졸업 논문
지도교수 양환석

클라우드 환경에서의 탄력적 허니넷

Elastic honeynet in cloud environment

중부대학교 정보보호학과
임민성
2022. 11

차 례

1장 서론

1.1 연구 배경과 목적

악성코드의 수는 몇 년 전부터 꾸준히 증가세를 기록해왔다. 이러한 악성코드 수의 급격한 증가와 수많은 우회기법들의 탄생은 사회적으로 심각한 문제다. 안티멀웨어 제품을 연구, 테스트하는 독일의 연구기관 AV-Testorg는 신종 악성코드의 수가 2005년에는 33만 3천개, 2006년 97만 2천개로 급격하게 증가했다고 보고했다. 최근에는 매달 100만건의 신규 악성코드 샘플이 나타나고 있으며 [그림 1]에서 볼 수 있듯이 악성코드 샘플이 이미 2010년도에 3천만 개에 달하는 것을 볼 수 있다.

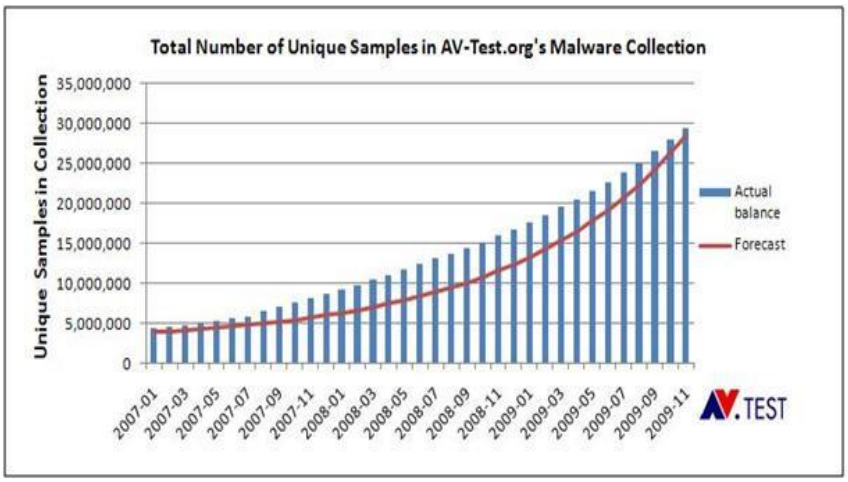

그림 1. 증가하는 악성코드의 종류

통신망을 통한 악성코드는 전통적으로 보안상 허점을 가진 포트를 공격한다거나 전자메일의 첨부파일을 통한 방식에서, 최근에는 웹 사이트의 은닉된 코드 삽입을 통한 방식으로 변화하였다. 이런 웹 사이트를 통한 공격 방식은 ISP(Internet Service Provider)에서 방화벽이나 침입차단 시스템을 사용하는 것이 보편화 되면서 기존 방식의 공격이 어려워 졌기 때문이다. 대신 악성코드를 웹 페이지의 iframe 태그, 자바 스크립트, META 태그, 어도비 플래시 콘텐츠 등에 은닉한 해당 웹 페이지에 접근하는 사용자를 공격하는 방식이 많이 이용되고 있다. 이런 방식은 방화벽이나 침입방지 시스템이 대처할 수 없으므로 공격 효과가 크다고 할 수 있다. 악성코드 은닉사이트의 수는 전 세계적으로 증가하는 추세이며, 이러한 유선 통신망을 통한 악성코드 위협에 더해서 최근에는 스마트 폰을 통한 모바일 악성코드의 위험이 점차 커지고 있다. 스마트

폰의 경우 사용자가 기존 컴퓨터 사용자에 비해 악성코드에 대한 경계심이 낮고 아직 관련 백신 프로그램도 미비한 상태이기에 관련 위험이 크다고 할 수 있다. 이런 모바일 악성 코드로는 단순 개인정보 유출 응용에서부터 SMS 과금형, 가짜 백신 응용, 악의적 관리자 권한 응용에 이르기 까지 점차 다양해지고 있고 그 수도 빠르게 증가하고 있다. McAfee의 보고서에 따르면 2010년, 2011년 각 각 약 900건과 약 1,800건의 모바일 악성 코드 사례가 있는 반면, 2012년 에는 약 13,000건의 모바일 악성 코드가 수집될 것으로 예측하고 있다. 즉, 2011년에 비해 7배의 빠른 모바일 악성코드의 증가가 예상되고 있다. 이처럼 전통적인 유선 통신망뿐만 아니라 무선 통신망 에서도 빠르게 증가하고 있는 악성코드에 대처하기 위한 방법이 매우 필요한 시점이라 할 수 있다. 이러한 악성 코드에 대처하기 위해서는 단순한 로그 분석 기법만으로 는 한계를 갖기 때문에 악성코드를 조기에 파악하고 관련 정보를 수집하여 이를 사용할 수 있는 방법이 필요하다 하겠다. [1]

위와 같은 상황으로 인해 생겨난 것이 허니팟이다. 해커의 공격으로부터 내부 자원을 보호하기 위한 허니팟 시스템은 크게 두 가지로 구분된 다. 하나는 내부 정보자원을 보호하기 위해 크래커의 공격을 유인하는 목적의 허니팟이며, 다른 하나는 방어기법을 연구하기 위해 크래커의 공격을 유도한 후 공격기법을 로그기반으로 수집하는 허니팟이다. 하지만, 최근의

