최종보고서

데이터베이스의 취약점 분석과 해결책

제출일자 : 2012년 5월 00일
과 목 명 : 캡스톤 디자인
팀　　명 : DIS
팀　　장 : 강연준
팀　　원 : 90816542 강연준
　　90710000 조응철
담당교수 : 양환석 교수님

목 차

1. 서 론

1.1. 연구의 필요성

1.1.1 데이터베이스 보안의 개념

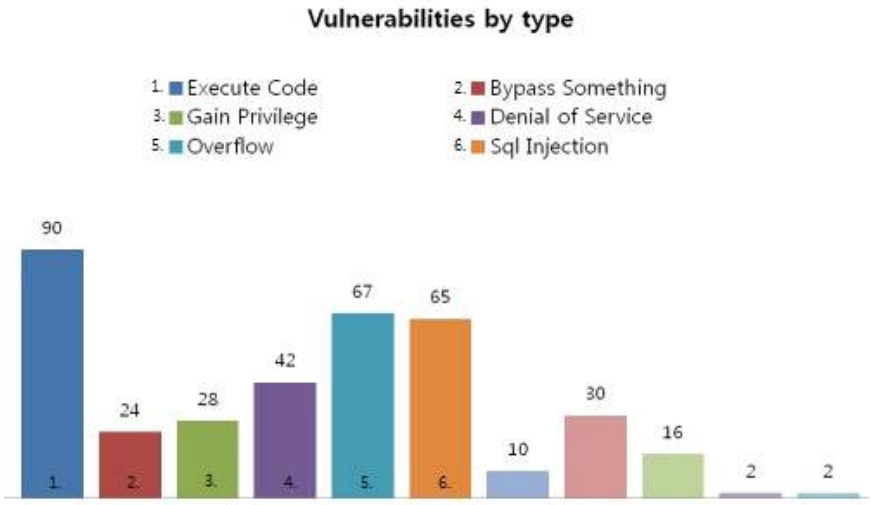
데이터베이스 보안(Database Security)은 의도하지 않은 활동으로부터 데이터베이스를 보호하는 시스템, 프로세스, 프로시저이다. 의도한 것이 아닌 활동은 권한 오용, 악의 있는 공격 또는 부주의한 관리로 분류될 수 있다.

데이터베이스 보안을 평가하는 중요한 절차는 데이터베이스에 대한 취약점 점검을 수행하는 것이다. 취약점 점검은 데이터베이스 침입에 사용될 수 있는 취약 부분을 찾는 것이다. 데이터베이스 관리자나 정보 보안 관리자는 데이터베이스 소프트웨어 내에 알려진 취약점과 더불어 위에 언급된 계층 간 통제 구성의 오류를 찾기 위해 취약점 스캔을 실시한다. 스캔 결과는 침입자들의 위협을 줄이고 데이터베이스를 견고히 하기 위해 사용되어야 한다.

1.1.2 데이터베이스 보안의 필요성

데이터베이스는 우리가 매일 가장 많이 사용하고 있는 어플리케이션 중의 하나이다. 데이터베이스 보안은 크게 접근제어(access control), 감사(auditing), 인증(authentication), 암호화(encryption), 무결성(integrity) 등 많은 주제를 포함하고 있다. 본 작품에서는 위 주제 중 인증에 국한하여 데이터베이스 보안 문제를 다루고자 한다. 현재 많은 어플리케이션이 데이터베이스와 연동을 해야 하는데 이를 손쉽게 하기 위해 개발된 스크립트언어가 PHP이다. PHP에서는 데이터베이스 연결에 필요한 정보를 다로 하나의 파일(DB connect)로 모아서 일일이 데이터베이스에 연결해주는 코드를 쓸 필요 없이 바로 DB connect 파일을 포함함으로써 데이터베이스 연결을 한다. 그런데 이 DB Connect 파일은 정상적인 프로그램에 의해서 쓰일 뿐만 아니라, 만일 누구나 접근 가능한 루트 디렉토리에 놓이게 되면 공격자가 이를 악용 할 수 있다는 문제가 있다.

본 작품에서는 보안이 취약한 가상 사이트 환경을 구축하고 이의 문제점을 분석하고 보완할 수 있는 방안을 제시한다.



< 오라클 사에서 발표한 데이터베이스 취약점 공격 빈도수 >

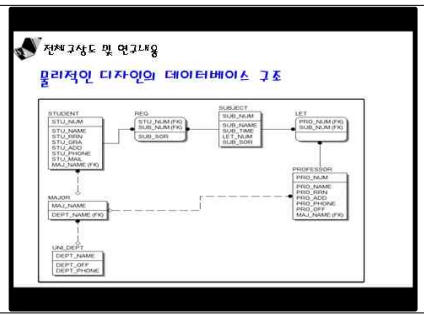
1.2. 팀 구성원

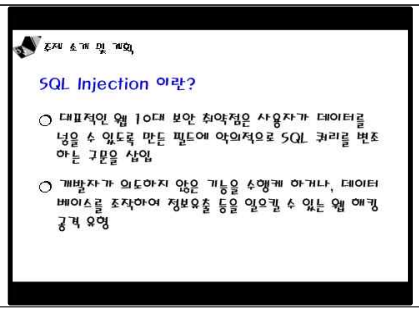
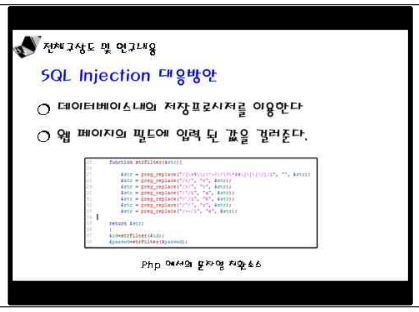
1.2.1 구성원

구성원			역할분담
2조	조장	강연준	웹페이지 구축, 데이터베이스 구축, SQL Injection 대응방안 연구 및 적용, 전체총괄
	조원	조응철	자 료 수 집

1.2. 주간활동 보고서

1.2.1 조별주간 활동 보고서

내용 / 주기	주간활동사항
1주차	· 조원의 역할분담 및 일정계획, 자료수집
2주차	· 기초적인 데이터베이스 지식 습득
3주차	
4주차	· 데이터베이스의 구조구상 및 설계 구축 
5주차	· 웹페이지 구축을 위한 지식습득 및 구축
6주차	
7주차	· 웹 페이지과 데이터베이스 연동

