웹 서버 및 홈페이지 취약점 점검 가이드

2014. 8

교육사이버안전센터
Education Cyber Security Center

Contents

제 1장 개요

제 2장 홈페이지 보안취약점 사례 및 대응

제 3 장 홈페이지 개발 보안 방안

부록

- 본 자료가 개인 블로그 또는 개인 홈페이지 등 전산망에 공개되지 않도록 주의하시기 바랍니다.
- 예제로 삽입된 그림은 임의 수정하여 실제와 다를 수 있습니다.

제 1장

개요

1. 배경 및 목적

01 개요

1. 배경 및 목적

지난해 3.20, 6.25 사이버테러, 올해 KT 홈페이지 해킹 등과 같이 최근 사이버공격은 대부분 홈페이지 보안 취약점을 악용한 해킹을 통해 정보시스템 파괴, 개인정보 유출, 홈페이지 위·변조 등의 피해를 발생시켜 정보시스템을 운영하는 기관의 대외 신뢰 하락과 많은 손실을 끼치고 있습니다.

이에 따라, 홈페이지 관리자는 홈페이지 및 웹서버에서 발생하는 보안취약점에 대한 점검과 대응방안에 대해 숙지하고 미리 제거해 홈페이지 서비스의 안전성과 신뢰성을 확보하는 것이 매우 중요합니다.

본 가이드는 홈페이지 보안 관련 다양한 서적을 참고하여 교육(행정)기관 홈페이지 관리자가 홈페이지 해킹 등의 사고 예방을 위해 수행하여야 할 보안취약점 점검 항목과 대응 방안을 담고자 노력하였습니다.

아울러, 한국교육학술정보원의 교육사이버안전센터에서는 홈페이지 보안취약점자가 점검시스템(cyber.ecsc.go.kr)을 운영하여 교육(행정)기관 홈페이지 관리자가 홈페이지 보안취약점을 스스로 점검하고 보완 조치할 수 있도록 지원하고 있습니다.

홈페이지 관리자께서는 본 가이드를 활용하여 수시로 소관 기관의 정보시스템의 보안 취약점을 점검하고 보완하여 국민들이 안전하게 믿고 신뢰할 수 있는 정보시스템을 구축·운영하여 주시기 바랍니다.

웹 서버 및 홈페이지 취약점 점검 가이드

제 2장

홈페이지 보안 취약점 사례 및 대응

1. 관리자페이지 노출 취약점
2. 디렉터리 나열 취약점
3. 시스템관리 취약점
4. 불필요한 Method 허용 취약점
5. 취약한 파일 존재 취약점
6. 계정 관리 취약점
7. 실명인증 취약점
8. 전송 시 주요정보 노출 취약점
9. 파일 다운로드 취약점
10. 파일 업로드 취약점
11. 소스코드 내 중요정보 노출 취약점
12. 공개용 웹 게시판 취약점
13. 크로스사이트스크립트(XSS) 취약점
14. SQL Injection 취약점
15. 권한인증 취약점
16. 에러처리 취약점

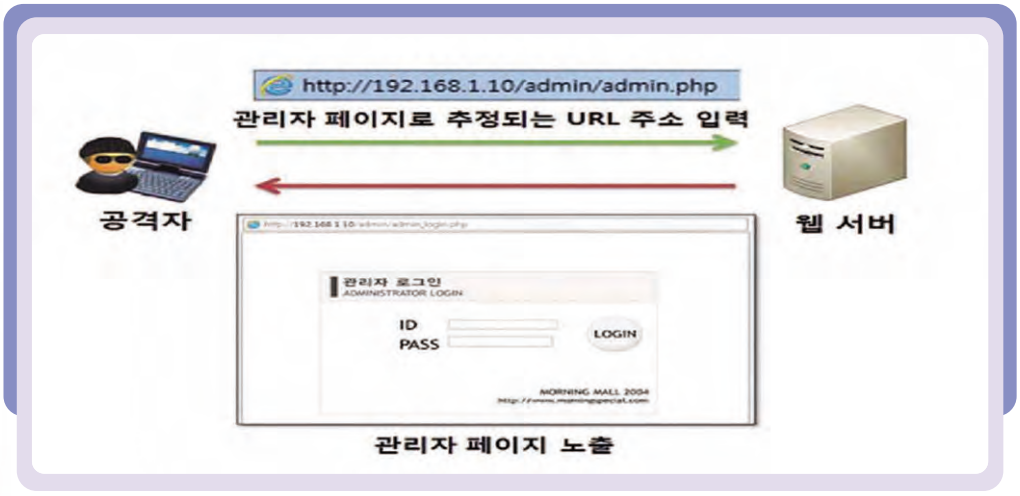
02 홈페이지 보안취약점 사례 및 대응

1. 관리자페이지 노출 취약점

가. 취약점 설명 및 사례

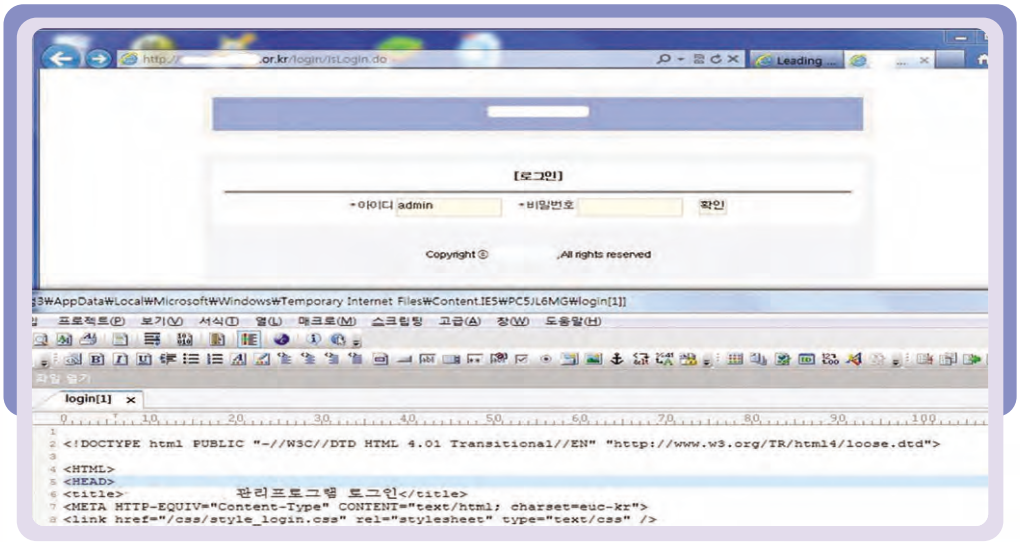
1) 취약점 설명

관리자페이지가 인터넷을 통해 접근 가능할 경우, 공격자의 주 타켓이 되어, 공격자의 SQL 인젝션, Brute-Force 공격 등 다양한 형태의 공격의 빌미를 제공하게 되는 취약점



