Journal of The Institute of Electronics and Information Engineers Vol.53, NO.2, February 2016 http://dx.doi.org/10.5573/ieie.2016.53.2.076
ISSN 2287-5026(Print) / ISSN 2288-159X(Online)

논문 2016-53-2-10

실행계획 분석을 이용한 SQL Injection 공격 대응방안

(Counter Measures by using Execution Plan Analysis against SQL Injection Attacks)

하 만 석*, 남궁 정 일*, 박 수 현*

(Man-Seok Ha, Jung-Il Namgung, and Soo-Hyun Park©)

요 약

최근 들어 급증하고 있는 보안 관련 사고들로 인하여 개인정보 및 기업정보의 관리에 대한 대책 마련이 시급한 가운데 있다. 보안 관련 사고 가운데 SQL Injection 공격은 가장 널리 악용되고, 오래된 전통적인 해킹 기법 중 하나이다. 최근까지도 웹 해킹을 시도하는 유형 중에서 높은 비중을 차지하고 있으며 그 공격 형태 또한 복잡해지고 있다. 많은 site에서 SQL Injection 공격에 대한 보완을 하여 이전보다 피해가 많이 줄어들기는 했으나 SQL Injection 공격에 의한 악의적인 관리자 권한 획득 및 비정상적인 로그인 등으로 인하여 여전히 많은 피해가 발생하고 있다. 더욱이 향후 사물인터넷 및 센서 빅데이터 환경이 널리 보급되면 수많은 디바이스들과 센서들이 연결되고 데이터의 양이 폭발적으로 증가하게 될 것이다. 그렇게 되면 현재보다 SQL Injection 공격에 의한 피해 규모는 더욱 커질 것이다. SQL Injection 공격에 대응하기 위해서는 많은 시간과 비용이 발생하게 되므로 시스템의 성능을 떨어뜨리지 않으면서도 신속정확하게 SQL Injection 공격을 판별하여 방어해야 할 것이다. 본 논문에서는 SQL Injection 공격에 대응하기 위하여 데이터 분석 및 기계학습을 통하여 웹로그 데이터를 검사하여 비정상적인 패턴의 입력값인 경우 SQL 명령어의 실행 계획을 분석하여 정상적인 SQL 명령어와 비정상적인 SQL 명령어를 판별하는 방안을 제시한다. 실험 및 성능 평가를 위해 사용자의 입력 또는 SQL Injection 공격툴에 의하여 입력되는 값을 실시간으로 실행계획을 분석하여 효과적으로 차단할 수 있음을 보여주었다.

Abstract

SQL Injection attacks are the most widely used and also they are considered one of the oldest traditional hacking techniques. SQL Injection attacks are getting quite complicated and they perform a high portion among web hacking. The big data environments in the future will be widely used resulting in many devices and sensors will be connected to the internet and the amount of data that flows among devices will be highly increased. The scale of damage caused by SQL Injection attacks would be even greater in the future. Besides, creating security solutions against SQL Injection attacks are high costs and time-consuming. In order to prevent SQL Injection attacks, we have to operate quickly and accurately according to this data analysis techniques. We utilized data analytics and machine learning techniques to defend against SQL Injection attacks and analyzed the execution plan of the SQL command input if there are abnormal patterns through checking the web log files. Herein, we propose a way to distinguish between normal and abnormal SQL commands. We have analyzed the value entered by the user in real time using the automated SQL Injection attacks tools. We have proved that it is possible to ensure an effective defense through analyzing the execution plan of the SQL command.

Keywords : 실행계획(Execution Plan), SQL Injection, 웹 해킹

Ⅰ. 서 론

SQL Injection 공격은 가장 널리 악용되고 오래된 전통적인 해킹 기법 중 하나이며 최근까지도 웹 해킹을 시도하는 유형 중에서 높은 비중을 차지하고 있으며 그 공격형태 또한 갈수록 교묘하고 복잡해지고 있다.[1~2]

SQL Injection 공격은 가장 고전적이고 다른 공격 방

* 정회원, 국민대학교 비즈니스IT전문대학원
(Kookmin University Graduate School of Business IT)
© Corresponding Author(E-mail: shpark21@kookmin.ac.kr)
※ 이 논문은 2015년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임.(NRF-2013R1A1A2012461)
Received ; December 22, 2015 Revised ; January 13, 2016
Accepted ; January 22, 2016

법에 비해 악용되기 쉬운 해킹방법이지만 아직도 여전히 많은 국내외의 웹사이트들은 SQL Injection 공격에 취약한 상태로 노출되어 있어 피해사례가 지속적으로 발생하고 있다.[3]

SQL Injection 공격은 악성 SQL문 Injection 공격으로도 불리며, 악성 데이터 값을 이용하여 기존의 웹 애플리케이션에 고정된 SQL 질의 구문을 새로운 악의적인 SQL 질의 구문으로 만들어 비정상적으로 데이터베이스에 데이터를 요청하고 처리하는 공격방법이다.[4]

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.[5~8]

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.[9~11]

Ⅱ. SQL Injection 공격 대응방안의 관련 연구

SQL Injection 공격은 주로 웹 사이트의 입력가능한 url에 get 또는 post 방식의 데이터 전달을 통하여 취약한 변수에 SQL 명령어를 삽입하는 형태로 이루어지고 있다. 또한 자동화된 툴을 이용하여 대량으로 악성 명령어를 삽입하는 공격 방법들로 계속 진화해가고 있다. 데이터베이스에 악성의 SQL 명령어를 자동화된 공격툴을 이용하여 대량으로 지속적으로 삽입하는 경우가 많다. 이러한 SQL Injection 공격들을 "Mass SQL Injection"이라고도 한다.[10]

본 논문에서는 SQL Injection 공격에 대한 대응과 연관하여 수행된 기존 연구들을 입력값 문자열 분석, 기계학습을 통한 SQL 질의 분석, 소스 코드 정적 분석, 가상의 공격 등 네 가지로 분류하였다.

1. 입력값 문자열 분석

Lashkaripour는 소스 코드의 정적 SQL 명령과 run time의 SQL 명령의 구조를 비교하여 비정상적 입력 패턴을 판별하는 방식을 제안하였다.[12] 사용자의 입력값을 제거한 후 정적 SQL 명령과 비교하여 구조가 다른 경우 비정상 입력 패턴으로 판별하게 된다.