8주차	· 취약점에 대한 자료수집과 분석	
9주차	· 취약점에 대한 대응방안 연구	
10주차	· 중 간 점 검	
11주차	· 취약점에 대한 대응방안 연구	
12주차	· 취약점에 대응방안 적용	
13주차	· 소스 내 오류 검토 및 수정	
14주차	· 발표 준비 및 선 테스트	
15주차	· 제 작 발 표 ·	

2. 관 련 연 구

2.1. SQL injection

2.1.1 SQL Injection의 정의

현재 대부분의 웹사이트들은 사용자로부터 입력받은 값을 이용해 데이터베이스 접근을 위한 SQL Query를 만들고 있다. 사용자 로그인 과정을 예로 들면, 사용자가 유효한 계정과 패스워드를 입력했는지 확인하기 위해 사용자 계정과 패스워드에 관한 SQL Query문을 만든다. 이때 SQL injection 기법을 통해서 정상적인 SQL query를 변조할 수 있도록 조작된 사용자 이름과 패스워드를 보내 정상적인 동작을 방해할 수 있다.

입력문에 구조화 조회 언어(SQL)문에 대한 필터링이 없을 경우 해커가 SQL문으로 해석될 수 있는 입력을 시도하여 데이터베이스에 접근할 수 있는 보호 취약점. 웹 브라우저 주소(URL)창 또는 사용자 ID 및 패스워드 입력 화면 등에서 데이터베이스 SQL문에 사용되는 문자 기호(‘ 및 “)의 입력을 적절히 필터링하지 않은 경우에 해커가 SQL문으로 해석될 수 있도록 조작한 입력으로 데이터베이스를 인증 절차없이 접근, 자료를 무단 유출하거나 변조할 수 있다. 예를 들어 관리자 ID와 패스워드에 아래 문자열을 입력했을 때 로그인되면 취약점이 존재한다.

2.1.1 SQL Injection 공격유형

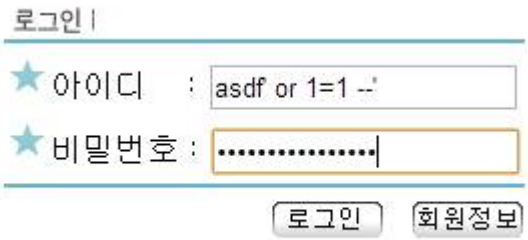
ⅰ) 사용자 인증을 비정상적으로 통과할 수 있다.
ⅱ) 데이터베이스에 저장된 데이터를 임의로 열람할 수 있다.
ⅲ) 데이터베이스의 시스템 명령을 이용하여 시스템 조작이 가능하다.

이러한 취약점을 SQL Injection 취약점이라고 하며, 사용자가 데이터 입력이 가능한 수많은 웹페이지 상에 이러한 취약점이 존재할 수 있다.

2.1.2 취약성 판단

ⅰ) 검색어 필드 및 로그인 ID, PASSWD 필드에 큰따옴표("), 작은따옴표 ('), 세미콜론(;) 등을 입력한 후, DB error가 일어나는지 확인한다.
ⅱ) 로그인 모듈 점검
Oracle인 경우: ID 필드에 ['or 1=1 --], 비밀번호 필드에는 아무 값이나 입력한 후 로그인을 시도한다.
기타의 경우
ID 필드에 ['or ''='], 비밀번호 필드에 ['or ''=']을 입력한 후 로그인을 시도한다.

※ 위 방법 외에도 다양한 방법이 가능하기 때문에, 로그인 및 사용자 입력 값을 사용하는 소스에서 DB Query 생성 방식을 직접 점검해야 한다.



< 그림 2 > SQL Injection

2.2 PHP

2.2.1 PHP의 정의

PHP란 하이퍼텍스트 생성 언어(HTML)에 포함되어 동작하는 스크립팅 언어. 별도의 실행 파일을 만들 필요 없이 HTML 문서 안에 직접 포함시켜 사용하며, C, 자바, 펄 언어 등에서 많은 문장 형식을 준용하고 있어 동적인 웹 문서를 빠르고 쉽게 작성할 수 있다. ASP(Active Server Pages)와 같이 스크립트에 따라 내용이 다양해서 동적 HTML 처리 속도가 빠르며, PHP 스크립트가 포함된 HTML 페이지에는 .php, .php3, .phtml이 붙는 파일 이름이 부여된다. 처음에는 'Personal Home Page Tools'이라 불렸으며, 공개된 무료 소스이다.

2.3 Apache

2.3.1 Apache란?

1995년 처음 발표된 월드와이드웹(WWW:World Wide Web) 서버용 소프트웨어이다. NCSA(National Center for Supercomputing Applications:미국국립수퍼컴퓨터활용센터) 소속 개발자들이 개발한 NCSA httpd 1.3 웹서버를 자신들이 개량한 것으로 소스코드까지 공개되고 있다.

NCSA httpd 1.3 서버에 패치(patch)파일을 제공했던 개발자들이 'A PAtCH server'라는 용어에서 아파치라는 이름을 따왔다. 1995년 3월 18일 공개된 아파치0.2가 NCSA httpd 1.3에 패치파일을 제공하였다.

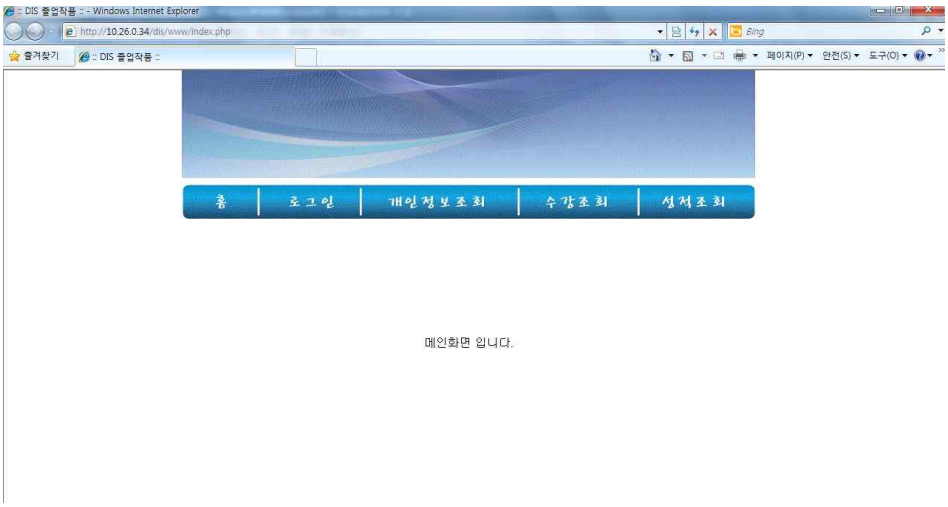
패치파일을 꾸준히 개선해 제공하고 있으며, 최고 수준의 성능을 발휘하기 때문에 월드와이드웹 서버용 소프트웨어로 가장 많이 사용되고 있다. 오픈소스(open source) 라이선스에 따라 무료로 배포되어 원하는 사람들이 자유롭게 사용할 수 있다. 유닉스·윈도 등을 비롯해 거의 모든 운영체제와 시스템에서 운용이 가능하다