공격은 크래커로 인한 공격보다는 불특정 다수를 공격하기 위해 대량의 악성코드를 통한 공격이 주를 이루고 있다. 따라서, 허니팟의 유형도 변화가 필요하게 되었다. 악성코드에 대한 방어기법을 연구하는 Anti-Virus 연구소에서는 최근의 악성코드 공격으로부터 시스템을 보호하기 위해서는 악성 코드를 조기에 수집하는 것이 주요 이슈로 등장하게 되었다. 악성코드 수집을 위한 허니팟은 기존 허니팟과 다른 특징을 가지고 있으며, 이러한 특징을 고려하여 개발되어야 한다. [2]

하지만 온프레미스 환경에서 악성코드 수집을 위한 허니팟을 구현하기에는 몇 가지 한계점이 존재한다. 첫 번째 한계점은 보안이다. 악성코드 수집을 위해서는 보안에 취약하게 고의적으로 허니팟과 해당 인프라를 설계해야 하는데, 그렇게 되면 해당 인프라는 정상적인 기능으로는 사용할 수 없고 또한 실제로 사용하는 네트워크와는 완전히 분리되어야 한다. 허니팟의 특성 상 고의적으로 보안에 취약하게 설계하기 때문에 실제 사용하는 네트워크에 영향을 미치면 안되기 때문이다.

두 번째 한계점은 수집되는 악성코드 파일과 데이터에 대한 처리이다. 허니팟을 구현하고 오랫동안 운영하면 로그 데이터와 악성코드 수집 데이터가 쌓이기 마련이다. 해당 데이터들을 관리하기 위해 저장매체를 계속해서 추가해야한다. 이 때 계속해서 필요한 새로운 저장매체와 해당 작업을 진행해야하는

관리적인 측면에서도 리소스가 계속 필요하게 된다.

세 번째 한계점은 수집된 데이터의 안정성이다. 오랫동안 노력을 들여 대용량의 악성코드 데이터와 로그 데이터를 쌓아도 일종의 오류나, 전력망 손상으로 인해 쌓아놓은 데이터가 한꺼번에 없어질 수도 있다. 이렇게 되면 몇 년간 허니팟을 운영한 결과 자체가 없어지는 것이기 때문에 백업이나 데이터의 버전 관리를 겸행해야하는데, 해당 작업 또한 리소스가 배로 들어가기 때문에 굉장히 번거롭다는 단점이 있다.

위에서 언급한 현재 허니팟의 여러가지 한계점을 극복하기 위해서 본 논문에서는 클라우드 서비스를 사용하여 클라우드 환경에서의 허니넷을 구현할 것을 제안한다. 클라우드 서비스에서 제공하는 리소스들을 효율적으로 사용하여, 추가적인 리소스가 투입되지 않아도 자동적으로 확장, 축소를 통해 최적화된 리소스로 허니팟을 운영할 수 있다.

아래 [그림 2]를 보면, 클라우드 서비스는 2010년도 중반부터 꾸준히 증가해 왔으며, 2020년에 들어서는 클라우드 컴퓨팅을 사용하지 않은 환경에서의 서버 운영을 보기 힘들 수준으로 클라우드 서비스를 많이 이용하고 있다. 이러한 흐름에 맞춰 서비스를 제공하는 서버뿐만 아니라 악성코드를 수집하거나 연구용으로 만들어진 서버도 클라우드 컴퓨팅을 사용하여 보다

비용효율적이고 가용적으로 운영할 수 있는 것이다.

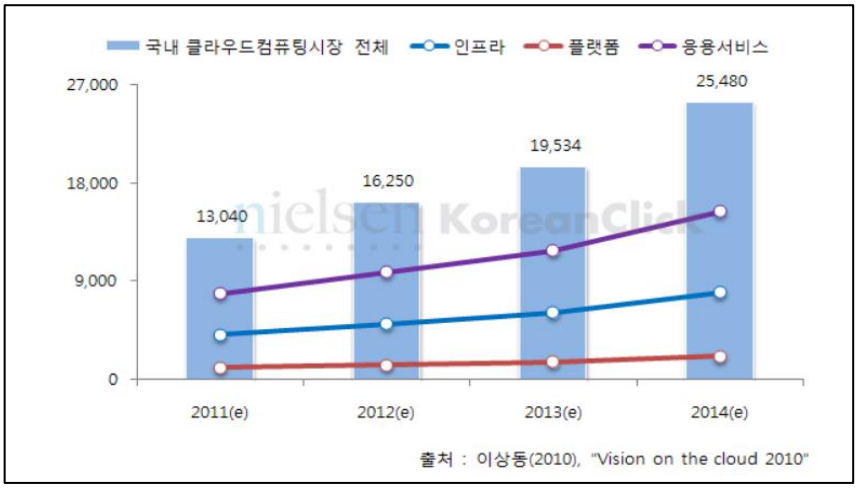

그림 2. 고속으로 성장중인 클라우드 시장

1.2 연구 구성과 내용

허니팟은 OWSAP의 프로젝트 중 하나인 OWASP/Honeytrap 오픈소스를 사용하였다. Honeytrap 은 아래 [그림 3]과 같이 총 3계층 인프라로 구성되어있다. 1계층은 크래커(Cracker)를 유인할 취약하게 구성된 웹 서버와 Modsecurity(웹방화벽)가 구성된다. 취약한 웹 서버에 접근하는 로그를 ModSecurity가 룰셋을 사용하여 공격 로그를 수집 할 것이다. 이렇게 수집된 로그 데이터들은 2계층에 구성된 ELK Stack 서버로 전송된다. ELK는 Elasticsearch, Logstash, Kibana의 세 가지 인기 있는 프로젝트로 구성된 스택을 의미하는 약어이다.

이 3가지 프로젝트를 통합해 데이터들을 시각화하고 분석하기 용이하게 만들 수 있다. 추후에 허니팟으로 모아놓은 공격 데이터나 로그 데이터를 사용하여 프로젝트를 구축할 시에 ELK Stack에 저장된 데이터를 사용할 것이다. 마지막으로 모든 공격 데이터를 3계층에 있는 MISP 서버로 전송하여 악성코드 공유 플랫폼에 보고되게 할 수 있도록 구성된다.

