ELK Stack을 활용한 SQL Injection 로그 탐지

민송하 · 유현재[1] · 임문주 · 김종민*

동신대학교

Detecting SQL Injection Logs Leveraging ELK Stack

Song-ha-Min · Hyun-jae-Yu · Moon-ju-Lim · Jong-min Kim*

Dongsin University

E-mail : thdgk1052@naver.com / tvman1214@naver.com / topun1234@naver.com / dyuo1004@dsu.ac.kr

요 약

SQL Injection 공격은 오래된 공격기법 중 하나로 웹 서비스에 대한 해킹 시도 유형 중에서도 높은 비중을 차지하고 있다. SQL Injection 공격은 데이터 노출 및 권한획득 등의 방법으로 현재까지도 해킹 시도가 많이 발생하고 있으며, 본 논문에서는 오픈소스인 ELK Stack을 활용하여 실시간으로 SQL Injection 공격 대응할 수 있는 로그 분석시스템을 구현하였다. 구현한 시스템을 통해 SQL Injection 공격에 대한 로그 데이터를 시각화하여 제공함으로써, 사용자는 공격의 위험도를 쉽게 파악할 수 있으며 신속하게 공격에 대비할 수 있을 것으로 기대한다.

ABSTRACT

SQL Injection attacks are one of the older attack techniques and are the dominant type of hacking attempts against web services. There have been many attempts to hack SQL injection attacks by exposing data or obtaining privileges. In this paper, we implement a log analysis system that can respond to SQL injection attacks in real time using the open source ELK Stack. did. By providing a visualization of SQL injection attack log data through the implemented system, it is expected that users will be able to easily grasp the degree of attack risk and quickly prepare for attacks.

키워드

ELK stack, SQL Injection, Log Analysis System, Snort, Web Hacking

Ⅰ. 서 론

IT 환경이 계속해서 발전해 나가면서 전 세계적으로 사이버 공격이 기하급수적으로 늘고 있으며 웹 해킹 중 오래된 공격 기법의 하나인 SQL Injection은 최근까지도 웹 서비스에 대한 해킹시도의 유형에서 높은 비중을 차지하고 있다. SQL Injection은 SQL 질의 구문을 통해 비정상적인 데이터를 요청하고 처리하여, 정상적인 동작과는 다른 동작을 하게 함으로써, 고객정보 유출이나 DB 삭제 등의 방법으로 공격이 이루어지고 있다.

이러한 공격은 기업의 개인정보 유출로 통해 기업 이미지에도 큰 타격일 뿐 아니라, 개인에게 피싱과 같은 다양한 형태로 제2의 피해자를 발생시킬 수 있다.

따라서 본 논문에서는 SQL Injection에 실시간으로 대응하기 위해 오픈소스인 ELK Stack을 활용하였다. Logstash를 이용하여 로그를 수집하고 Elasticsearch을 이용해 로그를 분석한 다음 Kibana를 통해 시각화함으로써, 대시보드를 통해 트래픽 이상 유무를 빠르게 파악 할 수 있는 시스템을 제안하고자 한다.

Ⅱ. 이론적 배경

2-1. SQL Injection

SQL Injection 공격은 악성 SQL문 Injection 공격으로도 불리며, 악성 데이터 값을 이용하여 기존의

* corresponding author

웹 애플리케이션에 고정된 SQL 질의 구문을 새로운 악의적인 SQL 질의 구문으로 만들어 비정상적으로 데이터베이스에 데이터를 요청하고 처리하는 공격방법이다.

예를 들어 아래와 같은 Java 언어로 작성된 코드가 있다고 가정한다.

```
String sql = "select name from member where userid='"+userid+"' and passwd='"+ passwd +"'";
```

사용자가 아이디에 test' or 1=1-- , 비밀번호에 1234 라는 값을 입력하면 DBMS (DataBase Management System)에 아래와 같은 명령어가 전달되게 된다.

```
select name from member where userid='test' or 1=1 --and passwd='1234'
```

위의 SQL 문장의 결과는 항상 true가 되므로 잘못된 계정임에도 불구하고 로그인이 되는 문제가 발생한다.

만약 사용자가 아이디에 ';drop table contents-- , 비밀번호에 1234 라는 값을 입력하면 DB서버에 아래와 같은 명령어가 전달되게 되어 contents 테이블이 삭제되는 문제가 발생할 수 있다.

```
select name from member where userid=''; drop table contents--' and passwd='1234'
```

위와 같은 기법의 웹 응용 프로그램에 대한 SQL Injection 공격 취약성은 아직도 많은 웹 사이트들에서 발견되고 있으며 데이터베이스에 악성코드를 삽입하는 사고 사례가 발견되기도 하였다.

2-2. ELK Stack

ELK Stack은 Elasticsearch, Logstash, Kibana 및 오픈소스 소프트웨어 조합을 의미한다.

Elasticsearch는 분산 환경을 지원하는 JAVA 기반의 오픈 소스 검색엔진 라이브러리인 (Lucene1)을 기반으로 구현된 분산 검색엔진으로 설치와 서버 확장이 편리한 기술이다. Elasticsearch는 검색 기능뿐만 아니라 검색한 데이터를 집계할 수 있는 기능이 있어 분석 엔진으로도 활용할 수 있다.

