Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.25, NO.5, Oct. 2015 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2015.25.5.1027

패스트 데이터 기반 실시간 비정상 행위 탐지 시스템*

이 명 철,† 문 대 성,‡ 김 익 균
한국전자통신연구원

Real-time Abnormal Behavior Detection System based on Fast Data*

Myungcheol Lee,† Daesung Moon,‡ Ikkyun Kim
Electronics and Telecommunications Research Institute

요 약

최근, Verizon(2010), 농협(2011), SK컴즈(2011), 그리고 3.20 사이버 테러(2013)와 같이 소중한 정보가 누출되고 자산에 피해가 발생한 후에야 보안 공격을 인지하는 APT (Advanced Persistent Threat) 공격 사례가 증가하고 있다. 이러한 APT 공격을 해결하고자 이상 행위 탐지 기술 관련 연구가 일부 진행되고 있으나, 대부분 알려진 악성 코드의 시그너쳐 기반으로 명백한 이상 행위를 탐지하는데 초점을 맞추고 있어서, 장기간 잠복하며 제로데이 취약점을 이용하고, 새로운 또는 변형된 악성 코드를 일관되게 사용하는 APT 공격에는 취약하여, 미탐율이 굉장히 높은 문제들을 겪고 있다. APT 공격을 탐지하기 위해서는 다양한 소스로부터 *장기간에 걸쳐 대규모 데이터를 수집, 처리 및 분석하는 기술*, 데이터를 수집 즉시 *실시간 분석하는 기술*, 그리고 개별 공격들 간의 *상관(correlation) 분석 기술*이 동시에 요구되나, 기존 보안 시스템들은 이러한 복잡한 분석 능력이나 컴퓨팅 파워, 신속성 등이 부족하다. 본 논문에서는 기존 시스템들의 실시간 처리 및 분석 한계를 극복하기 위해, *패스트 데이터 기반 실시간 비정상 행위 탐지 시스템*을 제안한다.

ABSTRACT

Recently, there are rapidly increasing cases of APT (Advanced Persistent Threat) attacks such as Verizon(2010), Nonghyup(2011), SK Communications(2011), and 3.20 Cyber Terror(2013), which cause leak of confidential information and tremendous damage to valuable assets without being noticed. Several anomaly detection technologies were studied to defend the APT attacks, mostly focusing on detection of obvious anomalies based on known malicious codes' signature. However, they are limited in detecting APT attacks and suffering from high false-negative detection accuracy because APT attacks consistently use zero-day vulnerabilities and have long latent period. Detecting APT attacks requires *long-term analysis* of data from a diverse set of sources collected over the long time, *real-time analysis* of the ingested data, and *correlation analysis* of individual attacks. However, traditional security systems lack sophisticated analytic capabilities, compute power, and agility. In this paper, we propose *a Fast Data based real-time abnormal behavior detection system* to overcome the traditional systems' real-time processing and analysis limitation.

Keywords: APT, Fast Data, Abnormal Behavior Detection, Host Process Behavior, Apache Storm

접수일(2015년 6월 2일), 수정일(1차: 2015년 8월 22일, 2차: 2015년 9월 24일), 게재확정일(2015년 9월 24일)
* 본 논문은 2015년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임. (B0101-15-1293, 다중소스 데이터의 Long-term History 분석 기반 사이버 표적공격 인지 및 추적 기술 개발)
† 주저자, mclee@etri.re.kr
‡ 교신저자, daesung@etri.re.kr(Corresponding author)

I. 서 론

최근, Verizon(2010), Stuxnet(2010), 농협(2011), SK컴즈(2011), 그리고 3.20 사이버 테러(2013.03.20)와 같이 이미 소중한 정보가 누출되고 자산에 피해가 발생한 후에야 보안 공격을 인지하는 APT (Advanced Persistent Threat) 공격 사례가 증가하고 있다. 2010년의 Verizon 데이터 유출 보고서[1]에 따르면, 86%의 유출 케이스는 데이터 유출의 흔적이 로그에 기록되어 있음에도 불구하고, 유출 탐지 메커니즘이 제대로 동작하지 못 해서 보안 경고를 발생시키지 못한 것으로 보고가 되었다.

APT 공격은 주요 정부 기관 또는 기업 등으로부터 군사 비밀, 민감한 산업 및 고객 정보를 훔치거나 산업 제어 시스템을 마비시키기 위한 목적으로 실시되며, 결과적으로 막대한 물리적 피해를 유발한다. 일상적인 APT 공격 시나리오에서는 공격자가 사회공학적 방법을 포함하는 다양한 방법을 동원해서 대상 기관에 은밀히 침투하고, 수주~수개월에 걸친 장기간의 준비 과정을 거쳐서 최종 공격에 이르게 된다[2,13].

웜, 바이러스, 트로이 목마 등의 malware를 불특정 다수에게 뿌리는 기존 공격과 달리, APT 공격은 일시적 공격이 아니라 장기간에 걸쳐서 이루어지고, 다양한 악성 코드 및 공격 루트가 사용되기 때문에, 이를 사전에 탐지하고 대응하는 것은 어려운 일이며, 대다수 기관에서는 변변한 침입 탐지 시스템을 갖추지 못해서 보안 분석가의 직관 및 수작업에 의존해서 탐지를 하고자 하나 역부족인 실정이다.

이상 행위 탐지를 통해 APT 공격을 해결하고자 하는 연구가 일부 진행 되고 있으나, 대부분 알려진 악성 코드의 시그너쳐 기반으로 명백한 이상 행위를 탐지하는데 초점을 맞추고 있어서, 장기간 잠복하며 제로데이[26] 취약점을 이용하고, 새로운 또는 변형된 악성 코드를 일관되게 사용하는 APT 공격에는 취약하여, 미탐율이 굉장히 높은 문제들을 겪고 있다.

APT 공격을 탐지하기 위해서는 네트워크, 호스트/서버, 보안 장비 등 다양한 소스로부터 장기간에 걸쳐 대규모 데이터를 수집, 처리 및 분석하는 *대규모 분석 기술*[3]과, 데이터를 수집 즉시 실시간 분석하는 *실시간 분석 기술*, 그리고 은밀한 공격들이 최종 공격의 성공까지 계속 시도되기 때문에, 개별 공격의 의미를 이해하기 위한 공격과 공격 간의 *상관(correlation) 분석 기술*이 동시에 요구되나, 막대한 데이터 규모 때문에 전통적인 침입 탐지 기술로는 처리, 분석은 물론 저장도 못 하는 실정이다.

최근 Hadoop[4]을 포함한 빅데이터 처리 기술의 발전과 함께, 대규모 고속 입력 데이터를 효과적으로 처리할 수 있는 컴퓨팅 능력을 보유하게 되어 과거에는 불가능했던 규모의 데이터를 저장, 처리, 분석할 수 있게 되어, 다양한 응용 도메인에서 빅데이터 기술의 급격한 도입이 이루어지고 있다[9].

최근에는, 사람들이 데이터의 규모를 넘어서 실시간성에 주목하기 시작하면서, Apache Storm[5]과 같이 빅데이터의 3V(volume, velocity, variety) 속성 중에서 velocity(데이터 유입 및 처리 속도)에 중점을 둔 패스트 데이터(Fast Data)[27] 기술들에 많은 관심이 집중되고 있다.

보안 분야에서도 APT 공격을 조기에 탐지 및 대응하여 공격자가 초기 침입 이후에 시스템 상에서 들키지 않고 활동할 수 있는 시간을 단축하고, 결과적으로 물리적 피해가 발생하기 전에 해결하기 위해서는, 패스트 데이터 기반의 실시간 보안 분석 기술에 관심을 가질 때다.

본 논문에서는 전술한 APT 공격과 관련한 기존 시스템들의 실시간 처리 및 분석 한계를 극복하기 위해서, 정상/비정상 행위 탐지를 위한 프로세스 행위 기반의 특성 인자 모델을 정의하고, 대표적인 패스트 데이터 시스템인 Apache Storm을 사용하여 기관 내에서 발생하는 알려지지 않은 비정상 행위를 실시간 탐지하는 패스트 데이터 기반 비정상 행위 탐지 시스템을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서 기존 빅데이터 기반 지능형 보안 기술을 살펴보고, 3장에서는 호스트 기반 비정상 행위 기술 방법에 대해서 설명하고, 4장에서는 제안하는 패스트 데이터 기반의 실시간 비정상 행위 탐지 방법 및 시스템을 설명한다. 5장에서는 제안 시스템의 성능 평가 결과를 기술하며, 마지막으로 6장에서 결론 및 향후 연구 방향을 제시한다.