2) 사례

가) OO기관의 경우 관리자 외의 IP에서 관리자페이지로 접근이 가능함. 소스보기로 관리프로그램 로그인 페이지 확인



나) OO병원은 유추하기 쉬운 URL을 사용하여 관리자페이지의 임의 접근이 가능

나. 점검방법

직접 점검하는 방법으로 점검하고 구글 웹사이트를 활용하여 추가 점검

1) 직접 점검하는 방법

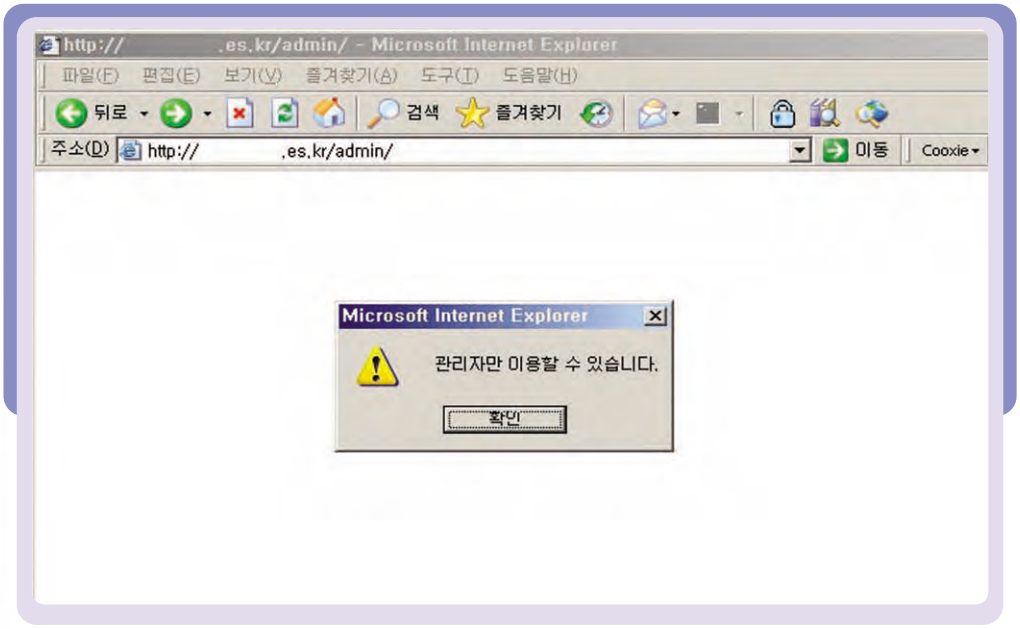
가) 관리자 페이지 위치를 알지 못할 경우 일반적으로 많이 사용하는 관리자 페이지 명을 입력하여 관리자 페이지가 존재하는지 점검

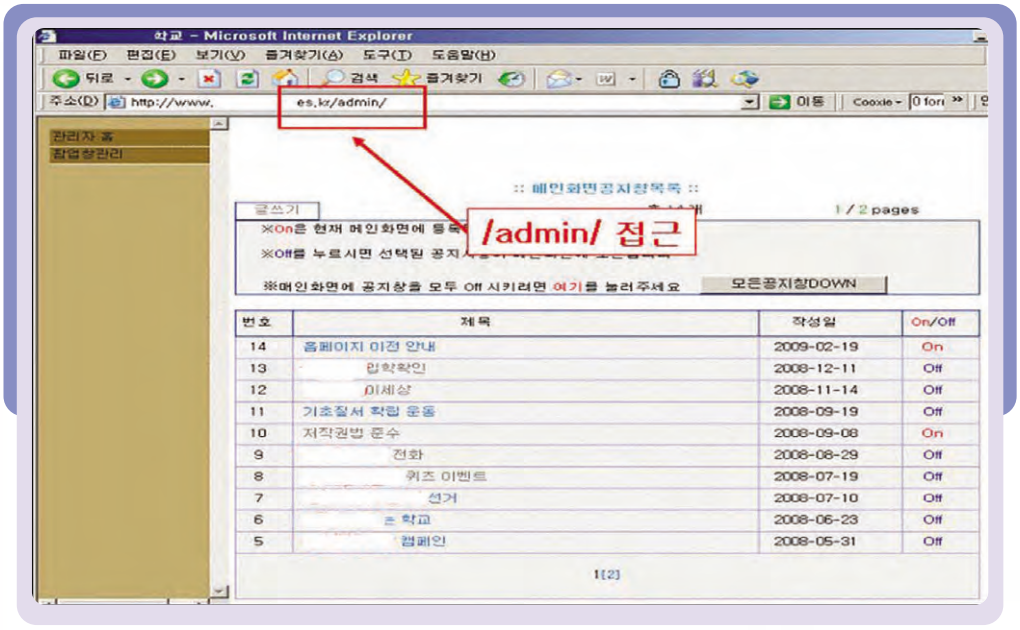
관리자 페이지 주소 예)

http://admin.ecsc.es.kr
http://www.ecsc.es.kr/admin/
http://www.ecsc.es.kr/manager/
http://www.ecsc.es.kr/master/
http://www.ecsc.es.kr/system/
http://www.ecsc.es.kr/adm/

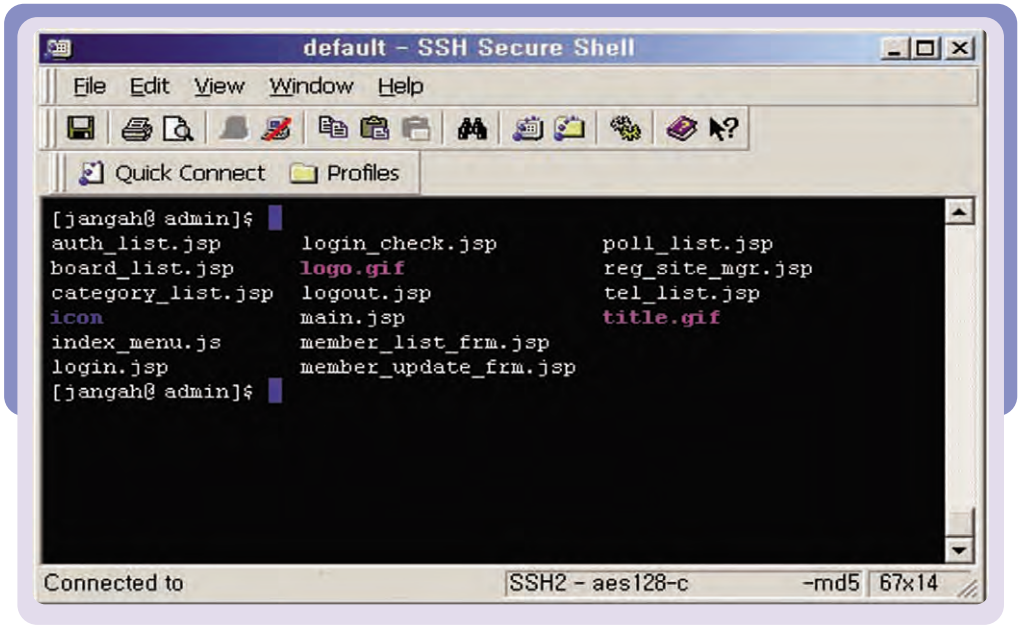
02 홈페이지 보안취약점 사례 및 대응

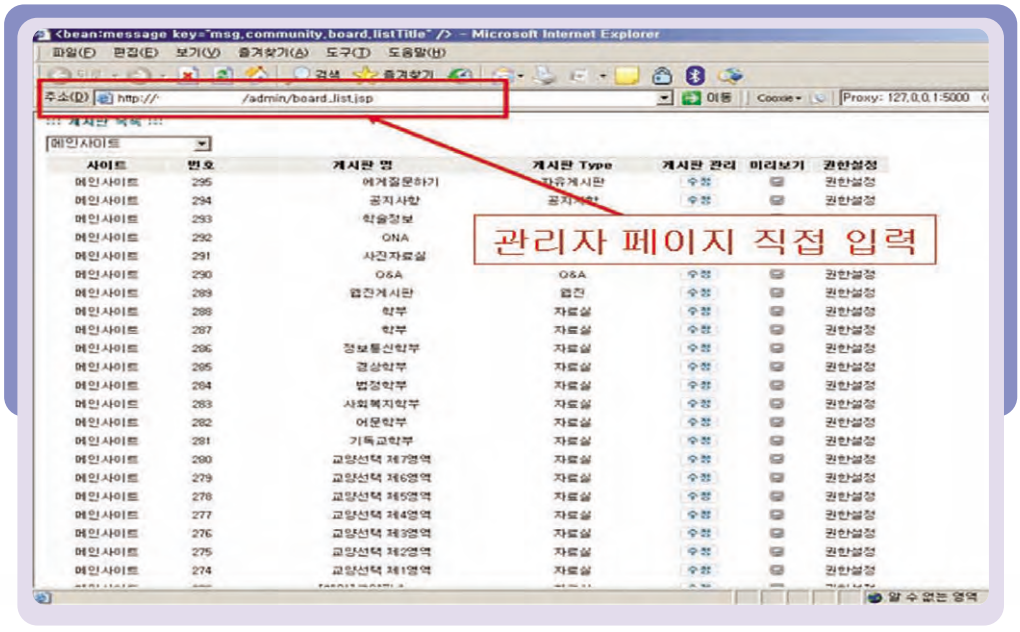
사용자 인증을 통과하여 페이지에 접속한 후 인증과정 없이 중간 페이지에 접속을 시도하여 접속이 가능한지 점검





