KIPS Tr. Comp. and Comm. Sys.
Vol.4, No.3 pp.97~104 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2015.4.3.97

Design of the MS-SQL Password Vulnerability Checking Function Using OLE Remote Connection

Seung Ju Jang†

ABSTRACT

This paper will feature designs for security vulnerability based on MS-SQL Database and OLE connectivity by checking the MS-SQL database password policy, the user account password access attempts, a user without password, and password does not be changed for a period of time. This paper uses the MS-SQL database and C++ linkage in order to use the OLE DB function. The design module should judge presence or absence of security vulnerability by checking database password policy, the user account password access attempts, a user without password, password does not be changed for a period of time. The MS-SQL database password associated with a feature, judging from the many features allows you to check for security vulnerability. This paper strengthen the security of the MS-SQL database by taking the advantage of the proposed ability.

Keywords : Vulnerability, MS-SQL Database, Password Vulnerability, Password Checking, OLE Remote Connection

OLE 원격 접속 기능을 이용한 MS-SQL 패스워드 취약점 점검 기능 설계

장 승 주†

요 약

본 논문은 MS-SQL 데이터베이스의 환경과 OLE 원격 접속 기능, C++ 환경을 바탕으로 MS-SQL 데이터베이스 암호정책과 사용자 계정 패스워드 접속 시도, 패스워드 없는 사용자 계정, 패스워드를 일정 기간 이상 변경하지 않은 경우 등에 대한 점검을 통해서 보안취약점 기능을 설계한다. MS-SQL 데이터베이스와 C++의 연동을 위해서는 OLE DB 기능을 사용한다. OLE DB 연동을 통해 계정마다 암호정책 강제 적용의 유무를 확인하고, 계정별 패스워드 접속 실패 유무, 패스워드 없는 사용자, 패스워드를 일정 기간 이상 변경하지 않은 경우 등을 종합적으로 판단하여 보안취약점 유무를 판단한다. MS-SQL 데이터베이스 패스워드 기능과 관련하여 여러 가지 기능들을 판단해서 보안취약점을 점검할 수 있도록 한다. 본 논문에서 제시하는 기능을 활용하여 MS-SQL 데이터베이스 보안을 강화하고자 한다.

키워드 : 보안취약점, MS-SQL 데이터베이스, 패스워드 보안취약점, 패스워드 체크, OLE 원격 접속

1. 서 론

IT 기술이 발전함에 따라 사용자들의 데이터베이스 정보 보호가 더욱 중요시되고 있다. 최근 데이터베이스 사용자 정보 유출 사고가 자주 발생하고 있다. 데이터베이스 정보의 보호에서 가장 중요한 점은 사고가 발생한 후가 아닌 발생하기 전에 미리 예방과 보호를 하는 것이다. 사고 발생 후 데이터베이스 정보 복구는 어느 정도 가능하지만 이미 손실된 데이터는 복구하기가 힘들거나 불가능한 경우가 많다. 그러므로 사용자 정보를 보호하기 위해서는 주기적인 점검을 통하여 보안취약점이 존재하는지를 확인하는 것이 필요하다. 본 논문에서는 MS-SQL 데이터베이스의 암호정책 강제 적용과 패스워드 실패 횟수를 확인하여 만약 기준점 이상 실패 횟수가 있을 경우, 사용자들이 손쉽게 데이터베이스 정보를 보호할 수 있도록 경고 메시지를 보내주는 기능을 모듈로 개발하고자 한다.

본 논문에서는 MS-SQL 데이터베이스에서 패스워드 취약점 점검 기능을 설계한다. MS-SQL 데이터베이스의 암호정책 강제 적용 상태와 SQL Server 로그인 접속 패스워드

※ 이 논문은 2014학년도 동의대학교 교내연구비에 의해 연구되었음(과제번호 : 2014AA018).
† 정 회 원 : 동의대학교 컴퓨터공학과 교수
Manuscript Received : August 25, 2014
First Revision : November 20, 2014; Second Revision : December 17, 2014
Accepted : December 17, 2014
* Corresponding Author : sjjang@deu.ac.kr

실패 횟수가 특정 기준점 이상 점검, 패스워드 없는 사용자 체크, 패스워드 사용 기간 체크 등을 종합적으로 체크하여 보안취약점이 존재할 가능성을 점검한다. 본 논문은 MS-SQL 데이터베이스에서 암호정책 강제 적용 여부 등이 설정되었는지를 우선 점검하게 된다. 암호정책 강제 적용이 되어있도록 설정되었을 경우 데이터베이스가 보다 안전하다고 판단할 수 있다. 또한, MS-SQL 보안취약점이 있다고 판단하기 위한 기능으로 패스워드 접속 실패 횟수를 점검한다. 이 기능은 악의적인 사용자가 MS-SQL 데이터베이스에 접속을 시도했다고 판단할 수 있다. 그리고 패스워드 없는 사용자 계정을 체크한다. 이 경우는 MS-SQL 데이터베이스에 치명적인 보안 약점이 될 수 있다. 또한, 패스워드 사용 기간 체크를 통해서 오랜 기간 동안 패스워드 변경 등이 없을 경우 외부 공격에 노출될 가능성이 있다. 본 논문은 MS-SQL 데이터베이스에서 외부 공격 등에 취약할 수 있는 부분들을 사전에 점검하여 보안을 강화하고자 개발되었다.