예를 들어 select * from member where userid='"+userid+"' and passwd='"+passwd+"'; 와 같은 명령어가 있을 경우 따옴표 내부에 입력되는 변수값을 빈값으로 초기화시키면 아래와 같은 명령어로 변환된다.

select * from member where userid='' and passwd='';

그런데 select * from member where userid=''or 1=1--' and passwd='1234'; 와 같이 SQL Injection 공격이 들어올 경우 변수값을 제거한 후 다음과 같은 코드로 변환된다.

select * from member where userid=''or 1=1--''1234';

위의 코드는 select * from member where userid='' and passwd=''; 와 비교할 때 다른 구조의 명령어이므로 비정상적 패턴으로 판별하게 된다.

또한 select * from member where userid = 'a';drop table member;' and passwd=''; 와 같은 코드가 있을 경우 변수값을 제거하면 다음과 같은 코드로 변환된다.

select * from member where userid = '';drop table member;''';

이 명령어는 처음의 SQL 명령어와 다른 구조이므로 비정상적 패턴으로 판별하게 된다. 이 방법은 비정상적인 패턴이 아닌 정상적인 패턴의 SQL 명령도 항상 분석하게 되어 성능 저하 요인이 될 수 있다. 이 문제를 해결하기 위해서는 기계학습을 통하여 충분한 입력값들을 학습시켜 정상적인 패턴인 경우 검사하지 않도록 처리하는 것이 한가지 해결책이 될 수 있다.

2. 기계 학습을 통한 SQL 질의 분석

Valeur는 웹 애플리케이션에서 사용자가 입력하는 명령어 가운데 SQL 명령어가 포함된 키워드를 기계학

습에 의하여 탐지하는 방안을 제안하였다.[13] 사용자의 입력값을 사전에 학습된 모델과 비교하여 SQL Injection 공격 여부를 판단한다. 기계학습에 의한 방법은 학습 데이터셋의 분량이나 성격에 따라 다른 결과가 나올 수 있는 단점이 존재한다.

WAVES는 기계학습 방법을 이용하여 웹 애플리케이션의 취약점을 찾은 후 패턴 목록과 공격 기법들을 바탕으로 공격코드를 구성하여 SQL Injection 공격에 대한 취약점을 찾는 방안을 제시하였다.[14] 이 방법은 기존의 전통적인 공격 코드 생성 방법을 개선하기는 했지만 모든 취약점을 찾아주지는 못하는 단점이 여전히 존재한다.[4]

3. 소스 코드 정적 분석 및 안전한 SQL문 자동 생성

박주화는 자바 프로그램의 소스 코드를 분석하여 따옴표가 포함된 취약한 인라인 SQL 명령어 코드를 매개변수를 사용하는 안전한 코드로 변환하는 방안을 제안하였다.[15] 또한 따옴표의 처리가 용이하며 사용자의 입력값에 의해 SQL 문장의 구조가 변경되지 않으므로 SQL injection을 근본적으로 방지할 수 있는 장점이 있다.

4. 가상의 공격

Huanget은 웹 애플리케이션의 SQL Injection 공격에 대한 취약점을 분석하기 위하여 블랙박스 검사 방식으로 가상의 공격을 통해 취약점을 식별하는 웹 애플리케이션 보안 평가 구조를 설계하였다.[14] 이 방식은 현재 상업적으로 사용되는 방법이기도 하지만 검사를 통해서는 근본적으로 보안 취약성의 부재를 증명하지는 못하는 한계점을 가진다.[9]

기존 연구들을 종합하여 볼 때 SQL Injection 공격에 효율적으로 대응하는데 필요하다고 생각되는 요소들은 다음과 같다.

첫째, 로그 파일에 대한 분석 방식의 경우 웹서버에서 기본적으로 제공하는 로그 파일의 경우 get 방식 호출에 대해서만 분석이 가능하므로 post 방식 호출에 대해서도 분석하고자 할 경우 별도의 프로그램을 사용하거나 개발해야 한다.

둘째, 오탐지 또는 미탐지되는 비율을 낮추고 탐지율을 높이기 위해서는 런타임에서 복잡한 패턴들에 대한 검사를 실시해야 하므로 서버의 성능이 저하될 수 있다. 향후 보안 문제가 더욱 큰 이슈가 될 사물인터넷 환경이나 센서 빅데이터 환경에서 발생하는 대량의 데이터들 가운데 비정상적인 공격 패턴을 찾아내는데 많은 비용이 소모되리라 예상된다.

셋째, 미리 지정된 패턴에 대해 검사하는 경우 새로운 패턴의 공격에 대한 적응 능력이 부족할 수 있으므로 지속적인 패턴 업데이트가 필요하다.

Ⅲ. 실행계획 분석을 이용한 SQL Injection 공격 대응방안

1. 실행계획 분석의 필요성

본 논문에서는 지금까지 나열한 방법론들을 종합하여 로그 파일의 내용 및 SQL 명령어의 실행계획을 분석하여 본래의 목적과 다른 실행계획이 나오는 경우 비정상적인 SQL 명령어로 판단하여 차단시키는 방법을 제안한다.

기존 연구들을 통해 SQL Injection 공격에 안전하다고 입증된 매개변수를 사용하는 PreparedStatement 보다는 SQL Injection 공격에 취약한 따옴표가 포함된 SQL 명령문을 대상으로 연구를 수행하였다. 또한 Lashkaripour[12]의 연구에서 SQL 명령어의 구조를 비교하는 아이디어에 착안하여 SQL 명령어의 실행계획을 비교하는 방향으로 연구를 수행하였다.

본 논문에서 제안하는 방식은 로그 파일 및 SQL 명령어의 실행계획에 대한 분석을 통하여 SQL Injection 공격으로부터 시스템을 보호하는 것이다. 로그 파일에 대한 정기적인 데이터 분석 및 기계학습을 통하여 입력 데이터의 특성을 분석하여 정상적인 패턴과 비정상적 패턴을 분류한 후 실제 서비스 환경에서 사용자의 입력값에 대해 비정상적인 패턴의 값이 입력된 경우 SQL 명령어의 실행계획을 검사하여 정상적인 값이 입력된 경우와 비교하여 다른 실행계획이 나올 경우 실행을 차단시켜 시스템을 안전하게 보호하고자 한다.

모든 SQL 명령어의 실행계획을 실시간으로 일일이 검사하는 것은 처리 속도의 저하 및 서비스 성능의 저하를 가져올 수 있으므로 데이터 분석을 위해서 로그 파일을 HDFS(Hadoop Distibuted File System)에 업로드하여 Hive 테이블로 변환한 후 텍스트 및 스팸 분류 등에 많이 사용되고 있는 Naive Bayes 알고리즘을 이용하여 기계학습과정을 통해 정상적 패턴과 비정상적 패턴을 분류하였다.

