opcode	Goodware	Kernel RK	User RK	Tools	Bot	Trojan	Virus	Worms
mov	25.3%	37.0%	29.0%	25.4%	34.6%	30.5%	16.1%	22.2%
push	19.5%	15.6%	16.6%	19.0%	14.1%	15.4%	22.7%	20.7%
call	8.7%	5.5%	8.9%	8.2%	11.0%	10.0%	9.1%	8.7%
...								
xor	1.9%	1.1%	2.3%	2.1%	3.2%	2.7%	2.1%	2.3%
and	1.3%	1.5%	1.0%	1.3%	0.5%	0.6%	1.5%	1.6%



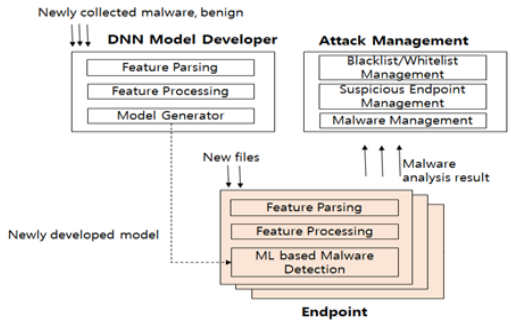
(Figure 1) Four Stages to PE Classification

이러한 정적 정보들의 경우에는 정적 분석기술의 한계를 극복하기 위하여 동적 분석기술과 hybrid 하여 상호 보완적으로 운용되게 된다. 마찬가지로, 이러한 기법에도 필연적으로 연산량 증가 등을 통한 성능 저하 문제가 발생하기 때문에 anti-debugging 기술에 대응하기 위한 부가적인 작업은 endpoint에서 악성코드 탐지 정책을 수립하기 어렵게 만들기 때문에 EDR 기술의 장애요소로 작용할 수 있다. 본 논문은 이러한 기존의 분석기술들의 문제점들을 개선하기 위하여 악성코드의 대다수를 차지하는 PE 파일의 구조와 정적 정보들의 의미를 분석하고 동시에 악성코드와 정상파일의 통계적 의미를 해석하는데 특히, anti-debugging 기술에 영향이 적으면서 endpoint에 적용 가능한 가볍고 고속의 악성코드 자동화 탐지 정책을 제안한다. 이는 탐지 정책이 기존에 주로 사용되던 정적 정보 외에 큰 의미를 부여하지 않았던 정적 정보들을 사용함과 동시에 기존의 정적 feature들의 빈도와 순서 정보 등과 같은 통계적 수치를 직접 머신러닝에 적용하지 않고 label간의 수치 분포를 고려한 영역화로 가능하다. 다음 장에서는 본 논문에서 제안하는 메커니즘을 자세한 제안모델을 제시한다.

3. 제안모델

3.1 System Overview

일반적인 사이버 침투공격은 웹/이메일 등을 통해 조직내부에 악성코드 감염PC를 만들고, 공격자는 악성코드 감염PC와 지속적으로 통신하면서 내부망 파악/추가 악성코드 설치/중요 시스템 접속정보 획득 등을 이뤄내고 적절한 시점에 시스템 파괴/개인정보유출/DDoS 공격 등이 발생하게 된다. 본 논문에서는 조직 내의 PC가 머신러닝 기술에 기반하여 악성코드에 감염되지 않도록 하여 사이버 침투공격에 대응하고자 한다. Figure 2는 전체 시스템의 동작 절차를 나타낸다.



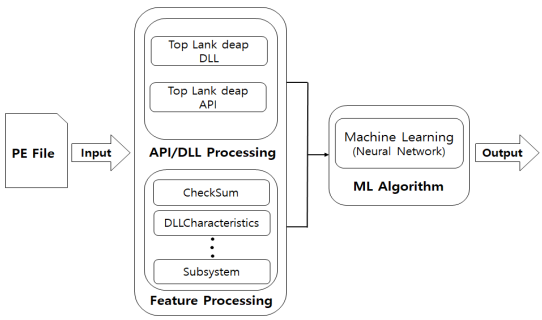
(Figure 2) System Overview

제안 시스템은 우선 제안하는 정적 정보들을 통해 이들 각각의 의미를 해석한 통계적 분포를 산출하고 그 분포를 가공하는 정책을 적용 후, 머신러닝을 이용하여 분석대상 파일에 대하여 악성유무를 자동으로 판별한다. 이때, 머신러닝 알고리즘으로는 데이터 분포에 대한 가공으로 모델을 좀 더 견고하게 구성할 수 있는 DNN(Deep Neural Network) 채택하였다. 다음으로 3.2절 에서는 endpoint에서 우선 탐지할 악성코드를 효과적으로 대응하기 위하여 API, DLL의 의미에 대한 해석과 데이터 분포에 따른 의사결정 정책을 제시하고 분석을 통하여 우선순위를 결정하는 절차를 제안한다. 또, PE header에 존재하는 정적 정보들의 의미와 특징을 전수 분석하여 그 분포를 가공하는 절차를 제시하여 유의미한 feature를 선별

해 내는 과정을 보인다.