3. 테스트 결과

3.1. 웹사이트 구축

3.1.1. PHP 웹사이트 구축

ⅰ) HTML과 PHP를 이용하여 웹사이트를 구축하였다. 로그인, 회원가입, 수강조회, 성적조회 메뉴 란을 만들었다.



< 구축된 웹 사이트 >

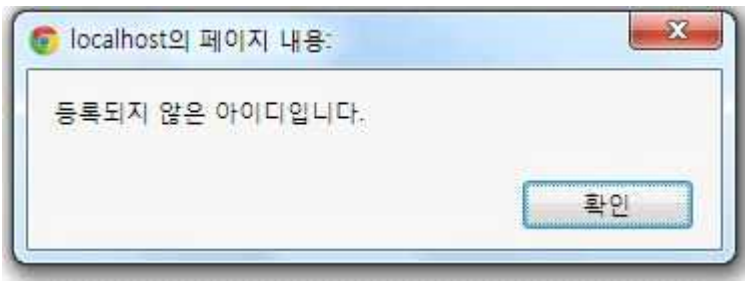
ⅱ) 로그인 페이지에서 정상적인 아이디와 비밀번호를 입력하고 로그인 한다.

< 로그인 페이지 >

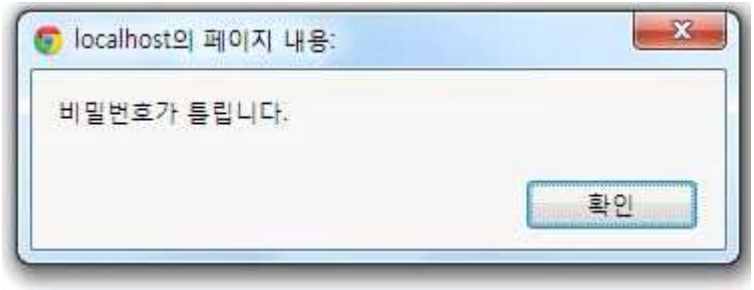
iii) 아이디와 비밀번호가 일치할 경우 정상적으로 로그인이 된다.

< 정상적인 방법으로 로그인을 하였을 경우 >

iv) 아이디와 비밀번호가 일치하지 않을 경우에 경고 메시지를 띄워주고 다시 로그인 페이지로 돌아가게 된다.



< 아이디가 올바르지 않을 경우 >

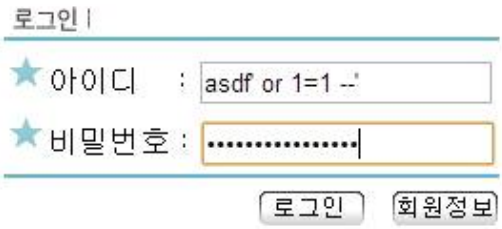


< 비밀번호가 올바르지 않을 경우 >

3.2. SQL Injection 대응방안 적용

3.2.1. SQL Injection 공격

i) SQL Injection 공격을 실행하기 위해서 본문에 설명한 바와 같이 'or 1=1--' 이라는 문구를 입력하고 로그인을 시도한다. 아이디 입력부분에 문구를 넣는 방법과 비밀번호에 문구를 넣는 방법 이 두 가지 방법을 통하여 테스트 해보았다.



< 아이디와 비밀번호를 모르는 상태에서의 SQL Injection 공격 >

< 아이디와 비밀번호를 모르는 상태에서의 SQL Injection 공격 >

ii) SQL 쿼리 문을 파괴하는 문구를 넣고 로그인을 시도한 결과로 아이디와 비밀번호가 일치하지 않았음에도 불구하고 로그인 되는 것을 확인 할 수 있다.

< 취약점을 파고들어 아이디와 비밀번호가 일치하지 않음에도 로그인이 되었다. >

3.3. SQL Injection 대응방안 적용

3.3.1. SQL Injection 공격의 대응방안 적용

SQL Injection 공격의 대응 방안으로는 두 가지 방안이 있다.
그 중 하나는 데이터베이스내의 프로시저를 통하여 입력받은 값을 프로시저 내에서 치환 함수를 통해 필터링 하여 구문이 변조, 파괴가 안 되도록 하는 방법이 있고, 또 다른 방법으로는 PHP소스 내에서 입력받은 파라미터 값을 앞에서 설명한 바와 같이 치환방법을 통하여 필터링을 해주는 것이다.

3.3.2. 대응방안 적용

i) 데이터베이스와 연동을 하는 스크립트의 모든 파라미터들을 점검하여 사용자의 입력 값이 SQL injection을 발생시키지 않도록 수정한다.

ii) 사용자 입력이 SQL injection을 발생시키지 않도록 사용자 입력 시 특수문자(' " / ₩ ; : Space -- +등)가 포함되어 있는지 검사하여 허용 되지 않은 문자열이나 문자가 포함된 경우에는 에러로 처리한다.

```
25      function strFilter($str){
26
27          $str = preg_replace("/[\+%\\;\^~|\!\?\*$#\[\]\{\}]/i", "", $str);
28          $str = preg_replace("/</", "<", $str);
29          $str = preg_replace("/>/", ">", $str);
30          $str = preg_replace("/'/i", "a", $str);
31          $str = preg_replace("/`/i", "b", $str);
32          $str = preg_replace('/"/', "c", $str);
33          $str = preg_replace("/--/i", "e", $str);
34
35      return $str;
36      }
37      $id=strFilter($id);
38      $passwd=strFilter($passwd);
```

< 입력받은 값을 PHP 내에서 치환해주는 함수 >

아이디와 패스워드에 입력 받은 값이 치환 함수를 거쳐 다른 기호로 바뀌게 되고 데이터베이스와 일치하지 않기에 로그인이 거부가 되는 것을 알수 있다.