나) 웹서버 내부 파일명을 알고 있을 경우 웹서버에서 관리자 페이지로 이용되는 웹페이지 (파일) 목록을 확인하여 웹 브라우저를 통해 직접 접속을 시도





02 홈페이지 보안취약점 사례 및 대응

2) 구글 검색엔진을 통한 점검 방법

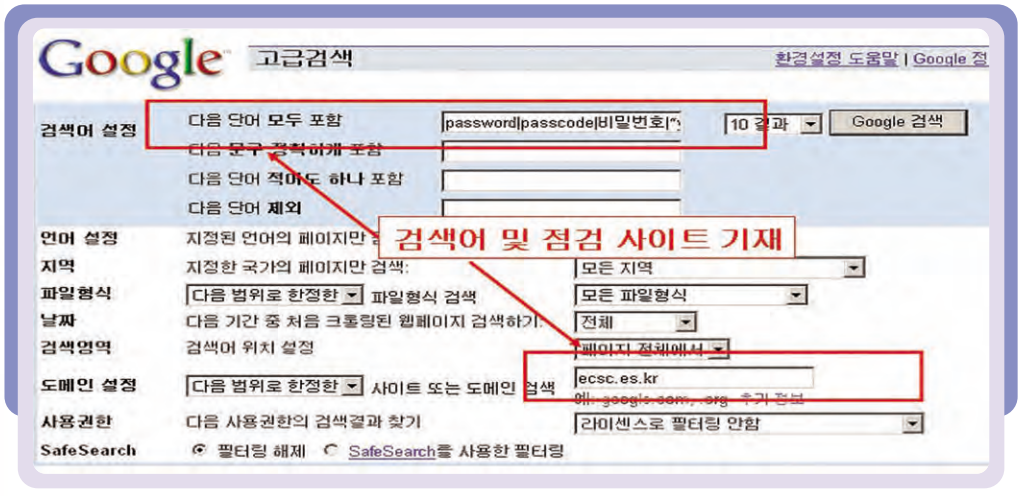
가) 구글(www.google.co.kr) 사이트에 접속 후 고급 검색으로 이동

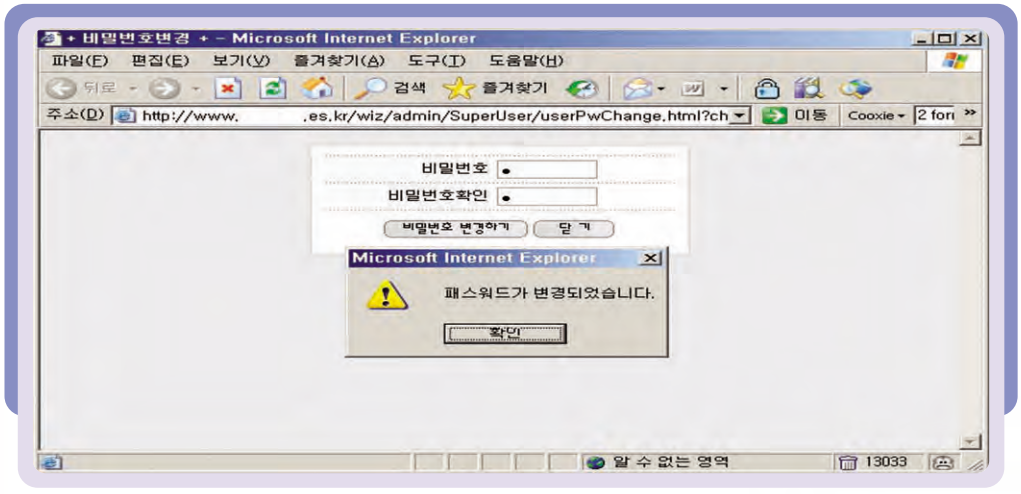
나) 도메인 설정에 해당 웹서버 주소를 입력하고 검색어 입력란에는 다음의 검색어를 각각 입력하여 ID/PW 및 관리자 웹서버(관리자 로그인 페이지 등) 노출 페이지를 검색

비밀번호 검색 예)

login|logon
password|passcode|비밀번호|"your password is"|"당신의 비밀번호는"
admin|administrator

※ 검색어는 공백(빈칸)이 포함되지 않아야함. 공백을 포함하기 위해서는 인용부호(")를 이용함
(예. "your password is")





다. 대응방안

1) 웹 서버 내에서의 조치

가) 홈페이지 관리자 페이지는 관리용으로 지정된 디렉터리에만 보관하여 운영

나) 홈페이지 관리자 페이지에 임의의 사용자가 접근할 수 없도록 접근권한을 설정하여, 접근할 수 있는 권한을 가진 단말기만 접근 가능 하도록 설정

① 윈도우즈 서버의 『인터넷 서비스 관리자(IIS)』조치방법

[설정] → [제어판] → [관리도구] → [인터넷 서비스 관리자]실행 → [인터넷정보 서비스]에서 관리자(admin)디렉터리를 선택 후 마우스 오른쪽 클릭 → [등록정보] → [디렉터리보안] → [IP주소 및 도메인 이름제한] → [편집]을 통해 관리자 IP만 등록 [확인]을 통해 임의 사용자 접근을 제한함

② 유닉스 및 리눅스의 『아파치(Apache)』조치방법

아파치 웹서버의 설정 파일인 httpd.conf 파일에서 "Directory"내의 AllowOverride를 찾아 AutoConfig 또는 All을 추가함
예) /usr/local/www/admin 폴더를 192.168.10.10만 허용하고 모두 차단할 경우

```
〈Directory "/usr/local/www/admin/"〉
    Order allow,deny
    Deny from all
    allow from 192.168.10.10
〈/Directory〉
```

가) 웹서버의 웹 관리자 메뉴의 접근을 특정 네트워크 대역으로 제한하여 IP주소까지 인증 요소로 체크하도록 웹 관리자 사용자 인터페이스를 개발

나) 홈페이지 관리자 페이지는 주소를 직접 입력하여 인증과정 없이 접속하지 못하도록 관리자 페이지 각각에 대하여 관리자 인증을 위한 세션 관리를 수행

2) 구글 검색기에 노출된 경우의 조치

가) 구글에 노출된 홈페이지 관리자 페이지 정보의 캐시 삭제를 요청

나) 웹서버에 노출 방지 표준(인터넷검색엔진배제표준)을 이용하여 개인정보가 포함된 주소를 지정하는 robots.txt 파일을 만들어 가상서버의 최상단 폴더에 저장하거나 해당 페이지의 HTML 안에 메타태그를 입력