사용자가 입력한 get/post 방식의 입력값을 로그 테이블에 저장하고 사용자의 입력에 대하여 코드상의 정적 SQL과 run time의 SQL 문장의 구조를 비교하여 다

표 1. 실행 계획의 사용 예시
Table 1. examples of the Execution Plan.

```
explain plan for
select * from member where userid='kim';
```

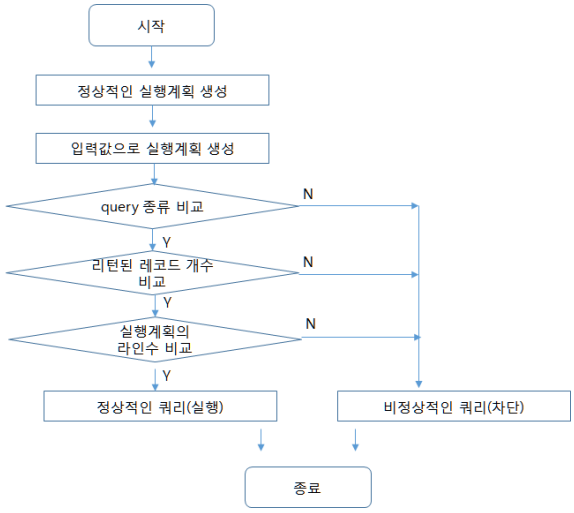

그림 1. SQL 명령어의 실행계획 분석 순서
Fig. 1. Flowchart for the Execution Plan analysis.

르면 비정상적 패턴의 입력으로 의심하여 실행되지 않도록 처리하고자 한다. 단, 이 경우 모든 SQL 명령에 대해 검사하면 서버의 성능이 떨어질 수 있으므로 취약한 inline SQL 문장인 경우에는 반드시 검사하고 사전에 데이터 분석 및 기계학습을 통해 정상적 패턴으로 분류된 명령에 대해서는 검사를 생략하고 비정상 패턴에 대해서만 검사 로직을 실행함으로써 서버의 리소스를 절약하고자 한다.

기존의 SQL Injection 공격을 분석하기 위해서는 결국 사람이 판단해야 하는 작업이 필요한 단점이 있다. 그런데 최근의 SQL Injection 공격 방법은 교묘하여 사람의 노력만으로 모두 대응하기는 어려운 부분이 있다. 또한 SQL Injection 공격에 대한 방어 방법들을 해커들이 너무나 잘 알고 있으므로 얼마든지 새로운 방법으로 우회할 수 있다. 그리고 미탐지 및 오탐지율을 낮추기 위해서는 모든 입력값에 대해 자세하고 다양한 검사를 수행해야 하는데 그렇게 되면 시스템이 느려지고 서비스의 만족도가 떨어질 수 있다. 향후 사물인터넷(Internet of Things) 환경이 대중화되게 되면 데이터의 양이 기하급수적으로 증가하고 로그 파일이 계속 커지므로 신속 정확한 분석을 위해서는 데이터 분석 및 기계학습 기법을 도입하는 것이 반드시 필요하다고 생각

표 2. 정상적인 입력값이 들어간 경우의 실행계획
Table 2. Normal Execution Plan.

```
explain plan for select * from member where
userid='kim' and passwd='1234';


Plan hash value: 3922500957
-------------------------------------
| Id  | Operation | Name | Rows  | Bytes | Cost (%CPU)| Time
-------------------------------------
|0| SELECT STATEMENT |  |  1 |  20 | 1(0)| 00:00:01 |
|*1|TABLE ACCESS BY INDEX ROWID| MEMBER | 1 | 20 | 1(0)| 00:00:01 |
|*2|INDEX UNIQUE SCAN|SYS_C007275|1| |0(0)| 00:00:01 |
-------------------------------------
Predicate Information (identified by operation id):
-------------------------------------
   1 - filter("PASSWD"='1234')
   2 - access("USERID"='kim')
```

된다.

2. SQL 명령어의 실행계획(Execution Plan)

SQL 명령어를 DBMS에서 실행하기 위해서는 몇가지 과정을 거치게 된다. 우선 SQL 문장을 파싱(Parsing)한 이 후 SQL 문법에 대한 오류를 확인한다. 데이터 딕셔너리를 통해 권한이나 객체에 대한 확인을 한 후 옵티마이저(Optimizer)는 해당 SQL 명령어를 어떻게 최적화하여 실행할 것인가에 대한 결정을 하게 되는데 이를 실행 계획(Execution Plan)이라 한다. 개발자가 실행 계획을 볼 수 있도록 제공하는 명령어는 Explain Plan 이다. Explain Plan 명령어를 통해 SQL문이 실행될 때의 상태 정보를 확인 할 수 있으며 실행계획은 실행 계획 테이블(PLAN_TABLE)에 저장된다.[16]

실행계획의 사용 예는 다음과 같다.

SQL Injection 공격은 사용자의 입력값을 비정상적인 값으로 입력하여 SQL 명령어의 구조를 교묘하게 변경시키게 된다.

마찬가지로 SQL 명령어에 입력된 사용자의 입력값을 봐서는 인코딩이 되거나 교묘한 패턴으로 입력되는 경우 또는 대량으로 입력되는 경우, Blind SQL Injection과 같이 대량으로 입력되는 경우, 오랜 기간에 걸쳐 치밀하게 준비하고 공격을 하는 경우 등에 있어서 관리자들이 육안으로 식별해 내기는 어렵다.

본 논문에서 제안하는 실행계획 비교는 그림 1과 같

표 3. 비정상적인 입력값이 들어간 경우의 실행계획
Table 3. Abnormal Execution Plan.

```
explain plan for select * from member where
userid="or 1=1 --' and passwd='1234';

Plan hash value: 3441279308
-------------------------------------
|Id|Operation |Name | Rows  | Bytes | Cost(%CPU)|Time  |
-------------------------------------
|0|SELECT STATEMENT| |3|60 |     3   (0)| 00:00:01 |
|1|TABLE ACCESS FULL|MEMBER|3|60|3  (0)| 00:00:01 |
-------------------------------------
```

표 4. 비정상적인 입력값이 들어간 경우의 실행계획
Table 4. Abnormal Execution Plan.

```
explain plan for select * from member where
userid='admin' --' and passwd='1234';

Plan hash value: 3922500957
-------------------------------------
|Id|Operation|Name| Rows  | Bytes | Cost (%CPU)| Time
-------------------------------------
|0|SELECT STATEMENT| |1 |20 |     1   (0)| 00:00:01 |
|1|TABLE ACCESS BY INDEX ROWID|MEMBER|1|20|1(0)| 00:00:01 |
|*2|INDEX UNIQUE SCAN|SYS_C007275|1| |0(0)| 00:00:01 |
-------------------------------------
Predicate Information (identified by operation id):
-------------------------------------
   2 - access("USERID"='admin')
```

이 수행된다.