II. 관련 연구

기존 지능형 보안을 위한 SIEM(Security Information and Event Management) 제품들이 보안 이슈에 대해 기존의 방어적 탐지 및 대응 전략에서 탈피하여, 적극적으로 대응함으로서 사전 예방을 달성하기 위한 목적으로 빅데이터 분석 기술과

점차 통합되어 가고 있는 추세이다.

2.1 빅데이터 (Big Data)

빅데이터(Big Data)라는 용어는 기존 데이터 저장 및 처리 기술의 능력을 넘어서는 대규모 데이터를 저장, 처리, 분석하는 온갖 기술[16]을 통틀어 얘기한다. 이미 많은 응용 도메인에서 빅데이터 기술을 활용하여 데이터의 규모(volume), 데이터 생성 및 처리 속도(velocity), 데이터 다양성(variety) 측면에서 다양한 문제를 해결하고자 하고 있다.

얼마 전까지 모든 기업들이 빅데이터의 규모에 대해서만 주목을 하였으나, 최근에는 고속으로 입력되는 데이터로부터 보다 빨리 문제에 대응하거나 또는 가능하면 사고를 사전에 예방하고자 하는 실시간성 욕구가 증대되면서 속도(velocity)에 초점을 맞춘 패스트 데이터(Fast Data) 라는 용어가 널리 쓰이고 있다.

빅데이터 처리 기술은 크게 배치 처리와 스트림 처리 기술로 구분할 수 있다. 가장 대표적인 시스템으로는 각각 Hadoop 과 Storm을 들 수 있다. Apache Hadoop은 가장 유명한 오픈 소스 배치 처리 시스템이며, 최근 Spark, Flink 등의 새로운 시스템들이 Hadoop 대비 향상된 처리 성능 덕분에 주목을 받고 있다. 스트림 처리 시스템으로는 Apache Storm이 가장 많이 알려진 시스템이며, Spark Streaming, Flink Streaming 등이 역시 최근에 Storm 대비 좋은 성능으로 관심을 받고 있다.

2004년 Apache Hadoop 시스템이 출현한 이후, 모든 사람들의 관심은 배치 처리 뿐이었으나, 최근 데이터가 생성되는 즉시 처리, 분석하여 즉각적인 의사 결정을 원하는 많은 요구 사항이 증가하고 있다. 이에 따라, 배치와 스트림 처리를 결합하여 빅데이터 및 패스트 데이터를 모두 처리할 수 있는 소프트웨어 구조로서 람다 구조(Lambda Architecture)[17]가 주목을 받고 있다. 람다 구조는 과거에 수집한 데이터 뿐만 아니라, 현재 입력되는 데이터도 같이 분석하여 종합적인 가치를 창출하고자 하는 소프트웨어 패러다임이다.

최근 대표적인 빅데이터 관리 시스템인 VoltDB 등이 람다 구조와의 호환성을 주장하면서, 람다 구조의 위상이 크게 높아지고 있다.

람다 구조에 빠른 인메모리 데이터베이스와 대규모 SQL-on-Hadoop 스토어를 더한 시스템이 빅데이터 및 패스트 데이터를 빨리 처리하여 종합적이면서도 데이터가 입력되자마자 즉각적인 의사 결정을 수행할 수 있는 강력한 빅데이터 분석 프레임워크로서 다양한 분야에서 활용될 수 있을 것으로 예상된다.

2.2 Big Data Security Intelligence

빅데이터 기술이 활발히 접목되고 있는 다른 응용 도메인에서와 마찬가지로 보안 분야에서도 빅데이터 기술을 활용하여 다양한 대규모 보안 데이터를 처리하여, 이를 통해 보안 침해를 기존 방법보다 효율적으로 (또는 기존에 못하던 수준까지) 탐지, 대응, 예방할 수 있다.

특히, 지능형 보안 분야에서 빠르게 생성되는 대규모 보안 로그를 분석할 수 있는 수단으로서 빅데이터 처리/분석 기술이 주목을 받으면서, 기존 규칙(rule) 기반의 알려진 위험을 탐지하는 방법들이 빅데이터와의 접목을 통해 APT 공격과 같이 미지의 또는 아직 실현되지 않은 위험들을 탐지하고 예방하는 효과적인 수단으로 교체되고 있다.

지능형 APT 공격에 대응하기 위해서는, 호스트, 네트워크, 보안 장비 등으로부터 생성되는 다양한 이벤트 정보에 대한 종합적인 분석이 필요하다. 특히 APT 공격자들은 일부 악성 코드를 호스트 PC에 초기에 잠입시킨 후, 나중에 그들의 최종 목적을 달성한다. 만일 이러한 호스트 PC 상의 악성 코드들을 일부라도 미리 발견할 수 있다면, 실제 피해가 발생하기 전에 APT 공격을 발견하고 조치를 취할 수 있을 것이다.

SIEM 선두 기업들이 지능형 보안을 적용하여 새로운 공격에 대응하기 위하여 빅데이터 분석 기술을 도입하여 현재 활용하고 있으며, 다양한 보안 장비, 서버, 네트워크 장비 등에서 발생하는 통계 정보 및 보안 이벤트 정보를 통합하고 그러한 정보 간의 상호 관계를 분석함으로서 보안 문맥(security context)을 인식하는 기능을 제공한다[5].

한편 HP, IBM, McAfee 등은 각각 ArcSight, Q1 Labs, NitroSecurity 등을 인수하여 빅데이터 기술을 적용한 토탈 지능형 보안 솔루션을 제공하고 있다[19,20,21]. 특히, Splunk는 자사의 강력한 빅데이터 분석 기술을 활용해서 “Splunk for Security”[6]라는 제품으로 지능형 보안 솔루션 시

장에서도 점유율을 확대하고 있다.

대부분의 지능형 보안 솔루션들이 주장하는 침입 탐지 성능을 EPS (Events Per Second, 초당 처리 호스트/네트워크/레거시 데이터 처리 이벤트 개수)로 나타내면, 대략 5,000 EPS ~ 50,000 EPS 정도[28]이며, 대표적인 지능형 보안 솔루션들의 실제 성능은 다음과 같이, IBM QRadar[29]의 경우 40대의 어플라이언스로 최대 100,000 EPS, 즉 노드당 2,500 EPS의 성능을 보이고, ArcSight ESM 7425[30]의 경우, 최대 12,500 EPS의 성능을 보이며, Splunk[31]는 full density 일 때 대략, 22,000 EPS의 성능을 보이는 한계를 갖고 있다. 이러한 실시간 처리 성능 한계를 극복하기 위해서는 빅데이터 기술에 기반한 대규모 수집 데이터에 대한 실시간 처리 및 분석의 확장성을 확보해야 할 것이다.

기타 다양한 빅데이터를 활용한 지능형 보안 관련 연구 프로젝트들[8]이 진행 중이다. 주목할 만한 프로젝트로는 2010년 이후 미국 DARPA에서 수행한 CINDER (Cyber-INsiDER) 와 Cyber Genome 프로그램[7]이 있다.

RSA Lab에서 개발한 APT 탐지 시스템인 Beehive [24]는 일상적인 패턴에서 벗어나는 사용자 행위를 발견하기 위한 행위 프로파일링에 기반 해서 APT 탐지를 달성하는 시스템이다. 각 이상 행위별로 이상 유무를 탐지하는 “anomaly sensor”를 다수 배치하여, 각 센서가 이상 유무를 탐지하고 이벤트를 발생하도록 한다. 관리자가 여러 센서를 다양한 패턴으로 조합하여 보다 복합적인 침입 탐지를 수행할 수 있도록 한다. 또한 서로 관련 없어 보이는 다양한 APT 공격 단계에서의 행위 간의 상관 분석을 통한 탐지 기능을 제공한다. 그러나, 빅데이터 기술에 기반한 대규모 수집 데이터에 대한 실시간 처리 및 분석의 확장성 측면에서는 한계가 존재한다.

빅데이터 기술인 Hadoop MapReduce를 활용해서 IP, TCP, HTTP, Netflow 데이터 등 다양한 대규모 보안 로그를 수집하고 배치 분석하고자 하는 연구들[22,23]이 다수 수행되어 기존 시스템의 데이터 규모 한계를 넘어 서는 효과를 보고 있으나, 아직 실시간 분석을 통한 실시간 APT 공격 탐지 기술에 대한 연구는 미진하다.

