Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.5, Oct. 2016 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2016.26.5.1313

스마트 제조 산업용 네트워크에 적합한 Snort IDS에서의 전처리기 구현

하 재 철†‡
호서대학교

Preprocessor Implementation of Open IDS Snort for Smart Manufacturing Industry Network

Jaecheol Ha†‡
Hoseo University

요 약

최근 인터넷을 통한 공공 기관이나 금융권에 대한 바이러스 및 해킹 공격이 더욱 지능화, 고도화되고 있다. 특히, 지능형 지속 공격인 APT(Advanced Persistent Threat)가 중요한 사이버 위협으로 주목을 받았는데 이러한 APT 공격은 기본적으로 네트워크상에서 악성 코드의 유포를 통해 이루어진다. 본 논문에서는 스마트 제조 산업에서 사용할 수 있도록 네트워크상에서 전송되는 PE(Portable Executable) 파일을 효과적으로 탐지하고 추출하여 악성코드 분석을 효과적으로 할 수 있는 방법을 제안하였다. PE 파일만 고속으로 추출하여 저장하는 기능을 공개 침입 탐지 툴인 Snort의 전처리기단에서 구현한 후 이를 하드웨어 센서 장치에 탑재하여 실험한 결과, 네트워크상에서 전송되는 악성 의심 코드인 PE 파일을 정상적으로 탐지하고 추출할 수 있음을 확인하였다.

ABSTRACT

Recently, many virus and hacking attacks on public organizations and financial institutions by internet are becoming increasingly intelligent and sophisticated. The Advanced Persistent Threat has been considered as an important cyber risk. This attack is basically accomplished by spreading malicious codes through complex networks. To detect and extract PE files in smart manufacturing industry networks, an efficient processing method which is performed before analysis procedure on malicious codes is proposed. We implement a preprocessor of open intrusion detection system Snort for fast extraction of PE files and install on a hardware sensor equipment. As a result of practical experiment, we verify that the network sensor can extract the PE files which are often suspected as a malware.

Keywords: Smart Manufacturing Industry Network, APT attack, PE file, Open IDS Snort

I. 서 론

최근 정부에서는 제 4차 산업 혁명기를 맞이하여 ICT와 제조업이 융합되어 산업 기기와 생산 과정이 모두 네트워크로 연결되어 상호 소통하면서 전사적인 최적화를 달성하기 위한 제조업 혁신 전략을 가동하고 있다. 이러한 제조업 경쟁력 강화 전략의 핵심이 스마트 팩토리(smart factory) 구현이며 이를 통하여 현재의 부분 자동화 단계를 넘어 제조 생태계 차원의 고도화와 변화를 유도하고 있다. 스마트 팩토리란 IT 기술과 결합된 자동화된 제조 설비를 갖추고 제품의 디자인, 제품 생산 공정, 조달 및 물류,

Received(09. 07. 2016), Modified(10. 04. 2016), Accepted(10. 05. 2016)
† 주저자, jcha@hoseo.edu
‡ 교신저자, jcha@hoseo.edu(Corresponding author)

관리 체계를 연결하여 제조 산업의 경쟁력을 향상시키기 위한 전략이다. 따라서 스마트 제조 공장 구축의 핵심 원천이 되는 기술은 모든 사물을 연결할 수 있는 사물 인터넷(Internet of Things)과 방대한 제조 과정의 데이터를 분석할 수 있는 빅데이터(big data) 기술이라고 할 수 있다[1,2].

그러나 스마트 제조 산업 시스템에서는 인터넷과 같은 개방형 망과의 연결, 무선 설비 사용, 자동화를 위한 원격제어 기능 그리고 업무망과 제어망의 연결과 같은 특징들이 보안 위협을 가중시키는 요소가 되고 있다. 특히, 스마트 팩토리에서는 관련 설비와 장치가 대부분 네트워크를 통해 주 제어 시스템과 연결되어 원격 제어가 가능한 환경이므로 필연적으로 보안 문제가 발생할 수 밖에 없다. 만약, 스마트 팩토리나 산업 제어 시스템에서 보안 체계가 무너지면 고급 산업 기술 정보의 유출은 물론, 외부의 원격제어를 통한 오동작 유발 및 시스템 마비, 주요 설비의 파괴로 인해 대형 사고로 이어질 수 있는 중대한 위험에 직면하게 된다.

스마트 제조 산업 현장에서는 사물 인터넷과 같은 통신 기능이 필수적인데 반해 컴퓨터 기반의 정보통신망에 대한 보안 위협은 지능화, 다양화, 그리고 대형화되고 있으며 피해 규모도 늘어나고 있다. 특히, 악성 코드를 유포한 후 지속적으로 시스템의 중요 정보를 탈취해 가는 지능형 지속 위협(Advance Persistent Threat)인 APT 공격이 새로운 보안 위협으로 큰 주목을 받기도 하였다[3, 4]. 또한, Drive-by Download 공격은 웹 브라우저 또는 플러그인의 취약점을 이용하여 악성 사이트 접속 시 사용자도 모르게 악성 코드를 감염시키는 공격으로서 웹 기반 공격으로 알려져 있다[5].

이러한 사이버 공격을 방어하기 위한 대응책은 크게 네트워크를 통한 공격자의 침입을 탐지하기 위한 침입 탐지 시스템을 설치 운영하는 기술[6]과 유포되는 파일의 코드 분석을 통해 악성 코드 여부를 판별하는 분석 기술로 크게 나누어 볼 수 있다[7, 8, 9]. 전자는 공격자의 악의적인 행위 시도 여부를 탐지하는 것으로 특정 파일 검색이나 서비스 거부 공격 방지책과 같은 네트워크 기반의 대응책이라 볼 수 있으며, 후자는 이미 시스템에 유입되어 있는 코드에 대해 유해성 여부를 체계적으로 분석하여 악성 코드인 경우 이를 제거하는 방어 기술이라 할 수 있다. 방어 시스템은 구현 방법에 따라 이 두 가지 기술을 하나의 시스템에서 통합하는 경우도 있으나 구현과 동작 효율성을 고려하여 분리하는 경우도 있다.