본 논문의 구성은 다음과 같다. 2절에서 관련 연구를 언급하고, 3절에서는 설계 시스템 환경에 대해 설명한다. 4절에서는 암호정책 강제 적용 및 패스워드 실패 횟수를 이용한 보안취약점 점검 기능을 설계한 내용을 설명한다. 5절에서는 실험 내용을 설명하고, 6절에서는 결론을 내린다.

2. 관련 연구

사이버 공격이 확산됨에 따라 데이터베이스 보안이 더욱 중요한 문제로 대두되고 있다. 특히 데이터베이스는 중요한 정보들을 보관하고 있기 때문에 사이버 범죄의 주요 표적이 되고 있으며, 해커들은 서버를 파괴하고 데이터베이스를 빼앗아 이익을 얻으려 한다.

보안취약점(vulnerability)이란 보안정책 위반으로 귀결될 수 있는 보안 절차, 설계, 구현 단계에서의 결점(weakness)으로, 보안 사고를 유발하는 근원이다. 정보기술이 발전하면서 IPTV, 스마트폰 등의 신기술 활용이 일반화되고 소셜미디어(SNS), 모바일 행정서비스 등의 신규 서비스도 빠르게 확산되어감에 따라 기술 적용 및 서비스 보급 확대 과정에서 새로운 보안취약점들이 나타난다. 이들 취약점들을 악용한 개인정보 침해 및 도용, 서비스 거부 등의 침해 사고가 발생할 가능성이 더욱 증대되었다. 국내의 경우 국가 차원의 취약점 분석이나 DB(Database) 관리 등의 보안취약점 관리체계가 구축되어있지 않아 침해사고에 따른 큰 피해가 발생될 수 있다. 이러한 침해사고에 대응하고 피해를 예방하기 위해서는 국내 환경에 적합한 보안취약점 관리체계를 구축하는 것이 중요하다[1-3].

DB(Database) 보안은 외부 혹은 내부자로부터 개인 또는 조직의 주요기밀 정보 유출에 대한 방어를 목적으로 한다. DB 보안의 위협들은 사용자의 실수, 오용, 내부자의 권한 남용, DB에 대한 알려진 취약점 등으로부터 기인하게 된다. 오늘날의 정보 자산은 무형의 자산에 그치지 않고 실질적인 재화이다. 이러한 정보 자산 가치의 증대로 인하여 주요 정보가 집중되어있는 DB에 대한 위협은 날로 증가하고 있다[2, 9-13].

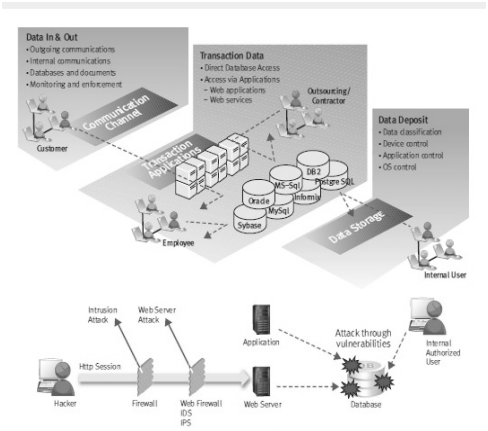

Fig. 1. MS-SQl DB Access Path

많은 매체들을 통하여 중요한 개인정보를 활용하는 서비스들이 많아지면서 개인정보들이 무분별하게 사용되고 있다. 해킹이나 내부 인가자의 유출에 의한 사건사고가 증가하면서 지난 2011년 우리나라에서만 약 1억 건의 개인정보가 유출되었다고 한다. 개인정보 유출을 막기 위해 정부에서도 개인정보보호법을 제정하여 개인정보 관리의 중요성을 알리고 있으며, 개인정보를 많이 가지고 있는 대기업, 공공기관 등에서도 점차 관심을 가지고 대응하고 있다[1-3].

본 논문에서는 데이터베이스의 보호를 위해 개발된 모듈을 통하여 MS-SQL 암호정책 적용 유무와 패스워드 실패 횟수를 통하여 보안취약점이 존재할 경우 사용자에게 경고메시지를 보내 보안의 문제점을 미리 예방하고 해결한다[4-6]. 아래 Fig. 2는 DB 보안취약점 유형을 정리한 내용이다.

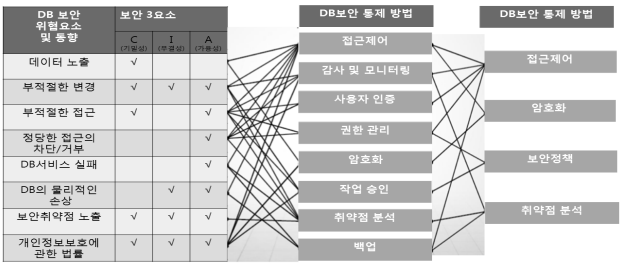

Fig. 2. DB Vulnerability Factory

Fig. 2에서 DB 보안 위협 요소들에서 보안취약점 노출이나 부적절한 접근 등이 본 논문에서 제안하는 DB 보안과 관련한 부분이라고 할 수 있다.