3.2 Deep Neural Network 기반 악성코드 분석

DNN 기반 악성코드 분석은 endpoint 환경에서 독립적으로 운영 가능한 악성코드 탐지 정책으로 악성코드 내의 정적 정보들을 가용하여 악성코드에 대응하게 된다. Figure 3은 DNN 방식의 악성코드 분석 절차를 나타낸다. 해당 정책은 다수의 PE 파일내의 정적 정보들을 이용해서 학습모델을 만든 뒤 해당 모델을 이용해서 분석 대상 파일에 대해 악성여부를 판별하게 된다. PE 파일 내에서 정적 정보를 이용하기 위해 파일 구조내의 feature들을 추출하고 garbage값을 제거하는 파싱과정 및 시스템 전반적 성능을 고려한 feature 가공 절차가 수반된다. 다음 절에서는 본 메커니즘에서 사용되는 정적 정보들에 대해 기술하고 feature 후보군 전수 조사를 통한 feature 가공 정책을 제시한다.

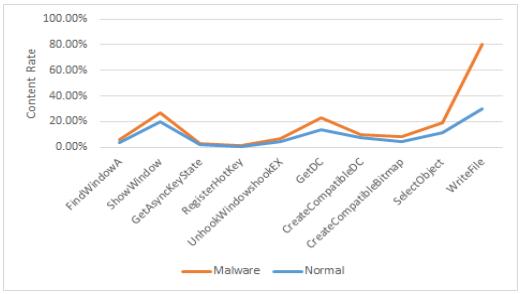


(Figure 3) DNN based Malware Analysis Overview

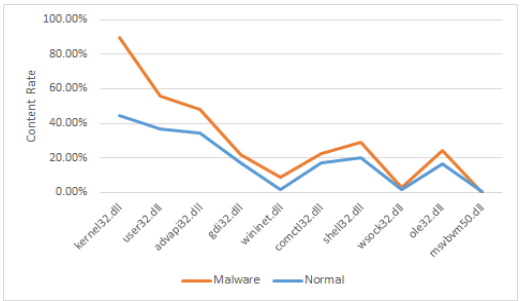
3.2.1 DLL/API feature 분석

어플리케이션의 기능을 수행하는 특징을 가지고 있는 API는 그 자체의 특성으로 인해 기존에 악성코드 분석에 주요하게 이용되었다. 하지만, 악성코드의 anti-debugging, packing, encryption, polymorphic/metamorphic 기술의 출현으로 단순한 dll/api에 기반한 분석의 실효성은 많이 떨어졌다. Figure 4는 상용 환경에서 사용되는 정상파일과 악성코드 샘플간의 API 출현빈도를 나타낸다.

DLL은 다양한 함수를 내재하고 있는 PE 파일로서 DLL 바인딩을 통하여 다른 프로그램이 DLL 내의 함수를 사용할 수 있게 하는 동적 라이브러리로서 악성코드 탐지기술의 정적정보로 앞서 제시한 API 와 유사한 특징을



(Figure 4) API Frequency Analysis between Malware and Benign



(Figure 5) DLL Frequency Analysis between Malware and Benign

가지고 있다. Figure 5는 자주 사 용하는 DLL들의 통계적 분포를 나타낸다. Figure 4와 Figure 5와 같이 API/DLL을 그대로 사용하면 악성코드 분석에 큰 도움이 되지 않고, 오히려 오탐지만 높이는 결과를 초래할 수 있다. Table 2는 악성코드 그룹 별 주로 사용하는 API 패턴[10]으로 그룹 별 기능적 관점에서 유사한 패턴을 나타내고 있는 것을 확인할 수 있다. 하지만 이러한 API 정적 정보는 그 자체로서의 크게 유의미하다고 판단할 수 없는데 이는 악성코드 제작자가 이러한 패턴기반 분석 기술을 회피하기 위한 기법들을 악성코드에 적용하기 때문이다. 본 논문에서는 해당 feature들의 사용 유무에 따라 분포 영역을 결정하고 각 feature들의 rank를 통한 가공으로 DLL/API 정적 정보의 유의미한 정책 선정 과정을 제시한다.