LogStash는 수집할 데이터를 선정한 후 대상 서버에 전송 및 분석을 담당하며, 데이터를 전달하기 전에 반드시 분석이 용이한 형태로 데이터를 정제하는 코딩 과정을 거쳐야 한다. Logstash의 코딩 과정은 '입력-> 필터-> 출력'의 세 단계로 구성되어 있으며, 입력은 이벤트를 생성, 필터는 이벤트를 수정하고, 출력은 다른 곳으로 필터링된 데이터를 전달한다.

Kibana는 시각화를 위한 오픈소스 기술이고. 시계열 분석 등 다양한 그래프를 이용해서 가시성을 보장하고 ELK Stack을 전체적으로 구성 및 기능을 통합 관리한다. 시계열 데이터의 경우 분석한 결과를 시각화하여 각 데이터의 시계열 적인 연관성을 찾아주며, 로그 상태에서는 볼 수 없었던 특이점을 찾아내거나, 과거와 실시간으로 발생하는 로그를 동시에 시각적으로 보여주기 때문에 누적된 과거 데이터를 통해 실시간 데이터 분석을 함으로써 미래를 예측하게 해준다.

Filebeat는 로그 데이터를 전달하고 중앙화하기 위한 경량의 Producer이다. Beats는 로그 분석을 목적으로 PC 혹은 서버에 에이전트로 설치되어, End 단에서 발생하는 시스템 로그를 비롯한 각종 이벤트성 데이터를 LogStash와 ElasticSearch로 전달하는 기능을 한다. 오프소스로 가볍게 개발된 데이터 수집 및 전송용 플랫폼이며 서버에 에이전트로 설치되는 Filebeat는 지정한 로그 파일 또는 위치를 모니터링하고 로그 이벤트를 수집한 다음 인덱싱을 위해 Elasticsearch 또는 Logstash로 전달한다. 그림 1.은 ELK를 통한 시각화 과정의 구성도이다.

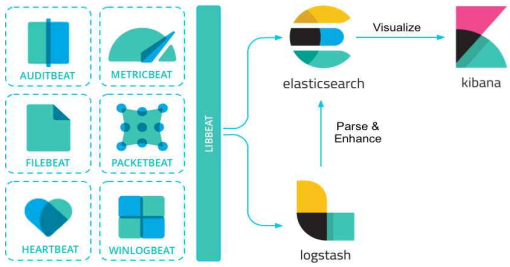

그림 1. ELK 시각화 구성도

III. 시스템 구현

3-1. 환경구성

3-1-1. ELK Stack

SQL Injection에 대해 실시간으로 대응하기 위해 ELK Stack을 구현하였으며, 표 1.은 ELK Stack에 대한 환경 구성이다.

표 1. ELK Stack 환경

구분	version
OS	CentOS 7
Elasticsearch	7.17.5
Logstash	7.17.5
Kibana	7.17.5
Filebeat	7.15.5

표 2.는 SQL Injection을 수행하기 위한 Target으로서 Web Server에 대한 환경 구성이다.

표 2. Target Web Server 환경

구분	version
OS	CentOS 7
Apache	2.2.17
MySQL	5.1.73
PHP	5.2.17

3-2. 시스템 프로세스

그림 2.는 시스템 구성도이다. 먼저 SQL Injection 공격을 진행하고 snort의 local.rules에 SQL Injection rules를 설정하여 SQL Injection 공격으로 인한 로그들을 생성하였으며, 생성된 로그들을 Filebeat로 수집하여 Logstash에 전달해 준다. 전달된 로그 데이터는 Logstash에 미리 설정한 필터링을 통해 전처리가 이루어지고, Elasticsearch에 전송 및 저장된다. Kibana는 Elasitcsearch의 실시간에 가까운 검색기능을 통해 데이터들을 가져오고, 보기 쉬운 대시보드 구성을 통해 시각화한다.

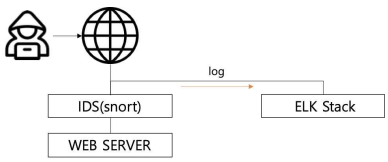

그림 2. 시스템 구성도

Ⅳ. 실험결과

SQL Injection 공격을 탐지하기 위해 Snort에 탐지 Rule을 작성하였다. 그림 3.은 local.rules에 SQL Injection 탐지 Rule을 추가한 모습이다.

```
#------------
# LOCAL RULES
#------------
#alert icmp any any -> 192.168.10.20 any (msg:"ICMP test"; classtype:icmp-event; sid:10000001; rev:001;)
#####SQL#########
alert tcp any any -> 192.168.10.20 any (msg:"SQL injection [single quote]"; content:"GET"; sid:10000070; rev:1;)
alert tcp any any -> 192.168.10.20 any (msg:"SQL Injection[union based]"; content : "%27"; content : "union"; content : "select";
content : "%23"; sid:10000070 rev:1;)
alert tcp any any -> 192.168.10.20 any (msg : "SQL Injection[union based select column_name]"; content : "%27"; content : "union";
content : "select"; content : "information_schema"; content : "column_name", sid : 10000070 rev:1;)
```

그림 3. Snort Rule

Snort를 실행하고 쇼핑몰 사이트에 SQL Injection 공격을 수행 하였으며, 그림 4.와 같다.