3. 시스템 설계 환경

MS-SQL 암호 복잡성 정책은 가능한 암호의 수를 늘려 문자 조합을 이용한 공격(brute force attacks)을 방지하도록

설계한다. 암호 복잡성 정책이 강제 적용된 경우 새 암호는 다음 지침을 준수해야 한다. 암호는 사용자의 계정 이름을 포함하지 않고 암호의 길이는 최소 8자 이상이어야 하며 암호는 다시 4가지 범주 중 세 범주의 문자를 포함해야 한다. 라틴어 대문자(A-Z), 라틴어 소문자(a-z), 기본 숫자 10가지(0-9), 느낌표(!), 달러 기호($), 숫자 기호(#) 또는 퍼센트(%)와 같은 영·숫자 이외의 문자이다. 암호 길이는 128자까지 가능하며 되도록 길고 복잡한 암호를 사용해야 한다.

암호 만료 정책을 사용하여 암호의 수명을 관리한다. MS-SQL Server에서 암호 만료 정책을 강제로 적용하면 사용자에게 기존 암호를 변경할 것과 암호가 만료되어 해당 계정을 사용할 수 없게 됨을 알려준다.

암호정책 적용은 각 MS-SQL Server 로그인마다 별도로 구성할 수 있다. ALTER LOGIN(Transact-SQL)을 사용하여 MS-SQL Server 로그인의 암호정책 옵션을 구성할 수 있다[4-7].

OLE DB(Object Linking and Embedding DataBase)는 RDBMS(Relational DataBase Management System)에 접근할 수 있는 통로를 제공하는 역할을 담당하는 기술이다. 그리고 마이크로소프트의 UDA(Universal Data Access) 전략을 구현하는 기술 중 하나이다. UDA는 광범위한 데이터들에 접근할 수 있도록 하는 마이크로소프트의 기술로 UDA 전략이라고도 표현한다. 이것은 기술 전략적인 이름일 뿐 실제로 이를 구현하는 것은 ADO(ActiveX Data Objects)와 OLE DB이다. OLE DB를 사용하여 광범위하게 위치한 데이터들을 처리하게 하는 기술이 UDA이다. ODBC(Open DataBase Connectivity)는 이 기종의 관계형 데이터베이스에 접근할 수 있게 해주지만 OLE DB는 이 기종의 관계형 데이터베이스뿐만 아니라 비관계형 데이터들에게도 접근할 수 있게 한다. 즉, MS-SQL 서버나 오라클과 같은 RDBMS뿐만 아니라 메일 폴더, 각각의 메일 자체, 디렉토리, 웹사이트 등 비관계형인 계층 구조적인 데이터들에도 접근할 수 있게 한다.

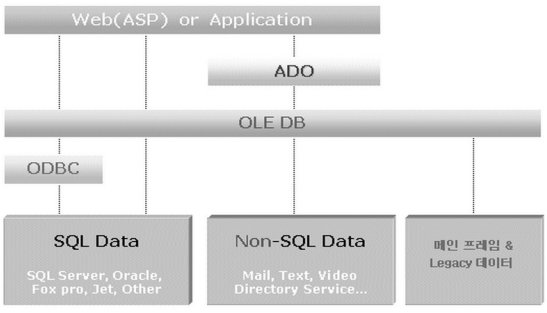

Fig. 3. MS-SQL UDA System Architecture

UDA 기술은 ADO와 OLE DB를 사용하여 SQL 데이터뿐만 아니라 비SQL적인 데이터들에도 접근을 가능하게 한다. 데이터를 처리할 수 있고 OLE DB를 통해 기존의 ODBC를 통한 데이터의 연결도 가능하다. 이처럼 OLE DB를 통한 접근은 기존 ODBC의 부족했던 점을 완벽히 보안하며, ODBC에 비해 접근 속도 면에서도 많은 향상을 보이고 있다. 기존 기술을 더욱 발전시키자는 큰 목적을 두고 있는 것이 OLE DB 기술이다[5-9].

4. OLE 원격 접속을 통한 패스워드 보안취약점 설계

기존의 데이터베이스 보안을 위해서는 데이터베이스의 암호화 기법이 일반적으로 많이 사용되고 있다. 데이터베이스 암호화 기법에는 모든 데이터베이스에 대한 파일 수준 암호화와 애플리케이션 수준 암호화가 있다. 하지만 데이터베이스 암호화는 몇 가지 단점이 있다. 인덱스나 키 필드를 암호화하는 경우 성능에 큰 악영향을 미치게 되고, 응용 계층에서 수정을 해야 하는 경우, 비용이 추가될 수 있다. 또한 범위나 부분 검색을 요청하는 경우, 모든 암호 데이터가 복호화 되어야 하기 때문에 심각한 성능 저하가 생기게 된다. 이와 같이 데이터베이스 암호화는 아직은 단점이 더 많아서 기술적으로 사용하는 데 한계가 있다[13]. 본 논문에서는 기존 데이터베이스 보안 방법의 문제점에 대한 대안으로 OLE 원격 접속을 통한 패스워드 보안취약점 설계 기능을 통하여 근원적인 보안취약점을 없애도록 한다.

본 논문은 이러한 문제점을 개선하여 데이터베이스의 원격 접속을 통한 계정들의 패스워드 취약점을 점검하여 악의적인 의도를 가진 사용자가 해킹을 하는 것을 방지 및 예방을 하는 정보 보호에 목적을 둔다. 원격으로 데이터베이스 계정들의 암호정책 강제 적용 점검을 하여 제어함으로써 정보 보호를 하고, 추가적인 비용이 들지 않는다. 그리고 계정마다 패스워드 상태를 점검하여 실패 횟수 제한 기능을 구현했기 때문에 성능 저하가 없고, 악영향을 미치지 않는다.