2.3 침입 탐지 방법

침입 탐지 방법은 분석 방법을 기준으로 오용 탐지(misuse detection)와 비정상 행위 탐지(anomaly detection)로 구분한다. 오용 탐지는 이미 알려진 악성 코드 및 악성 행위에 대해 행위 규칙(behavior rule), 시그너쳐 등을 생성하고 탐지 대상에 대해 패턴 매칭을 수행하여 탐지하거나, 악성 코드에서 발생하는 네트워크 및 호스트 정보를 수집하여 통계적인 방법을 통해 특정 악성 코드의 규칙을 생성하고 규칙을 만족할 경우 악성 코드로 분류하는 방법이다. 반면, 비정상 행위 탐지는 악성코드가 아닌 정상 행위에서 발생되는 네트워크 및 호스트 데이터를 수집하여 정상 행위를 나타내는 모델을 생성하고, 정상 행위 모델에 포함되지 않는 행위를 악성 행위로 탐지하는 방법이다. 정상 행위 모델 생성에는 데이터 마이닝, 통계적 방법 등의 방법들이 사용된다[14,15].

오용 탐지는 악성 행위를 기준으로 하기 때문에, 기존 악성 코드에 대해 높은 탐지율과 빠른 탐지 속도를 보장하지만, 제로데이 악성 코드와 같이 알려지지 않은 악성 행위에 대한 대응이 불가능하다. 비정상 행위 탐지는 정상 행위를 기준으로 탐지하기 때문에, 정상 행위에서 벗어나는 제로데이 악성 코드에 대해서도 탐지가 가능한 반면, 높은 오탐율과 오용 탐지에 비해 계산량이 많은 단점이 있다.

APT 공격과 같이 최근에 발생된 사이버 공격들은 제로데이 악성 코드를 사용하는 등 점차 지능화된 공격 기법을 사용하기 때문에 오용 탐지만으로는 적절한 대응이 어려우며, 비정상 행위 탐지를 기반으로 하면서 비정상 행위 탐지 기술이 갖는 단점인 오탐율을 낮추고, 높은 계산량도 극복할 수 있어야 한다.

III. 호스트 기반 비정상 행위 탐지

본 논문에서는 패스트 데이터 기술을 활용하여 고성능 실시간 APT 공격 탐지를 가능하게 하는 호스트 기반 비정상 행위 탐지 방법을 제안한다. 본 논문의 호스트 기반 비정상 행위 탐지 방법은 미탐율과 오탐율을 낮추고자 수행된 기존 연구[10]의 프로세스 행위 및 특성 인자 정의를 기반으로 실시간 처리 성능을 제공하기 위하여 확장된 탐지 방법이며, 본 논문에서는 이해를 돕기 위해 패스트 데이터 기반으로 구현된 실시간 비정상 행위 탐지 시스템에서의 구

현 내용 위주로 설명한다.

3.1 프로세스 행위

APT 공격자가 내부 시스템에 잠입한 후에 취하는 모든 행동들은 결국 호스트 PC 상에서 프로세스 형태로 실행이 된다. 따라서, 모든 대상 호스트 PC의 프로세스 행위를 모니터링하고, 프로세스가 수행한 일련의 행위들 간의 상호 관계 분석을 통해 이상 행위를 탐지할 수 있다면, 결국에는 은밀히 진행되는 APT 공격을 조기에 탐지하고 물리적인 피해를 예방할 수 있다.

프로세스 행위에 기반 하여 APT 공격을 탐지하기 위해, 프로세스가 수행 중에 갖는 주요 속성들을 추출하여 프로세스 행위 정보로 정의한다. 호스트 PC에서 수집된 모든 정상 및 비정상 프로세스 행위 정보는 Table 1.의 자료 구조 형태로 관리된다.

각 프로세스가 수행하는 개별 행위 이벤트(즉, API 시스템 호출 정보)들은 Table 2.에서 보는 바와 같이 47개의 특성 인자(feature)로 분류한다.

모든 프로세스가 호스트 PC에서 실행되는 동안 특성 인자에 해당하는 이벤트가 발생하면 관련 데이터를 수집하게 되며, 모든 발생 이벤트는 Table 1. 과 같은 구조로 수집된다. 해당 프로세스가 동작하는 호스트 ID, 프로세스를 실행한 사용자 ID, 프로세스 ID, 이벤트 발생 시간, 실행 파일(또는 프로세

Table 1. Data Structure for Process Behavior Information

Process Property	Data Type	Description
Index	int	event occurrence index
HostID	String	host IP address
FileName	String	file(=process) name
PID	long	process ID
PPID	long	parent's PID
UserID	String	user ID
StartTime	String	the time when first event was collected
LastTime	String	the time when latest event was collected
Child LastTime	String	LastTime of children/descendants
Start EventID	int	first event ID (Feature ID)
Last EventID	int	latest event ID (Feature ID)
Type	int	event type: normal(0), attack(1)
Feature Vector	int[94]	94-dimensional feature vector

Table 2. Process Behavior Features

Category	Index	Feature
Process	1	CreateProcess
	2	ExitProcess
	3	TerminateProcess
	4	OpenProcess
	5	SearchProcess
	6	ProcessDepPolicy
	7	InformationProcess
Thread	8	CreateLocalThread
	9	CreateRemoteThread
	10	ExitThread
	11	TerminateThread
	12	OpenThread
	13	SuspendThead
	14	ResumeThead
Memory	15	ReadProcessMemory
	16	WriteProcessMemory
	17	HeapCreate
	18	VirtualAlloc
	19	VirtualProtect
File	20	CreateFile
	21	CopyFile
	22	MoveFile
	23	DeleteFile
	24	OpenFile
	25	ReadFile
	26	WriteFile
	27	SearchFile
	28	FileInformation
Registry	29	CreateRegistry
	30	DeleteRegistry
	31	OpenRegistry
	32	ReadRegistry
	33	WriteRegistry
Network	34	Connect
	35	Listen
	36	Send
	37	Recv
	38	Download
Service	39	CreateService
	40	DeleteService
	41	OpenService
	42	StartService
Misc.	43	CreateMutex
	44	OpenMutex
	45	LoadLibrary
	46	WindowShook
	47	SetSecurity

스) 이름, Table 2.에서 정의된 특성 인자 등의 정보가 함께 수집된다.

프로세스 행위 정보는 프로세스가 생성되어 소멸되기까지 모든 행위 이력을 47 차원의 특성 인자 벡터(feature vector)에 누적 빈도수 형태로 관리한다. 프로세스의 자식/자손 프로세스에서 발생하는 특성 인자 정보도 추가적인 47 차원의 특성 인자 벡터에 포함하여 특정 프로세스에 의해서 발생되는 모든 행위 정보를 모두 94차원의 특성 인자 벡터에 표현할 수 있다.

3.2 프로세스 행위 특성 인자

본 연구에서는 악성 코드의 정적 분석을 통한 시그너쳐 대신 악성 코드가 호스트 PC에서 실행되는 동안 발생하는 동적 프로세스 행위 로그를 이용한 악성 코드 특성 인자 추출 및 탐지 방법을 사용한다. 대상 악성 코드는 윈도우즈 시스템에서 동작하는 프로그램들로 한정하였으며, 윈도우즈 API 호출 이벤트로부터 특성 인자를 추출한다.

본 연구에서는 윈도우즈 API 호출을 특징지을 수 있도록 8 종류의 카테고리(프로세스, 쓰레드, 메모리, 파일, 레지스트리, 네트워크, 서비스, 기타 정보 등)에 총 47개의 특성 인자를 정의하고, 모든 프로세스 행위 이벤트에 대해 47개의 특성 인자 중 하나를 부여한다. 47개의 특성 인자들은 특정 악성 코드의 행위에 의존적인 내용이 아니며, 정상 실행 파일에서도 빈번하게 발생하는 특성 인자들로 구성된다.

각각의 특성 인자는 동일한 행위 이벤트라도 서로 다른 여러 가지 윈도우즈 API가 호출될 때 발생할 수 있기 때문에, 호스트에서 발생하는 모든 특성 인자를 수집하기 위해서는 47개보다 많은 윈도우즈 API를 모니터링 한다. 예를 들어, SearchProcess 특성 인자의 경우, FindWindowA(), FindWindowW(), FindWindowExA(), FindWindowExW() 등 다양한 윈도우즈 API 호출에 의해 해당 이벤트가 발생된다.

IV. 실시간 비정상 행위 탐지 시스템

본 논문에서 제안하는 실시간 비정상 행위 탐지 시스템은 사이버 타겟 공격 대응 시스템인 SINBAPT (Security Intelligence techNology for Blocking APT) 시스템의 일부로 구현되었다. SINBAPT 시스템은 람다 구조에 대규모 SQL-on-Hadoop 스토리지와 인메모리 MySQL 클러스터 데이터베이스를 함께 활용하여 장기적인 분석과 실시간 분석을 모두 지원하고, 이를 통해 보다 진보하면서도 실시간 지능형 보안을 달성하고자 하는 시스템이다[18].