네트워크를 통해 유포되는 바이러스나 악성 코드들을 판별하여 APT 공격이나 제로데이(Zero-Day) 공격들을 방어하기 위해서는 우선 공격 의심 파일만을 수집하는 센싱 기술이 필요하다. 즉, 네트워크로 유입되는 모든 파일을 대상으로 악성 코드 여부를 검사하는 것은 거의 불가능하므로 의심이 되는 특정 파일만 탐지하여 추출하는 기술이 필요하다. 예를 들어 악성 코드는 일반적으로 PE(Portable Executable) 파일 형태[10, 11]로 제작되어 유포되는데 네트워크로 유입되는 PE 파일만 탐지하여 추출할 수 있는 센서 장비가 필요하다. 특히, 네트워크가 대형화되고 유입되는 정보가 너무 많은 경우에는 악성 여부를 판별하기 위한 대상 파일을 고속으로 추출하여 저장하는 것이 무엇보다도 중요하다.

악성 코드가 의심되는 PE 파일을 고속으로 추출하기 위해서는 전용 센서 장비를 구비하는 것도 중요하며 이를 효과적으로 운영하기 위한 보안 체계도 필요하다. 또한, 센서 장비에서 추출한 파일이 별도의 악성 분석 시스템을 통해 악성 코드로 판별되면 공격자를 역추적하기 위한 정보가 저장되어 있어야 한다. 이 외에도 동일한 PE 파일을 연속적으로 보내 센서 장비를 무력화시키기 위한 DoS(Denial of Service) 공격에 대비한 데이터 관리도 필요하다.

본 논문에서는 스마트 제조 산업용 네트워크에서 저비용으로 사용할 수 있도록 네트워크 탭(tap) 기능을 활용하여 악성 코드로 사용되는 PE 파일만 고속으로 추출할 수 있는 센서 장치를 직접 구현하고자 한다. 즉, 악성 코드 분석을 위해 특정 형식의 네트워크 패킷을 탐지하고 이를 효과적으로 파일화시켜 저장하기 위한 장치를 개발한다. 이를 위해 공개 침입 탐지 시스템(Intrusion Detection System, IDS)인 Snort[12-14]를 개선하여 전처리기 단계에서 특정 파일만 추출하도록 구현하였다. 논문에서 구현한 탐지 센서는 PE 파일을 고속으로 추출하는 방법에 관한 것이며, 악성 코드 여부를 판별하는 것은 별도의 분석 시스템에서 처리한다고 가정한다.

스마트 제조 산업 현장에서는 각 회사가 필요로 하는 침입 탐지 기능들을 유연하게 운용할 수 있으며 대량의 데이터를 고속으로 처리할 수 있는 시스템을 사용하는 것이 효과적이다. 이러한 측면에서 공개 IDS인 Snort를 사용하면 회사가 필요한 기능을 모듈별을 자유롭게 추가하거나 불필요한 기능을 제거함으로써 고속 구현을 실현할 수 있다.

본 논문의 2장에서는 IDS인 Snort의 개발 환경 및 PE 파일의 구조에 대해 살펴보고 3장에서는 PE 파일을 탐지하고 이를 추출하기 위한 알고리듬을 제안한다. 4장에서는 Snort의 전처리기에서 PE 탐지 및 추출 알고리듬을 구현한 것과 실제 파일 탐지 센서 장비를 구축하여 실험한 결과를 제시한다.

II. Snort 및 PE 파일 구조 분석

2.1 Snort 구조 분석

Snort는 패킷을 로깅하거나 실시간으로 트래픽을 분석하는 기능은 물론 네트워크 기반 침입 탐지 기능을 갖춘 소프트웨어 기반의 툴로서 소스 코드가 공개되어 있다. Snort는 크게 스니퍼(sniffer), 전처리기(preprocessor), 탐지 엔진(detection engine), 경고 및 로깅(alert and logging)의 4가지 모듈로 구성되어 있으며 이더넷 환경에서 TCP/IP 프로토콜을 중심으로 개발되었다. 또한, Snort는 악의적인 패킷이 있으면 탐지 엔진 부분에서 시그니처(signature)와 매칭되는지 여부를 판단하는 오용(misuse) 탐지 기법을 사용하여 패킷의 유해성을 판단한다. Fig. 1은 Snort의 기본 구조를 나타낸 것이다.

스니퍼에서는 네트워크를 수집(promiscuous) 모드로 동작시켜 이더넷 카드로 들어오는 모든 패킷을 수집하는 기능을 수행한다. 스니퍼에서는 패킷 수집 라이브러리인 Libpcap을 이용하여 패킷을 캡처한다. 또한, 네트워크 트래픽을 감시하면서 이를 분석하여 사용자가 보기 쉬운 형태로 변환하는 디코딩 기능도 수행한다.

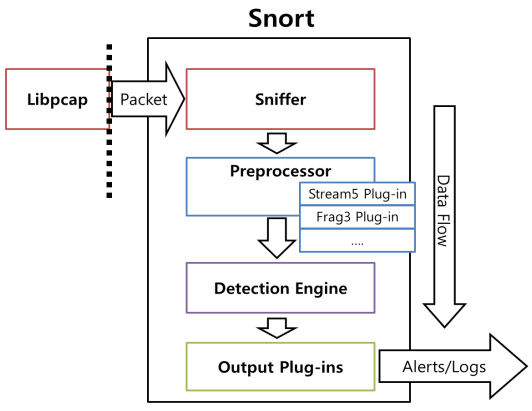

Fig. 1. Basic structure of Snort

전처리기에서는 분할된 패킷을 재조합한 후 기본적인 악성 행위를 탐지한 후 그에 따른 경고 및 로그를 남긴다. 전처리기에서 구현된 기능은 플러그인(plug-in) 형태로 적용되기 때문에 IDS의 처리 능력 및 용량에 따라 전처리 기능을 활성화시키거나 비활성화 시킬 수도 있는 장점이 있다. 따라서 유입되는 패킷에 대한 특별한 사전 처리가 필요한 경우에는 전처리기에서 별도의 플러그인을 만들어 추가할 수 있다.