본 논문에서 제안하는 설계 내용은 Microsoft Visual Studio 2008의 C++, Microsoft SQL Server 2008 R2의 환경에서 개발된다. 서버 접속 구성은 SQL Server OLE DB를 이용하여 해당 서버에 접속한다. MS-SQL 쿼리문을 이용하여 is_policy_checked로 암호정책 강제 적용 유무를 판단하고, BadPasswordCount를 이용하여 반복적인 패스워드 실패 횟수를 알 수 있다. 그리고 패스워드 없는 사용자 계정, 패스워드 사용 기간 판별 등을 쿼리문으로 작성한 후 C++로 OLE DB를 이용한 데이터베이스 서버 환경을 구축한다.

```
char Cnnstr[200]="Provider=SQLOLEDB.1;Persist Security Info=False;"\
                 "User ID=sa; Initial Catalog=master;Data -\
                 "Source=220.73.230.252;
.....
main()
{
        Database db;
        Table tbl;
        if(!db.open("sa", "1234", CnnStr);
        {
                print error message;
        }
        if(!db.Execute("쿼리리문 입력", tbl))
        {
                print error message for query;
        }
        return 0;
}
```

Fig. 4. Remote Access Source Code in the MS-SQL using OLE Feature

OLE DB를 이용해서 MS-SQL에 연결하는 방법으로 MS-SQL Server는 로컬에 설치되어있는 것으로 사용할 데이터베이스명을 정하고, 접근하기 위한 사용자 아이디와 패스워드를 생성한다. 사용 환경은 Fig. 4와 같다. Provider는 OLE DB의 인터페이스를 구현하고 있는 컴포넌트이다. 여기서 MS-SQL Server에 연결하기 위하여 Provider는 SQLOLEDB를 적용한다. Persist Security Info는 SQL Server 인증을 가지고 로그인 정보로 데이터베이스 서버에 연결하는 것이며 로그인 정보가 ConnectionString 내에서 사용되면 자동적으로 설정되는 사항이며 생략 가능하다. User Id(UID)는 데이터베이스에 연결하고자 하는 사용자의 이름이고, Password(PWD)는 사용자의 패스워드를 나타낸다. 본 논문에서는 SQL Server의 인증을 받은 'sa'이라는 관리자(로그인명)가 있고 그것의 패스워드가 '1234'이다. Initial Catalog=master는 사용자 정보로 연결을 해서 사용할 수 있는 기본적인 데이터베이스 이름이다. 이 속성은 MS-SQL Server가 데이터베이스 위주의 스키마를 가진 RDBMS이기 때문에 설정 가능한 항목이며 Database=master;로 해도 된다. 마지막으로 Source는 해당 주소로 접근을 가능하게 한다.

본 논문에서는 OLE DB를 이용하여 MS-SQL 데이터베이스의 SQL Server 사용자 계정 암호정책 강제 적용을 권고한다. 만약 암호정책 강제 적용이 되어있지 않다면 보안 취약점이 있는 것으로 판단한다. 디폴트 ID를 제외한 모든 계정을 검사하여 암호정책 강제 적용 유무를 체크하고, 계정마다 패스워드 실패 횟수를 확인하여 로그인 접속 패스워드 실패 횟수가 일정 수치 이상 넘어가면 보안취약점이 있는 것으로 판단한다. 그리고 패스워드 없는 사용자 계정, 패스워드 사용 기간을 체크하여 보안취약점 유무를 판단하게 된다. Fig. 5는 본 논문에서 개발한 MS-SQL 암호정책 강제 적용 및 접속 패스워드 실패 횟수 보안취약점 프로그램을 실행한 첫 화면이다.

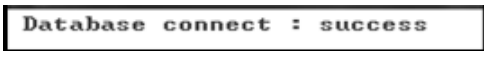

Fig. 5. Access to the Remote Database Using MS-SQL OLE Success Screen

먼저 프로그램을 실행하게 되면 관리자 계정이 정상적으로 접속을 했는지 체크한다. 접속에 성공하게 되면 'Database connect : success' 메시지를 보내고, 관리자 계정이 잘못되어 접속에 실패할 경우 Error 메시지가 발생된다.

Fig. 6. OLE Connection Fail Screen in the MS-SQL

접속에 성공하게 되면 MS-SQL Server 계정명, 암호정책 강제 적용 유무 확인, 접속 실패 횟수, 패스워드 없는 사용자 계정, 패스워드 사용 기간 등에 대해서 확인할 수 있다.

```
main()
{
        DB connection;
        while(!tbl.ISEOF())
        {
                if(tbl.get("name", id))
                        cout << "MS-SQL Server 계정명 : " << id << \n";
                else {
                        print error message; break;
                }
                tbl.MoveNext();
        }
        return 0;
}
```

Fig. 7. MS-SQL Server Account Name Check Source Code

Fig. 7은 MS-SQL Server 계정명을 확인하는 소스 코드이다. name값으로 계정명을 주고 출력하게 된다. 설계된 프로그램에서 계정마다 계정명을 확인할 수 있고, 그 계정에 해당하는 암호정책 강제 적용과 패스워드 접속 실패 횟수, 패스워드 없는 계정, 패스워드 사용 기간 등을 확인할 수 있다.

```
main()
{
        DB connection;
        .....
        int policy;
        if(!tbl.ISEOF()) tbl.MoveFirst();
        while (!tbl.ISEOF()) {
                if(tbl.Get("is_policy_checked", policy);
                        cout "policy set";
                else {
                        print error message;
                }
                if(policy > 1) cout "암호정책강제적용이 되어있습니다." << "\n";
                else {
                        cout "암호정책이 적용되어 있지 않습니다.\n"
                                "보안 취약점이 발견되었습니다." << "\n"
                }
                tbl.MoveNext();
        }
}
```