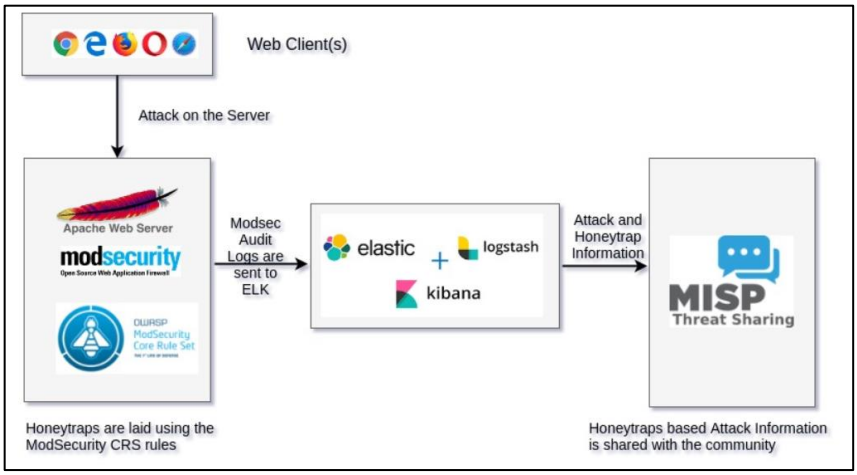

그림 3. OWASP HoneyPot 프로젝트의 구성

해당 오픈소스 프로젝트를 클라우드 서비스를 사용하여 구축할 것이다. 사용한 클라우드 서비스는 AWS를 사용했으며, 2개의 EC2 인스턴스와 1개의 CloudWatch, 1개의 ECS(Elastic Container Service)를 사용하여 구축했다. 각각의 인스턴스가 오픈소스 프로젝트의 계층을 담당하며, 공격자의 공격을 받을 웹서버는 비용효율적인 운영을 위해 작업 단위 수로 비용이 청구되는 ECS로 구현하였다. 최종 인프라 구성은 아래 [그림 4]와 같이

구성하였다.

웹 서버를 통해 Access 되는 로그들을 최종적으로 전부 모아놓기 위해서 Cloudwatch logs 서비스를 활용하여 백업 로그를 따로 저장하여 만약 ELK Stack 서버가 손실되더라도 백업 로그 데이터가 남도록 구성했다.

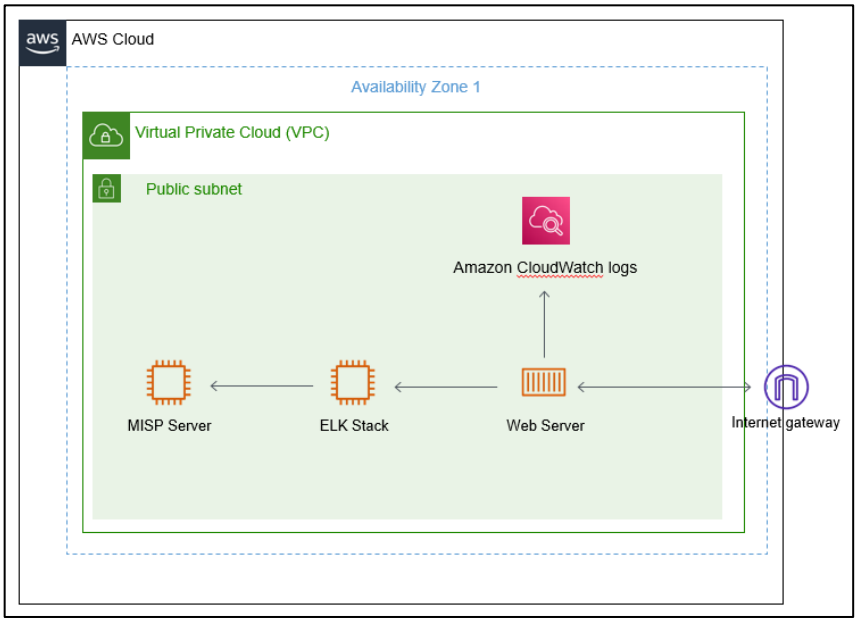

그림 4. 클라우드 허니팟의 구성도

1.3 용어 정의

'허니팟'

허니팟(honeypot) 또는 허니 포트(honey pot)[1]는 비정상적인 접근을 탐지하기 위해 의도적으로 설치해 둔 시스템을 의미한다. 예를 들어, 네트워크 상에 특정 컴퓨터를 연결해 두고 해당 컴퓨터에 중요한 정보가 있는 것처럼 꾸며두면, 공격자가 해당 컴퓨터를 크래킹하기 위해 시도하는 것을 탐지할 수 있다.

허니팟은 네트워크에 공격이 있는지를 알아채는 도구로 사용할 수 있으며, 또한 스팸 메일과 같이 기계적인 공격의 패턴을 파악하는 데에도 사용이 가능하다.

'클라우드 컴퓨팅'

클라우드 컴퓨팅(영어: cloud computing)은 사용자의 직접적인 활발한 관리 없이 특히, 데이터 스토리지(클라우드 스토리지)와 컴퓨팅 파워와 같은 컴퓨터 시스템 리소스를 필요 시 바로 제공(on-demand availability)하는 것을 말한다. 일반적으로는 인터넷 기반 컴퓨팅의 일종으로 정보를 자신의 컴퓨터가 아닌 클라우드에 연결된 다른 컴퓨터로 처리하는 기술을 의미한다.