표 2는 정상적인 입력값이 들어간 경우의 실행계획이다. 이 쿼리는 정상적인 값이 입력될 경우 1개의 row만을 리턴해야 한다.

표 3과 같이 userid 입력란에 비정상적인 값이 입력될 경우에는 SQL 명령어의 구조가 변경되므로 실행계획이 달라지게 되며 1개의 row가 아닌 여러 개의 row가 리턴되므로 전혀 다른 쿼리임을 알 수 있게 된다.

위의 두 가지 실행계획을 비교해 보면 결과행의 숫자가 다르게 나타나게 되며 실행계획의 라인수에도 차이가 있음을 볼 때 다른 구조의 쿼리로 변경되었음을 알 수 있다.

그런데 아래와 같이 admin' -- 이라는 값이 입력될 경우는 row와 실행계획의 라인수가 같게 된다. 하지만 이 경우에는 Predicate Information (identified by operation id)가 위에서와 달리 2만 출력되므로 다른 쿼리로 구조가 변경되었음을 알 수 있다.

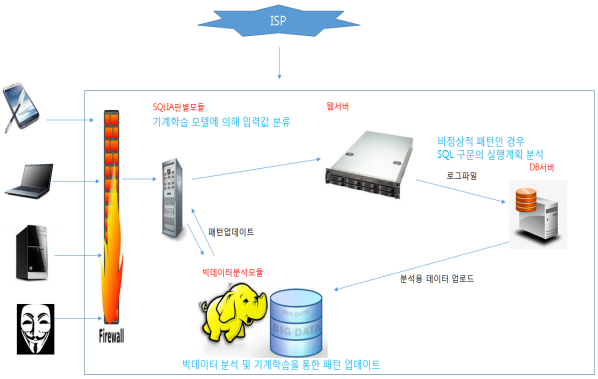

그림 2. 전체 시스템 구성도
Fig. 2. System Configuration.

IV. SQL Injection 공격 대응 프로세스 개발

1. 전체적인 구성

본 논문에서 구현하고자 하는 시스템의 전체적인 구성은 그림 2와 같다. 웹서버(상용서버)는 ISP(Internet Service Provider)와 연결되어 있다. 웹서버는 DBMS와 연동하여 웹서비스를 제공하고 있다. 사용자는 스마트폰, 노트북, 데스크탑 등의 다양한 기기를 통해 시스템에 접속할 수 있다. SQL Injection 공격을 방어하기 위해 방화벽과 웹서버 중간 단계에 SQL Injection 공격을 실시간으로 판별하여 차단시킬 수 있는 SQLIA(SQL Injection Attack) 판별 서버가 설치된다. 웹서버에 수집된 로그 정보는 데이터 분석 서버로 전달되어 데이터 분석 및 기계학습 기법을 이용하여 SQL Injection 공격의 다양한 패턴들을 인식하게 되며 수시로 공격 패턴을 업데이트하여 대응 능력을 높인다.

한편 사용자의 실시간 입력값이 들어올 경우 사전에 기계학습에 의하여 만들어진 모델과 비교하여 정상적 패턴과 비정상적 패턴을 분별하게 된다. 비정상적 패턴의 입력값이라고 판단될 경우 2차적으로 SQL 명령어의 실행계획을 검사하여 정상적인 실행계획과 비교하여 다른 경우 SQL 명령어의 실행을 차단시킨다.

시스템 관리자는 유무선 인트라넷을 통해 웹로그 파일 업로드, 웹로그 파일에 대한 데이터 분석 및 기계학습을 실시할 수 있고 취약점 분석 결과 등을 모니터링 할 수 있다. SQL Injection에 의해 계정 정보가 탈취되지 않도록 주기적으로 모니터링을 실시한다.

본 논문에서는 웹로그데이터를 데이터 분석 및 기계

그림 3. 사용자의 입력값 검증과 실행계획 분석
Fig. 3. Input validation and Execute Plan analysis.

학습을 실시한 후 방화벽과 웹서버 중간 단계에 SQL Injection 공격 판별서버를 설치하여 실시간으로 SQL Injection 공격에 대한 차단 기능을 제공하는 방안을 제안한다.

그림 3은 본 논문에서 제안하는 핵심적인 2가지 내용을 요약한 것이다. 방화벽과 기존의 대응 방법과 별도로 웹서버에서는 사용자의 입력값을 사전에 학습된 기계학습모델에 적용시켜 정상적 패턴인지 비정상적 패턴인지 분석하게 된다. 비정상적 패턴으로 분류된 경우 DB서버에서 최종적으로 실행하기에 앞서 실행계획을 분석하여 비정상적인 실행계획이 나오는 경우 실행을 차단시키는 방식이다.

SQL Injection 공격은 기존의 대응방법을 무력화시키기 위해 다양한 패턴으로 진화하고 있다. 예를 들면 아래의 예1), 예2)와 같이 연결연산자 || 를 사용하여 select와 user라는 키워드 검사를 우회할 수 있다.

예1) EXECUTE IMMEDIATE 'SEL' || 'ECT US' || 'ER'

예2) declare @x nvarchar(80);
set @x = N'SEL' + N'ECT US' + N'ER');
EXEC (@x)
EXEC SP_EXECUTESQL @x

또한 아래와 같이 16진수 인코딩을 통해서 기존의 키워드 검사 방식을 우회할 수 있다.

```
declare @x varchar(80);
set @x = 0x73656c656374204040766572273696f6e;
EXEC (@x)
```

위의 명령어들이 방화벽과 기존의 키워드 검사를 우회할 수는 있겠으나 실행 직전 단계에서 본 논문에서 제안하는 SQL 명령어의 실행계획을 분석하면 차단이 가능하다.