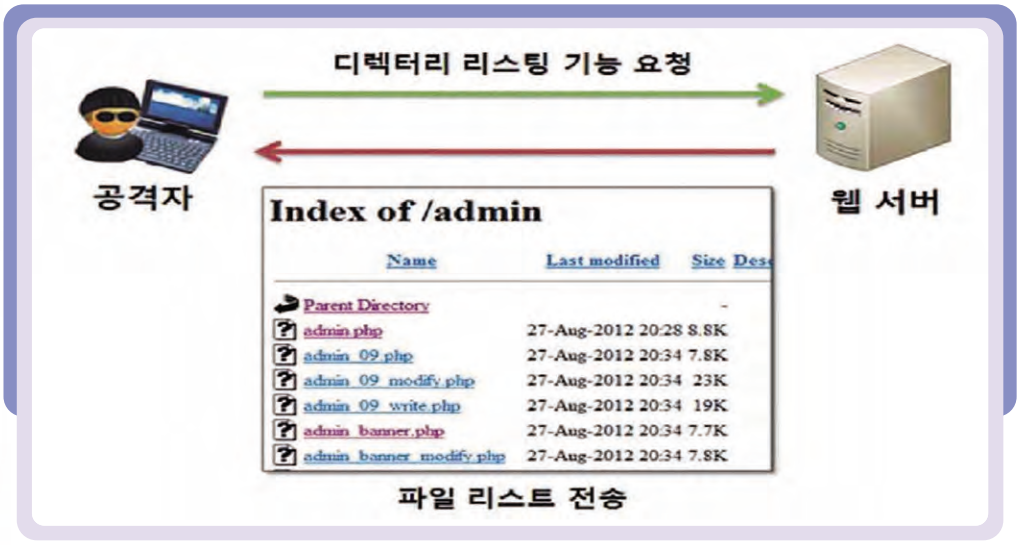
02 홈페이지 보안취약점 사례 및 대응

2. 디렉터리 나열 취약점

가. 취약점 설명 및 사례

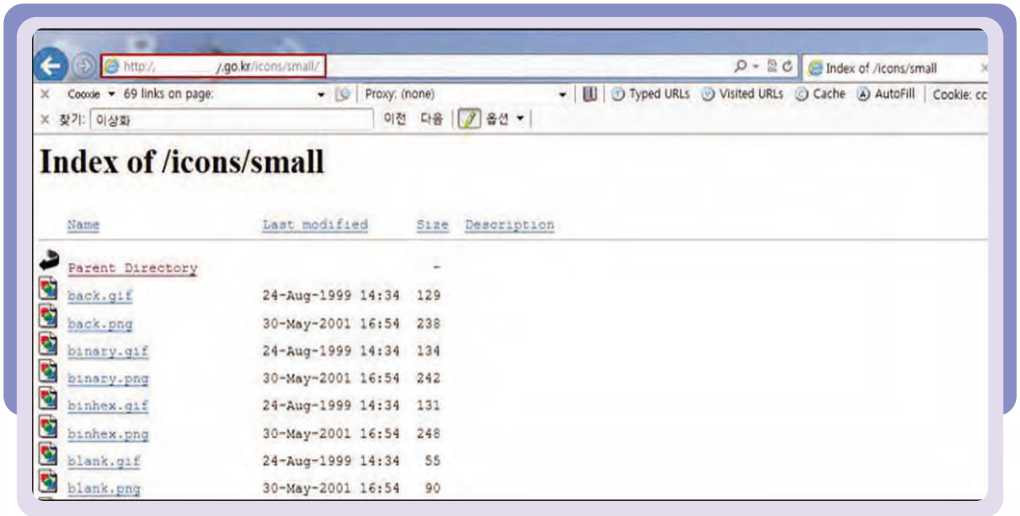
1) 취약점 설명

서버내의 모든 디렉터리 혹은 중요한 정보가 포함된 디렉터리에 대해 인덱싱이 가능하게 설정되어 중요파일 정보가 노출될 수 있는 취약점



2) 사례

가) OO기관의 일부 페이지에서 디렉터리 나열 취약점이 존재하여 내부구조를 확인할 수 있음



나) OO대학은 일부 페이지에서 디렉터리 나열 취약점이 존재하여 웹서버 내 파일 목록 열람 가능

나. 점검방법

1) 직접 점검방법

가) 점검 대상 웹 사이트의 하위 디렉터리 정보를 사전에 모두 확인

나) 웹 루트(Root)의 모든 하위 디렉터리에 대해서 웹 브라우저에 해당 주소를 입력하여 디렉터리 리스팅 취약점 존재여부를 점검

> ※ 참고사항
> http://ecsc.ms.kr/ 이란 웹서버의 웹 루트 밑에 "file"이란 디렉터리가 있다면 웹 브라우저의 URL 주소 입력란에 http://ecsc.ms.kr/file/ 을 입력함. 이때 "file" 디렉터리 하위 내용이 모두 화면에 출력된다면 디렉터리 리스팅 취약점이 존재함을 의미하며 반드시 맨 끝의 '/' 까지 입력함. 모든 디렉터리에 대해 디렉터리 리스팅 취약점 존재 여부를 확인함

2) 구글을 통한 점검

가) 구글 사이트에 접속 후 고급 검색으로 이동

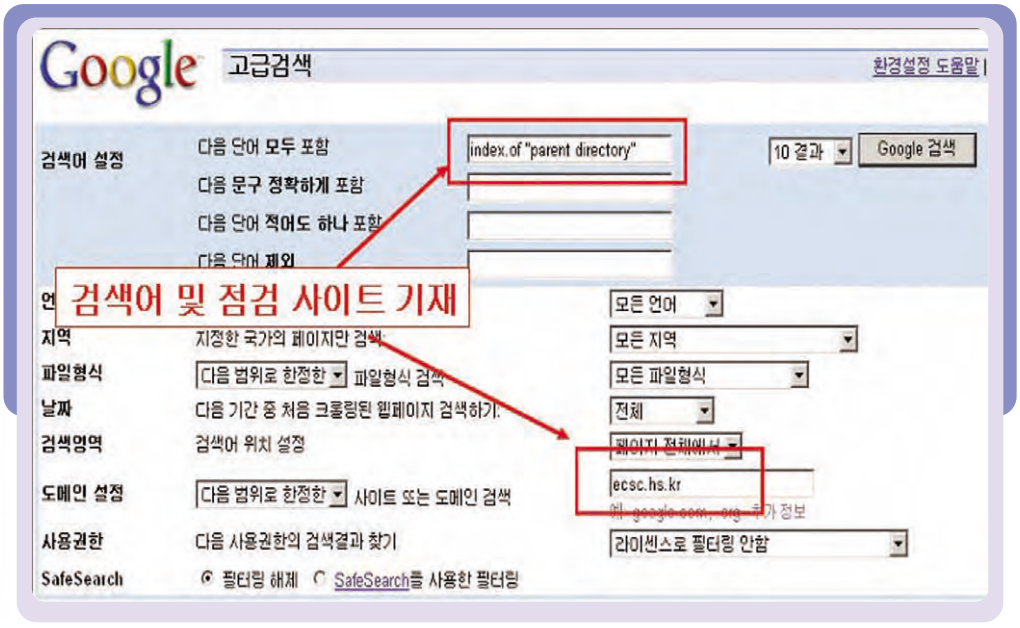
나) 도메인 설정 란에는 해당 사이트 주소를 입력하고, 검색창에는 다음을 입력하여 디렉터리 목록이 저장된 페이지를 검색

웹 어플리케이션(웹서버)	검색어
IIS	Parent Directory
Apache	Directory Listing
Tomcat	Directory Listing
기타	Index of

