파워쉘 공격 탐지방법에 대한 딥러닝 연구

조진호*, 박양훈*, 박광철*, 최선오*
*호남대학교 컴퓨터공학과
e-mail : whwlsgh1144@gmail.com, didgnsd16784@naver.com
ar16204@naver.com, suno@honam.ac.kr

Deep Learning Study on PowerShell Attack Detection Method

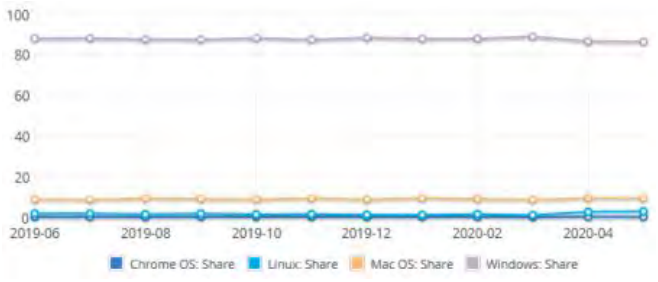
Jin-Ho Cho*, Yang-Hun Park*, Gwang-Cheol Park*, Sun-Oh Choi*
Dept. of Computer Engineering, Honam University

요 약

PowerShell은 닷넷 프레임 워크에 기반을 둔, 커맨드 라인 인터페이스 및 스크립트 언어의 특징을 가진 Microsoft의 명령어 인터프리터로, 이를 내장한 Windows 운영체제 점유율의 증가와 강력한 기능으로 PowerShell을 이용한 공격은 계속 증가하고 있다. 이연구에서는 딥러닝을 사용하여 악성코드를 효과적으로 탐지하는 방법과, 이에 필요한 시퀀스 데이터 추출의 방법을 찾아내는 것에 목적이 있다.

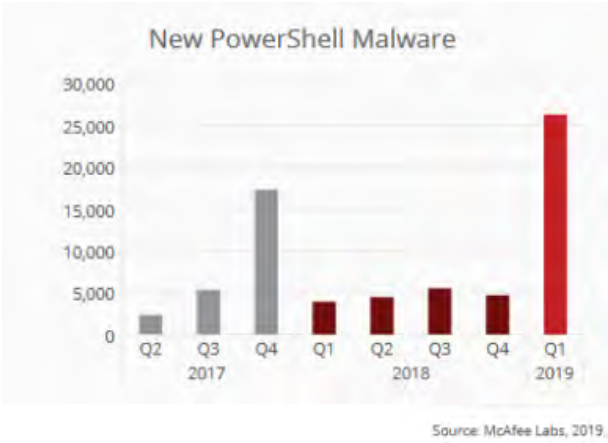
1. 서론

파워쉘(PowerShell)은 마이크로소프트가 개발한 커맨드 라인 인터페이스(CLI) 셸 및 스크립트 언어를 특징으로 하는 명령어 인터페이스 이다. 기본적으로 닷넷 프레임워크(NET Framework)를 기반하여 COM, WMI, XML 등 다양하고 강력한 기능을 제공하나, [1] 최근 닷넷 코어(.NET Core)를 기반으로 한 파워쉘 코어(PowerShell Core)의 도입으로 리눅스, Mac OS 등 다양한 플랫폼에서 사용이 가능하게 되었다.[2] 2020 년 6 월기준 IT 조사기관인 넷 마켓쉐어(Net Marketshare)에 의하면 MS 윈도우의 점유율은 87.97%이다. 이 중 파워셸이기본으로 내장된 윈도우 7 이상 점유율은 86.34%[3]으로 이는 별도의 프로그램 없이 실행 가능함을 의미한다.



(그림 1) Net Marketshare 에서 조사한 운영체제 별 점유율

파워쉘 멀웨어(Malware)는 Powelikes 나 Kovter 을 시작으로 계속적으로 증가하고 있으며, 2019 년 1 분기엔 2018 년 4 분기에 비해 5 배 이상 증가하는 모습을 보인다[4].