탐지 엔진에서는 전처리기로부터 패킷을 받아 정해진 기본 규칙이나 사용자가 정의한 규칙과 비교하여 일치하는 패킷이 있으면 경고 및 로그를 남긴다. 탐지 엔진에서 사용하는 규칙은 규칙 헤더와 규칙 옵션으로 구분된다. 규칙 헤더에는 패킷 탐지 시 취할 행동(alert, log, pass 등), 네트워크 패킷의 종류(TCP, UDP 등), 그리고 출발지와 목적지 IP 주소, 포트 주소 등이 포함되어 있다.

경고 및 로깅 단계는 전처리기 혹은 탐지 엔진에서 이상 패킷이 탐지 되었을 때 마지막으로 패킷을 처리하는 곳이다. 즉, 유입 패킷이 유해하다고 판단되면 설정된 규칙에 따라 syslog를 이용하거나 특정 폴더에 파일 및 관련 로그를 남기고 경고를 발생하게 된다.

2.2 PE 파일 분석

PE는 "Portable Executable"의 약어로서, Win32 플랫폼 하에서 공통적으로 이식하여 사용할 수 있는 실행 파일 형식을 말한다. PE 포맷은 마이크로소프트(Microsoft)의 윈도우 3.1 버전부터 지원되었으며 EXE, SCR, DLL, OCX, SYS, OBJ 등의 확장자를 가지고 있다. 이러한 PE 파일들은 컴퓨터에서 실행되는 파일이므로 대부분의 악성 코드는 이러한 PE 파일 형식으로 다운로드 되어 PC 자체를 감염시키거나 좀비 PC화시킬 수 있도록 제작된다. 따라서 탐지 센서를 통해 네트워크상에서 전송되는 수많은 파일 중에서 PE 파일만을 추출하여 분석하기 위해서는 기본적으로 PE 파일 구조 분석이 필요하다.

다음 Fig. 2는 PE 포맷의 전체 구조를 나타낸 것이다. PE 파일은 크게 파일이 실행되기 위해 관련 정보를 가지고 있는 헤더 부분과 실제 데이터, 리소스를 가지고 프로그램이 실행되는 부분인 섹션 부분

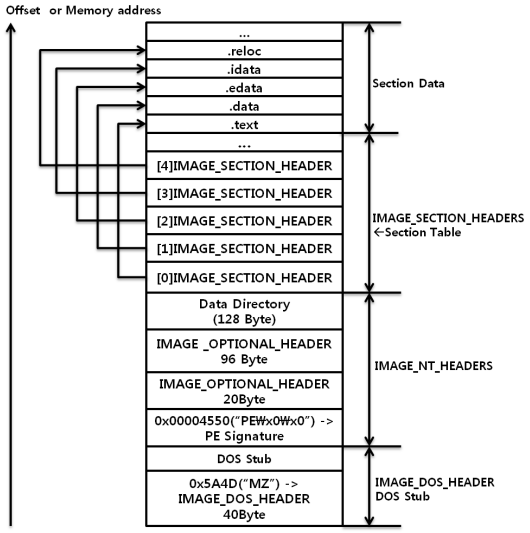

Fig. 2. PE file format

으로 나눌 수 있다. 헤더 부분은 다시 세부분으로 나누어지는데 DOS와의 호환성을 위해 사용되는 DOS 헤더, 실제 프로그램 실행에 중요한 NT 헤더, 그리고 각 세션에 대한 파일상의 위치 정보, 실제 가상 메모리상에서의 위치 정보 등을 담고 있는 섹션 헤드로 나누어진다.

PE 포맷은 반드시 DOS Header로 시작하는데 이 헤더는 모두 64바이트로 구성되어 있다. 여기에는 실행 파일을 나타내는 시그니처(signature)인 "MZ" 문자가 아스키 코드 값으로 저장되어 있으며 마지막 4바이트는 실제 PE 코드를 실행하는 주소로 구성되어 있다. 스텁(stub) 코드 부분은 OS가 PE 파일 형식을 알지 못할 때 경고문을 실행하기 위한 문구가 들어 있는 부분이다. 그리고 NT 헤더는 PE 시그니처, FileHeader 그리고 OptionalHeader로 구성되어 있다. PE 시그니처는 PE 파일을 실행하는 시작 부분을 나타낸 것으로 "PE" 문자가 아스키 코드 값으로 저장되어 있다. FileHeader는 20바이트이며 이 안에는 섹션의 수, 시간, OptionalHeader의 크기와 파일의 특성 등이 포함되어 있다. OptionalHeader는 36종류의 기본 필드로 구성된 96바이트 크기의 구조체와 IMAGE_DATA_DIRECTORY 배열을 가진다.

섹션 헤더는 섹션의 수에 따라 여러 개를 가질 수 있는데 각각 40바이트이고, 각 섹션에 대한 메모리상에서의 크기, 파일상에서의 크기 그리고 시작 오프셋 값 등을 가진다. 섹션 부분은 프로그램에 따라 조

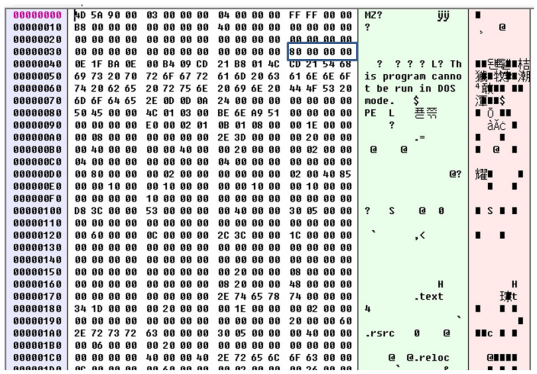

Fig. 3. An example of PE file format

금씩 다르지만 보통 프로그램의 실행 코드가 있는 .text 섹션, 초기화된 전역 변수 값이나 문자열 함수명 등을 가지고 있는 .data 섹션, 아이콘이나 커서 등 윈도우 어플리케이션 데이터들이 담겨져 있는 .rsrc 섹션 등이 있다.