SINBAPT 시스템은 보안 장비, 네트워크 트래픽, 통계 정보, 모든 호스트 발생 행위 정보 등에서 발생하는 보안 이벤트 간의 상관(correlation)을 분석하여 비정상 행위를 탐지하고 결과적으로 사이버 타겟 공격을 탐지하는 기능을 제공한다. 또한, 이러한 분석 결과를 바탕으로 공격의 근원지를 역추적하는 기능을 제공한다.

APT 공격자는 내부 네트워크에 침입하여 비밀 정보 누출, 시스템 장애 발생 등 최종 공격을 감행하기 위하여 여러 가지 준비를 하는 잠복기를 거친다. SINBAPT 시스템은 최종 공격이 실행되기 전에 비정상 행위를 사전에 탐지하여 이러한 사이버 타겟 공격에 대응할 수 있다.

본 논문은 SINBAPT 시스템의 서브시스템으로서 내부 호스트 PC에서 발생하는 비정상 행위를 탐지하기 위한 패스트 데이터 기반의 실시간 비정상 행위 탐지 시스템에 대해서 설명한다.

4.1 시스템 구조

Fig. 1.은 본 논문에서 제안하는 실시간 비정상 행위 탐지 시스템의 구조를 나타낸다. 실시간 비정상 행위 탐지 시스템은 Apache Storm 토폴로지 형태로 구현되는 실시간 처리 모듈과 별도의 프로세스로 동작하는 호스트 기반 분석 모듈(Host-based Analysis Module)로 구성된다.

Storm은 다수의 호스트 PC에서 실시간 입력되는 프로세스 행위 이벤트 데이터로부터 프로세스 행위 정보를 추출하고 초기 처리 및 분석을 실시하는 실시간 처리 모듈을 수행하기 위해서 사용된다. Storm을 통해 프로세스 행위 이벤트 데이터 처리 및 분석을 분산 환경에서 이벤트 데이터가 시스템에 입력되자마자 즉시 처리할 수 있게 된다.

Fig. 1.에서 탐지 대상 호스트 PC에 설치되는 호스트 PC 에이전트는 각 호스트 PC에서 발생되는 모든 프로세스 행위 이벤트를 수집하여 중앙 공유 데이터 서버에 관리되는 호스트 데이터 수집 디렉토리에 저장한다. 호스트 데이터 수집 디렉토리는 모든

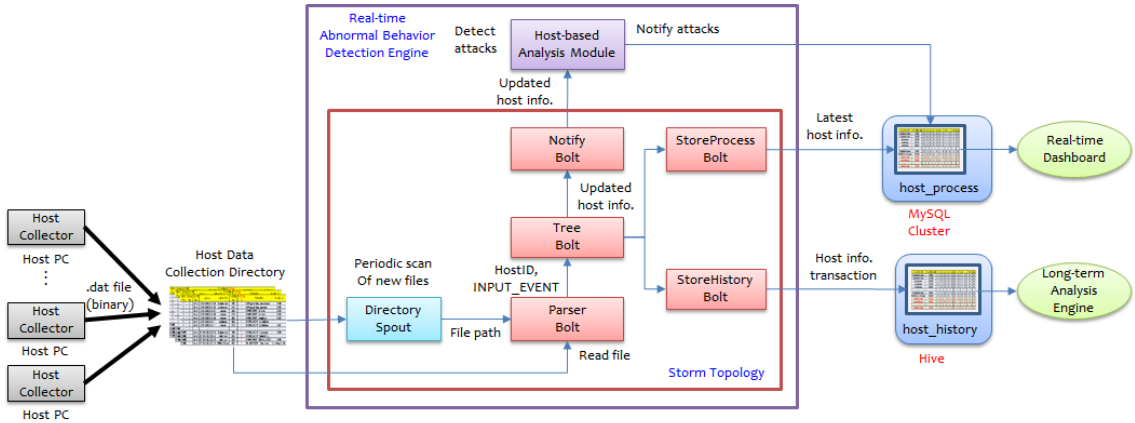


Fig. 1. Structure of Real-time Abnormal Behavior Detection System

Storm 수퍼바이저 머신에서 접근 가능하다.

DirectorySpout는 설정에 의해 정의된 호스트 데이터 수집 디렉토리를 주기적으로 스캔하여 새롭게 수집된 호스트 데이터 파일을 인지한다. 만일 새로운 호스트 데이터 파일이 수집되면 해당 파일의 파일 경로를 ParserBolt에 통지한다.

ParserBolt는 호스트 데이터 파일에 바이너리 형태로 저장된 프로세스 행위 이벤트 데이터를 구문 분석하여 Table 1.의 호스트 행위 정보 데이터 구조로 프로세스 행위 이벤트 정보를 추출하고, 추출된 프로세스 행위 이벤트 데이터는 변경 이력 관리를 위해 TreeBolt 로 전달한다.

TreeBolt는 프로세스 행위 상태 및 이력을 관리하고 추적하기 위해서, 프로세스 행위 특성 인자 벡터를 포함하는 프로세스 행위 정보 트리를 관리한다. 만일 신규로 발생한 이벤트가 이미 트리에 등록된 프로세스의 이벤트라면, 해당 프로세스의 특성 인자 벡터가 신규 이벤트 정보를 포함하도록 갱신된다.

모든 특징 벡터 트리에 대한 변경 사항은 NotifyBolt, StoreProcessBolt, StoreHistoryBolt에 통지되며, NotifyBolt는 TCP 서버로 동작하는 호스트 기반 분석 모듈에 특징 벡터 변경 사항을 통지한다.

StoreProcessBolt는 최신 프로세스 상태 정보를 MySQL 클러스터 (보다 빠른 저장 및 검색을 위한 MySQL 데이터베이스의 인메모리 버전)에 저장한다. StoreHistoryBolt는 모든 프로세스 행위 이벤트의 트랜잭션을 Apache HDFS (Hadoop Distributed File System) 파일에 저장한다. 이 파일들은 나중에 Apache Hive (SQL-on-Hadoop 시스템) 스토어에 저장되어 보다 진보하고 복잡한 이벤트 간의 상호 관계 기반 비정상 행위 탐지를 위한 장기적인 분석에 사용된다.

호스트 기반 분석 모듈은 NotifyBolt로부터 통지받은 변경 사항들에 대해 C4.5 결정 트리 알고리즘을 적용한 상관 분석을 수행하여, 공격인지 아닌지를 판단한다. 만일 공격으로 판단되면, MySQL Cluster의 host_process 테이블에 해당 프로세스가 "공격" 상태라고 표시한다.

이러한 프로세스 상태의 변화는 관리자에게 통지할 수 있도록 실시간 대시보드 또는 보안 분석 결과를 사용자에게 제공할 수 있도록 보안 가시화 도구에도 즉각 반영된다. 이를 통해 관리자 및 사용자는 보안 문맥을 보다 직관적으로 이해할 수 있게 된다. APT 공격 추적 엔진은 분석 결과를 바탕으로 공격 근원지를 추적할 수 있다.

4.2 프로세스 행위 정보 트리 구조

Table 2.의 모든 프로세스 행위 특성 인자들은 악성 코드 뿐만 아니라 정상적인 프로세스에서도 관찰이 되며, 하나의 특성 인자 이벤트만으로는 정상 행위인지 여부를 판단할 수 없다. 예를 들어, 프로세스 생성을 의미하는 CreateProcess 특성 인자 이벤트 1개로는 APT 공격이 발생했는지 판단할 수 없다.

따라서, 본 논문에서는 수집한 프로세스 행위 특성 인자 이벤트를 비정상 행위를 판단할 수 있는 자료 구조 형태로 구축하였다. 즉, 프로그램이 실행되는 동안 발생되는 모든 행위의 패턴 및 이력을 나타내기 위해서 동일한 특성 인자 이벤트가 발생한 빈도

수를 누적하여 프로세스 ID 별로 구축하였다.

일반적으로 하나의 실행 프로그램(정상 실행 파일 또는 악성 코드)은 여러 개의 자식 프로세스를 생성할 수 있다. 특히, 악성 코드의 경우는 하나의 악성 코드가 여러 가지 악성 행위를 수행하며, 각각의 악성 행위를 담당하는 자식 프로세스를 생성하는 경우가 발생한다. 따라서, 특정 프로그램의 실행 중 행위를 보다 정확히 표현하기 위해서는, 해당 프로세스 ID에 대한 특성 인자 이벤트의 관리 뿐만 아니라 자식 프로세스들의 특성 인자 이벤트에 대한 관리가 반드시 요구된다.