4. 결 론

본 작품에서 “ ‘or 1=1’ -- ”라는 문구를 넣어서 쿼리문을 의도적으로 ture 의 값이 되도록 하였고 그 결과 로그인을 강제적으로 되었다. 만약 로그인 된 아이디가 웹사이트를 관리하는 관리자 계정이라면 웹 사이트를 통제할 수 있는 대부분의 권한을 갖게 될 수 있다. 그리고 본 작품에서 보여지진 않았지만 update, delete, union 등 여러 가지 방법으로 Injection 공격을 해올 경우 데이터베이스 안에 있는 정보의 변조, 파괴, 엄청난 피해가 발생할 수 있다. 모든 프로그램 언어 그리고 SQL DB는 잠재적인 취약점을 가지고 있다. 이를 보호하기 위해서는 강력한 디자인, 정확한 입력 값 검증, 견고하게 서버를 운영해야 한다.

견고하게 서버를 운영하기 위해서는, DB 최소권한의 유저로 운영해야 하고 사용하지 않는 저장된 프로시저와 기능들은 제거하거나 관리자에게 제한된 접근 권한을 주어야 한다.
또한 모든 사용자 계정의 패스워드를 강화 시켜야 하며 웹 서버 접근만 허용해야 할 것이다. 또한 SQL 서버의 에러 메시지를 사용자에게 보여주지 않도록 설정한다. 공격자는 리턴되는 에러 메시지에 대한 분석을 통하여 공격에 성공 할 수 있는 SQL Injection 스트링을 알아 낼 수 있다. 따라서 SQL 서버의 에러 메시지를 외부에 제공하지 않도록 한다.

세계적으로 무수히 많은 기업들이 있다. 그 무수히 많은 기업들 중 데이터베이스를 사용하지 않는 기업은 없다고 봐도 무방하다 그만큼 데이터베이스의 의존도가 높은 사회이고 각 기업마다 데이터베이스에 기업의 기밀정보를 포함하여 많은 정보를 저장해 두고 사용하고 있다. 그만큼 데이터베이스의 보안은 중요하고 보안이 중요한 만큼 그 데이터베이스의 중요성을 다루고 있다. 이 작품을 통하여 우리 팀에겐 데이터베이스 보안에 대한 경각심을 심어주고 더욱 더 관심을 가질 수 있는 계기가 되었고 이 작품을 보는 이에겐 조금 더 경각심을 심어주고 관심을 가질 수 있는 계기가 되었길 바래본다.

5. 부 록

웹 사이트 전체 페이지 html 소스 index.php

```
<html>
 <head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
  <title>:: DIS 졸업작품 ::</title>
 </head>
  <frameset framespacing="0" border="0" frameborder="0" rows="210,*">
          <frame name="top" src="top.php" scrolling="auto" noresize>
          <frame name="main" src="main.php" scrolling="auto" noresize>
  </frameset>
</html>
```

웹사이트의 메인 페이지 html 소스 main.php

```
<html>
 <head>
  <meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
  <title> :: PHP 프로그래밍 입문에 오신 것을 환영합니다~~ :: </title>
  <link rel="stylesheet" href="style.css" type="text/css">
 </head>

<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<table width="776" align="center" cellspacing="0" cellpadding="0" border="0">
<tr height=150><td></td></tr>
<tr align=center>
<td>
    메인화면 입니다.
</td>
</tr>
</table>
</body>
</html>
```

홈페이지 메인화면 상단 메뉴 HTML , PHP 소스 top.php

```
<?session_start();
?>
<html>
 <head>
  <meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
  <title>:: PHP 프로그래밍 입문에 오신것을 환영합니다~~ ::</title>
  <link rel="stylesheet" href="style.css" type="text/css">
 </head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
    <table width="776" align="center" cellspacing="0" cellpadding="0" border="0">
```

```
        <tr>
          <td>
    <table width=776 cellspacing="0" cellpadding="0" border="0">
   <!--상단제목그림-->
        <tr>
          <td colspan="10">
            <img border="0" src="img/title.gif" width="776" height="146"></td>
        </tr>

        <tr>
          <td height="8" colspan="10">
            <img border="0" src="img/blank.gif" width="1" height="8"></td>
        </tr>


   <!--메뉴 시작-->
        <TR>
          <TD>
            <a href="main.php" target="main">
            <img SRC="img/menu_01.gif" WIDTH=104 HEIGHT=47 border=0 ALT=""></a></TD>
<?
    if (!$userid)
    {
        echo "
          <TD>
            <a href='login/login_form.html'   target='main'>
            <img SRC='img/menu_02.gif' WIDTH=138 HEIGHT=47 border=0 ALT=''></a></TD>
         ";
    }
    else
    {
        echo "
         <TD>
           <a href='login/logoff.php'   target='main'>
           <img SRC='img/menu_10.gif' WIDTH=138 HEIGHT=47 border=0 ALT=''></a></TD>
         ";
    }

    if (!$userid)
    {
        echo "
            <TD>
              <a href='login/member_form.html'   target='main'>
              <img SRC='img/menu_03.gif' WIDTH=214 HEIGHT=47   border=0 ALT=''></a></TD>
         ";
```

```
    }
    else
    {
        echo "
          <TD>
              <a href='login/modify_memberinfo.php'   target='main'>
            <img SRC='img/menu_03.gif' WIDTH=214 HEIGHT=47   border=0 ALT=''></a></TD>
         ";
          }
?>
          <TD>
              <a href="course/co_in.html"   target="main">
            <img SRC="img/menu_04.gif" WIDTH=162 HEIGHT=47 border=0 ALT=""></a></TD>
          <TD>
            <a href="score/sc_in.html"   target="main">
            <img SRC="img/menu_05.gif" WIDTH=158 HEIGHT=47 border=0 ALT=""></a></TD>
          </TR>
    </table>
          </td>
          </tr>
    </table>
<!--메뉴끝-->
</body>
</html>
```

데이터 베이스와 연결하는 PHP 소스 dbconn

```
<?
        $connect = oci_connect('dis_db', 'Q1w2e3r4', 'ORCL','AL32UTF8');
?>
```

로그인 페이지 PHP 소스 login.php

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<?session_start();
   // 이전화면에서 이름이 입력되지 않았으면 "이름을 입력하세요"
   // 메시지 출력
   if(!$id) {
     echo("
           <script>
             window.alert('아이디를 입력하세요.')
             history.go(-1)
           </script>
         ");
         exit;
   }
```