Table 3은 가장 많이 사용되는 15개의 DLL의 출현 빈도[7]를 나타낸다. 하지만 API와 마찬가지로, 자주 사용하는 DLL이 악성코드에서 주로 나타나는 패턴이라고 판단하기에는 무리가 있고 이러한 상황을 해결하기 위해서는 DLL 각각에 대해서 정량적으로 측정해야하는 문제가 있다. 이러한 점에 착안하여 어플리케이션의 기능적 특징을

(Table 2) General API Patterns by Malware Group

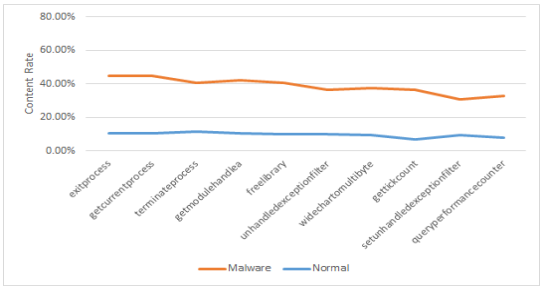
Malware Activity	API Pattern
Key Logger	FindWindowA, ShowWindow, GetAsyncKeyState, SetWindowsHookEx, RegisterHotKey, GetMessage, UnhookWindowsHookEx
Screen Capture	GetDC, GetWindowDC, CreateCompatibleDC, CreateCompatibleBitmap, SelectObject, BitBlt, WriteFile
Anti-debugging	IsDebuggerPresent, CheckRemoteDebuggerPresent, OutputDebugStringA, OutputDebugStringW
Downloader	URLDownloadToFile, WinExec, ShellExecute
DLL Injection	OpenProcess, VirtualAllocEx, WriteProcessMemory, CreateRemoteThread
Dropper	FindResource, LoadResource, SizeOfResource

(Table 3) Frequently Used TOP 15 DLL files

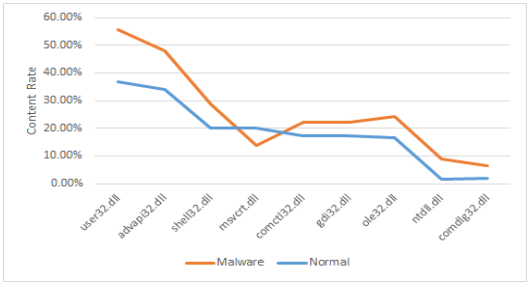
No	DLLs	Frequency
1.	kernel32.dll	3,467,435
2.	user32.dll	912,115
3.	advapi32.dll	441,914
	...	
14.	ntdll.dll	18,570
15.	urlmon.dll	16,117

나타내는 API의 본래의 특성을 벗어나 샘플 파일 내에 존재하는 string 관점에서 API 각각의 통계적 분포를 산출하고 그 분포에 따라 적절한 threshold를 설정하여 feature의 우선순위를 결정하면 다음 Figure 6과 같은 유의미한 결과를 나타낼 수 있다. 기존에 정형화되지 않았던 데이터 분포를 정성적으로 평가한 결과 Figure 6과 같은 통계적 분포 결과로 나타나게 되고 이는 정상파일과 악성코드간의 일반적인 API 사용 경향임에 동시에 두 범주를 구분지을 수 있는 특징으로 볼 수 있다.

Figure 5와 같이 정형화되지 않았던 데이터 분포를 정성적으로 평가하면 Figure 7과 같은 통계적 분포로 나타



(Figure 6) API Frequency with Feature Selection and Processing



(Figure 7) DLL Frequency with Feature Selection and Processing

나게 되는데 API의 정성정 평가 방법과 마찬가지로 이와 같은 분포 또한 정상파일과 악성코드간의 일반적인 DLL 사용 경향임에 동시에 두 범주를 구분 지을 수 있는 유의미한 통계적 분포라고 볼 수 있다. 따라서, 앞서 제안한 정책으로 도출한 통계적 분포는 기존 통계적 분포와는 다르게 악성코드에서 주로 사용하는 빈도뿐만 아니라 정상파일과 악성코드 내에서 구성 비율 및 각각의 비율 차를 고려하였고 상용 환경에서 탐지된 다수의 악성코드를 분석해서 유의미한 feature를 495개 선별하였다는 점에서 본 논문에서 제안하는 메커니즘에 가용 가능하다.