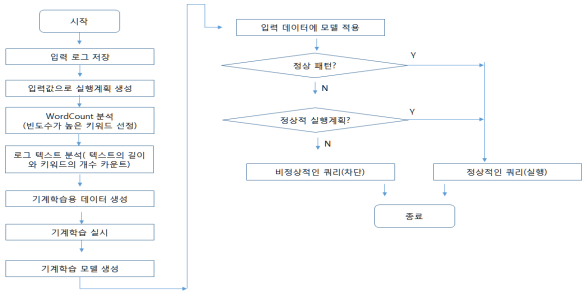

그림 4. 입력 로그 분석 절차
Fig. 4. Flowchart for log analysis.

2. 시스템의 주요 기능

시스템의 주요 기능은 크게 세 가지 부분으로 분류된다.

(1) 로그 저장모듈

웹서버에서는 get, post 방식으로 입력받은 값들을 미리 지정된 형식에 따라 데이터베이스에 저장한다. 데이터베이스에 저장된 로그 테이블은 주기적으로 csv 파일로 변환한 후 데이터 분석부에 업로드된다.

(2) 기계학습 모듈

HDFS에 업로드된 로그 파일을 Hive 테이블로 변환하여 저장한 후 Map-Reduce 및 Hive의 UDF(User Define Function)을 활용하여 빈도수가 높은 키워드들의 목록을 업데이트하고 기계학습을 위하여 입력로그텍스트의 길이 및 키워드의 갯수를 저장한다. 그리고 Naive Bayes 알고리즘을 적용시켜 기계학습을 수행하여 학습모델을 생성한 후 웹서버에 모델 파일을 업데이트한다.

(3) 입력값 판별 및 실행계획 검사 모듈

웹서버에서는 사용자의 실시간 입력값을 기계학습 모델에 적용시켜 정상 패턴인지 비정상 패턴인지를 판별한 후 비정상 패턴인 경우 2차적으로 SQL 명령어의 실행계획을 검사하여 비정상적인 SQL 명령어로 판별되면 실행을 차단시키게 된다.

3. SQL 명령어의 실행 계획 분석

웹서버에서는 사용자의 입력값을 다음과 같은 절차로 분석하게 된다.

먼저 사용자의 입력값을 기계학습을 통해 사전에 생성된 모델에 적용하여 사용자의 입력값을 정상/비정상 패턴으로 판별하게 된다. 비정상 패턴으로 판별된 경우 2차적으로 SQL 명령어의 실행계획을 검사하여 비정상

표 5. 프로그램의 실행 과정
Table 5. Running Process.

1. 사용자의 입력값을 데이터베이스에 저장
2. RDBMS에 저장된 데이터를 HDFS에 업로드한 후 Hive 테이블로 변환
3. Map-Reduce 및 Hive의 UDF을 활용하여 데이터 분석하여 기계학습에 필요한 데이터 생성
4. Naive Bayes 알고리즘을 적용하여 기계학습 실시
5. 사용자의 입력값을 정상/비정상 패턴으로 판별함
6. 비정상 패턴으로 판별된 경우 실행계획을 검사하여 비정상적인 경우 SQL 명령어의 실행을 차단시킴

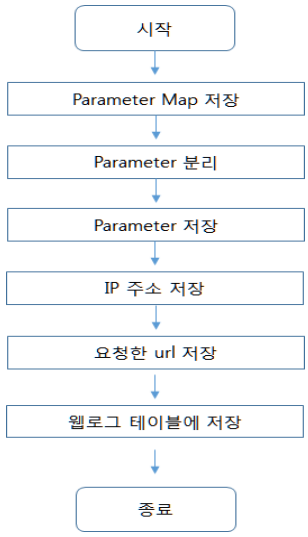

그림 5. 입력로그를 테이블에 저장하는 과정
Fig. 5. Flowchart for storing an input value.

적인 경우 SQL 명령어의 실행을 차단시키게 된다.

전체적인 실행 과정은 표 5에 나타나 있다.

입력로그를 테이블에 저장하는 과정은 그림 5에 나타나 있다. 사용자가 입력한 get, post 방식의 모든 parameter의 key와 value는 Parameter Map에 저장되어 있다. Parameter Map의 각 요소를 순회하면서 Parameter의 key와 value를 분리한다. value를 모두 모아 query_string 컬럼에 저장시키고 그 외에 필요한 ip 주소, 요청한 url 등의 정보를 테이블에 기록한다.

4. 입력 데이터의 분류를 위한 기계학습

사용자가 요청한 모든 SQL 명령어의 실행계획을 분석하는 것은 비정상적인 SQL 명령어를 가장 정확히 차단할 수 있는 장점이 있는 반면 속도의 저하 및 시스템의 성능에 지장을 초래할 수 있는 단점이 있다. 따라서 사용자가 입력한 값들을 저장해 두었다가 주기적으로 데이터 분석 기법을 활용하여 분석한 후 실제 서비스 환경에서는 의심스러운 명령어들만 필터링하여 실행계획을 분석하는 방식을 취하고자 한다.

SQL Injection 공격에 대응하기 위한 취약점을 분석하기 위한 방안의 하나로 모의 또는 실제 공격에 의한 동적검사 방안이 있다.

본 논문에서는 수집된 웹 로그 데이터를 데이터 분석 및 기계학습 기법을 이용하여 정상적 패턴과 비정상적 패턴으로 분류한 후 실제 서비스 환경에서 사용자의 입력값을 기계학습을 통해 만든 모델과 비교하여 비정상적 패턴으로 판별된 경우에 한하여 SQL 명령어의 실행계획을 분석하여 비정상적인 SQL 명령어로 분류된 경우 SQL 명령어의 실행을 차단시키도록 하였다.

SQL Injection 공격은 자동화된 툴을 이용하여 짧은 시간 안에 일정한 패턴의 로그가 자주 반복되는 경우가 많다. SQL 명령어가 곧바로 노출되는 경우도 있지만 인코딩 및 난독화 기법을 동원하여 갈수록 지능화되어 가고 있으므로 단순히 미리 입력된 패턴을 확인하여 차단하는 방법만으로는 한계가 있다. 따라서 로그 파일에 대한 데이터 분석 및 기계학습 기법을 활용하여 분류 패턴을 수시로 업데이트하여 오탐률 및 미탐률을 줄이고 급격히 진화하는 공격 패턴에 적응적인 차단 방법을 제공하고자 한다.