```
   if(!$passwd) {
     echo("
           <script>
             window.alert('비밀번호를 입력하세요.')
             history.go(-1)
           </script>
         ");
         exit;
   }

          function strFilter($str){

                    $str = preg_replace("/[₩+%₩₩;₩^~|₩!₩?₩*$#₩[₩]₩{₩}]/i", "", $str);
                    $str = preg_replace("/</", "<", $str);
                    $str = preg_replace("/>/", ">", $str);
                    $str = preg_replace("/'/i", "a", $str);
                    $str = preg_replace("/`/i", "b", $str);
                    $str = preg_replace('/"/', "c", $str);
                    $str = preg_replace("/--/i", "e", $str);
/*                  $str = preg_replace('/([₩x00-₩x08₩x0b-₩x0c₩x0e-₩x19])/', '',$str);
                    $str = preg_replace("/union[^₩x21-₩x7e]/i", "union ", $str);
                    $str = preg_replace("/select[^₩x21-₩x7e]/i", "select ", $str);
                    $str = preg_replace("/insert[^₩x22-₩x7e]/i", "insert ", $str);
                    $str = preg_replace("/drop[^₩x21-₩x7e]/i", "drop ", $str);
                    $str = preg_replace("/update[^₩x21-₩x7e]/i", "update ", $str);
                    $str = preg_replace("/and[^₩x21-₩x7e]/i", "and ", $str);
                    $str = preg_replace("/or[^₩x21-₩x7e]/i", "or ", $str);
                    $str = preg_replace("/if[^₩x21-₩x7e]/i", "if ", $str);

                    $str = preg_replace("/[^₩x21-₩x7e]union/i", " union", $str);
                    $str = preg_replace("/[^₩x21-₩x7e]select/i", " select", $str);
                    $str = preg_replace("/[^₩x21-₩x7e]insert/i", " insert", $str);
                    $str = preg_replace("/[^₩x21-₩x7e]drop/i", " drop", $str);
                    $str = preg_replace("/[^₩x21-₩x7e]update/i", " update", $str);
                    $str = preg_replace("/[^₩x21-₩x7e]and/i", " and", $str);
                    $str = preg_replace("/[^₩x21-₩x7e]or/i", " or", $str);
                    $str = preg_replace("/[^₩x21-₩x7e]if/i", " if", $str);

                    $str = preg_replace("/&/", "&", $str);
                    $str = preg_replace("/&amp/", "&", $str);
                    $str = preg_replace("/ /", " ", $str);
                    $str = preg_replace("/&lt/", "<", $str);
                    $str = preg_replace("/&gt/", ">", $str);
                    $str = preg_replace("/&#39/", "'", $str);
                    $str = preg_replace("/&#96/", "`", $str);
                    //$str = preg_replace("/&quot/", "'", $str);
                    $str = preg_replace("/&#95/", "_", $str);
*/
```

```
        return $str;
        }
        $id=strFilter($id);
        $passwd=strFilter($passwd);


        include "../dbconn.php";

        $sql = "select * from member where id='$id'";
        echo $id;
        $result = oci_parse($connect, $sql);
        oci_execute($result);
/*   if(!@oci_execute($result))
   error("SQL구문에러");
   exit;
*/

        $match = oci_fetch_all($result, $num_rows, 0, -1, OCI_FETCHSTATEMENT_BY_ROW);

        if($match==0)
        {
                echo("
                        <script>
                        window.alert('등록되지 않은 아이디입니다.')
                        history.go(-1)
                        </script>
                        ");
        }
        else
        {
                $sql2 = "select * from member where id='$id' and passwd='$passwd'";
                $result2 = oci_parse($connect, $sql2);
                oci_execute($result2);
                $match2 = oci_fetch_all($result2, $num_rows2, 0, -1,
OCI_FETCHSTATEMENT_BY_ROW);

                echo $match2;
                if($match2==0)
        {
                        echo("
                                <script>
                                window.alert('비밀번호가 틀립니다.')
                                history.go(-1)
                                </script>
                        ");

                        exit;
                }
                else
```

```
                {
                        $sql3 = "select id from member where id='$id'";
                        $result3 = oci_parse($connect, $sql3);
                        oci_execute($result3);
                        $row1 = oci_fetch($result3);

                        $sql4 = "select name from member where id='$id'";
                        $result4 = oci_parse($connect, $sql4);
                        oci_execute($result4);
                        $row2 = oci_fetch($result4);


                        $userid = $row1;
                        $username = $row2;

                        session_register(userid);
                        session_register(username);

                        echo("
                                <script>
                                top.location.href = '../index.php';
                                </script>
                        ");
                }
        }

?>
```

ID를 체크하는 PHP 소스 check_id.php

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<?php
   if(!$id)
   {
      echo("아이디를 입력하세요.");
   }
   else
   {
      include "../dbconn.php";

      $sql = "select * from member where id='$id'";

      $result = oci_parse($connect, $sql);
            oci_execute($result);
            $rows = oci_fetch_all($result, $num_rows, 0, -1, OCI_FETCHSTATEMENT_BY_ROW);
```

```
        if ($rows>0)
        {
            echo "아이디가 중복됩니다.<br>";
            echo "다른 아이디를 사용하세요.<br>";
        }
        else
        {
            echo "사용가능한 아이디입니다.";
        }
        oci_close($connect);
    }
?>
```

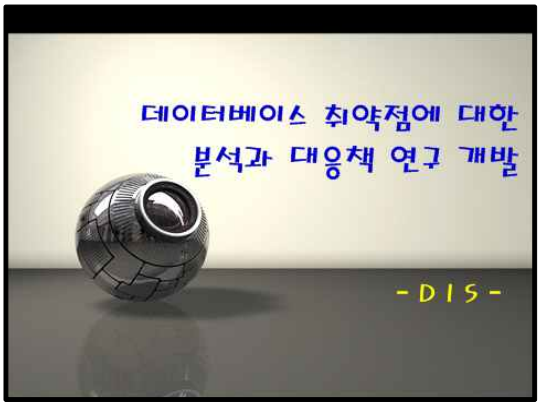
로그인 페이지 HTML 소스 login_form.html

```
<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
</head>
 <body>
   <link rel="stylesheet" href="../style.css" type="text/css">

      <form method=post action=login.php>
    <table align=center>
        <tr><td><img src="img/login.gif"></td></tr>
        <tr height=1 bgcolor=#5AB2C8><td></td></tr>
        <tr><td></td></tr>
        <tr><td><img src="img/star.gif">아이디    :
        <input type="text" name="id" size="20" maxlength="20"></td>
        </tr>
        <tr>
          <td><img src="img/star.gif">비밀번호 :
          <input type="password" name="passwd" size="20" maxlength="20">
          </td>
        </tr>
        <tr height=1 bgcolor=#5AB2C8><td></td> </tr>
        <tr><td></td></tr>
        <tr>
          <td align=right>
          <input type=image src="img/login_on.gif" border=0>
          </a>  
          <a href="member_form.html"><img src="img/member.gif" border=0></a>
          </td>
        </tr>
    </table>
      </form>

 </body>
</html>
```