3.2.2 PE header feature 분석

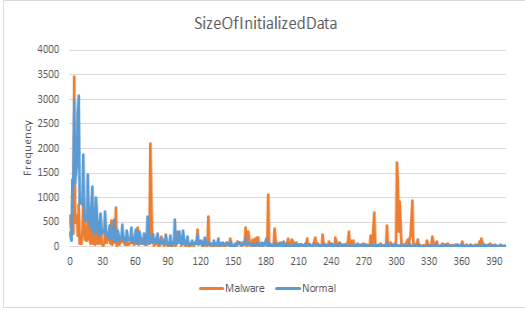
이번 절에서는 PE 구조를 가지는 악성코드 내의 구조적 정보에 의미를 분석하고 데이터에 대한 해석을 병행하여 악성코드의 패턴 변화에 noise가 적은 feature들을 제안하고자 한다. PE 파일의 구조는 각각 파일이 실행되기 위한 기능별로 영역이 구분되어 있다. 각각의 영역에는 그 기능을 위한 실제 값들이 포함되어 있는데 따라서 API와 DLL 정적정보와 마찬가지로 그 자체로도 유의미하다고 볼 수 있다. 이러한 특징으로 인해 해당 정적정보는 악

(Table 4) Feature Processing Table

Feature Name	Area 1	Area 2	Area 3	Area 4	Area 5
SizeOfInitializedData	0	1~50	51~250	251~400	over 401
DllCharacteristics	0~30k	32768	32769~34816	over 34817	non
MajorImageVerseion	0	1	2~15	16	over 17
...					
AddressOfEntryPoint	0	0~1.5k	1.5k~9k	over 9k	non
ImageBase	4194304	16777216	4.29E+09	5.37E+09	the rest

성코드 정적 분석기술의 장점을 포함하면서 anti-debugging 기술에 noise가 적고, 악성코드를 구분 짓는 정보로 가용할 수 있다. PE 파일에는 용도별로 영역이 나뉘어져 있고 그것은 section이라 칭하는 구조로 관리된다. unknown_sections_por은 PE 파일 구조에서 기본적으로 정의하고 있지 않은 section 명의 비율을 나타내는데 악성코드의 특징을 감안하면 feature로 가용 가능하다. sizeofinitializeddata의 경우, initialize를 위한 모든 section의 크기를 합한 값을 의미하는데 악성코드의 분류별 특징을 고려하면, 예를들어 trojan 범주의 악성코드의 경우에는 정상파일의 유사한 특징을 가지면서 악성행위를 시도하는 기능을 포함하여야 하므로 정상파일의 크기에 비해 상대적으로 클 것으로 판단할 수 있다.

Figure 8은 sizeofinitializeddata의 빈도 분포를 나타낸다. Figure 8과 같이 정상파일과 악성코드간의 통계적 분포를 상정해보면 두 범주가 서로 다른 빈도 영역에 주로 분포하는 것을 알 수 있다. 하지만 값의 범위가 넓고 정확한 threshold를 정하기에 어려움이 있기 때문에 해당 수치만을 가지고 악성코드의 특징을 구분 짓기 위해서는 해당 분포를 가공하여야 한다. PE 파일 내의 구조적 정보를 정량적으로 평가하게 되면 일정 수준의 유의미함을 나타내지만 수치 자체로는 악성코드의 특징을 명확히 하는 것이 어렵고 머신러닝 기법 등 실 환경에 적용하기 위해서는 해당 분포를 가공하는 정책이 필요하다. 다음과 같은 이유로 본 논문에서는 feature들의 분포를 정성적으로 평가하여 영역 별로 구분 짓는 정책을 제안한다. Figure 8의 sizeofinitializeddata 와 같이 다른 feature의 통계적 분포를 산출하고 정성적으로 평가하여 도출한 영역 구분의 threshold는 Table 4와 같다. Fig. 9.는 가공된 데이터의 분포를 나타낸다. 본 연구에서는 제시한 방법을 이용하여 PE header에 있는 다수의 feature 후보군들에 대해서 전수 분석하였고, 그 중에서도 유의미한 feature를 선별하였는



(Figure 8) Frequency Analysis of SizeOfInitializedData

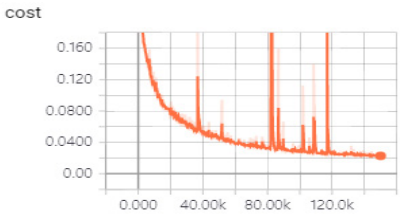
데, 위와 같은 정책을 적용하여 재구성한 정상파일과 악성코드의 분포 차이는 그 특징을 명확히 구분할 수 있고 상대적으로 머신러닝 등의 기법을 통해 고성능의 결과를 기대할 수 있는 동시에 시스템 복잡도 또한 낮아서 상용환경에 적용하기 용이하다.

4. 실험결과

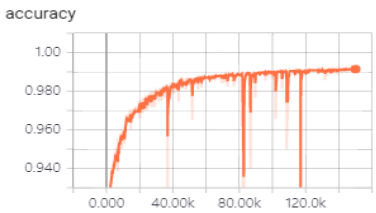
본 논문에서 제안한 기술은 실제 상용환경에서의 운영을 감안하여 일반적인 PC 환경인 2.30GHz 듀얼코어 CPU, 8GB ram과 windows 10 환경에서 구축하였다. 4.1절에서는 제안 메커니즘으로 사이버 공격침투 공격에 대한 기능적 분석 결과를 제시하고, 4.2절에서는 endpoint에서의 성능부담없이 고속으로 처리가능한지에 대한 성능 분석결과를 제시한다. 4.3절에서는 현 시스템을 상용 환경에 적용할 수 있고 지속적으로 연구개발하기 용이하도록 구성한 상용 환경에서의 운영 모델을 제시한다.