이에 따라, 호스트 PC로부터 수집된 모든 프로세스 행위 정보는 TreeBolt가 인메모리 관리하는 프로세스 행위 정보 트리에 삽입되면서, 프로세스 별로 각 특성 인자 이벤트의 발생 횟수를 누적해서 표현할 수 있는 47차원 특성 인자 벡터 형태로 구축된다. 이때, 자식 프로세스가 존재할 경우 자식 프로세스에서 발생되는 특성 인자 이벤트의 발생 횟수까지 함께 별도의 47차원 특성 인자 벡터에 표현함으로써, 모두 94 차원의 특성 인자 벡터를 이용해서 해당 프로세스의 시작 시점부터 종료 시점까지 발생되는 프로세스 행위 정보 및 이력 정보를 특성 인자 벡터로 유지 관리 할 수 있다.

모든 최신 프로세스 행위 특징을 나타내는 빈도수는 TreeBolt에 의해 인메모리 누적 관리되고, 또한 MySQL 클러스터에 영속 저장되어 관리되며, 만일 프로세스 생성 또는 종료, 빈도수 변경 이벤트 등이 발생하면 실시간으로 호스트 기반 분석 모듈에 통지된다.

Fig. 2.는 프로세스 행위 정보 트리의 한 예를 나타낸다. 프로세스 행위 정보 트리는 프로세스가 수행되는 호스트의 IP 주소 별로 관리가 되며 TreeBolt

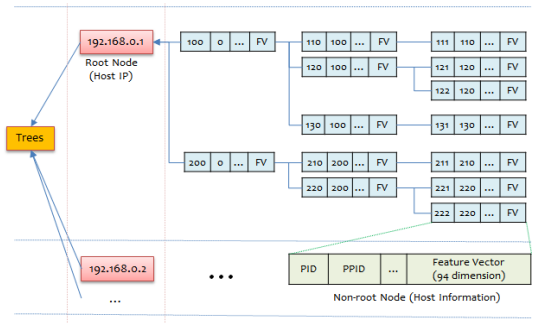


Fig. 2. An Example of Process Behavior Information Tree

```
set trees = process behavior information tree

procedure getRoot(hid : host id)
  set root = trees.get(hid)
  return root
end procedure

procedure addEvent( hid : host id, pid : process id, ppid
: parent process id, f : feature id )
  set root = getRoot(hid)
  set node = root.findNode(pid) // traverse tree
  if (node is null) then              // if not found
    if (ppid is 0) then               // if no parent exists
      set node = root.addNode(pid, ppid, f)
    else
      set parent = root.findNode(ppid)
      set node = parent.addNode(pid, ppid, f)
    end if
  end if
  set v = node.getVector()

  // update feature vector of current process
  set curr_idx = getIndex(f, false)
  set v[curr_idx] = v[curr_idx] + 1
  // update feature vector of ancestor processes
  set parent = node.getParent()
  while (parent is not null)
    set p_v = parent.getVector()
    set p_idx = getIndex(f, true)
    set p_v[p_idx] = p_v[p_idx] + 1
    set parent = parent.getParent()
  end while
end procedure
```

Fig. 3. Procedure for Process Behavior Information Tree Management

는 전체 프로세스 행위 정보 트리 목록을 관리한다. 각 트리의 노드는 프로세스의 〈PID(Process ID), PPID(Parent PID), 기타 프로세스 정보, 그리고 94 차원 특징 벡터〉로 구성된다.

이벤트 발생에 따라 프로세스 행위 정보 트리에 노드가 삽입되고 이벤트 발생 회수가 누적되는 과정은 Fig. 3.에서 보는 바와 같다. 부모가 없는 모든 프로세스(즉, PPID가 0인)는 각 트리의 루트 노드(즉, 호스트 IP 주소를 나타내는)의 자식 노드로 삽입된다. 부모가 있는 모든 노드는 부모 프로세스를 나타내는 노드의 자식 노드로 삽입된다. 만일 이미 삽입된 프로세스에 발생한 새로운 이벤트가 수집되면, 해당 이벤트가 가진 모든 정보는 트리의 노드에 반영된다. 또한 노드가 가진 특징 벡터에 신규로 수집된 이벤트의 특징이 누적되어 관리가 된다.

4.3 프로세스 행위 특성 인자 벡터 표현

프로세스 행위 특성 인자 이벤트의 빈도수를 누적

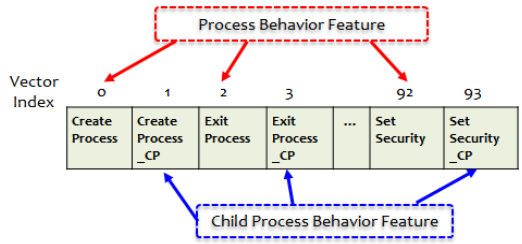


Fig. 4. Structure of Feature Vector

관리하는 47차원과, 자식/자손 프로세스들의 행위 특성 인자 이벤트 빈도수를 누적 관리하는 47차원의 총 94 차원으로 구성되는 프로세스 행위 특성 인자 벡터의 구조는 Fig. 4.와 같다.

Fig. 4.에서 특성 인자 벡터의 인덱스 위치에 대한 이름은 각 인덱스 위치가 가리키는 〈특성 인자의 이름 + (_CP)〉로 구성된다. 접미어 _CP는 해당 인덱스 위치가 자식/자손 프로세스로부터 파급된 특성 인자 값을 갖고 있음을 의미한다. 예를 들어, 0번 인덱스 위치의 CreateProcess는 해당 프로세스가 타 프로세스를 생성한 빈도수를 나타내며, 1번 인덱스 위치의 CreateProcess_CP는 해당 프로세스의 자식/자손 프로세스들이 타 프로세스를 생성한 빈도수의 누적 합을 의미한다.

Table 2.의 특성 인자 인덱스로부터 특성 인자 벡터 인덱스를 계산하는 방법은 다음 Fig. 5.와 같다.

예를 들어, 특성 인자 벡터에서 4번 특성 인자인 OpenProcess의 누적 빈도수가 관리되는 인덱스 위치는 (4-1)*2 = 6 이고, 자식/자손 프로세스로부터 누적된 4번 특성 인자인 OpenProcess의 누적 빈도수가 관리되는 인덱스 위치는 (4-1)*2+1 = 7 이다.

```
procedure getIndex( f : feature id, cp : is child process or not? )
  if (cp is false) then
    set index = (f-1)*2
  else
    set index = ((f-1)*2)+1
  end if
  return index
end procedure
```

Fig. 5. Procedure for Calculating Feature Vector Index

4.4 프로세스 행위 정보 트리 관리

만일 신규로 수집된 특성 인자 이벤트가 CREATE_PROCESS 이면, 트리에 해당 프로세스를 나타내는 신규 노드가 삽입이 되고, 해당 프로세스의 모든 부모/조상 노드에도 CREATE_PROCESS_CP 특성 인자 (즉, CREATE_PROCESS 특성 인자의 인덱스+1 위치의 특성 인자)의 값이 1 증가된다. 이는 해당 부모/조상 프로세스 입장에서 보면, 자신이 살아 있는 동안에 자식/자손 프로세스를 몇 개나 생성했는지 프로세스 생성 이력을 관리할 수 있도록 해 준다. 이를 통해 임의의 프로세스가 비정상적인 활동을 하고 있는지 탐지할 수 있는 기본 자료가 수집된다.

프로세스 행위 정보 트리에서 특성 인자 벡터에 대한 변경 절차 및 변경의 파급 절차는 Fig. 6.에서 보는 바와 같다.

만일 노드의 특성 인자 벡터에 변경이 발생하면, 모든 변경 사항은 Fig. 7.에서 보는 바와 같이 모든 부모/조상 노드에 반영이 된다. 예를 들어서, 123 프로세스 노드의 32번 특성 인자의 값이 1 증가하면, 부모/조상 프로세스인 122, 120, 100 프로세스

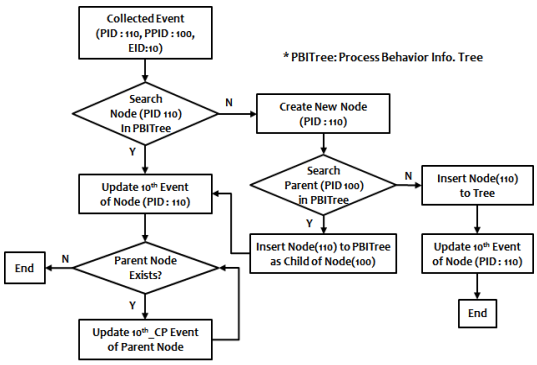


Fig. 6. Procedure for Feature Vector Update

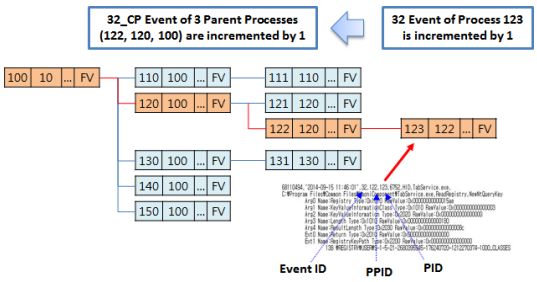


Fig. 7. An Example of Feature Vector Update

노드의 32_CP 특성 인자의 값도 1 증가한다. 이를 통해서 특정 프로세스 및 해당 프로세스에서 생성된 모든 자식/자손 프로세스의 누적된 프로세스 특성 인자 이벤트 발생 상황을 관리하고 이를 모니터링하여 프로세스의 정상 및 비정상 상황을 파악할 수 있다.