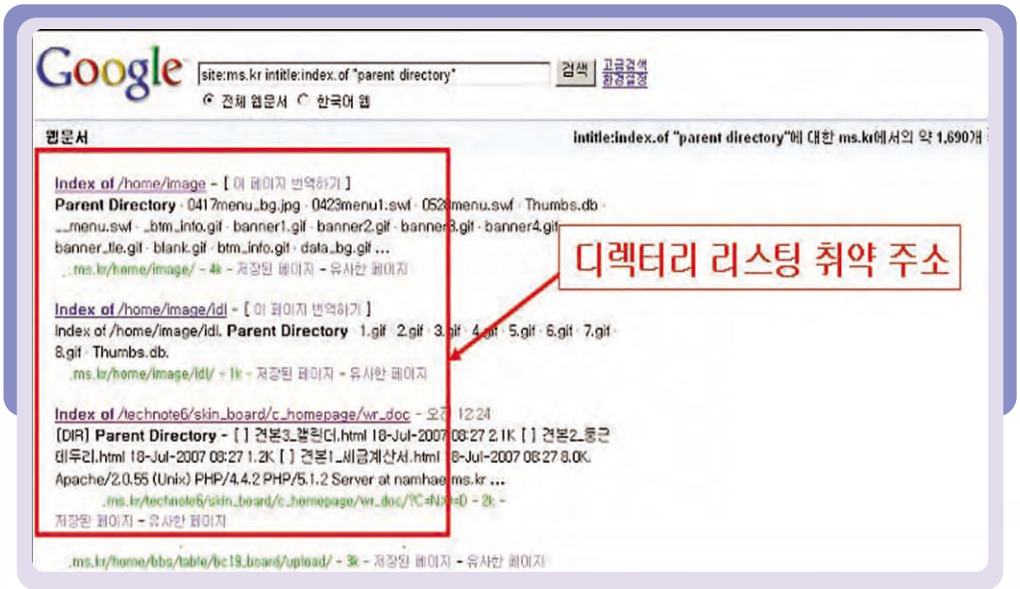
02 홈페이지 보안취약점 사례 및 대응

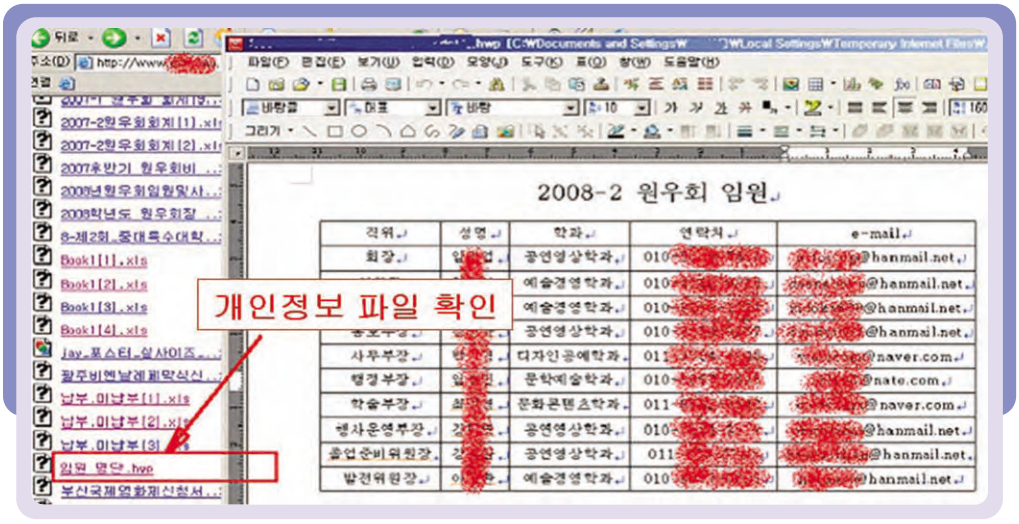
IIS 웹서버 검색 예)

index.of "Parent Directory"



다) 검색 결과에서 해당 사이트의 디렉터리가 노출 되었는지 확인





다. 대응방안

취약점이 발견된 웹서버의 설정 값을 수정하여 취약점을 제거하고 구글 검색사이트에 취약점을 재점검하여 노출 되었을 경우 정보 제거를 요청

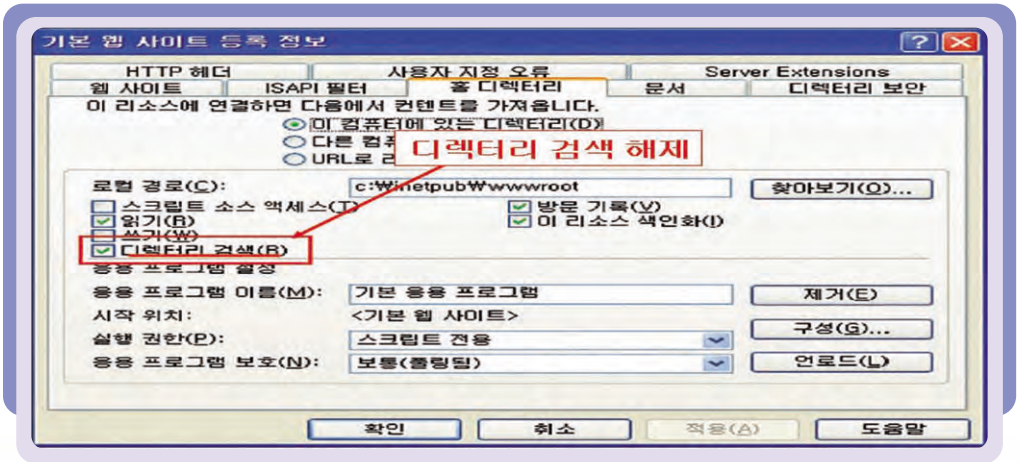
1) 웹 서버 내에서의 조치

가) Windows의 IIS 조치

윈도우 2000 서버 계열 및 윈도우 2003 서버 계열의 운영체제로서 '인터넷 정보서비스(IIS)'를 이용하는 웹서버이며 아래 작업을 통해 디렉터리 리스팅 취약점을 차단 가능

① 〔제어판〕 〔관리도구〕 〔인터넷 서비스 관리자〕메뉴에서〔기본 웹 사이트〕를 마우스 오른쪽 클릭, '속성' '기본 웹 사이트 등록 정보'를 선택

② '기본 웹 사이트 등록 정보'에서 '홈 디렉터리' 부분을 선택, '디렉터리 검색(B)'의 체크를 해제



02 홈페이지 보안취약점 사례 및 대응

나) Unix, Linux의 Apache 조치

유닉스 및 리눅스 운영체제로서 아파치(apache)를 이용하는 웹서버이며 아래 작업을 통해 디렉터리 리스팅 취약점을 차단 가능

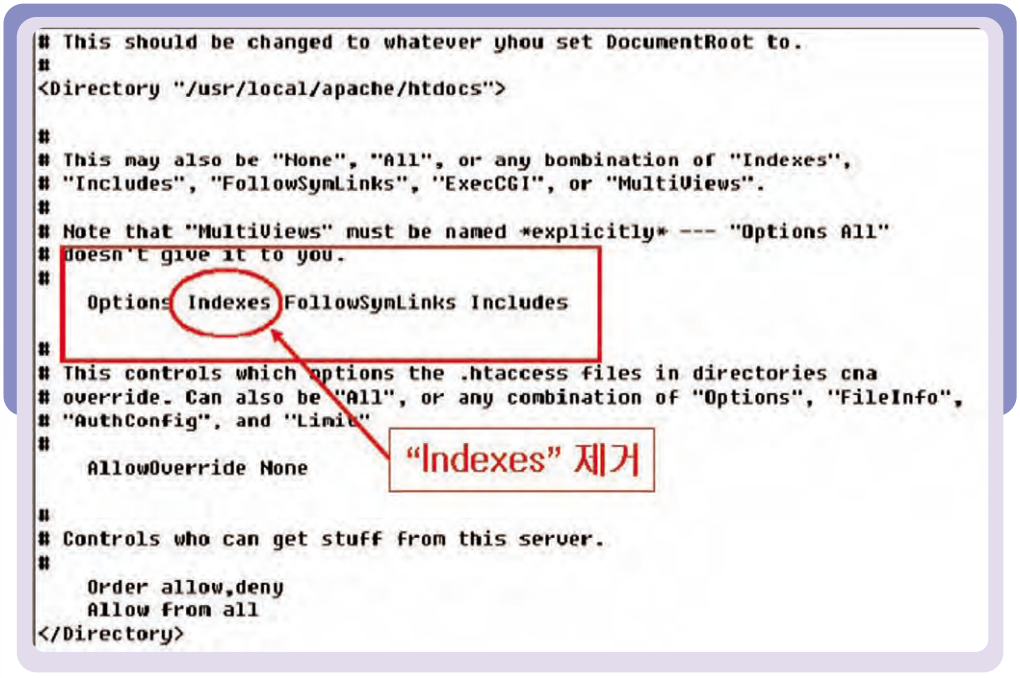
① 서버에서 아파치 서버의 설정파일인"httpd.conf"파일을 검색

IIS 웹서버 검색 예)

index.of "Parent Directory"