Fig. 8. MS-SQL Password Policy Setting Check Source Code

Fig. 8은 암호정책 강제 적용 확인을 하는 소스 코드이다. is_policy_checked값으로 암호정책 강제 적용을 판단하게 된다. MS-SQL에서는 쿼리문 is_policy_checked을 이용하여 0과 1을 비교함으로써 암호정책 강제 적용 유무를 판단할 수 있다. 만약 is_policy_checked가 0이면 암호정책 강제 적용이 되어있지 않은 상태이고 1이면 적용된 상태이다. 만약 policy가 1이거나 1보다 크면 (policy >= 1) 강제 적용이 되어있는 상태로 확인할 수 있다.

```
main()
{
        DB connection;
        ....
        char id[100];
        int day;
        if(!tbl.ISEOF()) tbl.MoveFirst();
        while (!tbl.ISEOF()) {
                if(tbl.Get("Time", id) then cout "현재 시간 : " << day << "일 \n";
                else {
                        print error message;
                }

                if(tbl.Get("modify_days", day);
                        cout "패스워드 변경하지 않은 기간 : " << id << "\n";
                else {
                        print error message;
                }


                if(day > 일정기간) cout "오랫동안 패스워드 변경을 하지 않은 계정이 있습니다." << "\n";
                                "보안 취약점이 발견되었습니다." << "\n"
                else {
                        cout "보안 취약점이 발견되지 않았습니다." << "\n"
                }
                tbl.MoveNext();
        }
        return 0;
}
```

Fig. 9. Feature Design of MS-SQL Password Change Period Inspecting

Fig. 9는 MS-SQL 사용자 계정들의 패스워드 변경 기간을 확인하는 소스 코드이다. Time은 현재시간을 보여주고 modify_date는 마지막 패스워드 변경 시간, 그리고 modify_days는 패스워드 변경하지 않은 기간을 나타낸다. 현재 시간과 마지막 패스워드 변경 시간의 차를 계산하여 패스워드 변경을 하지 않은 기간을 확인할 수 있게 된다. 현재 기준일인 day >= 10으로 설정을 하여 10일 동안 패스워드를 변경하지 않았다면 보안취약점이 존재하는 것으로 판단을 하여 경고메시지를 보내주게 된다.

```
main()
{
        DB connection;
        .....
        char id[100];
        int no_pw;
        if(!tbl.[SEOF()) tbl.MoveFirst();
        while (!tbl.[SEOF()) {
                if(tbl.Get("Name", id) no_pw++;  cout "패스워드 없는 사용자 계정 : " << id << " \n";
                else {
                        print error message;
                }
                tbl.MoveNext();
        }
        if(no_pw > 0) cout "패스워드가 없는 사용자 계정이 있습니다. 보안 취약점이 있습니다. \n";
        else cout "보안 취약점이 없습니다. \n";
        return 0;
}
```

Fig. 10. Design of the MS-SQL no Password Account

Fig. 10은 MS-SQL 사용자 계정들의 패스워드가 존재하는지 확인하는 소스 코드이다. 계정명을 확인하여 해당 계정의 패스워드 존재 유무를 판단한다. PWDCOMPARE문을 이용하여 SQL Server의 사용자 패스워드를 확인하고, 조건문을 이용하여 패스워드가 없는 사용자가 하나하도 존재한다면 보안취약점이 존재하는 것으로 판단하고, 패스워드가 없는 사용자가 존재하지 않는다면 보안취약점이 없는 것으로 판단을 한다. sys.sql_logins 테이블에서 id 계정을 받아와서 패스워드가 없는 계정을 출력하고 1씩 증가하여 모든 계정을 검사하게 된다. 모든 계정의 검사를 마친 후 패스워드가 없는 계정 no_pw가 한 개라도 나오게 되면 보안취약점이 존재하는 것으로 판단하게 된다.

```
main()
{
        DB connection;
        .....
        char id[100];
        int bad_policy;
        if(!tbl.[SEOF()) tbl.MoveFirst();
        while (!tbl.[SEOF()) {
                if(tbl.Get("BadPasswordCount", bad) cout "접속실패 횟수 : " << bad << " \n";
                else {
                        print error message;
                }
                if(bad > 일정 횟수) cout "로그인 여러번 시도한 사용자가 있습니다. 보안 취약점이 있습니다. \n";
                else cout "보안 취약점이 없습니다. \n";
                tbl.MoveNext();
        }
        return 0;
}
```

Fig. 11. Design of the MS-SQL Password Retrial

Fig. 11에서 BadPasswordCount는 패스워드 접속 실패 횟수 소스 코드로 기준점을 3으로 지정을 하고 만약 3번 이상 패스워드 접속에 실패할 경우 보안취약점이 있는 것으로 판단하는 소스 코드이다. BadPasswordCount 쿼리문으로 데이터베이스 값을 받아와서 bad 이름으로 기준점 3과 비교한다. 현재 패스워드 접속 실패 횟수 기준점이 3 이상(bad >=3)이지만 만약 더 큰 값을 주게 된다면 문제가 발생할 수 있다. 기준점을 100으로 지정할 경우 악의적인 의도를 가진 사용자가 99번 접속시도를 하여 성공할 가능성이 기준점 3보다 높기 때문이다. 따라서 접속 실패 횟수를 3으로 두어 설계하였다.