또한 이러한 모든 특성 인자 벡터의 변경 사항은 호스트 기반 분석 모듈에 전달이 되어 결정 트리 알고리즘을 거쳐서, 최종적으로 공격/정상 여부를 판단하게 된다. 이때 변경이 발생한 노드는 물론 해당 노드의 부모/조상 노드의 변경 상태, 즉 갱신된 특성 인자 벡터 값도 같이 전달이 되며 상관 분석에 활용된다.

4.5 비정상 행위 탐지

MySQL Cluster에 관리되는 프로세스 최신 상태 정보 테이블인 host_process의 스키마는 다음 Fig. 8.과 같다. 호스트 ID(host_id) 필드와 프로세스 ID(pid) 필드가 기본 키 (primary key)로 사용된다. 대부분의 필드는 프로세스 행위 이벤트가 수집되면서 값이 정해지며, 정상/공격 여부를 표시하는 type 이벤트는 초기 정상 모드로 저장되고, 나중에 호스트 기반 분석 모듈의 결정 트리 알고리즘에 의해서 공격 여부가 결정된 후에 공격(1)으로 변경된다[10]. 이때 공격이 발생한 결정 트리의 이벤트 발생 경로값도 같이 path 필드에 반영이 된다.

```
CREATE TABLE host_process (
  no_events int unsigned,  # event occurrence index
  fname varchar(30),       # process name
  pid smallint unsigned,   # process ID
  ppid smallint unsigned,  # parent PID
  host_id varchar(15),     # host IP address
  user_id varchar(30),     # user ID
  start_time datetime,     # process start time
  last_time datetime,      # latest process update time
  last_time_cp datetime,   # last_time of child process
  analysis_time datetime,  # analysis time
  start_event_id smallint unsigned, # first event's ID
  last_event_id smallint unsigned,  # latest event's ID
  type int unsigned DEFAULT 0, # 0:normal, 1:attack
  vector varchar(512), # 94 occurrences +93 delimiters
  path varchar(256), # decision tree path:
                     create->send->recv
  PRIMARY KEY (host_id, pid)
) ENGINE=NDBCLUSTER default charset=latin1;
```

Fig. 8. Process Behavior Information Schema

호스트 기반 분석 모듈은 통지된 프로세스 행위 정보의 정상/비정상 여부를 결정하기 위해 결정 트리 알고리즘을 수행해서 APT 공격 여부를 판단하며, 이는 실험 결과 낮은 미탐율(5.8%)과 오탐율(2.0%) 정확도를 보였다[10].

V. 성능 시험

본 장에서는 본 논문에서 제안하는 실시간 비정상 행위 탐지 시스템의 프로세스 행위 이벤트 처리 성능을 측정하기 위하여 시험한 환경, 시험 결과, 그리고 최대 성능을 위하여 수행한 몇 가지 최적화 작업에 대해서 기술한다.

5.1 시험 환경

본 논문에서 제안하는 실시간 비정상 행위 탐지 시스템의 성능 시험은 Fig. 9.에서 보는 바와 같이 3대의 노드로 구성된 시험 환경에서 수행하였다.

우선, storm1 노드(Intel Core i7-3770K CPU @ 3.50GHz quad-core CPU, 16GB RAM, 256GB SSD)와 storm2 노드(Intel Core i7-2600 CPU @ 3.40GHz quad-core CPU, 8GB RAM, 256GB SSD)는 실시간 데이터 처리를 위한 Storm 클러스터, Storm 클러스터 관리를 위한 ZooKeeper 클러스터, 그리고 최대 12GB의 최신 프로세스 행위 상태 정보를 저장하는 MySQL

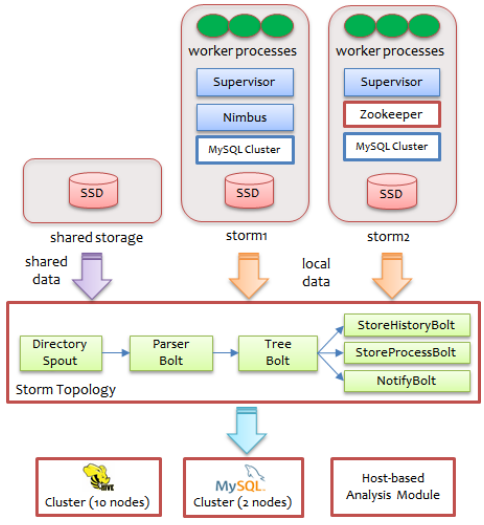


Fig. 9. Performance Test Environment

클러스터를 구성하며, 총 15TB의 공간을 갖는 10대의 외부 Hadoop/Hive 클러스터에 모든 수집된 프로세스 행위 정보 트랜잭션을 기록한다.

공유 스토리지(shared storage) 노드는 SSD 기반의 공유 스토리지 공간(shared)을 제공하고, storm1과 storm2 노드는 각각 SSD를 지역 스토리지 공간(local)으로서 갖고 있다.

운영 체제로는 Ubuntu 10.04 LTS를 사용하며, 각 노드에는 Zookeeper 3.4.6, Storm 0.9.2, MySQL Cluster 7.3.6 버전의 배포본을 설치하여 시험하였다.

성능 시험을 위한 데이터는 2개 호스트에서 정상 및 비정상 상황을 가상으로 만든 상태에서 자체 제작한 호스트 데이터 수집기 에이전트를 이용하여 Table 2.의 47개 특성 인자 관련한 프로세스의 모든 행위를 수집하였다. 수집한 호스트 프로세스 이벤트 데이터는 Table 3.에서 보는 바와 같이 2개의 호스트에서 수집한 총 7개 파일로 구성되며, 전체 크기는 약 4.1GB이고, 약 800만개의 호스트 프로세스 이벤트를 갖고 있다. 각 파일은 하나의 호스트에서 1분 동안 발생한 모든 프로세스 행위를 수집한 것이며, 설정 수집 주기에 따라 파일 크기 및 이벤트 개수는 달라질 수 있다.

Table 3. Performance Test Data

File Name	Size	#Events
Log_HID-PC_10.200.0.203_2014-09-15_114601.dat	570 MB	1,101,535
Log_HID-PC_10.200.0.203_2014-09-15_114701.dat	564 MB	1,090,444
Log_HID-PC_10.200.0.203_2014-09-15_114801.dat	567 MB	1,095,918
Log_HID-PC_10.200.0.203_2014-09-15_114901.dat	586 MB	1,130,746
Log_HID-PC_10.200.0.203_2014-09-15_115001.dat	562 MB	1,086,490
Log_HID-PC_10.200.0.203_2014-09-15_115101.dat	567 MB	1,095,777
Log_HID-PC_192.168.130.80_2014-09-29_135437.dat	711 MB	1,355,649
Entire Data	4,127 MB	7,956,559

5.2 시험 결과

시험은 지역 스토리지와 공유 스토리지 접근 속도 차이와 DB 트랜잭션 유무에 따른 성능 차이를 알아보기 위해서 다음과 같이 4가지 조건에서 수행하였다. 4가지 조건에서 각각 호스트 수집 바이너리 데이터 파일을 구문 분석하는 파서 볼트(ParserBolt)의 쓰레드 개수를 변경하면서, 노드당 프로세스 행위 이벤트 처리 성능(개수/초)과 최대 성능 발휘 시의 CPU 활용율을 측정하였다.

먼저, DB 트랙잭션 유무에 따른 성능을 알아보기 위해서, no-store-local 과 no-store-shared 조건에서는 파서 볼트(ParserBolt)가 입력 바이너리 호스트 데이터 파일 내의 레코드를 처리하고, 4개의 트리 볼트(TreeBolt)를 거쳐서, 최신 프로세스 행위 특성 인자 벡터 상태 정보를 생성하는 단계까지만 포함하여 시험하였고, store-local 과 store-share 조건에서는 MySQL 클러스터에 관리되는 process_history 테이블에 저장하는 4개의 프로세스 저장 볼트(StoreProcessBolt)를 포함하여 시험하였다.