② httpd.conf의 파일 내용 중 Options 항목 뒤'Indexes'라는 지시어를 지우고 저장

③ 설정한 정보를 적용하기 위해 웹 서비스의 데몬(daemon)을 재시작



다) Unix, Linux의 Tomcat 조치

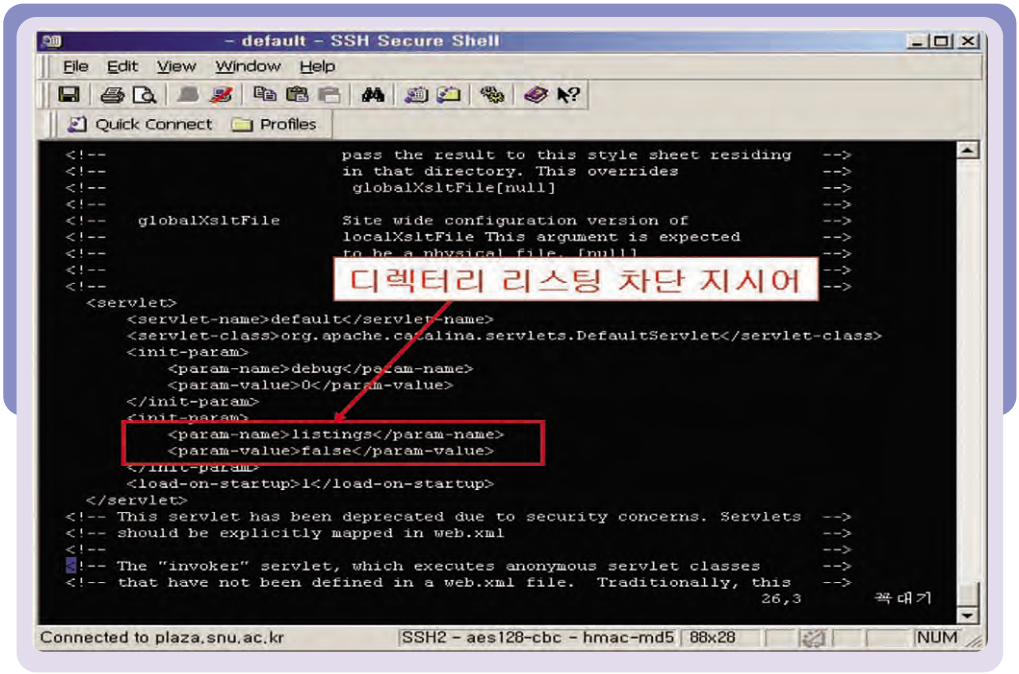
유닉스 및 리눅스 운영체제로서 톰캣(Tomcat)을 이용하는 웹서버이며 아래 작업을 통해 디렉터리 리스팅 취약점을 차단 가능

① 서버에서 Tomcat의 설정파일인'web.xml'파일을 검색

다) Unix, Linux의 Tomcat 조치

예) find / -name web.xml

② web.xml의 파일 내 디렉터리 리스팅을 차단하는 지시어(listings, false)를 설정
③ 설정한 정보를 적용하기 위해 웹 서비스의 데몬(daemon)을 재시작



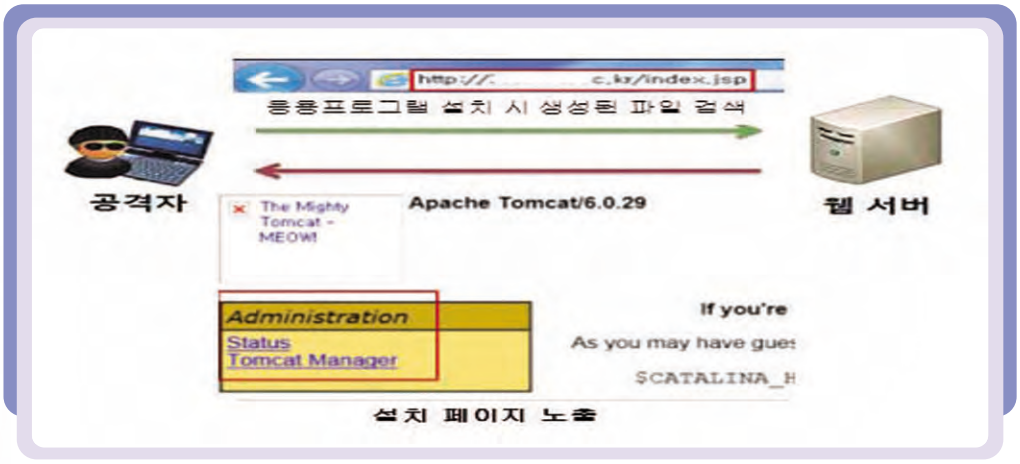
02 홈페이지 보안취약점 사례 및 대응

3. 시스템 관리 취약점

가. 취약점 설명 및 사례

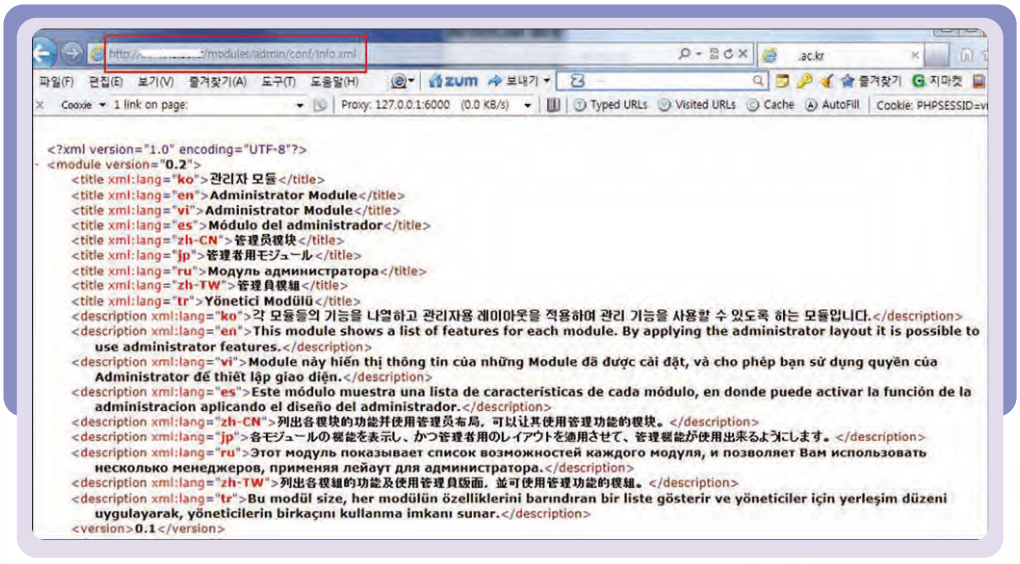
1) 취약점 설명

응용프로그램(Apache 등) 설치 중에 생성되는 설치 파일 및 임시 파일이 존재하거나 웹상에서 윈도우 로그인 창이 노출되는 등 시스템상 설정 미비로 인해 발생하는 취약점

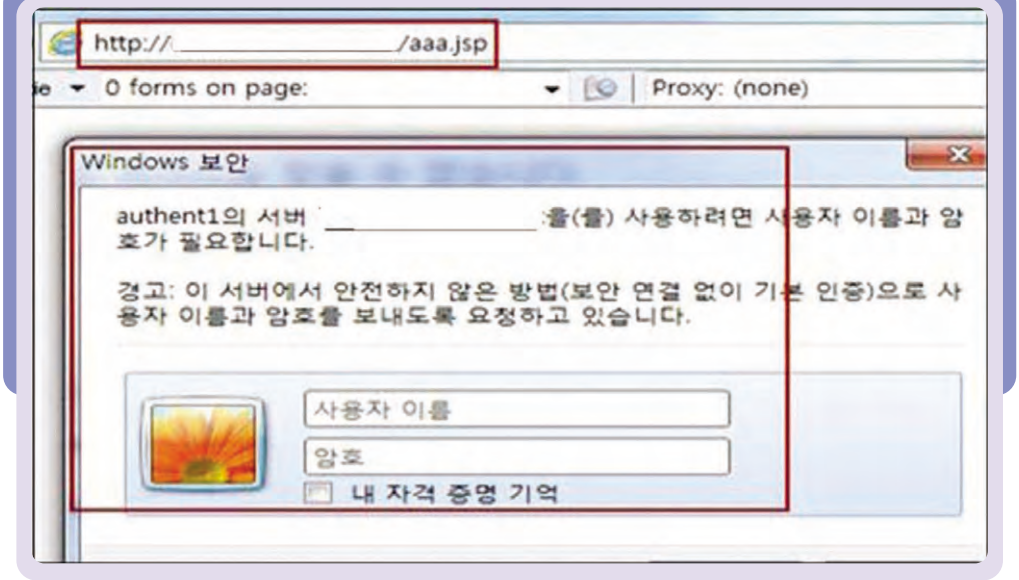


2) 사례

가) OO대학은 응용프로그램 설치 중에 생성되는 설치 파일 및 임시파일을 통해 응용프로그램의 취약점 정보 수집이 가능



나) OO기관은 웹에서 윈도우 원격 로그인 창이 노출되는 시스템 관리 취약점이 존재



나. 점검방법

1) 응용프로그램 설치 시 생성되는 기본 경로(ex. fckeditor, apache, phpMyAdmin 등)를 검색하여 불필요한 설치 파일이 있는지 확인