‘AWS’

아마존 웹 서비스(영어: Amazon Web Services, 약칭: AWS)는 아마존닷컴의 클라우드 컴퓨팅 사업부이다.

아마존 웹 서비스는 다른 웹 사이트나 클라이언트측 응용 프로그램에 대해 온라인 서비스를 제공하고 있다. 이러한 서비스의 상당수는 최종 사용자에 직접 공개되는 것이 아니고, 다른 개발자가 사용 가능한 기능을 제공하는 플랫폼을 제공하는 PaaS이다.

‘EC2’

아마존 일래스틱 컴퓨트 클라우드(Amazon Elastic Compute Cloud, EC2)는 아마존닷컴의 클라우드 컴퓨팅 플랫폼 아마존 웹 서비스의 중앙부를 이루며, 사용자가 가상 컴퓨터를 임대 받아 그 위에 자신만의 컴퓨터 애플리케이션들을 실행할 수 있게 한다. EC2는 사용자가 아마존 머신 이미지(AMI)로 부팅하여 아마존이 "인스턴스"라 부르는 가상 머신을, 원하는 소프트웨어를 포함하여 구성할 수 있게 하는 웹 서비스를 제공함으로써 스케일링이 가능한 애플리케이션 배치(deployment)를 장려한다. 사용자는 필요하면 서버 인스턴스를 만들고 시작하고 종료할 수 있으며, 실행 중인 서버에 대해 시간당 비용을 지불하므로 "일라스틱"(elastic, 탄력적인)이라는 용어를 사용하게 된다. EC2는 사용자에게

레이턴시 최적화와 높은 수준의 다중화를 허용하는 지리학적 인스턴스 위치에 대한 통제 기능을 제공한다.

‘ECS’

클러스터에서 컨테이너를 쉽게 실행, 중지 및 관리할 수 있게 해주는 컨테이너 관리 서비스다.

‘MISP’

악성코드 기반의 위협 데이터를 관리하는 악성코드 첩보 공유 플랫폼으로 오픈소스이다. 보안관제나 CERT, 또는 정보보안 부서에서 많이 활용하고 있는 것으로 알려져 있고, 오픈소스 TI 플랫폼중에서 CRITs, YETI에 비해 가장 적극적으로 릴리즈 되고 있다.

‘ELK Stack’

"ELK"는 Elasticsearch, Logstash 및 Kibana, 이 오픈 소스 프로젝트 세 개의 머리글자이다. Elasticsearch는 검색 및 분석 엔진이다. Logstash는 여러 소스에서 동시에 데이터를 수집하여 변환한 후 Elasticsearch 같은 “stash”로 전송하는 서버 사이드 데이터 처리 파이프라인이다. Kibana는 사용자가 Elasticsearch에서 차트와 그래프를 이용해 데이터를 시각화할 수 있게 해준다.

2장 관련 연구

2.1 기존 허니팟에 관한 연구

[5] 공격 유인목적의 허니팟은 고객사의 웹페이지로 위장하거나 매력적인 위장 콘텐츠를 제공함으로써, 공격 자의 유입을 적극적으로 유도하는 방식으로, 해킹과 같은 공격자의 침입을 능동적으로 유도하여 보호해야 되는 정보자산인 내부시스템을 보호하는 목적을 갖는다. 이러한 공격 유인 목적의 허니팟은 쉽게 해커에게 노출 되어야하고, 해킹이 가능한 것처럼 취약해 보여야한 다. 그렇기 때문에 허니팟 한 대로는 크래커의 유도가 쉽지 않다 따라서, 다수의 허니팟으로 구성된 네트워크 즉, 허니넷(Honeynet)을 구성하는 방법으로 구현된다. 이처럼 공격을 유인하기 위한 목적의 허니팟은 시스템을 통과하는 모든 패킷을 감시할 수 있어야 하며, 관리자는 허니팟 시스템에 접속하는 접속자를 확인할 수 있도록 구성되어야 한다.

다양한 악성코드로부터 정보자산을 보호하기 위해서 허니팟 시스템을 구축한다. 허니팟 시스템은 내부 시스템이 공격받지 않도록 공격을 유인하는 목적으로 설계되거나, 악성코드 정보를 수집하기 위한 목적으로 설계된다. 그러나 기존의 하니팟은 정보 수집을 목적으로 구축되었기 때문에 위장서버

혹은 위장 클라이언트 서버를 구축하거나 위장 콘텐츠를 제공하여 공격자의 유입을 적극적으로 유도하도록 설계되었다. 그러나 위장서버구축의 경우는 빈번한 디스크 입출력으로 약 1년 주기로 하드웨어를 재설치하여야 하고, 위장 클라이언트 서버를 구축하는 경우는 획득한 정보 분석의 자동화에는 한계가 있기 때문에 전문 인 력 확보와 같은 운영상의 문제가 있다.

2.2 클라우드 인프라에 관한 연구

[7] 클라우드 서비스는 유형별로 3가지로 분류할 수 있다. 첫 번째는 IaaS 유형이다. IaaS 서비스는 가상화 기반의 서버와 스토리지, 그리고 네트워크를 제공한다. 서버 가상화 기술은 운영체제 영역을 둘로 나누어 그 하단에 위치하는 하이퍼바이저(hyperviser)를 통해 하나의 물리적 서버를 여러 개의 가상 서버로 쪼개는 것이다. 이렇게 쪼개진 가상 서버 각각을 가상 머신(virtual machine, VM)이라고 하며, 각 VM은 독자적인 하드웨어를 갖는 하나의 독립적인 서버처럼 동작한다. 현재, 하나의 고성능 PC 서버는 대략 16개 정도의 VM으로 쪼갤 수 있으며 각 VM은 서로 다른 사용자들에게 할당될 수 있다. IaaS 서비스의 사용자는 고객 측 시스템관리자가 되며 하이퍼바이저 위 단의 모든 소프트웨어 관리를 책임진다. 사용자는 각 컴퓨팅 자원의 사용량과 사용시간에 비례하여 사용료를 지불하기 때문에 자원 구매, 운영공간과 관리자 확보