4.1 사이버 공격침투 분석 결과

연구결과물에 대한 시험은 상용 환경에서 확보한 115,000개의 정상파일과 악성코드를 대상으로 분석하였고 앞서 제시한 정적 정보들을 이용하여 DNN의 hidden layer 4, node 50, 학습횟수는 150,000번을 진행하였는데 메커니즘의 검증으로는 앞서 사용한 샘플 중 71,000개를 선택하였다. Figure 9와 Figure 10은 tensorboard를 이용하여 학습횟수에 따른 accuracy와 cost에 대한 분석결과를 나타낸다. Figure 9와 같이 15만번의 학습을 진행하면서 Cost는 0.02이하, Figure 10과 같이 accuracy는 99% 이상의 결과를 도출하였다. 실 환경 모델에서는 학습횟수를 더 늘려서 비용대비 효과적인 최적의 성능값을 얻을 수 있을 것으로 예상된다. Table 5는 본 논문에서 제안한 연구의 성능분석결과이다. 여러차례 시험한 결과, 99.13%의 accuracy, 99.26%의 precision, 99.09%의 recall 값이 안정적으로 산출되는 것을 확인하였다.



(Figure 9) Cost Analysis with Tensorboard

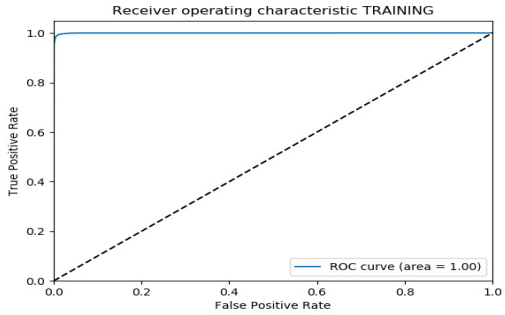


(Figure 10) Accuracy Analysis with Tensorboard

(Table 5) DNN based Malware Analysis Result

Accuracy	Precision	Recall
99.13%	99.26%	99.09%

Fig. 11은 분석결과를 오탐지율/탐지율 관점에서 분석하는 ROC(Receiver Operating Characteristic) curve를 나타낸다. ROC curve는 1에 가까울수록 좋은 성능을 나타내는데, 거의 1에 근접했음을 확인할 수 있다.

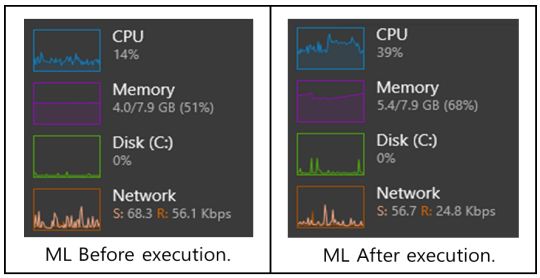


(Figure 11) ROC Curve with Malware Analysis

4.2 Endpoint에서의 처리성능 분석 결과

앞서 제시한 endpoint 환경에서 총 71,000개의 파일의 악성여부를 분석하는데 총 14142초가 소요되었다. 이는, 초당 5.02개 악성코드를 분석할 수 있는 성능임을 의미하는데, 실 환경에서의 분석성능 요구사항은 충분히 만족할 것으로 예상한다.

분석성능이 좋더라도 자원소모량이 많으면 현실적으로 운영할 수 없다. 제안 시스템의 실 적용을 위해서는 가볍고 효율적이어야 하는데 Figure 12는 endpoint에서 악성코드 분석을 진행하기 전과 후의 시스템 자원 소모량의 변화를 나타낸다. CPU의 부하가 약 25%, memory의 부하가 약 17% 증가한 것으로 나타났는데, 이는 7만여개의 파일을 한번에 분석해서 나타난 상황이다. 실제 환경에서는 파일분석 요구는 일반적으로 1분에 10개를 넘지 않을 것이므로, endpoint에 성능부담은 사실상 없을 것으로 판단된다.