※ 시스템관리 취약점은 시스템 설정에 관련된 취약점이므로 범위가 방대하고 응용프로그램 별로 기본 설치 경로가 상이하므로 각 환경에 맞는 진단 방법을 검색

구 분	설치시 생성되는 기본 파일 위치
아파치(Apache)	ServerRoot/cgi-bin/
톰켓(Tomcat)	TOMCAT_HOME/examples
웹투비(WebToB)	ServerRoot/cgi-bin/

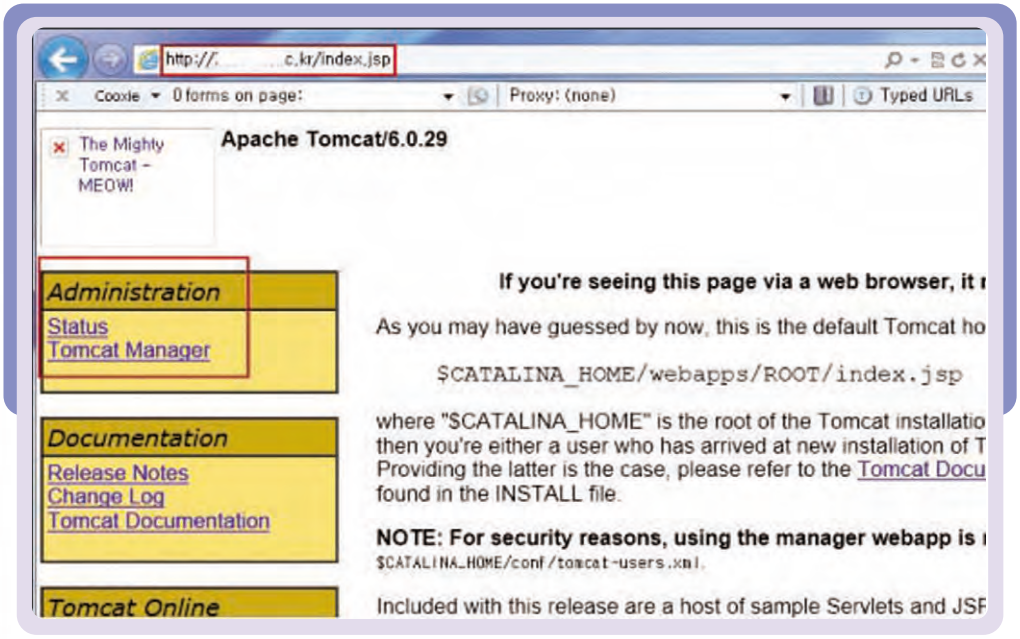
검색 방법

예) find /[웹서버디렉터리] –name "phpinfo.php"

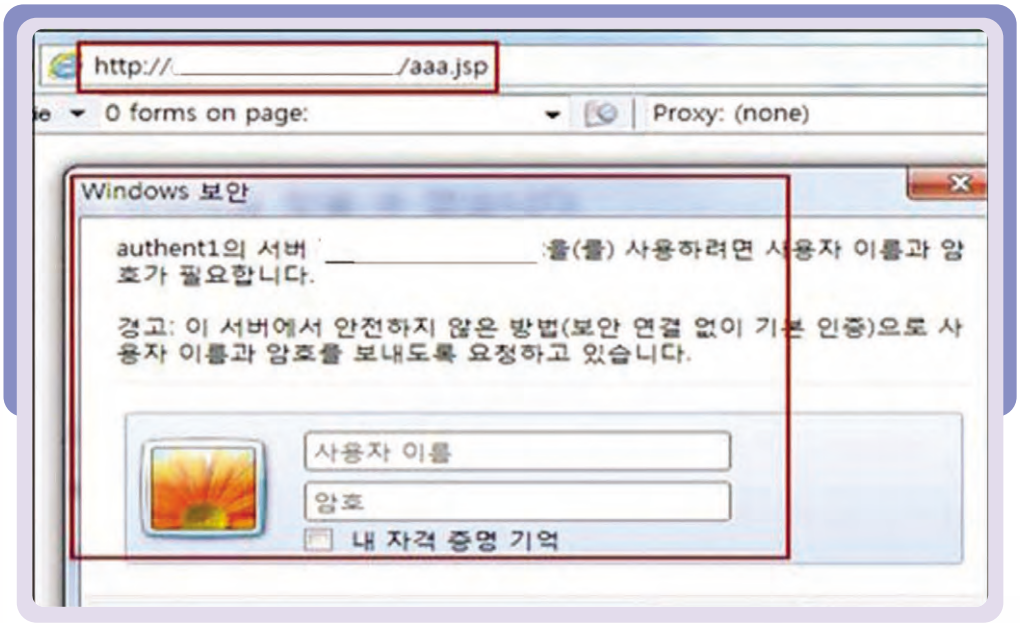
또는 모든 PHP 파일에 아래의 문자열이 포함되어있는 파일을 조회함
(단, 파일내의 문자열 검색 시 시스템에 과부하가 발생할 수 있음)

예) grep "phpinfo( )" *php

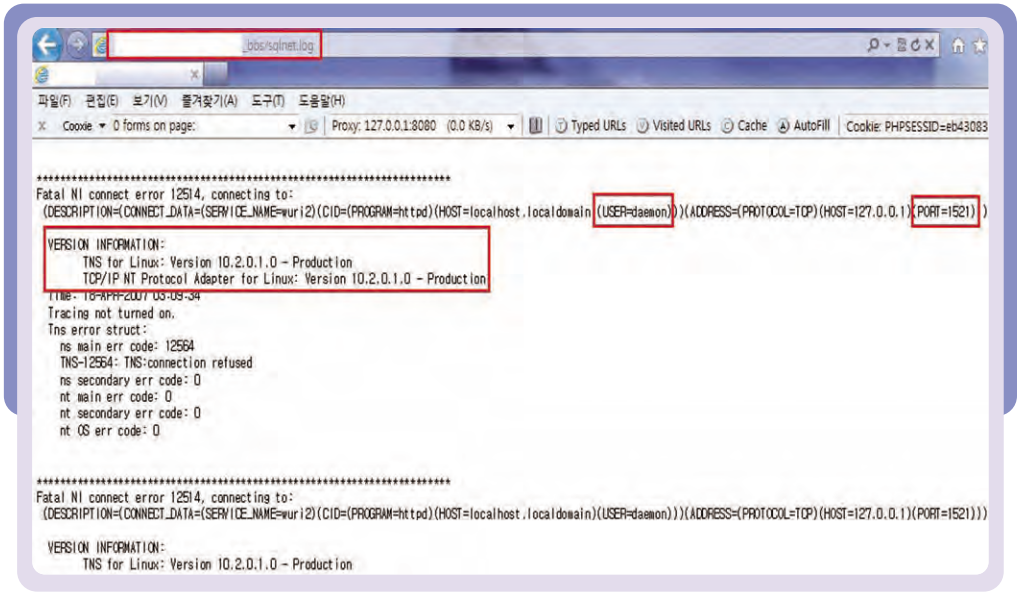
02 홈페이지 보안취약점 사례 및 대응



2) 존재하지 않는 파일 또는 디렉터리만을 요청 시 아래와 같은 로그인 페이지가 노출되는지 확인



3) SQL 로그 파일이 웹 서버에 존재하는지 검색(sqlnet.log)



4) 기타 시스템 환경에 따라 발견되는 취약점이 다양하기 때문에 운영하는 시스템에 맞는 취약점 점검 필요

다. 대응방안

1) 웹 서버 내에서의 조치

가) 웹 서버에 응용프로그램 설치 시 임시 생성되는 파일은 설치가 완료되면 즉시 삭제

나) 웹 서버 소스상에 설정이 잘못된 것은 없는지, 시스템 보안 설정이 미비한지 점검하여 해당 시스템에 가장 알맞게 설정

다) 정기적으로 웹서버의 불필요 파일을 검색하여 제거

■ 일반적으로 존재하는 백업파일 유형

구분	내용
*.bak	Edit plus, Ultra Edit 등의 작업을 할 경우 기본 설정에 의해 백업파일이 생성된다.
ws_ftp.log	WS FTP를 사용하여 어플리케이션을 업로드 한 경우 로그파일에 디렉터리 구조, 숨겨진 파일들을 알아낼 수 있다.
*.tar.gz	주로 웹 어플리케이션을 압축한 형태로 존재 한다.
*.zip	주로 웹 어플리케이션을 압축한 형태로 존재 한다.
파일명.날짜	main.jsp.20121111 와 같은 형식의 백업파일을 사용한다.
*.html.old	기존 파일을 백업하는 개념으로 old를 사용한다.