등의 높은 초기 투자 비용을 줄일 수 있다. 이와 더불어 VM 등 컴퓨팅 자원에 관리자로서 전적인 제어권을 행사할 수 있기 때문에 기존의 응용들을 클라우드 환경으로 이전시킬 수 있는 높은 이식성과 상호운용성을 제공할 수 있다. 하지만 기존의 응용들이 IaaS 클라우드에서도 실행되다 보니 기존 시스템이 갖는 보안 취약성 또한 클라우드 환경에서 노출될 수 있다. 이와 더불어 많은 VM들이 다양한상태로 존재하다 보면 비활동적인 꺼진 VM들에게는 보안 업데이트가 안 되는 경우도 있을 수 있고, 따라서 이들 VM의 재구동 시 보안에 허점이 발생할 수도 있다.

두 번째 유형은 PaaS 유형이다. PaaS는 개발자들에게 확장성 있는 응용을 개발할 수 있는 기반을 제공한다. 즉 PaaS 클라우드의 응용들은 필요한 만큼의 자원들을 이용하여 필요한 만큼의 데이터 처리를 하고, 바로 배치될 수 있으며, 큰 초기 비용 없이 사용량에 따라 점진적으로 과금 된다. 이는 잘 만들어진 PaaS응용은 동적으로 급변하는 수요 증감에 부드럽게 대처할 수 있음을 의미한다. 반면에 이러한 유연한 확장성을 가능하 게 하는 서비스들과 더불어 프로그램 개발의 효율성과 편리성을 위한 부가 서비스들은 PaaS 제공 업체에 따라 각각 특화되어 있기 때문에 제공 방식과 인터페이스가 서로 상이하다. 이는 서로 다른 PaaS환경에서 만든 응용서비스들 사이에는 상호운용성이나 이식성이 매우 부족하기 때문에 벤더에 종속될 가능 성이 크고, 따라서 PaaS 표준화에 대한 필요성이 부각되고

있다. 그러나 여러 다양한 PaaS 제품들을 표준화한다는 것은 각 PaaS 제품들이 가진 그들 만의 특화된 기능들이 희석된다는 것을 의미하기 때문에 표준화를 기대하기는 쉽지 않을 것으로 예상된다

세 번째 유형은 SaaS 유형이다. SaaS에서 제공되는 응용 소프트웨어는 클라우드 서비스 제공 자의 서버에서 실행되고, 사용자의 웹 브라우저는 사용자의 입 력 데이터를 받아 클라우드로 전송하고 처리된 결과를 받아 사 용자에게 전달하는 인터페이스 역할을 수행한다. 따라서 클라 우드와 사용자 웹 브라우저 간의 통신은 공유 키 값의 인증과 이 키 값을 이용한 암호화 방식에 따라 이루어진다. 그러나 암호화된 통신방식을 사용한다고 하여도 사용자의 웹 브라우저가 위험한 사이트를 방문하였다면 오염될 수도 있고, 이후 그 웹 브라우저로 SaaS 응용을 접근한다면 사용자 데이터 보안에 문 제가 생길 수도 있다. 이러한 문제를 해결하는 한 방안으로는 전용 웹 브라우저를 두어 중요한 SaaS 응용 접근에는 이것만을 사용하고, 일반적인 웹 서핑에는 다른 웹 브라우저를 사용하는 방식이 있을 수 있다. 전통적인 컴퓨팅 방식과 비교하면 SaaS 클라우드는 확장성 제공과 더불어 관리의 편의성, 그리고 효율성 측면에서 여러 장 점을 갖는다. 우선 웹 브라우저의 기능과 성능이 우수해짐에 따 라 별도의 클라이언트 프로그램과 복잡한 설치 절차 없이 SaaS 응용을 사용할 수 있으며, 따라서 고객의 컴퓨터에 설치된 기존 프로그램들과의 간섭영향도

최소화된다. 이와 더불어 소프트웨어 유통에 소요되는 비용을 줄일 수 있고, 서로 다른 시간대라 면 한 라이선스로 여러 컴퓨터에서 그 응용을 사용할 수 있으며, 소프트웨어는 제공자의 인프라에서 실행되기 때문에 저작 권 보호 등의 우려가 감소한다. 또한 클라우드 컴퓨팅 자원 제 공과 운영을 아웃소싱하는 경우라면 SaaS 제공자는 보안검사 와 백업, 그리고 재난복구 등의 전문가적 데이터관리 서비스를 제공할 수도 있다. 반면에, SaaS 응용의 가용도는 네트워크의 신뢰성과 연속성에 밀접한 관계를 갖는다. 한 예로, 공공 SaaS 클라우드라면 인터넷을 사용하기 때문에 네트워크 신뢰도는 보장이 되지 않는 반면에 전용망을 사용하는 사설 및 커뮤니티 SaaS 클라우드라면 비용이 증가하겠지만 네트워크 보안과 신뢰도는 유지될 것이다.