두 번째, 지역/공유 스토리지 접근 속도 차이에 따른 성능을 알아보기 위해서, store-shared 와 no-store-shared 조건에서는 모든 수집 호스트 데이터를 공유 저장소에 저장하고 ParserBolt가 원격에서 읽게 하였고, store-local 과 no-store-local 조건에서는 모든 수집 데이터를 각 처리 노드의 동일 위치에 복제해서 저장해 놓고 ParserBolt가 지역 읽기를 수행하도록 하였다.

보다 정확한 시험을 위해, 태스크가 실제 연산을 수행하기 전에 태스크 배치 및 태스크 간의 연결을 수립하기 위해 소비되는 초기 10초의 워밍업 시간을 제외하고 시비스 수행 시작 10초 후부터 1분 10초까지 총 60초 동안 처리한 결과로부터 초당 프로세스 행위 이벤트 처리 개수를 측정하였다.

Fig. 10.에서 보는 바와 같이, DB에 트랜잭션을 저장하는 경우(store-local, 파서 8개 시) 최대 17,220건/초/노드의 성능을 보였으며, 저장하지 않는 경우(no-store-shared, 파서 8개 시)에는 최대 41,262건/초/노드의 성능을 보였다.

일반적으로, 노드 수가 많은 경우 원격 노드에서 읽는 것보다 로컬 머신에 복사한 후 읽는 것이 대규모 병렬 IO 성능을 더 높일 수 있다[25]. 본 시험에서는 노드를 2대만 사용하였기 때문에, 로컬 머신에 데이터를 중복해서 저장한 후 읽는 방식(local)이 공유 저장소에 저장한 후 원격에서 읽는 방식(remote)과 비교해서 유의미한 성능 차이는 보이지 않았다. 향후 보다 대규모 클러스터 환경에서 시험한

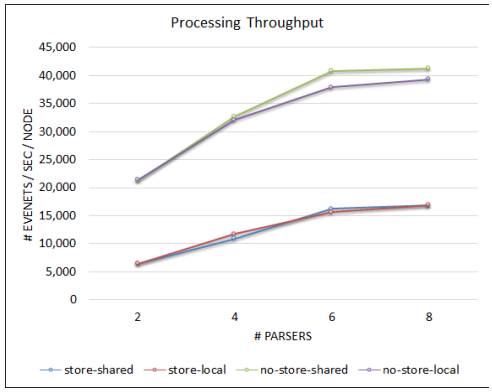


Fig. 10. Processing Throughput

다면, 네트워크 IO 부담을 각 노드의 디스크가 분산해서 담당하기 때문에 로컬 머신의 데이터를 읽는 방식이 더 성능이 좋게 나올 것으로 예측된다.

Fig. 11.은 상기 시험 시에 storm1 노드에서 측정한 CPU 활용률을 나타낸다. storm1 노드에서는 총 8개의 파서, 즉 각 노드에 4개의 파서를 동시 실행하는 경우 가장 CPU 활용률이 좋게 나왔다.

Fig. 12.는 상기 시험 시에 storm2 노드에서 측정한 CPU 활용률을 나타내며, storm2 노드에서는 총 6개의 파서, 즉 각 노드에 3개의 파서를 동시 실행하는 경우 가장 좋은 CPU 활용률이 측정되었다.

Fig. 13.은 전체 7,956,559 개의 프로세스 행위 이벤트를 처리하는데 걸린 시간을 나타낸다. DB에 트랜잭션을 저장하지 않는 경우(no-store-shared, 파서 8개 시), 108초의 처리 시간이 소요되어 가장 좋은 성능을 보였다.

DB에 트랜잭션을 기록하는 경우, 전체 파서 개수가 8개인 경우 StoreProcess 볼트의 DB 트랜잭션

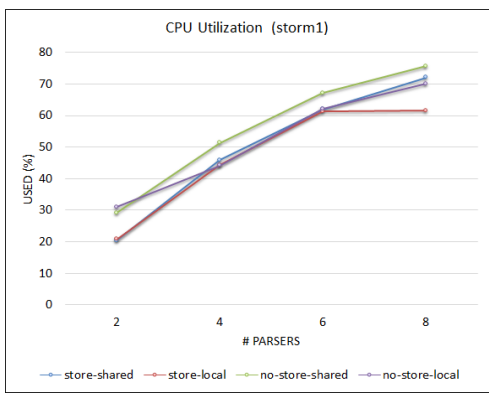


Fig. 11. CPU Utilization (storm1)

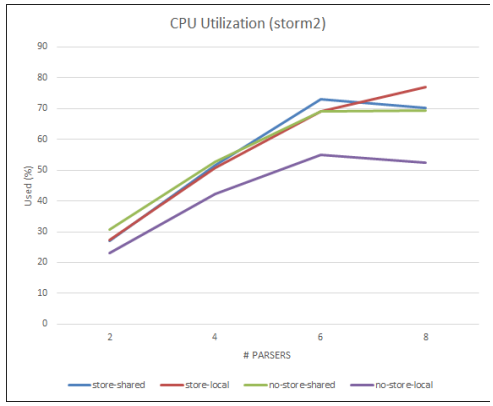


Fig. 12. CPU Utilization (storm2)

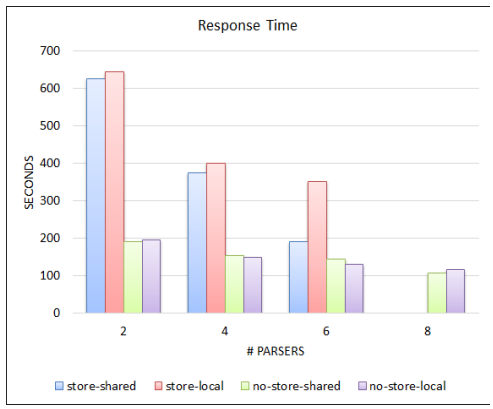


Fig. 13. Response Time

저장 지연으로 인해, Tree 볼트의 출력 버퍼에 데이터가 쌓이게 되고, 결국 출력 버퍼 오버플로로 인해 관련 태스크가 재시작 되어 정확한 처리 시간을 측정할 수 없었다.

본 시험에서는 시험 데이터가 총 7개의 바이너리 파일로 구성이 되어, 8개 파서 수행 시 파싱 1번 만에 모든 파일의 처리가 되지만, 4개와 6개의 파서 수행 시에는 파싱 2번 만에 모든 파일이 처리가 되어 파서 4개와 6개의 수행 완료 시간이 비슷하게 측정되었다.

5.3 최적화 작업

초기 시험 과정에서 Tree 볼트의 출력 대상으로 DB에 저장하거나(StoreProcess), HDFS에 파일을 쓰거나(StoreHistory), 실시간 상관 분석 모듈에 통지(Notify)하는 후행 볼트들이 연결되는 경우,

후행 볼트들이 입력 데이터를 선행 파서 및 트리 볼트와 같은 속도로 데이터를 처리하지 못 하여, 출력 버퍼 오버플로가 발생하고, 결국에는 태스크가 재실행되는 문제가 발생하였다.

일차적으로 이 문제를 해결하기 위한 방안으로 파서의 파싱 속도를 조절하는 설정 옵션을 추가해서 프로세스 행위 이벤트 처리 속도를 후행 볼트들이 감당할 수 있는 적절한 값으로 설정한 후 시험을 수행하였다.

이렇게 처리 속도를 강제로 느리게 한 후 시험한 결과, host_process 테이블에 최신 프로세스 상태를 오류없이 안정적으로 저장하는 성능이 최대 초당 2,000건 내외로 측정되었다. 이는 시험 환경에서 설정하여 사용한 인메모리 DB인 MySQL Cluster가 초당 저장 트랜잭션을 충분히 감당하지 못해서 발생한 문제이다.

이후, 다음과 같이 MySQL이 제공하는 배치 삽입(batch insertion) 구문을 사용해서 불필요한 DB 접근을 줄이는 최적화 작업을 수행해서 저장 성능을 최대 17,220 이벤트/초의 성능을 확보할 수 있었다.

StoreProcessBolt의 메모리 맵에 모든 트랜잭션이 일단 삽입되고, 주기적으로 메모리 맵 내의 트랜잭션이 Fig. 14.에서 보는 바와 같이 MySQL Cluster에서 지원하는 배치 삽입 구문으로 변환되어 저장된다. 이때 메모리 맵의 key는 "호스트 ID_프로세스 ID" 형태로 표현되며, 저장 주기 동안에 갱신되는 특징 벡터 값들은 MySQL Cluster에 저장되기 전에 메모리 내에서 갱신이 완료되기 때문에, MySQL 클러스터의 트랜잭션 오버헤드를 상당히 줄이는 효과가 발생한다.