그림 4. 쇼핑몰 사이트 SQL Injection 공격

그림 5.는 SQL Injection 공격 시 snort에서 탐지된 로그 목록이다.

```
[**] [1:3000001:0] SQL Injection[single quote] [**] [Priority: 0] {TCP} 192.168.10.20:80 -> 192.168.10.20:80
[**] [1:3000001:0] SQL Injection[single quote] [**] [Priority: 0] {TCP} 192.168.10.20:80 -> 192.168.10.20:80
[**] [1:3000002:0] SQL Injection[union based] [**] [Priority: 0] {TCP} 192.168.10.20:80 -> 192.168.10.20:80
[**] [1:3000001:0] SQL Injection[single quote] [**] [Priority: 0] {TCP} 192.168.10.20:80 -> 192.168.10.20:80
[**] [1:3000003:0] SQL Injection[union based select table_name] [**] [Priority: 0] {TCP} 192.168.10.20:80 -> 192.168.10.20:80
[**] [1:3000005:0] SQL Injection [**] [Priority: 0] {TCP} 192.168.10.20:80 -> 192.168.10.20:80
[**] [1:3000002:0] SQL Injection[union based] [**] [Priority: 0] {TCP} 192.168.10.20:80 -> 192.168.10.20:80
[**] [1:3000003:0] SQL Injection[union based select table_name] [**] [Priority: 0] {TCP} 192.168.10.20:80 -> 192.168.10.20:80
[**] [1:3000005:0] SQL Injection [**] [Priority: 0] {TCP} 192.168.10.20:80 -> 192.168.10.20:80
[**] [1:3000002:0] SQL Injection[union based] [**] [Priority: 0] {TCP} 192.168.10.20:80 -> 192.168.10.20:80
[**] [1:3000005:0] SQL Injection [**] [Priority: 0] {TCP} 192.168.10.20:80 -> 192.168.10.20:80
[**] [1:3000002:0] SQL Injection[union based] [**] [Priority: 0] {TCP} 192.168.10.20:80 -> 192.168.10.20:80
[**] [1:3000001:0] SQL Injection[single quote] [**] [Priority: 0] {TCP} 192.168.10.20:80 -> 192.168.10.20:80
```

그림 5. SQL Injection 로그 탐지 목록

Web Server 내에 Filebeat을 설치하였으며, Web Server로부터 로그 데이터를 수집하여 Logstash로 보내 필터링하고, Elasticksearch를 통해 로그 분석을 한 다음 Kibana를 통해 로그를 시각화 하였다. 이를 통해 SQL Injection 공격에 따른 로그 발생량을 빠르게 분석할 수 있었으며, 그림 6.은 Snort에서 탐지한 로그를 Kibana에서 시각화한 것이다.

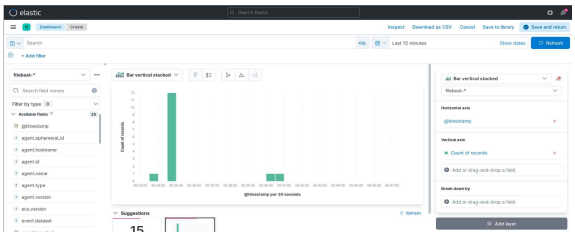

그림 6. Kibana를 통한 시각화

Ⅵ. 결 론

네트워크에서 모든 행동의 발자취인 로그 데이터는 네트워크가 존재하는 다양한 곳에 지속해서 발생하고 있다. 이 로그들은 수집 및 가공, 분석을 통해 활용 가능한 정보를 얻을 수 있는 중요한 자산이다. 이 자료를 활용하여 예상치 못한 보안 위협을 방어하고 해당 시스템의 영속성에 이바지할 수 있는 보안 로그 분석 시스템이 필요하다. 따라서 본 논문에서는 웹 모의 해킹 시 발생하는 로그들을 오픈소스인 ELK stack과 Snort를 통하여 만든 시스템을 통해 공격 로그들을 탐지하고 이를 시각화할 수 있는 보안관제 시스템을 구축해 보았다.

Acknowledgement

본 과제(결과물)는 2022년도 교육부의 재원으로 한국연구재단의 지원을 받아 수행된 지자체-대학 협력기반 지역혁신 사업의 결과입니다. (NRF-1345341782)

References

[1] J.H.Hyun, H.J.Kim, Security Operation Implementation through Big Data Analysis by Using Open Source ELK Stack, J*ournal of Digital Contents Society,* Vol. 19, No. 1, pp. 181-191, January, 2018

[2] Hyo Hyun Choi, Su Ji Kim. (2021). Development of intrusion detection System using Snort. Proceedings of the Korean Society of Computer Information Conference, 29(1), 207-208.