클라우드 컴퓨팅 도입 시 고려사항은 5가지로 나눌 수 있다. 첫 번째는 활용분야와의 적합성 검토이다. 클라우드 컴퓨팅은 네트워크와 밀접한 관계를 이루고 있다. 네트워크와의 연결이 장점이 되는 응용 분야와 단점이 되는 응용 분야에 따라 적합성이 달라진다. 클라우드 컴퓨팅 도입이 적합한 서비스는 네트워크 연결 때문에 가능한 상호 협업 응용이나 다양한 여러 환경들을 각각 구축하고 분산 테스트 및 개발하는 응용, 자주 실행되지는 않지만 일단 실행되면 많은 자원을 필요로 하는 응용 등이 있다. 실시간성 또는 성능에 매우 민감한 응용이나

유출 가능성이 매우 적어야 하는 민감한 데이터를 다루는 응용의 경우 클라우드 컴퓨팅 도입이 적합하지 않을 수 있다.

두 번째 고려 사항은 작업 및 데이터 이전 방안 수립이다. 클라우드를 도입 시 가장 먼저 고려되어야 할 사항은 추후 클라우드 환경으로부터 복귀하는 경우에 대한 대비책이다. 클라우드 제공자와의 계약 만료, 폐기, 제공사의 도산 등 여러 가능한 경우를 고려하여 클라우드에 있는 작업(워크로드)과 데이터를 고객의 기존 환경으로 회수 또는 다른 클라우드 제공자로 이전할 수 있는 방안과 절차가 사전에 수립되어 있어야 한다.

세 번째 고려 사항은 이식성 및 상호운용성 검토이다. 클라우드 컴퓨팅의 상호운용성과 이식성은 특정 클라우드 제공자에게 종속되는 것을 막기 위한 중요한 요소이다. 사유기술에 기반을 둔 제품보다는 최대한 오픈 소스에 기반을 둔 제품을 사용해야 추후 상호 운용성이나 이식성을 향상시킬 수 있다.

네 번째 고려 사항은 규정 준수 정책 검토이다. 클라우드 컴퓨팅의 도입에서 고객의 데이터가 저장된 데이터 센터가 위치한 국가의 규정을 지켜야 한다. 제공자가 규정을 잘 준수하고 있는지 확신시키기 위해 공신력 있는 제 3자로부터 주기적인 감사를 실시하는 것은 좋은 방법이 될 수 있다.

다섯 번째 고려 사항은 다중 소유이다. 클라우드 컴퓨팅은 제공자 측 자원의 공유를 통해서 상당한 경제적 이득을 취할 수

있다. IaaS 에서는 서로 다른 VM들 간에 하이퍼바이저를 통해 하드웨어를 공유 가능하고, PaaS 에서는 서로 다른 프로세스들 간에 운영체제와 데이터, 그리고 네트워크서비스들을 공유가 가능함으로써 경제적 이득을 취할 수 있다.

3장 클라우드 허니넷의 구성

3.1 ModSecurity and FireWall

첫 번째 계층은 공격 로그를 수집할 기준을 잡아줄 ModSecurity 방화벽의 룰셋과 공격자를 유인할 몇 가지의 미끼를 가진 웹서버로 구성된다. 미끼 중 공격자가 가장 처음으로 마주할 가능성이 높은 것은 고의적으로 취약하게 구성된 robots.txt 파일이다. 본래 robots.txt 파일은 검색 엔진으로 인한 노출이나 크롤링을 막기 위한 용도로 사용하는 설정 파일이다. 그러나 잘못된 보안 설정으로 인해 웹 서버의 중요 디렉터리나 파일 경로를 노출시킬 수 있다.

OWASP 허니팟에서는 해당 robots.txt 파일 내 db_backup 이라는 공격자가 좋아할만한 미끼를 던져놓았다. 해당 파일을 관찰한 공격자는 해당 파일의 경로로 접속을 시도할 것이다.(그림 5 참조)

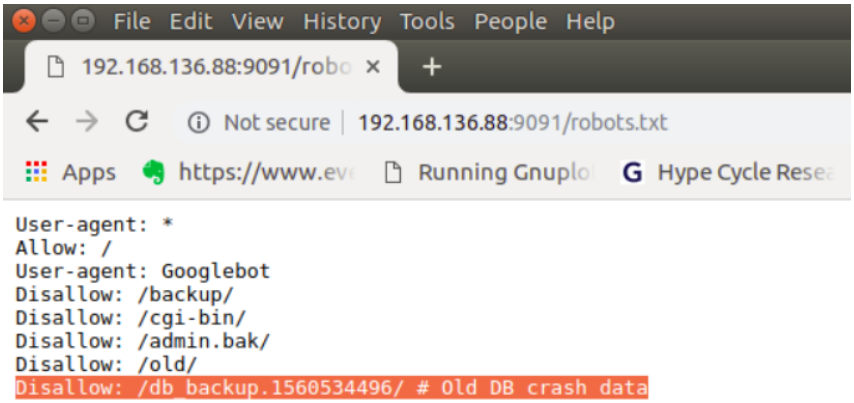

그림 5. 공격자 유인을 위한 미끼(가짜 db백업파일)

그리고 이어지는 접근 통제를 우회하거나 크랙하기 위해서 다양한 공격을 할 것으로 예상된다. (그림 6 참조)

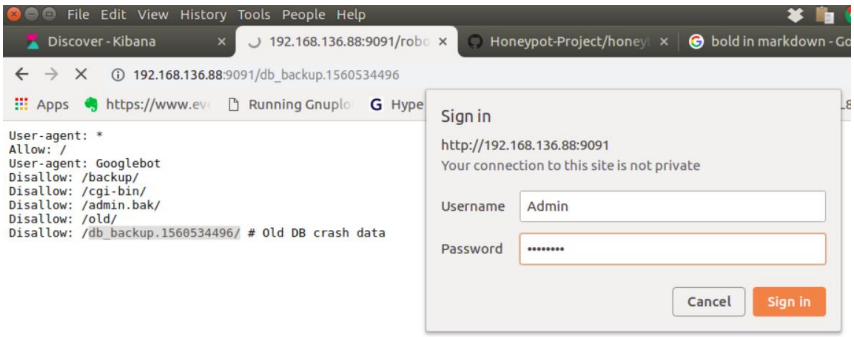

그림 6. 공격자 유인을 위한 미끼2 (기본계정으로 설정 되어있는 Admin 권한 검증)

그러면 ModSecurity 방화벽이 룰셋에 기반하여 해당 공격 로그들을 수집하여 ELK Stack이 작동하고 있는 서버로 로그를 전송한다.