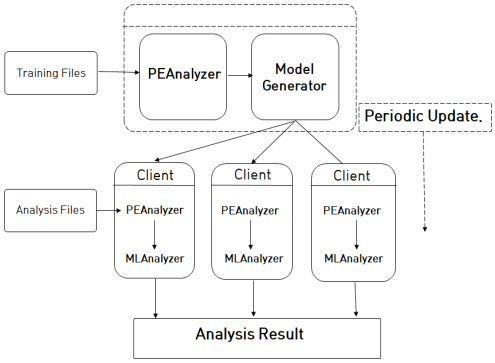


(Figure 12) Resource Usage with Malware Analysis

4.3 지속운영 Model

연구결과물에 대한 시험은 상용 환경에서 여기서는 본

논문에서 제안한 연구 결과물을 상용환경에서 활용하기 위한 지속운영 model을 제안하고자 한다. 중앙서버에서는 지속적으로 수집된 악성코드와 정상파일에 대해서 주기적으로 DNN에 기반한 학습모델을 생성한다. 생성주기는 정책에 다르지만 월1회, 분기별 1회 정도로도 무난할 것으로 판단된다. 이렇게 생성된 모델파일은 endpoint로 주기적으로 전송된다. 각각의 endpoint에서는 신규 유입된 파일에 대해 수신한 model에 기반하여 악성여부를 판단하는 방식으로 동작한다. Figure 13은 지속운영 model을 나타낸다. Figgure 13에서는 update된 model과 머신러닝 기반 모델만 설명하였는데, 각각의 endpoint에서 악성코드로 판단한 정보와 정상파일로 판단한 정보들은 중앙서버에서 통합 관리하여, 일괄적인 whitelist 및 blacklist 정책을 통해 효율적인 운영이 가능하다. 4.2에서 자원소모량을 측정하였는데, 실질적인 endpoint 환경에서의 분석대상 파일은 아무리 많아도 1분에 10개 이상을 넘지 않을 것이므로, 사실상 endpoint에 부담이 없을 것으로 판단된다.



(Figure 13) Resource Usage with Malware Analysis

5. 결 론

본 논문에서는 사이버 안보 전반에 큰 영향을 미치는 악성코드 침투공격 대응기술을 주로 다루었다. 특히, 기존의 악성코드 정적분석에서 주로 사용하였지만 난독화, packing 등으로 분석의미가 줄어든 API, DLL 정보를 재해석 및 의미를 부여하여 활용할 뿐 아니라, 중요하게 부각되지 않았던 PE 파일내의 여러 feature들의 의미를 해석하고 통계적 분포를 분석하여 악성코드 분석에 중요 factor로 활용한 악성코드 분석 및 탐지 기술을 제시하였다.

제안 모델에서는 기존의 API, DLL들의 각각의 기능적 특징을 벗어나 샘플 파일 내에 존재하는 string 관점에서 통계적 분포를 산출하고, 그에 따라 적절한 threshold를 설정하여 분포를 rank화 하고 PE header에 존재하는 feature을 전수 조사 및 선별하여 가공하는 시스템 복잡도가 낮은 정책을 적용하여 기존의 정적 분석기술의 경량성은 확보하며 anti-debugging에 영향이 적은 feature set을 구축하였다. 또한 선별한 feature들은 range화 함으로써 DNN 알고리즘의 높은 견고함도 얻을 수 있었다. 결론적으로 본 논문에서 제안하는 메커니즘으로 anti-debugging 기술에 영향을 크게 받지 않으며 기존 기법과 hybrid로 endpoint에 적용가능한 수준의 경량화된 악성코드 탐지 정책을 제안하였으며 더불어 사이버침투공격 전반에 대한 대응 방안을 제시하였다. 시험 결과에선 본 메커니즘의 악성코드 탐지율 및 ROC curve를 이용한 탐지 성능을 제시하고, 자원 소모량을 측정, 지속운영 model을 제시함으로써 제안하는 시스템이 endpoint 환경에 적용 가능함과 악성코드를 이용한 사이버 침투공격 전반에 대해 효율적으로 대응할 수 있음을 보였다.

차후 시스템의 복잡도를 고려하여 threshold 결정 정책과 머신러닝 알고리즘의 성능분석을 통하여 좀 더 높은 시스템의 성능을 나타낼 수 있을 것이라 기대하며 사이버 침투공격에 대응하기 위한 제안 기술의 시험결과는 의미와 해석을 추후 연구를 통해 지속함으로써 endpoint에서의 악성코드의 탐지에만 의존하지 않고 사이버 공격 행위 전반에 대한 효과적인 대응이 가능할 것으로 판단한다. 본 연구 결과물은 복잡한 sandbox 형태가 아닌 머신러닝 기법을 적용함으로써 endpoint 환경에서 독립적으로 운영 가능하며, 기존의 anti-virus 기술 및 여타 정적, 동적 분석기술과 연계하여 동작 시, 상호 보완적인 형태로 시너지를 낼 수 있을 것으로 판단되며 또한, 악성코드 변종 분석 및 최근 트렌드인 EDR 기술의 핵심 요소로 활용 가능할 것으로 예상된다.