Fig. 3은 특정 EXE 파일에 대한 PE 구조 데이터를 16진수 값으로 나타낸 것이다. 그림에서 보는 바와 같이 파일의 시작 부분에 "MZ" 시그니처 값인 0x4D, 0x5A 값이 저장되어 있으며 4번째 라인의 마지막 4바이트에 0x80번지가 보이는데 이것이 NT 헤더의 시작 주소이다. 그리고 NT헤더의 시작 주소에 보면 "PE" 시그니처 값인 0x50, 0x45 값이 저장되어 있음을 볼 수 있다.

III. 고속 PE 파일 탐지 및 추출 알고리듬 개발

3.1 탐지 센서 네트워크 구성

파일의 악성 여부를 분석하기 위해서는 먼저 PE 파일을 탐지하고 이를 저장하는 과정이 필요하다. 이러한 기능을 하는 장비를 의심 파일 탐지 센서라 하고 본 논문에서는 Fig. 4와 같은 네트워크 구조로 설계하였다. 그림에서 인터넷은 외부망을 나타내고, 스위칭 장비 아래는 내부망을 나타내는데 외부망에서 들어오는 패킷은 라우터를 경유하게 된다. 라우터는 보통 내부망의 스위치와 연결되는데 그 사이에 패킷을 탐지하기 위한 TAP 장비를 설치한다.

TAP 장비는 외부에서 들어오는 패킷과 내부에서 나가는 패킷을 탐지 센서로 보내면서 두 네트워크 간의 투명한 연결을 보장한다. 또한, 무선으로 연결된 내부망에서도 TAP 장비를 이용하여 탐지 센서로 패

킷을 전달할 수 있도록 전체적인 네트워크를 구성하였다.

PE 파일 추출하는 기능과 추출된 파일이 악성 코드인지 그렇지 않은 지를 분석하는 기능을 하나로 통합한 침입 탐지 시스템을 구축할 수도 있다. 그러나 이 경우에는 악성 코드 분석 시간이 크게 증가하여 처리 부하를 지연시킬 수도 있어 서비스 거부 공격의 표적이 될 수도 있다. 또한, APT 공격과 같이 지속적인 위협에 대해서는 PE 파일을 별도로 저장한 후 악성 코드 분석 시스템에서 별도로 분석하는 것이 효과적일 수도 있다. 따라서 본 논문에서는 이러한 점을 고려하여 탐지 센서와 악성 코드 분석 기능을 따로 분리하여 설계하였으며 탐지 센서에서 PE 파일을 고속으로 추출하는 기능만 구현하고자 한다. 즉, 이 센서 장비의 핵심 기능은 네트워크상에서 전송되는 모든 파일에 대해 PE 파일 여부를 검사한 후 PE 파일인 경우 이를 효과적으로 추출하여 저장한 후 악성 분석 시스템으로 전달하는 것이다.

본 논문에서 구현하고자 하는 악성 코드 의심 파일 탐지 센서의 기능은 다음과 같이 요약할 수 있다.

① 태핑(tapping)을 통해 패킷을 탐지하거나 추출하더라도 네트워크상에서 투명한 데이터 흐름을 유지할 수 있어야 한다.

② 전송되는 패킷량이 많더라도 탐지 및 저장을 고속으로 처리할 수 있어야 한다. 즉, 센서 장비의 처리 시간이 짧고 실시간 처리가 가능하도록 설계되어야 한다.

③ PE 파일이 최종적으로 악성 코드로 분석되었을 경우, 시스템 공격자나 파일 유포자를 추적할 수 있는 정보를 저장하여야 한다.

④ PE 파일 및 역추적 관련 정보를 악성 코드 분석 시스템에 전달하기 전까지 일시적으로 보관해야 할 저장 공간이 최소화되어야 한다.

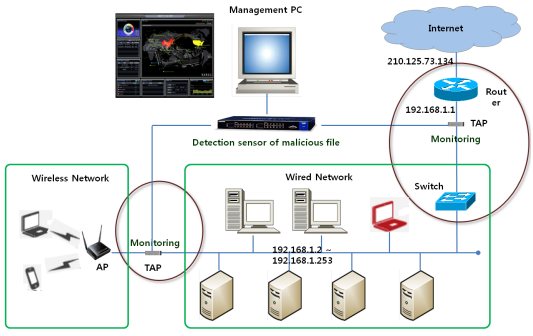

Fig. 4. Sensor network for PE file detection

3.2 PE 파일 탐지 알고리듬

네트워크상에서 라우터와 스위치 사이에서 태핑된 패킷은 탐지 센서 장비로 유입이 되며 탐지 센서는 PE 파일 여부를 결정하게 된다. 센서 장비가 PE 파일을 탐지하기 위해서는 먼저 PE 파일의 구조를 자동적으로 분석할 수 있어야 한다. 다음 Fig. 5는 PE 파일 탐지 알고리즘에 대한 흐름도이다.

그림에서 보는 바와 같이 처음 패킷이 유입되면 패킷의 DOS 헤더에서 "MZ" 시그니처가 있는지 확인을 하게 된다. 이와 같은 시그니처가 있으면 DOS 헤더 값을 복사하여 PE 코드의 실제 실행 주소를 저장하게 되는데 이 실행 주소를 참조하여 NT 헤더 영역으로 이동할 수 있다.