본 논문에서 실험에 사용한 웹로그데이터는 http://www.isi.csic.es/dataset 사이트에서 제공하는 웹로그 데이터를 정상로그와 악성로그로 구분하여 가공한 데이터를 다운로드받아 활용하였다.[17] 이 사이트에서는 정상적인 로그와 비정상적인 로그를 자바 프로그램을 활용하여 구분하였다. 본 논문에서는 그 중에서 정상적인 로그 84,000건을 활용하여 테스트하였고 비정상적인 로그는 sqlmap 툴에서 접속한 로그 84,000건을 저장하여 활용하였다. 생성된 웹로그 데이터는 DB서버에 저장되며 일정한 간격으로 데이터 분석 서버로 전달되어 Hive 테이블로 변환된다.

데이터 분석 서버에서는 전달받은 웹로그 데이터를 기계학습 알고리즘을 적용하여 생성된 패턴을 SQL Injection 공격 판별 모듈에 업데이트한다. 기계학습을 위한 알고리즘으로는 R에서 제공하는 klaR와 caret Naive 패키지를 설치하여 Naive Bayes 알고리즘을 활용하였으며 자바 프로그램에서 사용 가능한 JRI (Java R Interface) 라이브러리를 활용하여 jsp 웹사이트에서 사용자가 폼에 입력한 값이 서버로 전달될 때 미리 학습된 모델을 로딩하여 테스트하도록 설정하였다.

표 6. 빈도수가 높은 키워드 목록
Table 6. The high frequency of the keyword list.

키워드	빈도수	키워드	빈도수
insert	4438	"1"="1"	182
delay	1104	1=1	172
AND	424	1'='1'	164
FROM	352	'	158
%	204	DELETE	148
*	204	USERS	148
DROP	204	+	120
LIKE	204	|	114
SELECT	204	waitfor	94
TABLE	204	;	94
WHERE	204	OR	90

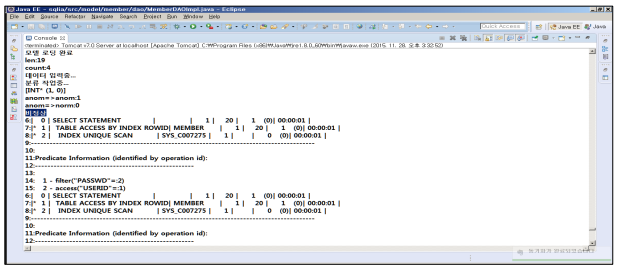

그림 6. 덧붙임 쿼리 공격에 대한 차단 예시
Fig. 6. Block Attack 1.

SQL Injection 공격 판별 서버는 클라이언트의 실시간 요청 내역에 대해 기계학습을 통해 만들어진 모델과 비교하여 비정상적 패턴으로 판별된 경우 2차적으로 실행계획을 검사하여 또한번 비정상적인 쿼리로 판별된 경우 SQL 명령어의 실행을 차단하고 안전한 패턴의 요청만 실행시킨다.

실험을 위하여 네임노드 1대와 데이터노드 4대 총 5대로 구성된 Hadoop 클러스터를 구성하고 설치하였다.

Hadoop 파일시스템에 분석용 웹로그 파일을 업로드하였다. 로그 파일을 쉼표(,)를 기준으로 나누고 사용자의 입력 내용이 들어있는 컬럼을 대상으로 WordCount 분석을 수행했다.

위와 같이 샘플 데이터셋의 웹로그 파일 내용과 sqlmap을 통해 얻은 로그를 종합하여 word count를 실행하여 실제로 SQL Injection에 관련되어 웹로그파일에서 자주 나타나는 키워드들의 목록을 작성하였다. 대체로 SQL 및 PL/SQL(Procedural Language extension to SQL) 명령어에 사용되는 연산자들과 예약어들이 주로 집계되었다.

다음으로 사용자의 입력값의 특성을 판별하기 위해서 문자열의 길이와 키워드의 빈도수를 계산하였다.

표 7. 분석 대상 키워드의 목록
Table 7. List of keywords to check.

delay	waitfor	cmd	and
from	exec	select	where
like	drop	table	or
dir	ls	delete	password
include	insert	into	values
union	all	update	set users
* % # $, ( ) < > - ; = '			

Hive의 UDF 기능을 활용하여 Hive에서 자바 함수를 호출하여 Map-Reduce를 대체하여 분석하였다.

최종적으로 작성한 키워드의 목록은 다음과 같다. 검사할 키워드들의 목록은 로그 파일을 주기적으로 분석하여 업데이트하는 것이 필요하다.

다음으로 분석할 로그 파일을 Hive 테이블에 저장하기 위하여 테이블을 생성한 후 load data 명령어를 실행하여 csv 파일의 내용을 Hive 테이블로 import 하였다.

다음으로 텍스트의 길이와 텍스트에 포함된 키워드의 갯수를 계산하기 위해서 Hive에서 프로그래밍 작업이 필요한 경우 사용하는 UDF(User Defined Function) 클래스를 작성하여 호출했다.

이 과정을 통해 생성된 csv 파일을 R에서 분석하여 기계학습을 수행한 후 학습모델을 생성하였다. 기계학습은 JRI 라이브러리를 활용하여 자바 코드상에서 R 명령어를 수행하여 수행하였다.

V. 실행계획 분석 및 공격 차단 실험

실험을 위하여 구축된 웹서버에 대하여 SQL Injection 모의 공격 실험을 수행하였다.

모의공격은 다섯 가지 항목에 대해 테스트되었으며 SQL Injection 공격에 대한 취약점을 분석하기 위해 가장 많이 사용되는 툴 중의 하나인 sqlmap과 jSQL 2가지의 툴을 사용하여 수행하였다.

1. sqlmap을 이용한 모의 공격1(덧붙임 쿼리 공격)

대상 url : login.do?userid=1

명령어 : sqlmap.py -u
"http://192.168.4.92/sqlia/login.do?userid=a"
stacked-test -v 3

그림 6과 같이 Naive Bayes 알고리즘에 의해 학습된 모델에 사용자의 입력값을 검증하여 비정상적인 패턴으

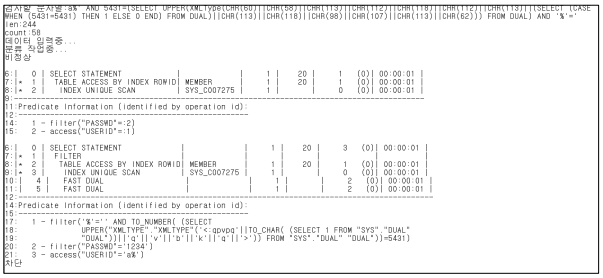

그림 7. 문법적/논리적 오류 쿼리 공격에 대한 차단 예시

Fig 7. Block Attack 2.