참고문헌(Reference)

[1] D. Keragala, “Detecting Malware and Sandbox Evasion Techniques”, SANS Institute InfoSec Reading Room, 2016.
https://scholar.google.co.kr/scholar?hl=ko&as_sdt=2005&sciodt=0%2C5&cites=11695446247611230975&scipsc=&q=Detecting+Malware+and+Sandbox+Evasion+Techniques&btnG=

[2] M. Asha. Jerlin, C. Jayakumar, “A Dynamic Malware Analysis for Windows Platform – A Survey”, Indian Journal of Science and Technology, Vol. 8, No. 27, pp.1-5, 2015.
https://doi.org/10.17485/ijst/2015/v8i27/81172

[3] H.V. Nath, B. M. Mehtr, “Static Malware Analysis Using Machine Learning Methods”, Communication in Computer and Information Science, pp.440-450, 2014.
https://doi.org/10.1007/978-3-642-54525-2_39

[4] N. Rafiq, Y. Mao, “Improving heuristics. Virus Bulletin Conference”, pp.9-12, 2008.
https://www.virusbulletin.com/virusbulletin/2008/08/improving-heuristics

[5] A. Stewart, “Malware Dynamic Behavior Classification : SVM-HMM applied to Malware API sequencing”, Whiting School of Engineering(Johns Hopkins University), 2014.
https://scholar.google.co.kr/scholar?hl=ko&as_sdt=0%2C5&q=Malware+Dynamic+Behavior+Classification+%3A+SVM-HMM+applied+to+Malware+API+sequencing.&btnG=

[6] R. Veeramani, R. Ni tin, “Windows API based Malware Detection and Framework Analysis”, International Journal of Scientific & Engineering Research, Vol. 3, No. 3, 2012.
https://scholar.google.co.kr/scholar?hl=ko&as_sdt=0%2C5&q=Windows+API+based+Malware+Detection+and+Framework+Analysis&btnG=

[7] U. Baldangombo, N. Jambaljav, SJ. Horng, “A Static Malware Detection System Using Data Mining Methods”, Cornell University, 2013.
https://scholar.google.co.kr/scholar?hl=ko&as_sdt=0%2C5&q=A+Static+Malware+Detection+System+Using+Data+Mining+Methods&btnG=

[8] D. Bilar, “Statistical structures : Fingerprinting Malware for Classification and Analysis”, Proceedings of Black Hat Federal, 2006.
https://scholar.google.co.kr/scholar?hl=ko&as_sdt=0%2C5&q=Statistical+structures+%3A+Fingerprinting+Malware+for+Classification+and+Analysis&btnG=

[9] C. Burgess, F. Kurugollu, S. Sezer, K. McLaughlin, “Detecting Packed Executables Using Steganalysis”, Visual Information Processing(5th European Workshop (EUVIP), pp.1-5, 2014.
https://doi.org/10.1109/euvip.2014.7018361

[10] S. Gupta, H. Sharma, S. Kaur, “Malware Characterization using Windows API Call Sequences”, International Conference on Security, Privacy, and Applied Cryptography Engineering“, pp.271-280, 2016.
https://doi.org/10.1007/978-3-319-49445-6_15

[11] L. Hyo-young, K. Wan-ju, N. Hong-jun, L. Jae-sung, “Research on Malware Classification with Network Activity for Classification and Attack Prediction of Attack Groups”, The Journal of Korean Institute of Communications and Information Science, Vol. 42, No. 1, pp.193-204, 2017.
https://doi.org/10.7840/kics.2017.42.1.193

[12] A. Javaid, Q. Niyaz, W. Sun, M. Alam, “A Deep Learning Approach for Network Intrusion Detection System”, Proceeding of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies, pp.21-26, 2016.
https://doi.org/10.4108/eai.3-12-2015.2262516

[13] L. Etienne, “Malicious Traffic Detection in Local Networks with Snort”, EPFL-SSC, pp.1-34, 2009.
https://scholar.google.co.kr/scholar?hl=ko&as_sdt=0%2C5&q=Malicious+Traffic+Detection+in+Local+Networks+with+Snort&btnG=

[14] C. Wang, J. Pang, R. Zhao, X. Liu, “Using API Sequence and Bayes Algorithm to Detect Suspicious Behavior”, International Conference on Communication Software and Networks, pp.544-548, 2009.
https://doi.org/10.1109/iccsn.2009.60