그리고 NT 헤더를 검사하여 헤더에 "PE" 시그니처가 있으면 지금 캡쳐된 패킷은 PE 파일을 구성하는 첫 패킷임을 알려주고 PE 파일 탐지 기능 실행은 마치게 된다. PE 파일이 탐지되면 연속적으로 유입되는 패킷이 PE 파일의 나머지 부분임을 인지하여 센서 장비에 저장하게 된다. 이와 같은 PE 패킷의 탐지 기능은 센서 장비 및 인터페이스 장치의 성능에 따라 달라지겠지만 실시간 패킷 처리를 위해 고속이면서 간단한 메커니즘으로 수행되어야 한다.

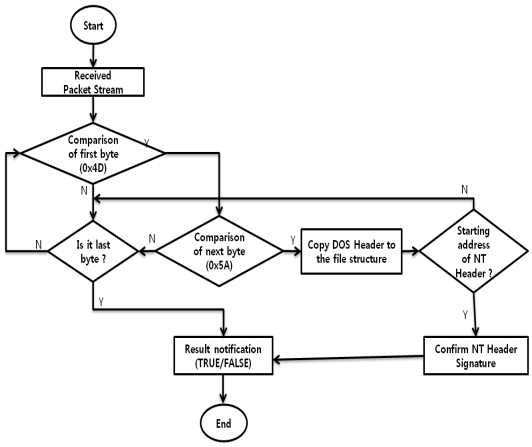

Fig. 5. Flowchart for PE file detection

3.3 PE 파일 추출 알고리듬

네트워크상에서 전달되는 파일 중 PE 파일의 첫

패킷이 탐지되면 이후의 패킷을 모두 모아 재조립 과정을 거쳐 센서 장비에 저장하여야 한다. 이 과정에서 나누어진 패킷이 연속적으로 들어올 수도 있지만 다른 일반 파일의 패킷들이 섞여 유입될 가능성이 높기 때문에 하나의 PE 파일을 연속적으로 추출하여 저장하는 것이 중요하다.

본 논문에서는 이러한 네트워크 환경을 고려하여 전송되는 모든 파일을 검사한 후, PE 파일만 추출하여 저장하는 기능을 가진 탐지 센서를 개발하기 위해 고속 알고리듬을 설계하였다. 특히, 이 파일 추출 알고리듬에서는 패킷 추출 과정을 고속화하기 위해 유입된 패킷을 응용 계층에서 처리하지 아니하고 TCP 계층에서 처리하도록 설계하였다.

전체적인 PE 파일 추출 알고리듬의 흐름을 나타낸 것이 Fig. 6이다. PE 파일 추출 알고리듬에서는 패킷의 효율적 처리를 위해 2개의 변수를 사용하였다. 그림에서 사용된 Flag는 패킷이 PE 파일의 마지막인지를 확인하는 변수이며, Dummy는 처음 PE 파일이 유입된 이후 이 파일과 관련 없는 패킷이 들어왔을 경우 이를 처리하는 변수이다. 즉, PE 파일의 저장을 완료하지 못하는 무한 대기 상태가 될 때, PE 파일과 관련없는 일정한 갯수 패킷이 유입될 때까지 기다린 후 이 상태를 강제로 벗어나도록 알고리듬을 초기화하기 위한 변수이다.

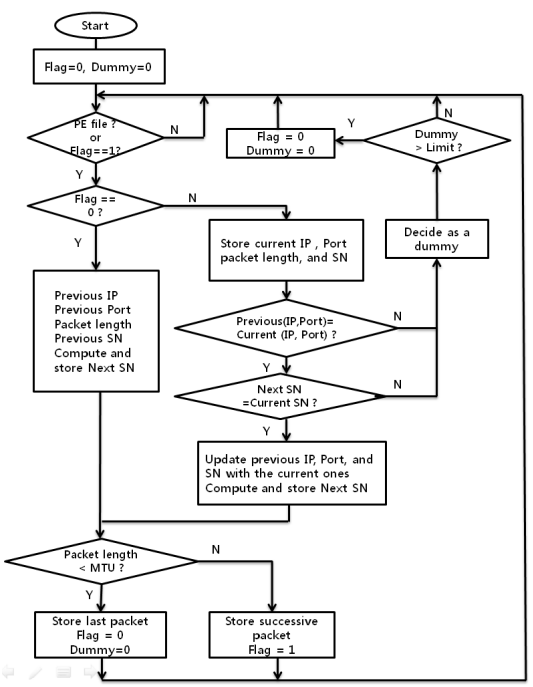

Fig. 6. Flowchart for PE file extraction

들어온 패킷이 PE 파일의 처음 패킷이면 우선 출발지와 목적지의 IP, 포트 번호 그리고 패킷 길이와 순서 번호(Sequence Number, SN)저장한다. 이 정보들은 다음에 유입되는 패킷이 현재 패킷의 다음 이어지는 패킷인지를 결정하는 중요한 정보가 된다. 또한, 현재 순서 번호와 패킷 길이를 이용해서 다음에 유입이 예상되는 패킷의 순서 번호를 미리 저장해 두는 것이 중요하다. 패킷의 다음 순서 번호는 이전 순서 번호와 패킷 길이 정보를 이용하여 아래와 같이 계산한다.

Next SN= Previous SN+패킷길이 - 40

즉, 다음 패킷의 순서 번호를 계산할 때에는 전체 패킷 길이에서 TCP 헤더 길이(20바이트)와 IP 헤더 길이(20바이트)를 빼 줌으로서 정확한 순서 번호를 산출할 수 있다. 그리고 처음 패킷이 들어오면 패킷 길이를 MTU 값과 비교하여 PE 파일의 최종 패킷인지 확인하고 최종 패킷이 아니면 패킷을 저장하고 Flag 값을 1로 셋팅함으로써 다음 패킷을 기다리게 한다.