3.2 ELK Stack 서버

3가지 엔진으로 구성된 ELK 서버는 다양한 로그들을 전송받아 자동으로 분류하고, 시각화하여 분석하는데 용이하게 만든다. 먼저 Logstash 엔진이 데이터를 받아 input, filter, output 3단계를 걸쳐 가공 후 데이터 저장소로 전송한다. Elasticsearch는 해당 데이터 저장소에 저장된 가공된 데이터를 검색 하는데 사용되는 검색 엔진이다. Elasticsearch는 문서 색인에 역 인덱싱(Inverted Indexing) 기법을 이용하여 빠른 검색 성능을 보인다. 역 인덱싱이란, 키워드가 어떤 문서에 있는지를 해시 테이블로 저장해 놓는 색인 방식을 말한다.

Elasticsearch는 문자, 숫자 외에도 메트릭, 위경도 등의 위치 정보에 이르기까지 다양한 정형 및 비정형 데이터에 대한 mapping을 통해 내부적으로 역 인덱싱이 가능하도록 지원한다. 따라서 다양한 형태의 데이터를 빠르게 검색할 수 있다는 특징을 보인다.

Kibana는 Elasticsearch에 있는 데이터를 시각화할 수 있도록 하는 웹 브라우저 기반의 시각화 플랫폼이다. Elasticsearch에

있는 인덱스의 패턴을 찾아서, 데이터를 확인하거나 시각화할 수 있게 한다. Elasticsearch와 REST API를 통해 통신하므로, HTTP 요청을 통해 필요한 데이터를 요청하고, 응답으로 온 데이터를 시각화한다.

따라서 이 3개의 오픈소스 엔진을 통해 [그림 7]와 같은 로그 모니터링 시스템을 구축할 수 있다. 본 논문에서 사용하고 있는 ELK Stack 로깅 서버도 [그림 7]와 같은 구조로 동작하고 있다.

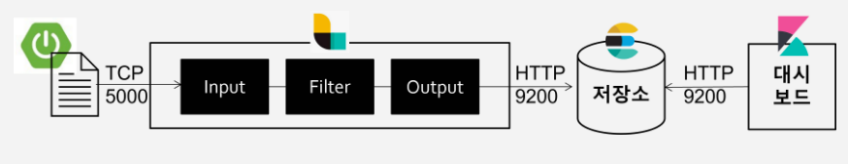

그림 7. ELK Stack 로깅 서버의 구조

3.3 다중 로깅 및 로그 백업

ELK Stack 서버의 저장소를 이용하여 로그 데이터를 저장하더라도 해당 서버의 오류로 인해서 오랫동안 모아놓은 데이터가 한번에 삭제될 수도 있기 때문에 항상 백업 데이터를 구축해야 한다. 본 논문에서는 AWS의 Cloud Watch Logs를 사용해 접속 로그들을 백업했다.

Cloud Watch는 ECS로 구동되고 있는 웹 서버 리소스에 대한 모든 로그를 기록한다[그림 8]. 특정 로그 그룹에 ECS 인스턴스를 연결하여 실시간으로 로그 기록이 가능하며, Log Insights 등의 서비스를 통해 해당 로그들을 시각화할 수도 있다.

▶	2022-08-24T21:02:06.405+09:00	2022-08-24T12:02:06.405Z ERROR pipeline/output.go:121 Failed to publish events: client is not connected
▶	2022-08-24T21:02:06.405+09:00	2022-08-24T12:02:06.405Z INFO pipeline/output.go:95 Connecting to backoff(async(tcp://54.183.236.86:5601))
▶	2022-08-24T21:02:06.409+09:00	2022-08-24T12:02:06.405Z INFO pipeline/output.go:105 Connection to backoff(async(tcp://54.183.236.86:5601)) established
▶	2022-08-24T21:02:06.410+09:00	2022-08-24T12:02:06.410Z ERROR logstash/async.go:256 Failed to publish events caused by: lumberjack protocol error
▶	2022-08-24T21:02:06.410+09:00	2022-08-24T12:02:06.410Z ERROR logstash/async.go:256 Failed to publish events caused by: lumberjack protocol error
▶	2022-08-24T21:02:06.410+09:00	2022-08-24T12:02:06.410Z ERROR logstash/async.go:256 Failed to publish events caused by: client is not connected
▶	2022-08-24T21:02:08.095+09:00	2022-08-24T12:02:08.095Z ERROR pipeline/output.go:121 Failed to publish events: client is not connected
▶	2022-08-24T21:02:08.095+09:00	2022-08-24T12:02:08.095Z INFO pipeline/output.go:95 Connecting to backoff(async(tcp://54.183.236.86:5601))
▶	2022-08-24T21:02:08.095+09:00	2022-08-24T12:02:08.095Z INFO pipeline/output.go:105 Connection to backoff(async(tcp://54.183.236.86:5601)) established
▶	2022-08-24T21:02:08.096+09:00	2022-08-24T12:02:08.096Z ERROR logstash/async.go:256 Failed to publish events caused by: lumberjack protocol error
▶	2022-08-24T21:02:08.096+09:00	2022-08-24T12:02:08.096Z ERROR logstash/async.go:256 Failed to publish events caused by: lumberjack protocol error
▶	2022-08-24T21:02:08.097+09:00	2022-08-24T12:02:08.097Z ERROR logstash/async.go:256 Failed to publish events caused by: client is not connected
▶	2022-08-24T21:02:09.700+09:00	2022-08-24T12:02:09.700Z ERROR pipeline/output.go:121 Failed to publish events: client is not connected
▶	2022-08-24T21:02:09.700+09:00	2022-08-24T12:02:09.700Z INFO pipeline/output.go:95 Connecting to backoff(async(tcp://54.183.236.86:5601))
▶	2022-08-24T21:02:09.700+09:00	2022-08-24T12:02:09.700Z INFO pipeline/output.go:105 Connection to backoff(async(tcp://54.183.236.86:5601)) established
▶	2022-08-24T21:02:09.703+09:00	2022-08-24T12:02:09.703Z ERROR logstash/async.go:256 Failed to publish events caused by: lumberjack protocol error