[15] P. Vinod, R. Jaipur, V. Laxmi, M. Gaur, “Survey on Malware Detection Methods(3rd Hackers)”, Workshop on Computer and Internet Security, Department of Computer Science and Engineering, Prabhu Goel Research Centre for Computer & Internet Security, IIT, Kanpur, pp.74-79, 2009.
https://scholar.google.co.kr/scholar?hl=ko&as_sdt=0%2C5&q=Survey+on+Malware+Detection+Methods&btnG=

[16] P. Natani, D. Vidyarthi, “Malware Detection Using API Function Frequency with Ensemble based Classifier”, Communications in Computer and Information Science, pp.378-388, 2013.
https://doi.org/10.1007/978-3-642-40576-1_37

[17] D. Ucci, L. Aniello, R. Baldoni, “Survey on the Usage of Machine Learning Techniques for Malware Analysis”, ACM, Vol. 1, No. 1, 2017.
https://scholar.google.co.kr/scholar?hl=ko&as_sdt=0%2C5&q=Survey+on+the+Usage+of+Machine+Learning+Techniques+for+Malware+Analysis&btnG=

[18] G. Liang, J. Pang, C. Dai, “A Behavior-Based Malware Variant Classification Technique”, International Journal of Information and Education Technology, Vol. 6, No. 4, pp.291, 2016.
https://doi.org/10.7763/ijiet.2016.v6.702

[19] J. Sexton, C. Storlie, B. Anderson, “Subroutine based Detection of APT Malware”, Journal of Computer Virology and Hacking Techniques, Vol. 12, No. 4, pp.225-233, 2015.
https://doi.org/10.1007/s11416-015-0258-7

[20] R. Perdisci, W. Lee, N. Feamster, “Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces”, USENIX NSDI, 2010.
https://scholar.google.co.kr/scholar?hl=ko&as_sdt=0%2C5&q=Behavioral+Clustering+of+HTTP-Based+Malware+and+Signature+Generation+Using+Malicious+Network+Traces&btnG=

[21] G. Gu, R. Perdisci, J. Zhang, W. Lee, “Botminer : clustering analysis of network traffic for protocol- and structure independent botnet detection”, USENIX Security 2008.
https://scholar.google.co.kr/scholar?hl=ko&as_sdt=0%2C5&q=Botminer+%3A+clustering+analysis+of+network+traffic+for+protocol-+and+structure+independent+botnet+detection&btnG=

[22] Tae-woo. K, Cae-lk. C, Man-hyun. C, Jong-sub. M, “Malware Detection Via Hybrid Analysis for API Calls”, Journal of the Korea Institute of Information Security and Cryptology, 2007.
https://scholar.google.co.kr/scholar?hl=ko&as_sdt=0%2C5&q=Malware+Detection+Via+Hybrid+Analysis+for+API+Calls&btnG=

[23] G. Berger-Sabbatel, A. Duda, “Classification of Malware Network Activity”, Multimedia Communications Services and Security, pp.24-35, 2012.
https://doi.org/10.1007/978-3-642-30721-8_3

[24] M. Zubair. Rafique, P. Chen, C. Huygens, W. Joosen, “Evolutionary Algorithms for Classification of Malware Families through Different Network Behaviors”, Genetic and Evolutionary Computation Conference, pp.1167-1174, 2014.
https://doi.org/10.1145/2576768.2598238

[25] K. Iwamoto, K. Wasaki, “Malware Classification based on Extracted API Sequences using Static Analysis”, Internet Engineering Conference, pp.31-38, 2012.
https://doi.org/10.1145/2402599.2402604

[26] I. Ahmed, L. Kyung-suk, “Classification of Packet Contents for Malware Detection”, Journal in Computer Virology, Vol. 7, No. 4, pp.279-295, 2011.
https://doi.org/10.1007/s11416-011-0156-6

◐ 저 자 소 개 ◑

황 준 호(Jun-ho Hwang)
2018년 호서대학교 정보보호학과(공학사)
2018년~현재 호서대학교 일반대학원(정보보호학) 석사과정
관심분야 : 안드로이드, 정보보호, 머신러닝
E-mail : hwangso93@gmail.com

황 선 빈(Seon-bin Hwang)
2013년~현재 호서대학교 정보보호학과 학부과정
관심분야 : 정보보호, 네트워크
E-mail : tjsqlsdl2@naver.com

김 수 정(Su-jeong Kim)
2018년 호서대학교 정보보호학과(공학사)
2018년~현재 호서대학교 일반대학원(정보보호학) 석사과정
관심분야 : 안드로이드, 정보보호, 머신러닝
E-mail : sjkim395@gmail.com

이 태 진(Tae-jin Lee)
2003년~2017년 한국인터넷진흥원 팀장
2017년~현재 호서대학교 정보보호학과 교수
관심분야 : 시스템 보안, 악성코드 분석, 침해사고 대응
E-mail : kingecs0@gmail.com