PE 파일의 두번째 패킷이 들어오면 현재 패킷의 IP, 포트 번호, 순서 번호 등을 저장한 후 이전 패킷의 연속인지를 확인한다. 두번째 패킷이 이전 패킷의 연속이라고 판단되면 현재의 패킷 정보를 이전 패킷 정보로 갱신한다. 패킷 정보를 갱신한 다음에는 이전과 동일한 방법으로 파일의 끝 부분인지 확인한 후 패킷 단위로 이전 파일에 붙여서 저장하게 된다.

만약, 두번째 들어온 패킷이 이전 패킷의 이어지는 부분이 아니라고 판단되면 PE 파일과 관련없는 패킷으로 판단하고 Dummy 카운터 값을 증가시키게 된다. 또, PE 파일이 완전히 저장되지 않은 상태에서 일정한 수의 쓸모없는 패킷이 유입되면 추출 알고리듬을 초기화함으로써 무한 루프 상태로 빠지는 것을 방지하도록 설계하였다.

이와 같이 PE 파일을 탐지하고 저장하는 과정은 대량의 패킷이 연속적으로 유입되는 환경을 가정한다면 고속 처리가 필수적이다. 따라서 본 논문의 센서 장비는 PE 파일 탐지 및 추출 시 응용 계층까지 파일을 검색하지 아니하고 전송 계층에서 파일을 추출하도록 설계하였다.

탐지 센서에서는 추출된 PE 파일을 저장할 뿐만 아니라 이후에 파일이 악성 코드로 판명될 때에 대비하여 파일의 유포지를 역으로 추적하기 위한 정보를 함께 저장하게 된다. 따라서 저장되는 PE 파일에 추

가적으로 송신자 및 수신자의 IP 주소, 포트 번호, 유입 시간 등 악성 코드 추적 정보를 같이 저장하게 된다.

한편, 공격자는 악의적인 PE 파일을 유포시키면서 대량의 동일한 파일을 네트워크에 전파시켜 탐지 센서 기능을 마비시킬 수도 있다. 즉, 대용량의 악성 PE 파일을 대량으로 받은 탐지 센서는 이 파일을 모두 저장하느라 원활한 서비스를 제공하지 못하는 경우가 발생할 수 있다. 따라서 탐지 센서는 유입된 PE 파일이 이전에 유입된 PE 파일과 동일한지를 검사하여 동일한 경우에는 파일 자체를 저장하는 것은 생략하고 악성 코드 추적 정보만 추가적으로 저장할 수 있도록 설계하였다. 이를 위해 각 PE 파일은 저장하기 전 해쉬 값을 계산하여 함께 저장하게 한다. 이 해쉬 값은 이후에 유입되는 PE 파일이 동일한지 여부를 확인할 수 있는 인덱스 값으로 사용될 수도 있으며 저장된 PE 파일의 무결성을 검사할 수 있는 증거 값으로 활용 가능하다.

IV. Snort에서 전처리기 구현 및 실험 결과

본 논문에서는 공개 침입 탐지 시스템인 Snort의 전처리기단에 PE 파일 탐지 및 추출 기능을 구현하였다. PE파일 탐지 방법은 정형화된 PE 구조의 특징을 이용하여 Snort 전처리기단에서 별도의 플러그인 형태로 개발하였다. 또한, 개발된 Snort 프로그램을 파일 탐지용 센서 장비에 장착하여 실제 네트워크 환경하에서 정상적으로 운영됨을 확인하였다.

4.1 Snort 전처리기 개발

공개 침입 탐지 소프트웨어인 Snort에 플러그인 형태로 PE 파일 탐지 및 추출 프로그램을 구현하였다. 이를 위해 전처리기단에서 구현한 주요 함수의 기능은 Table 1과 같다.

먼저 GetPEPreHeader( ) 함수는 모든 유입된 패킷에 대해 DOS Header의 시그니처를 확인한 후 그 위치를 전달하는 함수이다. 그리고 PE_check( ) 함수는 GetPEPreHeader를 통해 확인된 “MZ” 시그니처를 가지는 실행 파일 중에서 PE 포맷을 가지고 있는지 확인한 후 True 혹은 False를 반환해 주는 함수이다. 즉, NT Header 부분으로 이동하여 파일 내 “PE” 시그니처를 탐색함으로써 PE 파일 여부를 결정하는 기능을 수행한다.

Table 1. Functions implemented in preprocessor of Snort

Functions	Main role
GetPEPreHeader	Check “MZ” signature Check starting address of PE code
PE_check	Check “PE” signature Initialization file storing process
PE_Process	Store PE file Store information related with malicious code Compute and store a hash value of PE file

마지막으로 PE_Process( ) 함수는 PE 파일로 확인이 된 패킷을 따라 연속적으로 유입되는 패킷을 수집하며 이를 재조립하고 하나의 PE 파일을 완성하는 함수이다. 즉, PE_Process( ) 함수는 Fig. 6

```
...(synncopation)...
if(FLAG == 0){
chk = PE_check((const unsigned char *)p->payload, 0,
                    p->payload_size);
}
if(chk==1 || FLAG == 1){
    if(FLAG== 0){
      printf("%s\n", PE_MATCH_STR);
      _dpd.alertAdd(GENERATOR_SPP_PE,
                    PE_FILE_MATCH, 1, 0, 3,
                    PE_MATCH_STR, 0);
      previnfo.src_ip = (UINT) GET_SRC_ADDR(p);
      previnfo.dst_ip = (UINT) GET_DST_ADDR(p);
     previnfo.src_port=
           (UINT)ntohs(p->tcp_header->source_port);
     previnfo.dst_port =
           (UINT) ntohs(p->tcp_header->destination_port);
     previnfo.c_seq_num =
           (UINT) ntohl(p->tcp_header->sequence);
     previnfo.len =
           (UINT) ntohs(p->ip4_header->data_length);
     previnfo.n_seq_num =
          (UINT)(previnfo.c_seq_num + previnfo.len - 40);
..(synncopation)...
     printf("first packet ! \n");
     PE_Save(p->payload, p->payload_size);
     FLAG= 1;
     chk = 0;
 ..(synncopation)...
}else{
..(synncopation)...
    if((previnfo.src_ip==nextinfo.src_ip) &&
      (previnfo.src_port==nextinfo.src_port) &&
      (previnfo.n_seq_num==nextinfo.c_seq_num)) {
  ..(synncopation)...
```

Fig. 7. PE_Process( ) function

의 기능을 모두 구현한 함수로서 최종적인 출력 정보는 PE 파일 원본 및 역추적 관련 정보들이다. 다음 Fig. 7은 PE 파일 추출을 수행하는 PE_Process 함수의 일부분을 나타낸 것이다.