6. 참고문헌

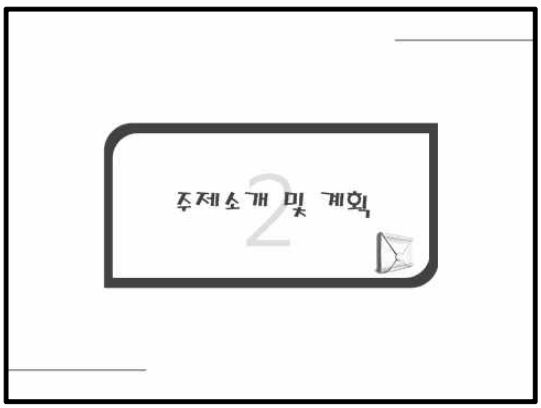
[1] 황성운, “데이터베이스 보안 취약 사이트 공격 및 대응 방안”,보안공학연구논문지 2011. 04
[2] KISA, “자동화된 SQL Injection 공격을 통한 악성코드 대량 삽입 수법 분석” 한국정보보호진흥원 2008. 12
[3] 문성기, “ PHP입문에서 최적화된 DB 연동까지 PHP, MySQL & Oracle” 다올미디어
[4] 조은백, “데이터베이스 보안”, 생능출판사
[5] 김태근, “업그레이드 된 쿼리로 만나는 오라클 SQL & PL/SQL”, 프리렉
[6] http://www.phpschool.com/ PHP school
[7] http://www.kisa.or.kr/ 한국인터넷진흥원

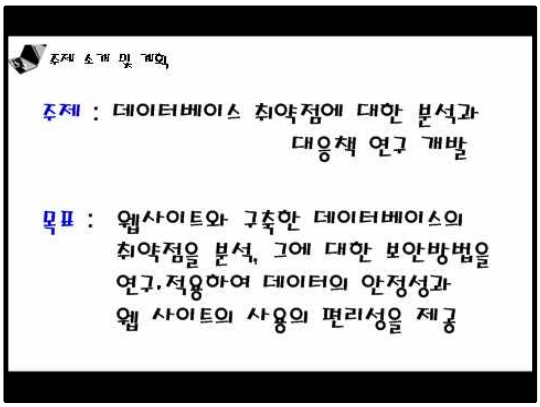
7. 발표 PPT


데이터베이스 취약점에 대한
분석과 대응책 연구 개발
- D I S -


목차
DIS 팀 소개
주제 소개 및 계획
전체 구상도 및 연구내용
결론 및 발전방향


조원소개
☆ 조 이 름 : D I S
(Database Information Security)
☆ 구 성 원 : 08 학 번 강 연 준
07 학 번 조 응 철
☆ 담당교수님 : 윤승재 교수님


주제소개 및 계획
2


주제 소개 및 계획
주제 : 데이터베이스 취약점에 대한 분석과
대응책 연구 개발
목표 : 웹사이트와 구축한 데이터베이스의
취약점을 분석, 그에 대한 보안방법을
연구,적용하여 데이터의 안정성과
웹 사이트의 사용의 편리성을 제공


주제 소개 및 계획
알려진 데이터 베이스의 취약점

주제 소개 및 계획

SQL Injection 이란?

- 대표적인 웹 10대 보안 취약점은 사용자가 데이터를 넣을 수 있도록 만든 필드에 악의적으로 SQL 쿼리를 변조하는 구문을 삽입
- 개발자가 의도하지 않은 기능을 수행케 하거나, 데이터베이스를 조작하여 정보유출 등을 일으킬 수 있는 웹 해킹 공격 유형

주제 소개 및 계획

SQL Injection 피해사례 1

지난 2009년 루마니아 해커 'Unu(우누)'는 SQL Injection 공격을 이용해 유럽의 대형 금융사인 ING, 덱시아(Dexia) 그리고 HSBC 등을 해킹했다. 그의 공격을 받은 금융사 들은 고객 e메일과 풀 네임 정보, 평문 패스워드 등이 유출됐다. 우누는 HSBC 프랑스에서 모든 데이터 베이스 접근 권한을 획득했고 파일 시스템 접근권한도 얻어냈다.
특히 이들 금융사 중 일부는 모바일 뱅킹 서비스까지 해킹 당했다. 고객 스마트폰이 해킹되면서 보안이 뚫린 것. 다행히 당시 우누는 일반에 해킹사실을 공개한 후 "기업이 자신들의 시스템을 보호하기 위해 더 많은 보안투자를 해야 한다는 걸 인식시켜주기 위해 해킹했다"고 밝혀 이 사건은 직접적인 피해로 까지 이어지진 않았다

주제 소개 및 계획

SQL Injection 피해사례 2

해커들이 최근 들어 가장 많이 사용하는 해킹 수단은 'SQL인젝션'이라는 기법인 것으로 조사됐다.
주요 외신은 지난 26일(현지시간) 보안 클라우드 호스팅 전문기업인 파이어호스트의 보고서를 인용, 미국과 유럽 데이터센터에서 SQL인젝션 공격이 지난 1분기 대비 2분기에 69%가 늘어났다고 보도했다.
이 회사는 SQL인젝션과 함께 크로스사이트스크립팅(XSS), 디렉토리 트래버설(Directory traversal), 크로스사이트 리퀘스트 포저리(CSRF) 등을 대상으로 이러한 결과를 내놨다.