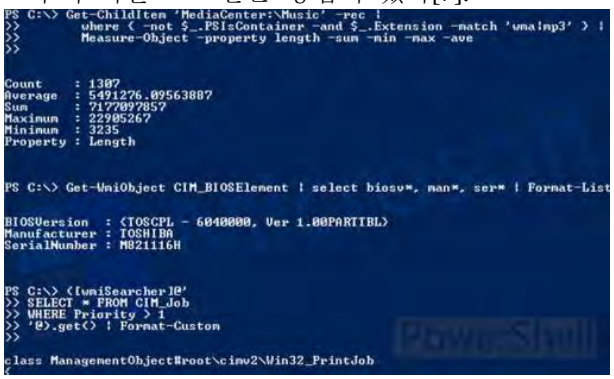
(그림 2) McAfee에서 조사한 새로 발견되는 파워쉘 멀웨어

또한 파워쉘은 유연한 문법과 리플렉션, 와일드카드 같은 다양한 난독화 기법이 존재하여 기존 안티바이러스의 시그니쳐 기반 탐지를 어렵게 한다[5]. 이에 본 논문은 다양한 샘플 데이터를 이용하여 파워쉘 시퀀스를 추출한 후 다양한 딥러닝 방법을 통해 악성 행위를 하는 멀웨어를 효과적으로 탐지하고자 한다.

2. Fileless Malware

파일이 존재 하지 않는 악성코드의 합성어로, 하드디스크에 저장하지 않고 시스템 메모리에 상주하는 악성코드의 변종이다[6]. 이 악성코드는 하드디스크 및 SSD 에 기록을 하지 않으므로 포렌식 작업을 어렵게 하며, 시스템 메모리상에 존재하므로 재 부팅시 소멸된다. 이번 연구에서 다룬 악성코드는 Script-based Technique

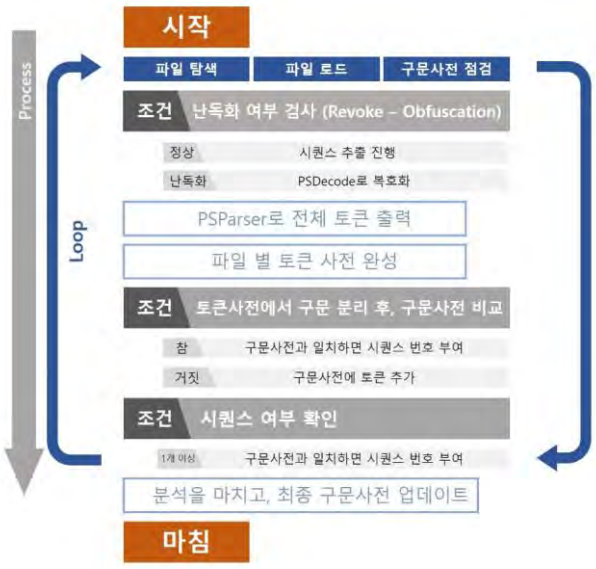
방식으로 PowerShell 에서 제공하는 Scripting Language 기능을 사용하는 것으로, 메모리에서만 실행되는 Mal Ware 를 전달하도록 악성 Scripts 를 만들어 사용하는 것이다. 이러한 Script 는 Registry 에 숨기거나, C&C 서버로부터 다운로드 받는 방법이 있다[7].



(그림 3) PowerShell Fileless Malware Scripts

3. 설계

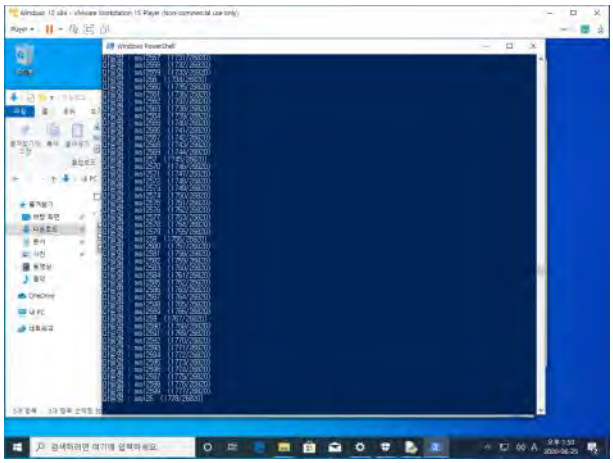
첫 번째, 분석할 파일목록을 스캔 후, 구문사전을 점검한다. 만약 파일이 일정 크기보다 크다면, 분석시간 최적화를 위해 파일을 일정크기 부분으로 자르게 설계하였다. 두 번째, PSParser로 전체 토큰을 출력하여 토큰사전을 완성한다. 이후 토큰 사전에서 필요한 내용분리후, 구문사전을 비교해 결과가 참 일 경우, 구문사전과 일치하면 시퀀스 번호를 부여하고, 거짓일 경우구문사전에 토큰을 추가한다. 세 번째, 시퀀스 여부를 확인하여 데이터가 1개이상일 경우 CSV파일에 기록하고 저장한다. 마지막으로 분석이 끝나면 최종 구문 사전을 업데이트한다.