4.2 PE 파일 탐지 및 추출 센서 실험

본 논문에서는 PE 파일을 탐지하기 위해 FTP와 E-mail을 이용하여 파일을 전송하는 과정에서 유입되는 PE 파일을 탐지하고 추출하는 센서 장치를 개발하였다. 앞서 3장에서 설명한 네트워크를 외부망과 내부망으로 나누어 구성하고 중간에 TAP 장비를 설치하여 전송되는 패킷을 태핑하였다. 또한, 패킷을 탐지하고 추출하기 위해서 TAP 장비와 탐지 센서를 연결하여 실험 네트워크를 구축하였다. 다음 Fig. 8은 실험을 위해 구성한 전체적인 네트워크 구성 화면이다.

그림의 왼쪽은 내부망 설치된 클라이언트 PC를 나타낸 것이며 가운데는 네트워크 장비 및 센서 장치를 설치해 둔 것이다. 외부망은 라우터를 통해 연결되며 라우터와 스위치 사이에 (주)RPA 네트워크의 ST-100이라는 TAP 장비를 설치하여 양방향으로 전송되는 패킷을 모두 태핑하였다. ST-100은 100Mbps 속도로 설치된 지점을 통과하는 네트워크 상의 모든 데이터를 네트워크 흐름에 전혀 영향을 미치지 않으면서 모니터링 할 수 있는 장치이다.

그림의 오른쪽이 탐지 센서 장비 및 모니터 화면이다. 탐지 센서용 하드웨어 장비는 (주)엔큐리티(ncurity)의 NCF-10000 제품을 사용하였다. 이 장비는 10Gbps급 방화벽용 데이터 처리 장비로서 자체 OS로 운영되고 있는데, 논문에서는 이 탐지 센서 장비에 기존의 Snort 시스템을 개선하여 PE 파일 탐지 및 추출 기능이 추가된 전처리기를 개발하여 탑재하였다. 의심 파일 센서 장치를 통해 추출된 파일은 특정한 폴더를 지정하여 PE 파일과 추적용 로그 정보 그리고 해쉬 결과 값을 함께 저장한 후 악성 코드 분석 시스템으로 전달되도록 구현하였다.

Fig. 8. Experimental sensor equipment for PE file detection and extraction

실험을 위해서 먼저 외부망에 FTP 및 E-mail 서버를 설정하였고 내부망의 클라이언트 PC를 이용하여 서버에 접속한 후 파일을 주고 받는 과정에서 탐지 센서가 PE 파일을 탐지할 수 있는지 확인하였다. 실험 결과, FTP나 E-mail을 통해 송·수신 되는 파일 중 PE 파일만 정확히 추출하여 센서 장비에 저장할 수 있음을 확인하였다. Fig. 9는 센서 장비가 PE 파일을 정확히 탐지하고 이를 저장하는 화면을 모니터에서 캡쳐한 화면이다. 예제 그림에서 보듯이 하나의 PE 파일은 모두 9개의 패킷으로 전송되고 있으며 원본과 동일하게 추출됨을 확인하였고 파일 관련 정보와 해쉬 값을 동시에 저장함을 알 수 있었다.

물론, FTP 파일이나 E-mail에 첨부되어 유입되는 파일이 아니라 다른 인터넷 응용 서비스를 통하여서 패킷이 혼재되어 들어온다 하더라도 PE 파일의 형태이면 센서 장비에서 출발지 IP, 포트 번호, 패킷 길이 그리고 순서 번호 등을 검사하여 파일로 저장하게 된다. 그러나 공격자가 하나의 악성 파일을 몇 개로 분할한 후 여러 대의 공격용 시스템에서 하나의 공격 목표로 전송하거나 충분한 시간차를 두고 나누어 전송한 후 재조립한다면 이 공격은 센서 장치에서는 탐지가 어렵다. 이 경우에는 의심 파일로 분류한 후 악성 코드 분석 시스템에서 면밀한 분석이

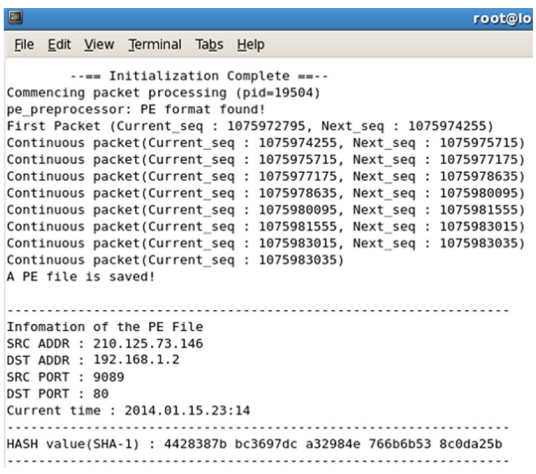

Fig. 9. PE File sensor monitor

이루어져야 한다.

기존의 PE파일 탐지 도구로는 PEiD나 MRC(Mandiant Red Curtain) 그리고 REMINDer[6, 9, 15] 등이 있었다. 특히, 문헌[6]에서는 PE 파일의 진입점 섹션의 엔트로피 통계와 속성 추출을 통하여 패킹된 PE 파일을 탐지하는 도구인 REMINDer를 개발하였는데 컴퓨터 성능에 따라 차이는 있겠지만 약 9G바이트의 데이터를 검색하는데 약 5분 정도(약 240Mbps)가 소요되는 처리 성능을 보이고 있다. 그러나 이러한 탐지 도구들은 탐지 능력, 악성 코드의 판별 여부, 오탐율 적용 범위, 시스템 성능, 네트워크 실험 환경 등이 서로 다르며, 특히 본 논문에 사용하는 네트워크 패킷 탐지 센서 장비와 다른 형태이므로 상대적인 성능 비교는 현실적으로 어려운 면이 있었다.