그림 8. CloudWatch의 접속 로그 기록들

오랫동안 로그를 쌓아두면 로그의 데이터가 엄청나게 커질 것이다. 이런 상황을 대비하여 AWS의 S3 나 DynamoDB 등의 스토리지 서비스를 사용했다. 데이터 수명 주기를 구성하여 일정 주기마다 자동으로 스토리지 인스턴스로 로그를 보내게 구성하여 비용효율적으로 운영할 수 있게 되었다. 추후에 로그들을 분석하거나 새로운 프로젝트에 이용 시 Amazon Athena같은 데이터 분석 서비스를 통해 쉽게 분석할 수 있을 것이다.

4 결론

4.1 기대 효과

기존에 온프레미스 환경에서 허니팟을 운영할 때에는 증가하는 로그 데이터에 따라 추가적인 스토리지 증설 작업, 서버의 유지보수 및 관리에 대한 리소스가 불필요하게 많이 투자되었다. 하지만 클라우드 서비스를 활용한 허니팟은 그런 불필요한 리소스를 전부 자동화시켜 비용효율적으로 운영이 가능하다.

예를 들어 온프레미스 환경에서의 경우 데이터가 늘어날 때마다 새로운 스토리지를 서버에 장착/해제해야 하는 번거로움이 있다. 또한 새로운 스토리지 구매에 들어가는 비용도 계속하여 증가한다. 이러한 한계점들을 클라우드 서비스로 해결할 수 있다. AWS의 S3나 DynamoDB 등과 같은 스토리지 서비스들은 자기 자신에 대한 Auto scaling이 가능하기 때문에 스토리지 관리를 자동화하여 늘어나는 데이터에 맞춰 비용을 지불할 수 있다.

또한 로그데이터에 대한 수명 주기 관리도 자동화할 수 있다. 오래 쌓여있거나, 사용을 자주 하지는 않지만 접근 시 빠른 접근을 원하는 로그 데이터는 S3 IA에 보관할 수 있고, 아예 접근을 자주 안하고 접근을 하더라도 고속의 응답이 필요없는 데이터의 경우 S3 Glacier에 데이터를 보관할 수도 있다. 이처럼 데이터의 특성에 따라 다양한 스토리지 서비스에 용도에 알맞게

비용효율적으로 데이터를 보관할 수 있다.

클라우드 서비스에 허니팟을 구현함으로써 우리는 해당 인프라에 대해 인력을 더 이상 투자하지 않아도 된다. 서버 관리자나 데이터베이스 기술자 등 추가 인력 필요없이 클라우드 인프라를 관리할 인력만 있으면 허니팟 시스템을 유지보수할 수 있기 때문에 유지보수면에 있어서도 강점을 가질 수 있다.

4.2 향후 개선 방향

현재 구현된 시스템은 로그를 Loastash서버의 로컬에 저장한다. 향후에 추가 구현을 통해 AWS 데이터 수명 주기 정책을 이용하여 분석이나 AI에 접목할 데이터들의 수명을 관리하여 자동적이고, 비용효율적으로 관리할 수 있도록 할 것이다.

사용처가 마땅치 않거나 필요가 없는 데이터들은 S3 Glacier를 사용해 저렴한 비용으로 장기 보관을 하고, 자주 액세스가 필요한 데이터들은 S3 Standard 등을 사용해 효율적으로 관리할 것이다.

참고문헌

[1] 이주화. "허니팟을 이용한 악성코드 수집과 이용 방법에 관한 연구". 2012

[2] 허종오. "악성코드 수집을 위한 글로벌 허니팟 시스템 구축에 관한 연구". 2010

[3] B. Endicott-popovsky, J. Narvaez, C. Seifert, D. A. Frincke, L. R. O'Neil, and C. Aval, "Use of deception to improve client honeypot detection of drive-by-download attacks," Proc. of the 5th Inter-national Conference on Foundations of Augmented Cognition (FAC), 2009

[4] C. Seifert, P. Komisarczuk, and I. Welch, "True Positive Cost Curve: A Cost-Based Evaluation Method for High-Interaction Client Honeypots", SECUREWARE, 2009

[5] 이문구. Journal of the Institute of Electronics and Information Engineers = 전자공학회논문지 v.51 no.11 , 2014년, pp.127 – 133

[6] M. Egele, P. Wurzinger, C. Kruegel and E. Kirda, "Defending browsers against drive-by downloads: Mitigating heap spraying code injection attacks," 2009. Available from http://www.iseclab.org/papers/ driveby.pdf; accessed on 15 May. 2010

[7] 김양우, 이승윤. 클라우드 컴퓨팅의 분석과 이해. 한국통신학회지(정보와통신). 2015. pp.87 - 92

[8] Peter Mell, Timothy Grance, "The NIST Definition of Cloud

Computing", National Institute of Standards and Technology, 2011

그림 및 표 차례