5. 실 험

본 논문에서는 암호정책 강제 적용 및 패스워드 실패 횟수 보안취약점 설계를 통하여 프로그램을 개발하고 실험을 하였다. MS-SQL 시스템에 대한 암호정책 강제 적용 및 패스워드 실패 횟수 보안취약점 실험을 위하여 사용자 계정 문자열이 짧은 사용자를 중심으로 수행하였다. 그리고 정상적인 실험이 이루어질 경우에 계정 문자열이 긴 것으로 실험을 수행하였고, 패스워드가 없는 사용자 계정을 찾는 기능에 대한 실험은 모든 계정에 대해서 수행하였다. 다음 Fig. 12는 실험을 위한 전체적인 시스템 환경 및 구조를 나타낸다.

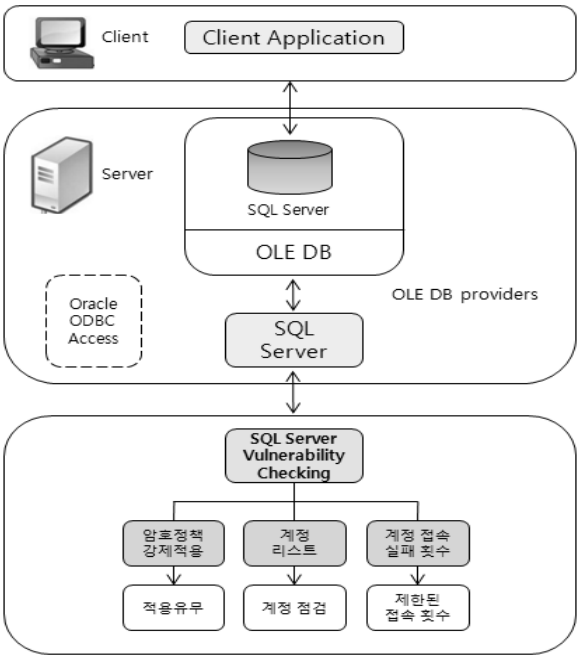

Fig. 12. System Structure and Environment for the Experiment

가장 먼저 프로그램을 실행하게 되면 관리자 sa 계정으로 접속을 하게 된다. 접속이 성공적으로 되면 success 메시지가 출력되고 계속해서 진행하며 모든 계정들을 점검한다. 아래 Fig. 13과 Fig. 14는 여러 계정 들 중 보안취약점이 없는 계정을 보여주고 있다.

```
Database connect : success

MS-SQL Server 계정명 : SJK
 - 암호정책강제적용이 되어있습니다.
 - 접속 실패횟수 : 0
 - SQL Server 로그인을 여러번 시도한 사용자가 없습니다.
   보안 취약점이 존재하지 않습니다.

MS-SQL Server 계정명 : OSLAB
 - 암호정책강제적용이 되어있습니다.
 - 접속 실패횟수 : 0
 - SQL Server 로그인을 여러번 시도한 사용자가 없습니다.
   보안 취약점이 존재하지 않습니다.
```

Fig. 13. No Vulnerability Password Access Retrial in MS-SQL

Fig. 13의 MS-SQL Server 계정 SJK와 OSLAB는 보안취약점이 없는 경우이다. 암호정책 강제 적용이 되어있고, 패스워드 접속 실패 횟수가 없기 때문에 보안취약점이 없는 것으로 판단한다. 설계 단계에서 패스워드 접속 실패 횟수를 3으로 제한을 두었기 때문에 현재는 접속 실패 횟수가 0이므로 보안취약점이 없다.

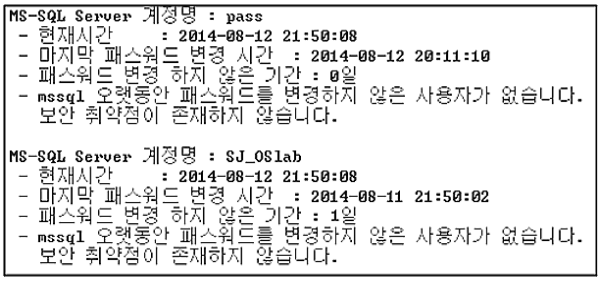

Fig. 14. No Vulnerability Password Change Period in MS-SQL

Fig. 14의 MS-SQL Server 계정 pass와 SJ_OSlab은 보안취약점이 없는 경우이다. 현재 시간은 2014년 8월 12일이고, 마지막 패스워드 변경 시간은 2014년 8월 12일과 11일이다. 현재 시간과 마지막 패스워드 변경 시간의 차를 계산하여 그 기간 동안 패스워드를 변경하지 않은 계정이 기준일 10일 이상 차이가 나면 보안취약점이 존재하는 것으로 판단하는데 현재 Fig. 14에서는 계정 pass는 0일 차, SJ_OSlab은 1일 차로 보안취약점이 존재하지 않는 것으로 판단하여 보안취약점이 없다.