(그림 4)설계 알고리즘 구성

3.1 환경 설정

이번 연구에서는 악성파일을 다룬다는 점을 착안하여 가상 머신(VMware)상에서 시행했으며, 게스트 OS 는 Windows 10 Pro (버전 2004), 사용된 언어는 Python 3.8 을 사용했다.



(그림 5) 가상머신(VMware)환경

4. 난독화

데이터 샘플 파일 중 일부는 난독화가 이루어져있는 파일이 존재했다. 기존 프로그램에서 작동 시 각종 인코딩 오류를 일으켜 이번 연구에서 PowerShell 스크립트로 제작한 Revoke-obfuscation모듈과, PSDecode 모듈을 사용하여 일부 샘플 파일들을 복호화 할 수 있었다.

4.1 Revoke-Obfuscation

이번 데이터 샘플 파일 중 난독화가 되어 있는 파일들은 여러 종류의 인코딩/압축 방식으로 이루어져 있다. 다음은 Powershell 스크립트 난독화 종류를 보여준다.

종류	설명
Random Case	대소문자 변화 (예 : tEsT)
Division	문자 분할 (예 : 'te'+'st')
Reorder	문자 순서 지정 (예 :'{1}{0}' -f 'te','st'
Back Ticks	Back Tick 문자 추가 (예 : `t`e`s`t)
Call Operator	스크립트 블록 사용 (예 : & (test))
Whitespace	문자 중간에 공백 추가 (예 :'te' +'st')
Ascii Char Assigns	아스키코드 사용(예 : [char]116+[char]101…)
Replace	Replace 기능 사용 (예:'aesa'.replace('a','t')
Base64 Encoding	Base64 인코딩 (예 : dGVzdA==)
Invoke-Obfuscation.ps1	파워쉘 난독화 도구

(그림 6)Powershell 스크립트 난독화 종류

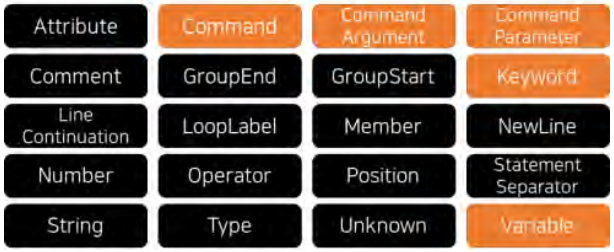
이중, 연구에서 사용한 [9]Revoke-Obfuscation 파워쉘 모듈은 난독화 된 PowerShell 명령 및 스크립트를 대규모로 감지하는 오픈 소스 프레임 워크로 PowerShell의 AST(AbstractSyntaTree)를 사용하여 모든 입력 PowerShell스크립트에서 수천 개의 기능을 신속하게 추출할 수 있다.

(그림 7). Revoke-Obfuscation

정상 데이터 파일은 대부분 난독화가 되어 있지 않은 파일로 분석되었으며, 악성 데이터 파일 3,856 개 중, 난독화 파일 개수가 54 개로 검출 되었다. 이는 대부분의 악성데이터 샘플 파일들이 인코딩 되어있는 것을 파악할 수 있었다.

5. 토큰화

PSParser를 사용하여 PowerShell코드를 토큰화 하면 코드 문자를 문자별로 읽고 문자를 의미 있는 단어인 토큰으로 그룹화한다. PSParser가 예상하지 못한 문자를 발견하면, 즉, 문자열이 이중 문자로 시작되지만, 단일 문자로 끝날 때 구문 오류를 생성한다. 이번 연구에서 20가지 토큰 유형 중 Attribute, Command Argument, Comman Paramter를 중점으로 연구를 진행하였다[8].