본 실험에서는 비록 클러스터 버전의 MySQL을 사용하였지만, 충분한 노드 수를 확보하지 못하였고, MySQL Cluster가 제공하는 테이블 파티션(partition) 기능을 충분히 활용하지 못하였다.

그리고, MySQL Cluster의 복제(replication) 기능을 사용하는 경우, 삽입 시에 여러 노드에 중복 저장이 되기 때문에 삽입 성능이 저하되어, 본 실험의 실시간 프로세스 이벤트 저장 측면에서 도움이 되지 못 했다.

향후 다양한 시험을 통해 MySQL 클러스터가 최적으로 동작할 수 있는 설정을 찾아낸 다면, 보다 좋은 저장 성능을 달성할 수 있을 것으로 기대한다.

```
INSERT INTO host_process VALUES
    (-- 1st process behavior info.'s insert clause --),
    (-- 2nd process behavior info.'s insert clause --),
    ...
ON DUPLICATE KEY UPDATE
    no_events=VALUES(no_events),
    last_time=VALUES(last_time),
    last_time_cp=VALUES(last_time_cp),
    analysis_time=VALUES(analysis_time),
    last_event_id=VALUES(last_event_id),
    vector=VALUES(vector);
```

Fig. 14. Batch INSERT Statement

VI. 결 론

본 논문에서는 APT 공격과 같은 사이버 타겟 공격을 사전에 탐지하기 위한 패스트 데이터 기반의 실시간 비정상 행위 탐지 시스템을 제안하였다.

제안 시스템은 알려지지 않은 공격을 실시간 탐지할 수 있도록, 호스트에서 수행되는 프로세스의 행위를 규정하기 위하여 47개의 특성 인자를 정의하고, 특성 인자를 포함한 모든 프로세스 행위 이벤트를 수집하고 94차원의 특성 인자 벡터 트리를 구축하여, 프로세스 실행 시간 동안의 모든 행위 실행 이력을 모니터링하고 프로세스 행위의 변경을 상관 분석하는 방법을 사용하였다.

제안 시스템은 대표적인 분산 스트림 처리 시스템인 Apache Storm 의 토폴로지 형태로 구현이 되었으며, 이를 통해 호스트 발생 프로세스 행위 이벤트 데이터를 수집하자마자 실시간 연속 처리하여, 실제 피해가 발생하기 전에 잠재한 위협을 조기에 발견할 수 있으며 또한 분산 처리를 통해 데이터 폭증 시에 노드 추가를 통한 유연한 대처가 가능하다.

성능 시험을 통해 프로세스 행위 이벤트 데이터 초당 처리 성능(throughput)을 측정한 결과, 호스트 행위 기반 탐지 기법의 높은 정확도는 유지하면서, 노드당 약 40,000 EPS(프로세스 행위 이벤트/초)의 처리 성능을 확인할 수 있었다.

향후에는, 현재 호스트 PC 수집 데이터를 공유 저장소에 일단 저장한 후 처리하는 방식을 벗어나서 제안 시스템이 보다 실시간 처리 환경을 제공하고, 보다 높은 프로세스 행위 이벤트 처리 성능을 제공할 수 있도록, 호스트 PC에서 수행되는 호스트 데이터 수집기 에이전트로부터 데이터를 네트워크로 직접 입력 받도록 처리 구조를 개선하려고 한다.

References

[1] Verizon, "2010 data breach investigations report," 2010.

[2] Colin Tankard, "Advanced persistent threats and how to monitor and deter them," Network Security, vol. 2011, no. 8, pp. 16-19, Aug. 2011.

[3] Paul Giura and Wei Wang, "Using large scale distributed computing to unveil advanced persistent threats," Science Journal, vol. 1, no. 3, pp. 93-105, 2012.

[4] Apache Hadoop Project, http://hadoop.apache.org/.

[5] Apache Storm Project, http://storm.apache.org/.

[6] Splunk, http://www.splunk.com/.

[7] General Dynamics, "Proposal for R&D support of DARPA cyber genome program," Mar. 2010.

[8] Sung-Hwan Ahn, Nam-Uk Kim, and Tai-Myoung Chung, "Big data analysis system concept for detecting unknown attacks," ICACT 2014, pp. 269-272, Feb. 2014.

[9] Kim Jonghyeon, et al., "Trend of cyber security technology using Big Data," Electronic Communication Trend Analysis, vol. 28, 3rd Ed., June 2013.

[10] Daesung Moon, Hansung Lee, and Ikkyun Kim, "Host based feature description method for detecting APT attack," Journal of The Korea Institute of Information Security & Cryptology, 24(5), pp. 839-850, Oct. 2014.

[11] M.A. Beyer, A. Lapkin, N. Gall, D. Feinberg, and V.T. Sribar, "Big data is only the beginning of extreme information management," Gartner, Apr. 2011.

[12] Ashish Thusoo, et al., "Hive - a warehousing solution over a Map-Reduce framework," VLDB 2009, vol. 2, no. 2, pp. 1626-1629, Aug. 2009.

[13] NIST, "Guide for conducting risk assessments," Special Publication 800-30 Revision 1, Sep. 2009.

[14] Symantec, "Symantec internet security threat report," Symantec, 2011.

[15] Art Coviello, "Open letter to RSA customers," June 2011.

[16] Gartner, "Big data," http://gartner.com/it-glossary/big-data.

[17] Lambda Architecture, http://lambda-architecture.net/.

[18] Hyunjoo Kim, Ikkyun Kim, and Tai-Myoung Chung, "Abnormal behavior detection technique based on big data," Lecture Notes in Electrical Engineering, vol. 301, pp. 553-563, Apr. 2014.

[19] IBM QRadar, http://www-01.ibm.com/software/tivoli/products/security-operations-mgr/.

[20] McAfee ESM, http://www.mcafee.com/us/products/enterprise-security-manager.aspx

[21] IBM Security Intelligence with Big Data, http://www-03.ibm.com/security/solution/intelligence-big-data/.

[22] Yeonhee Lee and Youngseok Lee, "Toward scalable Internet traffic measurement and analysis with Hadoop," ACM SIGCOMM Computer Communication Review, vol. 43, no. 1, pp. 6-13, Jan. 2013.

[23] Daesu Choi, Giljong Moon, Yongmin Kim, and Bongnam Noh, "Large quantity of security log analysis using MapReduce," Journal of the Korean Institute of Information Technology, vol. 9, 8th Ed., Aug. 2011.

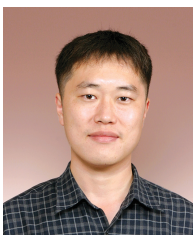
[24] Ting-Fang Yen et al., "Beehive: large-scale log analysis for detecting suspicious activity in enterprise networks," ACSAC 2013, pp. 199-208, Dec. 2013.

[25] Ioan Raicu et al., "Falkon: a fast and light-weight task execution framework," ACM/IEEE Conference on Supercomputing, no. 43, Nov. 2007.

[26] http://en.wikipedia.org/wiki/Zero-day_(computing)

〔27〕 Alissa Lorentz, "Big data, fast data, smart data," WIRED, Apr. 2013.
〔28〕 DCIG, "2014-2015 SIEM appliance buyer's guide," 2014.
〔29〕 IBM, "IBM Security QRadar SIEM - product overview," 2013.
〔30〕 NDM, "ArcSight ESM 7425," http://www.ndm.net/siem/arcsight/arcsight-esm, 2015.
〔31〕 Splunk, "Splunk performance guide v2.1," 2015.

〈저 자 소 개〉



이 명 철 (Myungcheol Lee) 정회원
1999년: 충남대학교 컴퓨터공학과(학사)
2001년: 충남대학교 컴퓨터공학과(석사)
2001년~현재: 한국전자통신연구원 데이터관리연구실 선임연구원
〈관심분야〉 빅데이터 처리 및 분석, 데이터베이스, 클라우드 컴퓨팅, 분산 컴퓨팅

문 대 성 (Daesung Moon) 정회원
1999년: 인제대학교 전산학과(학사)
2001년: 부산대학교 컴퓨터공학과(석사)
2007년: 고려대학교 전산학과(박사)
2000년~현재: 한국전자통신연구원 네트워크보안연구실 선임연구원
〈관심분야〉 네트워크 보안, 데이터마이닝, 영상처리, 지능형비디오감시, 바이오인식

김 익 균 (Ikkyun Kim) 정회원
1994년: 경북대학교 컴퓨터공학과(학사)
1996년: 경북대학교 컴퓨터공학과(석사)
2009년: 경북대학교 컴퓨터공학과(박사)
2004년~2005년: Purdue University 초빙 연구원
1996년~현재: 한국전자통신연구원 네트워크보안연구실 실장/책임연구원
〈관심분야〉 네트워크 보안, 컴퓨터 네트워크, 클라우드 보안, 빅데이터 분석