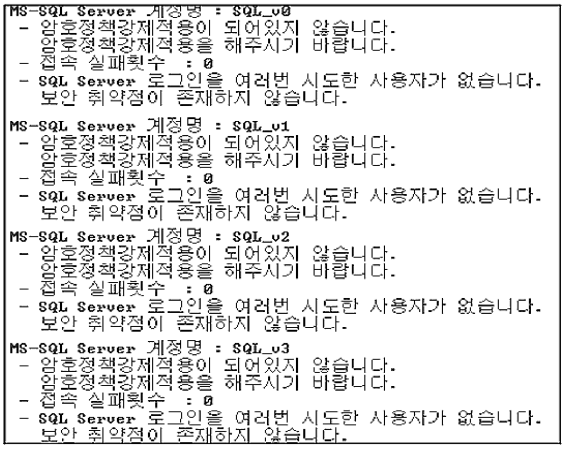

Fig. 15. No Enforcement Password Policy in MS-SQL

Fig. 15의 MS-SQL Server 계정 SQL_v0, SQL_v1, SQL_v2, SQL_v3는 MS-SQL의 접속 패스워드 실패 횟수는 없지만 암호정책 강제 적용이 되어있지 않은 경우이다. 일단 사용자의 계정이 다른 제3의 사용자가 접속 시도를 하지 않았기 때문에 패스워드 실패 횟수는 0으로 나타나지만 암호정책 적용을 하지 않았기 때문에 해당 계정은 보안의 위험도가 조금 높아진다. 데이터베이스의 정보 보호를 위해 해당 계정에 암호정책 강제 적용이 되어있지 않으므로 적용해줄 수 있도록 경고 메시지를 보내준다.

MS-SQL Server 계정 SQLServer_v2, MSTest_v1, MSTest_v2는 암호정책 강제 적용이 되어있고, 패스워드 접속 실패 횟수가 1인 경우이다. 본 논문에서는 기준점을 3으로 정하였기 때문에 접속 실패 횟수가 1인 계정은 보안취약점이 존재하지 않는 것으로 판단을 하여 메시지를 보내준다. 만약 패스워드 접속 실패 횟수가 기준점 3번을 넘어갈 경우 보안취약점이 있는 것으로 판단을 한다.

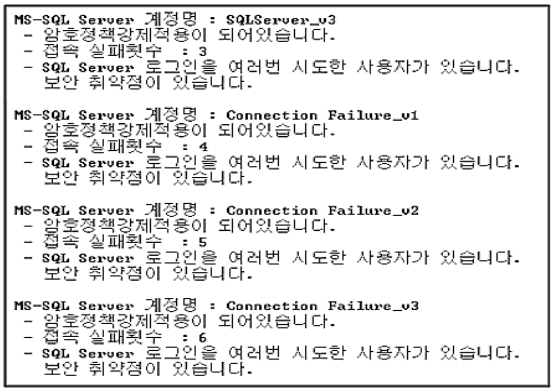

Fig. 16. Password Retrial in MS-SQL

Fig. 16의 MS-SQL Server 계정 SQL_v3, Connection Failure_v1, Connection Failure_v2, Connection Failure_v3는 암호정책 강제 적용이 되어있지만 패스워드 접속 실패 횟수가 기준점 3번을 넘었을 경우이다. 악의적인 의도를 가진 사용자가 해당 계정에 접속하기 위해 반복적인 패스워드 접속 시도를 했을 가능성이 크기 때문에 보안취약점이 존재하는 것으로 판단을 하여 경고 메시지를 보내준다. 본 논문에서는 기준점을 3으로 제한시키고 반복적인 패스워드 실패가 3 이상이지만, 만약 반복적인 패스워드 접속 실패 횟수가 더 많이 발생한다면 데이터베이스의 정보가 유출될 가능성이 더욱 커지게 된다.

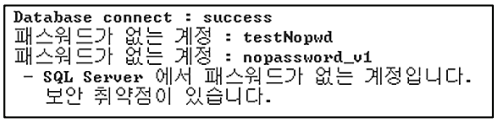

Fig. 17. No Password in MS-SQL

Fig. 17은 MS-SQL Server에서 사용자의 패스워드가 없는 계정 존재 유무를 판단한다. 패스워드가 없는 계정이 존재한다면 악의적인 의도를 가진 사용자가 해당 계정을 이용하여 접근 후 불법적으로 사용할 가능성이 크다. 데이터베이스 로그인이 가능한 계정들은 인증을 통해 데이터베이스에 접근이 가능하기 때문에 로그인 계정의 패스워드 부재로 인해 패스워드가 없는 계정이 존재할 경우 큰 피해를 입을 수 있다. 그러므로 패스워드가 없는 계정이 존재할 경우 경고메시지를 보내주어 패스워드 설정을 할 수 있도록 한다.

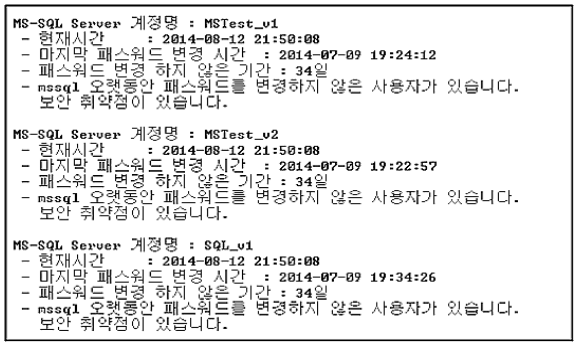

Fig. 18. No Password Change for a Period Time in MS-SQL

Fig. 18은 기준일 이상 패스워드가 변경되지 않아서 보안 취약점이 있는 것으로 판단을 하여 사용자에게 경고메시지를 보내준다. 패스워드를 주기적으로 바꾸어주지 않으면 악의적인 의도를 가진 사용자가 취약점 공격을 가할 수 있다. 유추하기 쉬운 패스워드나 여러 번의 시도를 통한 패스워드 접근 등은 주기적인 패스워드 변경으로 어느 정도 문제점을 해결할 수 있다. 그러므로 주기적인 점검과 확인을 통하여 미리 보안취약점 예방을 하고 더 큰 피해를 입지 않도록 사용자에게 실시간 경고 메시지를 보내줌으로써 데이터베이스의 정보를 보호할 수 있게 된다.