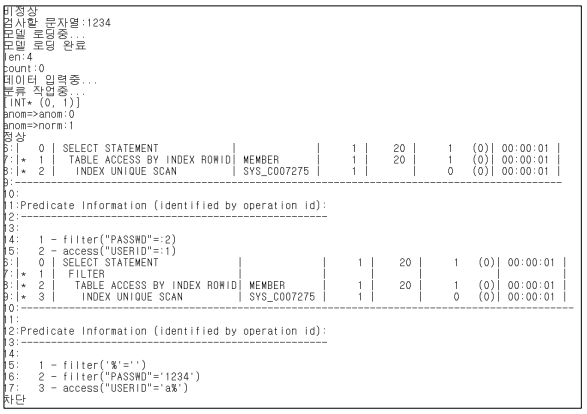

그림 8. Fingerprint 공격에 대한 차단 예시

Fig. 8. Block Attack 3.

로 분류한 후 SQL 명령어의 실행계획을 분석하여 비정상적인 SQL 문장으로 분류하여 실행을 차단하도록 하였다.

2. sqlmap을 이용한 모의 공격2(문법적/논리적 오류 쿼리 공격)

대상 url : login.do?userid=1

명령어 : sqlmap.py -u “http://192.168.4.92/sqlia/login.do?userid=a” time-test -v 3

그림 7과 같이 문법적/논리적 오류 쿼리 공격이 들어올 때 입력값을 기계학습모델에 검증한 결과 비정상적인 패턴으로 분류되어 실행계획을 검사했을 때 다른 실행계획임을 확인하고 실행을 차단하였다.

3. sqlmap을 이용한 모의 공격3(Fingerprint)

대상 url : login.do?userid=1

명령어 : sqlmap.py -u “http://192.168.4.92/sqlia/login.do?userid=a” --fingerprint -v 3

표 8. 실행계획 검사 실험 결과(sqlmap)

Table 8. Experiment result.(sqlmap)

Test name	공격 횟수	차단 횟수	차단률
Test 1	156	156	100%
Test 2	199	199	100%
Test 3	187	187	100%
Test 4	182	182	100%
평균	181.00	181.00	100%

표 9. 실행계획 검사 실험 결과(jSQL injection tool)

Table 9. Experiment result.(jSQL)

Test name	공격 횟수	차단 횟수	차단률
Test 1	426	426	100%
Test 2	416	416	100%
Test 3	468	468	100%
Test 4	480	480	100%
Test 5	436	436	100%
평균	445.20	445.20	100%

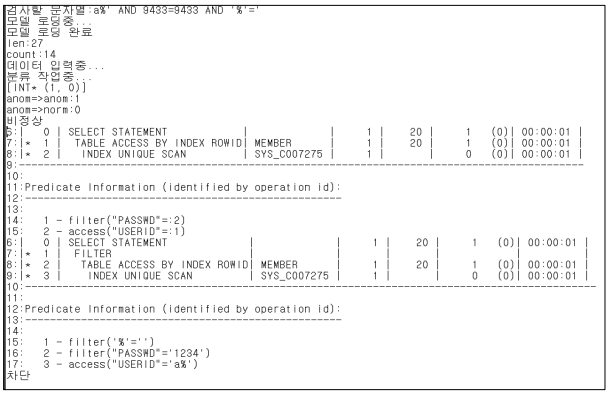

그림 9. Enumerate available databases 공격 차단 예시

Fig. 9. Block Attack 4.

4. sqlmap을 이용한 모의 공격4(Enumerate Available Databases)

대상 url : login.do?userid=1

명령어 : sqlmap.py -u “http://192.168.4.92/sqlia/login.do?userid=a” --dbs -v 3

이상 4가지 모의 공격 및 차단 실험을 실시한 결과 표 8과 같은 성능을 보여주었다.

두 번째로 jSQL Injection tool을 이용하여 테스트하였다. 이 툴은 2015년 12월 현재 http://github.com 사이트에서 SQL Injection 및 파일업로드, 웹쉘 공격 등 다양한 웹해킹 취약점에 대해 점검 기능을 제공하는 툴 중에서 가장 다운로드 횟수가 많은 툴이다.[18]

jSQL 툴을 이용하여 모의 공격 및 차단 실험을 실시

한 결과 표 9와 같이 우수한 성능을 보여주었다.

sqlmap과 jSQL 툴을 이용하여 실험한 결과 100%의 차단률을 얻을 수 있었는데 이는 기존 연구에서 제시된 다른 방안들을 전혀 사용하지 않고 SQL 명령어들을 실행하기 직전에 실행계획만을 비교하여 차단한 것이기에 더욱 큰 의미가 있다. 본 논문에서 제시한 방법을 기존의 연구에서 제시된 대응 방안들과 더불어 종합적으로 활용하게 되면 실제 환경에서 다양한 변종의 SQL Injection 공격에 대해서도 충분히 대응이 가능하리라고 생각된다.

VI. 결 론

본 논문에서는 SQL Injection 공격에 대응하기 위하여 SQL Injection 공격에 대응하기 위하여 웹로그 데이터를 검사하여 비정상적인 패턴의 입력값인 경우 SQL 명령어의 실행 계획을 분석하여 정상적인 SQL 명령어와 비정상적인 SQL 명령어를 판별하는 방안을 제시하였다.

이를 위해 SQL Injection 공격에 관련된 연구들을 살펴보고 본 연구와 밀접한 관계를 갖는 연구들에 대하여 살펴보았다.

실험 및 성능 평가를 위해 웹서버에서 수집한 로그 데이터를 데이터 분석 서버에 업로드하여 Hive의 테이블로 변환하고 Map-Reduce를 활용하여 데이터 분석 및 JRI 라이브러리를 활용하여 R에서의 기계학습을 통하여 입력값의 패턴을 분류한 후 사용자의 입력 또는 SQL Injection 공격툴에 의하여 입력되는 값을 실시간으로 실행계획을 분석하여 효과적으로 차단할 수 있음을 보여주었다.

sqlmap과 jSQL 툴을 이용하여 실험한 결과 100%의 차단률을 얻을 수 있었는데 이는 기존 연구에서 제시된 다른 방안들을 전혀 사용하지 않고 SQL 명령어들을 실행하기 직전에 실행계획만을 비교하여 차단한 것이기에 더욱 큰 의미가 있다.