파이어호스트는 지난 1분기에 27만7천770건의 SQL인젝션을 이용한 공격이 차단됐으나 2분기 들어 차단된 공격이 46만9천983건에 이르렀다고 밝혔다. 이 회사는 "데이터를 훔치기 위한 공격방식으로 자주 활용된다"고 설명했다.
SQL 인젝션은 보안이 취약한 웹사이트의 URL과 텍스트필드에 악성코드를 심는다. 이 공격의 목표는 데이터베이스의 내용을 훔치고 2차 범죄에 활용하기 위한 것이다.
지난해 해커그룹인 룰즈섹이 소니를 해킹할 때 이 같은 방법을 썼다. 외신은 "해커들이 이메일 주소와 비밀번호 등 개인정보를 훔칠 때 이 기법이 자주 사용된다"고 말했다.
지난 수개월 간 링크드인, 이하모니, 야후, 안드로이드 포럼, 폼스프링, 엔비디아 등은 모두 SQL 인젝션 공격이 있었던 것으로 추정되고 있다.
파이어 호스트의 선임 보안 엔지니어인 크리스 헝클리는 "많은 고객들이 이 방법을 통해 데이터를 잃고 있다"며 "SQL 인젝션 공격은 자동화됐고, 많은 웹사이트 이용자들이 정보유출의 위험에 처할 수 있다"며 주의를 당부했다.

2012.07.30 ZDNET Korea 손경호 기자

주제 소개 및 계획

진행 결과

	1월	2월	3월	4월	5월
데이터베이스 구축	■	■			
PHP 웹사이트와 연동		■	■	■	
취약점 분석과 보완		■	■	■	
SQL 대응책 기술 조사			■	■	■
보안기능 프로그래밍/시스템 완성				■	■