6. 결 론

데이터베이스 보안은 사용자로부터 조직 혹은 개인의 정보 유출에 대한 보호를 목적으로 한다. 데이터베이스의 정보를 지킬 수 있는 가장 효율적인 방법은 사용자가 데이터 정보의 중요성을 알고 미리 해킹 등의 공격 등이 발생하지 않도록 하는 것이 가장 좋은 방법이다. 악의적인 의도를 가진 사용자가 공격을 하게 된다면 사용자의 데이터베이스 정보가 손실 및 손상될 가능성이 아주 크다.

본 논문에서 제안하는 MS-SQL 데이터베이스 패스워드 보안취약점 점검 모듈은 MS-SQL 데이터베이스의 보안취약점 점검을 통하여 암호정책 강제 적용 유무를 판단하고, 사용자 계정의 패스워드 접근 시도 여부, 패스워드 없는 사용자 계정, 패스워드를 일정 기간 이상 변경하지 않은 경우 등에 대한 종합적인 판단을 통해서 보안취약점 존재 유무를 판단하도록 설계했다. 본 논문에서 설계한 내용을 중심으로 실험도 수행하였다. 실험 결과 본 논문에서 설계한 내용이 정상적으로 잘 동작됨을 확인할 수 있었다. 이러한 기능은 데이터베이스의 보안을 더욱 강화하고 문제점을 예방함으로써, 사용자들의 정보를 효율적으로 보호할 수 있다.

References

[1] Charlie Osborne, “The Top Ten Most Common Database Security Vulnerabilities,” http://www.zdnet.com/the-top-ten-most-common-database-security-vulnerabilities-7000017320/

[2] Vulnerability Analysis Quality Assurance DB database, http://www.dqc.or.kr/guideline/6-7-0.html

[3] Jin-Seong Jeong, “A Study on the Composition of a Secure Database,” 2013.

[4] SQL Server Password Policy (http://technet.microsoft.com/ko-kr/library/ms161959(v=sql.105).aspx)

[5] Tae-Young Kim, “about OLE DB,” http://www.taeyo.pe.kr/lecture/9_Board2001/Board2001_02.htm

[6] Yu-Kyung Kim, Seung-Cheol Sin, Jun-Seon Ahn, Wook-Sae Lee, Eun-Young Lee, and Hwan-Su Han, “Case study of software security vulnerabilities database,” *Journal of Information Science*, Vol.28, No.2, pp.20-31, Feb., 2010.

[7] Dong-Jin Kin, Dong-Woo Seo, Wan-Seok Lee, and Seong-Je Cho, “An Efficient Vulnerability Management System for Utilization of New Information Technologies-related Security Vulnerabilities,” *Korea Information Science Society Conference 2010 Korea Computer*, Vol.37, No.2(B), pp.66-71, 2010.

[8] Ji-Hong Kim, Huy-Kang Kim, “Automated Attack Path Enumeration Method based on System Vulnerabilities Analysis,” *Institute of Information Security*, Vol.22, No.5, pp.1079-1090, Oct., 2010.

[9] Hyun-A Park, Dong-Hoon Lee, and Taik-Yeong Chung, “Comperhensive Study on Security and Privacy Requirements for Retrieval System over Encrypted Database,” *Institute of Information Security*, Vol.22, No.3, pp.621-635, 2012.

[10] Seung-Ju Jang, Sung-Jin Kim “A Study of the Specific IP Vulnerability for the Oracle System”, The Comprehensive Winter Conference of Korea Information and Communications Society, pp.392-394, 2014.

[11] Woo-Seok Seo, Moon-Seog Jun, “The Management and Security Plans of a Separated Virtualization Infringement Type Learning Database Using VM(Virtual Machine),” *Korea Information and Communications Society*, Vol.36, No.8, pp.947-953, 2011.

[12] Seoung-Min Lee, Joon-Seok Oh, and Jin-Young Choi, “Comparative analysis on potential error-possibility and security vulnerability in software,” *Korea Information Science Society Conference 2010 Korea Computer*, Vol.37, No.1(D), pp.106-109, 2010.

[13] Cheon-Shik Kim, Hyoung-Joong Kim, and You-Sik Hong, “Technique of Range Query in Encrypted Database,” *Journal of the Institute of Electronics Engineers-CI*, Vol.45, No.3, pp.22-30, May, 2008.

장 승 주

e-mail : sjjang@deu.ac.kr

1985년 부산대학교 계산통계학과(전산학) (학사)

1991년 부산대학교 계산통계학과(전산학) (석사)

1996년 부산대학교 컴퓨터공학과(박사)

1987년～1996년 한국전자통신연구원(ETRI) 시스템 SW연구실

1993년～1996년 부산대학교 시간강사

2001년～2002년 Univ. of Missouri at Kansas City, visiting professor

현 재 동의대학교 컴퓨터공학과 교수

관심분야 : 운영체제, 임베디드 운영체제, 분산시스템, 시스템 보안, 스마트폰 시스템 운영체제, 자동차용 운영체제