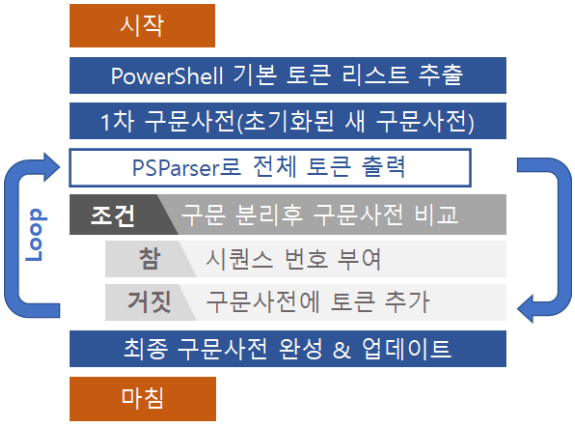


(그림 8) PSParser 의 20 가지 토큰 유형

5.1 구문 사전

5.1.1 부분 능동적 구문사전

각 분석파일은 서로 다양한 명령어들로 구성되어 있다. 분석파일에 존재하는 토큰이 구문사전에 포함되어 있지 않을 경우, 해당 토큰에 대한 시퀀스 데이터를 추출할 수 없어, 이로 인해 딥러닝시 정확도의 감소 등을 유발하는 장애요소가 될 수 있다. 그러하여, 분석할 파일에 존재하는 모든 토큰들이 구문사전 내에 존재 해야하며, 이에 맞는 구문사전을 만들기 위해 매 분석파일마다 구성된 명령어들을 조사하여 기존의 구문사전에 등록되지 않은 새로운 내용을 자동으로 추가한다. 이러한 방법으로 구문사전에서 특정 토큰에 대한 부재문제를 해결할 수 있다.

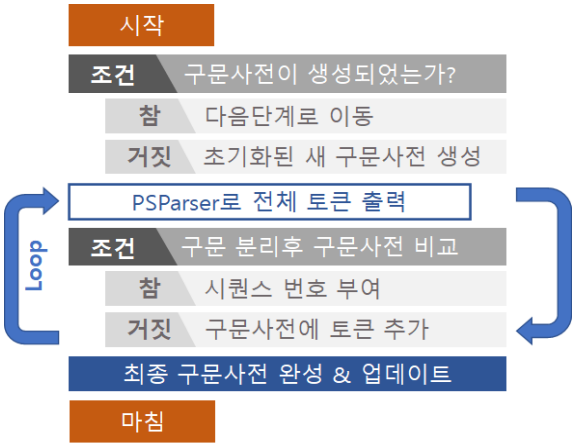


(그림 9)부분 능동적 구문사전 구성

5.1.2 완전 능동적 구문사전

위에서 소개한 '부분 능동적 구문사전'에서는 새로운 분석 파일에서 기존의 구문사전에 존재하지 않던 새로운 명령어가 발견될 경우 해당 내용을 구문사전에 추가하여 왔다. 이로 인해 부재문제는 해결할 수 있었으나, 분석을 거듭할수록 구문사전의 크기가 늘어나며, 실제로 사용하지 않는 내용이 증가함에 따라 전체 처리속도가 감소된다는 단점이 발생한다. 이런 문제를 해결하고자 기존 구문사전에 새로운 내용을 추가하는 대신, 구문사전을 분석할 파일에 맞게 재구성하는 방식을 사용한다.