논문에서는 탭 장치에 의해 통신 속도가 100Mbps로 제한되는 점을 고려하여 여러 개의 독립된 PE 파일이 유입되는 경우를 가정하여 단위 시간당 PE 파일 추출 성능을 실험하였다. 비록 실험에 사용된 네트워크 규모가 제한적이었고 수십 대의 컴퓨터를 이용하여 동시에 PE 파일을 생성할 수 있는 환경은 아니었지만 센서 장치는 연속적으로 유입되는 패킷들도 실시간으로 탐지하고 저장할 수 있음을 확인하였다.

IV. 결 론

본 논문에서는 스마트 팩토리와 같은 제조 산업 현장에서 사용할 수 있는 네트워크를 보호할 수 있도록 악성 코드 유포 공격에 대응하는 PE 파일 탐지 및 추출 센싱 시스템을 구현하였다. 이와 같은 기능을 하는 센서 장치는 실시간으로 유입되는 분할된 패킷을 탐지하여 하나의 PE 파일 형태로 재조립하는 것이 주요 기능이다. 따라서 고속의 패킷 처리 능력 및 대용량 저장 공간을 필요로 한다.

본 논문에서는 공개 침입 탐지용 툴인 Snort의 전처리기단에 PE 파일을 탐지하고 추출할 수 있는 기능을 플러그인 형태로 개발하고 이를 실제 센서 장비에 탑재하여 구현하였다. 이 센서 장비는 유·무선 네트워크상에서 TAP 장비를 이용하여 전송되는 모든 패킷을 모니터링할 수 있으며 PE 파일만을 추출하여 악성 코드 분석 시스템으로 전달함으로써 APT와 같은 악성 코드 유입 공격에 대해 신속하게 대응할 수 있다.

References

[1] S. D. Lee, "Global trend for smart factory and countermeasure strategy for standardization in Korea," KSA Policy Study 012 Issue Paper 2015-3, July 2015.

[2] J. M. Park, "Technology and issue on embodiment of smart factory in small-medium manufacturing business," The Journal of Korean Institute of Communications and Information Science, 40(12), pp. 2491-2502, Dec. 2015.

[3] K. C. Yang, S. Y. Lee, W. H. Park, K. C. Park, and J. I. Lim, "A study on analysis and detection method for protecting malware spreading via E-mail," Journal of Information and Security, 19(1), Feb. 2009.

[4] M. K. Daly, "Advanced Persistent Threat (or Informationized Force Operations)," 23rd Large Installation System Administration Conference(LISA'09), 2009. Available at http://www.usenix.org /event/lisa09/tech/slides/daly.pdf

[5] M. Cova, C. Kruegel, and G. Vigna, "Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code," Proceedings of the 19th international conference on World Wide Web(WWW'10), pp. 281-290, 2010.

[6] S. W. Han and S. J. Lee, "Packed PE file detection for malware forensics," The KIPS Transactions : Part C, 16(5), pp. 555-562, 2009.

[7] S. W. Kim, E. H. Kim, and J. Y. Choi, "A study of detection method about Android app file on the network," Journal of Security Engineering, 9(1), Feb. 2012.

[8] Y. S. Choi, I. K. Kim, J. T. Oh, and J. C. Ryou, "PE File Header Analysis-based Packed PE File Detection Technique,"

International Symposium on Computer Science and its Applications(CSA'08), pp. 28-31, Oct. 2008.
[9] Y. S. Choi, I. K. Kim, J. T. Oh, and J. C. Ryou, "Encoded executable file detection technique via executable file header analysis," International Journal of Hybrid Information Technology, vol. 2, no 2, pp. 25-36, April 2009.
[10] D. Devi and S. Nandi, "PE File Features in Detection of Packed Executables," International Journal of Computer Theory and Engineering, vol 4, no. 3, pp. 476-478, 2012.
[11] M. Z. Shafiq, S. M. Tabish, F. Mirza, and M. Farooq, "PE-Miner: Mining Structural Information to Detect Malicious Executables in Realtime," 12th International Symposium on Recent Advances in Intrusion Detection (RAID'09), LNCS 5758, pp. 121-141, 2009.
[12] J. Beale, J. C. Foster, J. Faircloth, and B. Caswell, Snort 2.0 Intrusion Detection, 2nd Ed., Syngress, Feb. 2003.
[13] R. U. Rehman, Intrusion Detection with SNORT: Advanced IDS Techniques Using SNORT, Apache, MySQL, PHP, and ACID, Prentice Hall, May 2003.
[14] A. R. Baker and J. Esler, Snort IDS and IPS Toolkit (Jay Beale's Open Source Security), 4th Ed., Syngress, Oct. 2010.
[15] R. McRee, "Mandiant Red Crutain: Malware identification for incident responders," Information Systems Security Association(ISSA) Journal, Dec., 2007. Available at https://holisticinfosec.org/toolsmith/pdf/december2007.pdf

〈저자소개〉

하 재 철 (Jaecheol Ha) 종신회원
1989년 2월: 경북대학교 전자공학과 졸업
1993년 8월: 경북대학교 전자공학과 석사
1998년 2월: 경북대학교 전자공학과 박사
1998년 3월~2007년 2월: 나사렛대학교 정보통신학과 부교수
2007년 3월~현재: 호서대학교 컴퓨터정보공학부 정보보호전공 교수
2013년 1월~현재: 한국정보보호학회 부회장
2009년 1월~현재: 한국산학기술학회 이사
〈관심분야〉 암호 알고리듬, 네트워크 보안, 부채널 공격