3 전체구상도 및 연구내용

전체구상도 및 연구내용

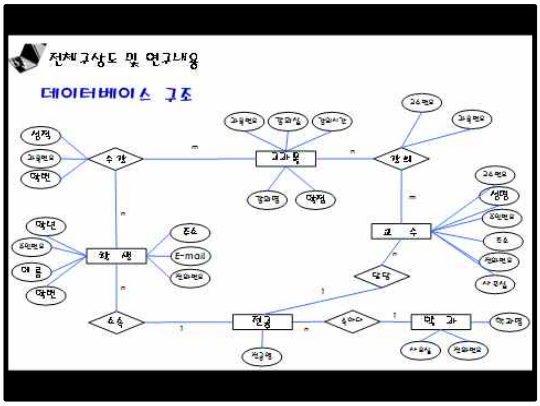
개발환경

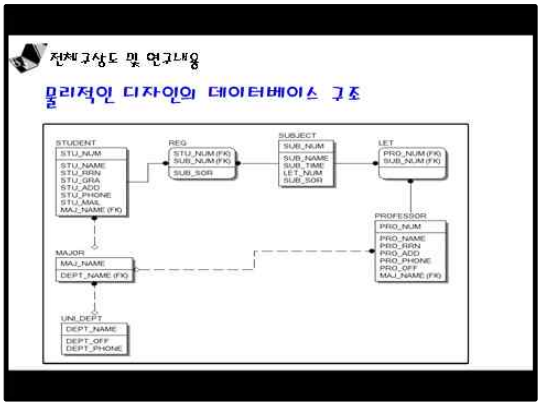
O S : Windows 7

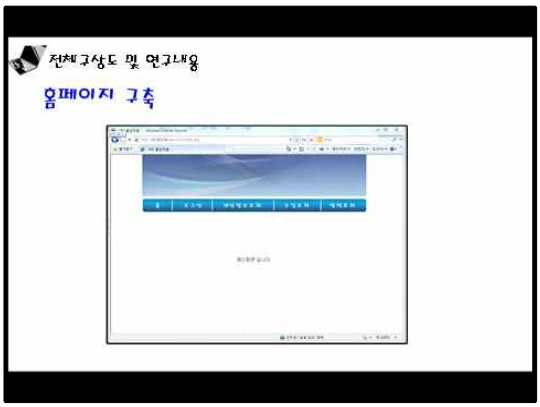
DataBase : Oracle 11g

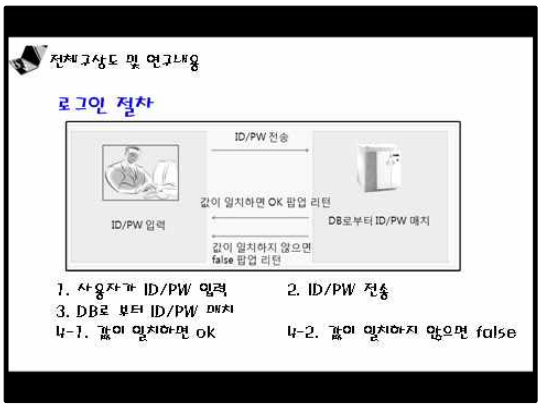
server : Apache2.2.2 , php5

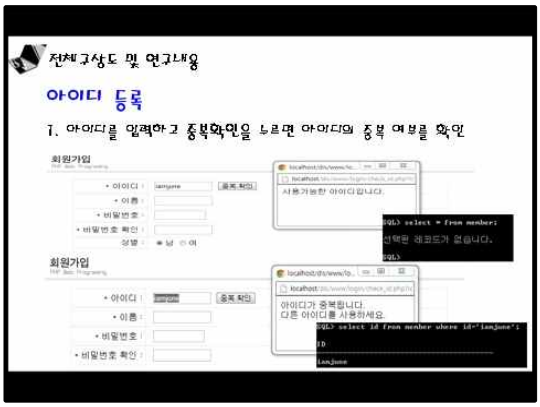
사용언어 : PHP, html

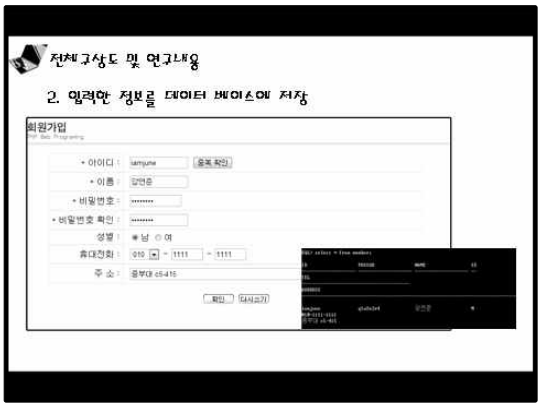

전체구상도 및 연구내용
데이터베이스 구조

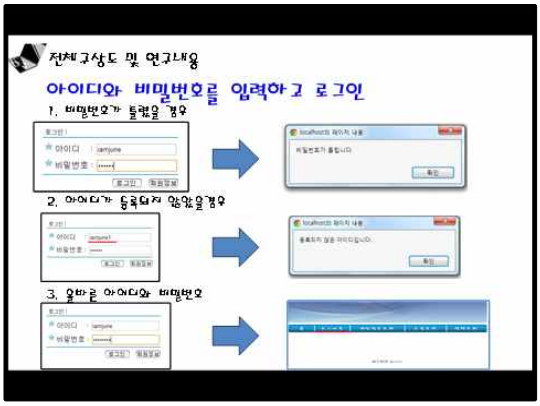

전체구상도 및 연구내용
물리적인 디자인의 데이터베이스 구조

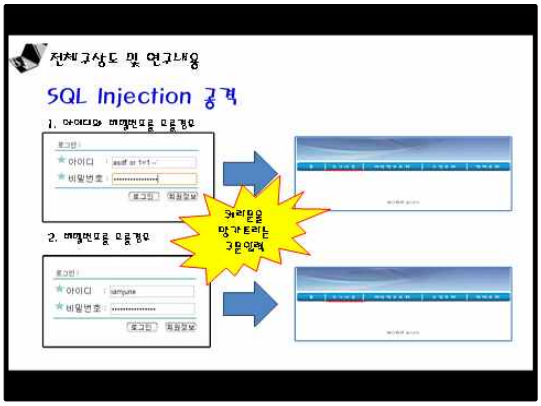

전체구상도 및 연구내용
홈페이지 구축

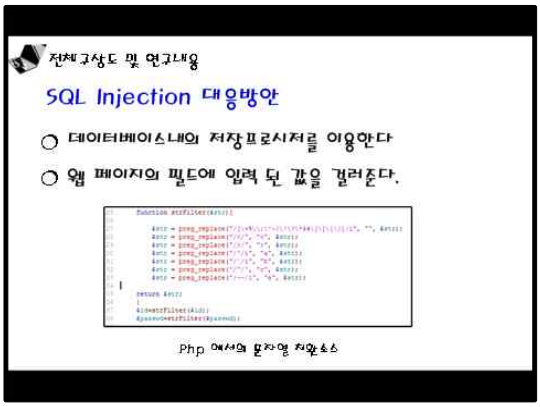

전체구상도 및 연구내용
로그인 절차
ID/PW 전송
값이 일치하면 OK 팝업 리턴
DB로부터 ID/PW 매치
ID/PW 입력
값이 일치하지 않으면
false 팝업 리턴
1. 사용자가 ID/PW 입력
2. ID/PW 전송
3. DB로 부터 ID/PW 매치
4-1. 값이 일치하면 ok
4-2. 값이 일치하지 않으면 false

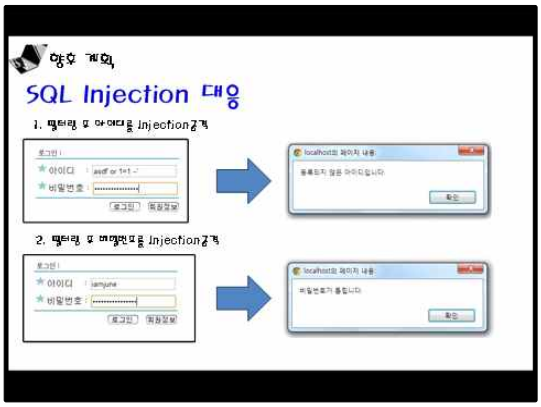

전체구상도 및 연구내용
아이디 등록
1. 아이디를 입력하고 중복확인을 누르면 아이디의 중복 여부를 확인


전체구상도 및 연구내용
2. 입력한 정보를 데이터 베이스에 저장

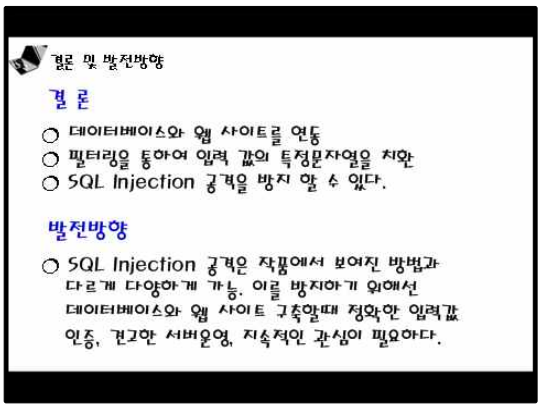

전체구상도 및 연구내용
아이디와 비밀번호를 입력하고 로그인
1. 비밀번호가 틀렸을 경우
2. 아이디가 등록되지 않았을경우
3. 알맞은 아이디와 비밀번호

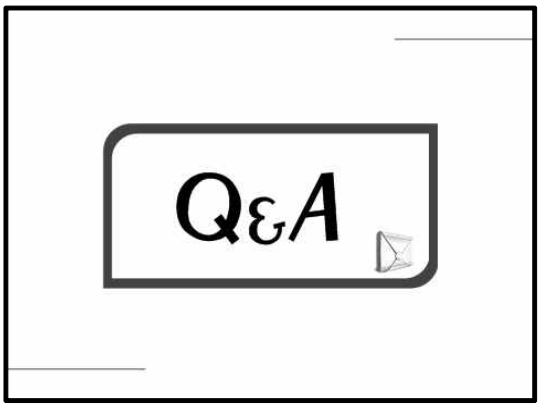

전체구상도 및 연구내용
SQL Injection 공격
1. 아이디와 비밀번호를 모를경우
2. 비밀번호를 모를경우
쿼리문을 망가트리는 구문입력


전체구상도 및 연구내용
SQL Injection 대응방안
데이터베이스내의 저장프로시저를 이용한다
웹 페이지의 필드에 입력 된 값을 걸러준다.
Php 에서의 문자열 치환소스


향후 계획
SQL Injection 대응
1. 필터링 및 아이디를 Injection공격
2. 필터링 및 비밀번호를 Injection공격


결론 및 발전방향


결론 및 발전방향
결 론
데이터베이스와 웹 사이트를 연동
필터링을 통하여 입력 값의 특정문자열을 치환
SQL Injection 공격을 방지 할 수 있다.
발전방향
SQL Injection 공격은 작품에서 보여진 방법과 다르게 다양하게 가능. 이를 방지하기 위해선 데이터베이스와 웹 사이트 구축할때 정확한 입력값 인증, 견고한 서버운영, 지속적인 관심이 필요하다.


Q&A


THANK YOU!