(그림 10) 완전 능동적 구문사전 구성

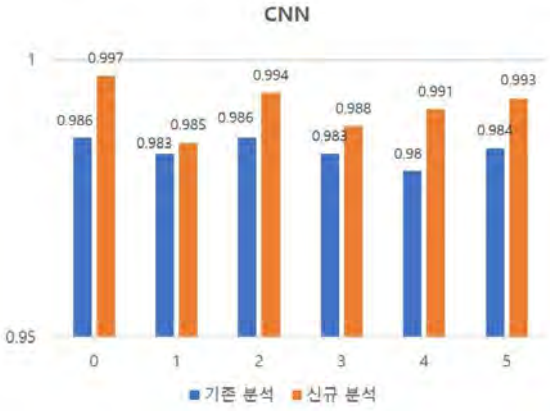


(그림 11) 하이브리드 구문사전 구성

이를 보완하기 위해, 부분 능동적과 완전 능동적의 특징을 결합하여 하이브리드 구문사전 방식을 고안하였다. 이 방식은 분석파일을 분할하여 분석할 때 일관성이 떨어지지 않도록 완전 능동적 구문사전의 초기화 기능을 배제하였다.

6. Deep Learning

아래 그래프는 이번 연구에 사용된 CNN, LSTM, CNN 과 LSTM 혼합 딥러닝 기술 적용 결과이다.



(그래프 1) CNN 모델 분석 결과

(그래프 2) LSTM 모델 분석 결과

(그래프 3)LSTM+CNN 모델 분석 결과

실험은 악성 3,856개, 정상 22,543개의 샘플 데이터를 사용하였으며, 정상 파일과 악성 파일의 개수를 맞추기 위해 6개의 파트로 나눠 진행하였다. 두 실험 모두 하이브리드 구문사전 알고리즘에 기반 하였으며, 기존 분석과 신규 분석의 차이점은 기존 분석은 난독화 탐지 후 복호화 작업이 진행되지 않았다. 기존 테스트 데이터의 난독화 파일들을 복호화 후 테스트 데이터에 포함 시켜 학습시킨 결과, 1~2%의 평균 값의 차이를 보였다. CNN(합성 신경망)은 정확도의 수치가 탐지율에 비해 상대적으로 낮게 나온 것으로 확인되었고, LSTM(순환신경망)은 정확도의 수치가 탐지율과 비슷함을 보여주지만, 전반적으로 조화평균의 값이 CNN 모델에 비해 떨어진다. 이것을 보안하여 LSTM+CNN 모델을 적용하였는데, LSTM에 비해 높은 탐지율과 조화 평균을 보이고 CNN과 거의 비슷한 양상을 보이는 것으로 확인 되었다.

7. 결론

이번 연구에서 구문사전 분석 프로그램을 구현 하면서 몇 가지 이슈가 발견 되었지만 가장 큰 문제는 난독화 된 파워쉘 코드이다. 특히 악의적 목적을 가진 파일 중 난독화가 되어있어 제대로 특징을 추출할 수 없는 문제가 발생되었다. 이를 해결하기 위해 Revoke-Obfuscation 과 PSDecode 모듈의 도움을 받아 복호화 작업을 진행하였다. 그 결과 딥러닝 정확도가 소폭 상승하는 결과가 나왔지만 해당 모듈이 모든 난독화에 대응되지 않는다는 단점이 있다. 이는 본 논문에서 사용되는 PSParser 기반 특징추출 기법이 가지는 한계이다.

참고문헌

[1] COM, OLE, NET Framework Concept Search https://sanseolab.tistory.com/49 (2020/06/27)
[2] Wikipedia, “PowerShell”, https://en.wikipedia.org/wiki/PowerShell, (2020/06/24)
[3] NET MARKETSHARE, “Operation System Market Share”, https://netmarketshare.com/operating-system-market-share.aspx, (2019/06~2020/05)
[4]McAfee, “McAfee labs threats report Augest 2019”, Augest 2019
[5] Seung-Hyeon Lee, Jong-Sub Moon. (2018). PowerShell-based Malware Detection Method Using Command Execution Monitoring and Deep Learning. Journal of The Korea Institute of Information Security & Cryptology, 28(5), 1197-1207
[6]PowerShell, fileless malware’s great attack vector, https://www.pandasecurity.com/mediacenter/malware/powershell-fileless-malware-attack-vector (2020/07/01)
[7]PowerShell&Fileless Malware, https://blog.naver.com/aepkoreanet/221322291834 (2020/07/01)
[8] Tokenzing Powershell Scripts https://powershell.one/powershell-internals/parsing-and-tokenization/simple-tokenizer(2020/05/23)
[9]Revoke-obfuscation https://github.com/danielbohannon/Revoke-Obfuscation