SQL Injection 공격은 다른 공격방법보다 쉽게 악용이 가능하고 대응 방안이 이미 잘 알려져 있으므로 종합적인 대응방법을 사용하는 것이 효과적이라고 생각된다. 본 논문에서 제안하는 방식을 단독으로 사용하기보다는 기존의 대응 방안과 함께 보완적인 방안으로 사용하면 더욱 효과적이라고 생각된다.

본 논문에서 제시한 SQL Injection 공격 대응 방안은 다음과 같은 한계점 및 향후 연구과제를 갖는다.

본 논문에서는 Java에서 Oracle 데이터베이스에 연동하는 환경에서 테스트하였다. 따라서 본 논문에서 실험을 위해 사용한 Java와 Oracle에 종속적인 단점이 있다. Java 외의 다른 언어와 오라클 외의 다른 데이터베이스에서도 실행계획의 출력 방법 및 그것을 가져오는 방법이 비슷하므로 충분히 적용이 가능하지만 세부적인 방법이 상이하므로 향후 표준화된 인터페이스를 제공해야 할 필요가 있다.

본 논문에서 SQL Injection 모의 공격에 사용한 로그인 관련 로직에는 비교적 짧은 데이터가 들어가기 때문에 입력 내용이 많은 다른 url에 공통적으로 적용하기는 어려움이 있다. 실제 서비스 환경에서는 입력 데이터의 양이나 특성에 따라 별도로 기계학습을 실시한 후 각각의 입력 데이터의 성격에 맞는 학습모델을 적용해야 할 필요가 있다.

또한 본 논문에서 사용한 웹로그 데이터는 실제 환경에서 분석을 하기에는 적은 양의 데이터를 사용하였다. 향후 실제 환경에서 사용되는 대량의 데이터를 활용하여 적용할 예정이다. 또한 다양한 기계학습 알고리즘을 적용하고 최근 급격한 성능 향상으로 주목을 받고 있는 딥 러닝 기법 등을 적용한 연구 등을 진행하고자 한다.

REFERENCES

[1] JUSTIN CLARKE, “SQL Injection : Attacks and Defense”, Elsevier Inc. 2009
[2] Seong-Ho Choi, "Defense method against SQL injection attack by hacking tools", Graduate School of Information&Technology, Sogang University, 2010
[3] M.Amutha Prabakar, M.KarthiKeyan, K. Marimuthu, “An Efficient Technique For Preventing SQL Injection Attack Using Pattern”, IEEE Computer Society, 2013.5
[4] In-Yong Lee, Jae-Ik Cho, Kyu-Hyung Cho, Jong-sub Moon, "A Method for SQL Injection Attack Detection using the Removal of SQL Query Attribute Values", Journal of The Korea Institute of Information Security and Cryptology Vol.18, No.5, pp.136-140, 2009.
[5] Jeom-Goo Kim, Si-Choon Noh, "A Study of Step-by-step Countermeasures Model through Analysis of SQL Injection Attacks Code", Journal of information and security Vol.12, No.1, 2012
[6] Lwin Khin Shar, Hee Beng Kuan Tan, “Defeating SQL Injection”, IEEE Computer

Society, 2013.3
[7] Mun-Joo Lee, “A study on the intrusion detection and countermeasure system for mass SQL injection”, Graduate School of Information, Chung-ang University, 2011
[8] Debabrata Kar, Suvasini Panigrahi, “Prevention of SQL Injection Attack Using Query Transformation and Hashing”, IEEE Computer Society, 2012.3
[9] Joon-Seon Ahn, Yeong-Min Kim, Jang-Wu Jo, “Development of a String Injection Vulnerability Analyzer for Web Application Programs”, The KIPS transactions. Part A. Vol.15-A, No.3, pp.181-182, 2008
[10] Korea Information Security Agency, “Research on inserting malicious code through an automated SQL Injection Attack”, 2008
[11] Incheon Paik, “Improving Malicious Web Code Classification with Sequence by Machine Learning”, IEIE Transactions on Smart Processing & Computing Vol.3 No.4, pp.319-324, 2014.10
[12] Z. Lashkaripour, “A Simple and Fast Technique for Detection and Prevention of SQL Injection Attacks”, International Journal of Security and Its Applications Vol.7, No.5, pp.53-66, 2013
[13] F. Valeur, D. Mutz, G. Vigna, “A Learning-Based Approach to the Detection of SQL Attacks”, In Proceedings of the Conference on Detection of Intrusious and Malware and Vulnerability Assessment, pp 123-140, 2005.
[14] Huang.Y, Huang.S, Lin.T, Tasi.C, “Web application security assessment by fault injection and behavior monitoring”, In Proceedings of the 12th international Conference on World Wide Web, pp.148-159, 2003
[15] Ju-Hwa Park, “A study of the preventive measures to counteract the SQL injection attacks on the web programed in Java”, Grauduate School of Sungkyunkwan University, 2010
[16] Jong-Cheol Lee, “Oracle Query tuning method using a Hint”, Graduate School of Technology Management, Kyunghee University, 2008
[17] http://users.aber.ac.uk/pds7/csic_dataset/csic2010http.html
[18] https://github.com/ron190/jsql-injection
[19] Young-Sun Kim, Choon-Weon Seo, “Design and Implimentation of Intrusion Detection System on Contents Security”, Journal of The Institute of Electronics and Information Engineers Vol.52, No.11, pp.164-168, 2015

저 자 소 개

하 만 석(학생회원)
1997년 국민대학교 경제학과 학사
2002년 국민대학교 정보관리학과 석사
2015년 국민대학교 비즈니스IT 전문대학원 박사과정 수료

<주관심분야 : 객체지향 프로그래밍, 모바일 프로그래밍, 데이터베이스>

남궁 정 일(정회원)
1995년 인천대학교 기계공학과 공학사
2005년 국민대학교 비즈니스정보통신 이학석사
2011년 국민대학교 비즈니스IT 전문대학원 이학박사
2014년~현재 국민대학교 금융정보보안학과 연구교수
<주관심분야 : M2M/IoT, 인공지능, 극한환경 통신프로토콜 등>

박 수 현(정회원)
1988년 고려대학교 컴퓨터학과 학사
1990년 고려대학교 전산학과 석사
1998년 고려대학교 컴퓨터학과 박사
2002년~현재 국민대학교 정보시스템전공 교수
<주관심분야 :유비쿼터스 네트워크, Underwater, Sensor Network>