02 홈페이지 보안취약점 사례 및 대응

4. 불필요한 Method 허용 취약점

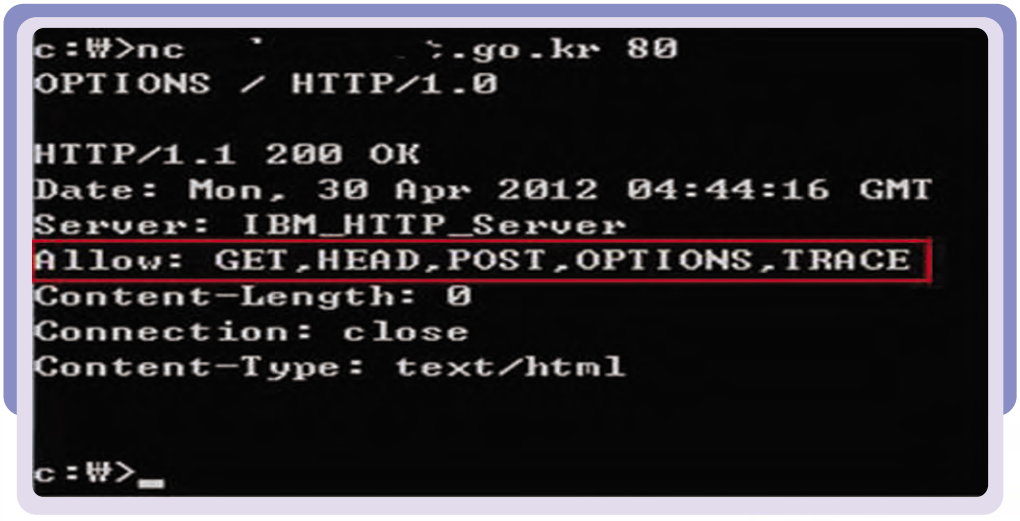
가. 취약점 설명 및 사례

1) 취약점 설명

웹 서비스 제공 시 불필요한 Method(PUT, DELETE, OPTIONS 등) 허용으로 공격자에 의해 악성파일을 업로드 하거나 중요파일 삭제가 가능해지는 취약점

2) 사례

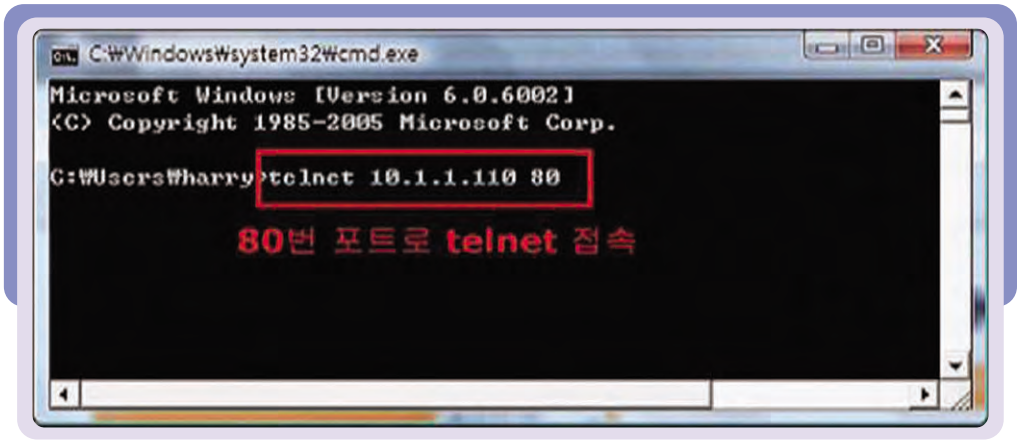
가) OO기관은 GET, POST 외에 Method를 허용하여 웹서버 정보 획득, 파일업로드 등의 공격이 가능함



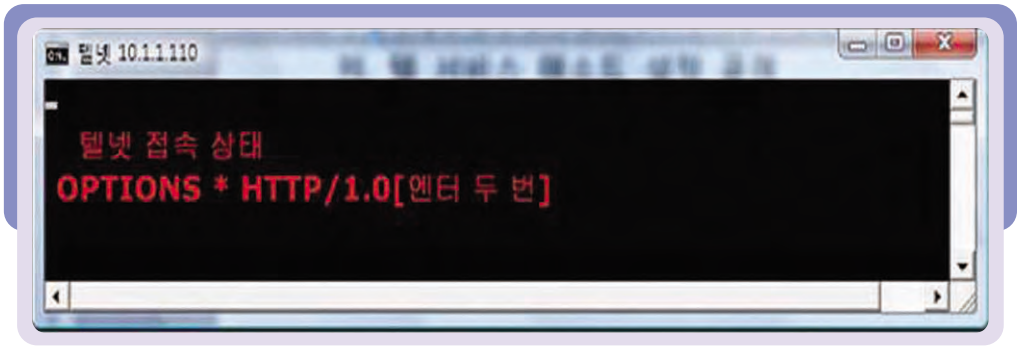
나. 점검방법

1) 텔넷에 접속하여 불필요한 Method가 활성화 되어 있는지 확인

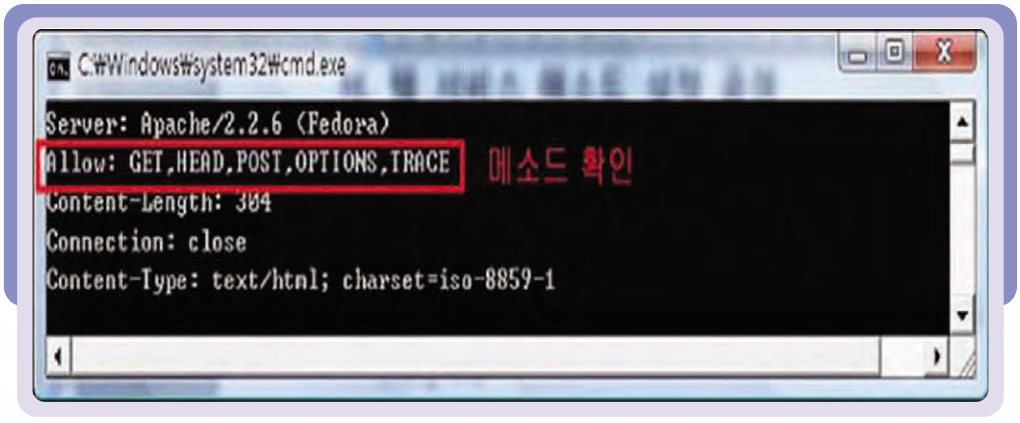
가) 텔넷 접속



나) Method 정보 요청



다) PUT, DELETE Method 활성화 여부 확인



다. 대응방안

1) 웹 서버 내에서의 조치

가) 홈페이지 운영에 불필요한 Method(PUT, DELETE, OPTIONS 등) 비활성화

※ 웹 서비스 제공 시 필요한 GET, POST 이외 사용되지 않는 Method 사용 제한

나) 웹 서버 홈 디렉터리 아래 conf 디렉터리의 web.xml파일에서 불필요한 Method 비활성화 (삭제 또는 주석)를 위한 보안설정

※ 〈http-method〉DELETE〈/http-method〉 : 차단할 Method를 등록
※ 모든 페이지(/*)에 대하여 PUT, DELETE, OPTIONS Method 사용 제한

안전한 서버 설정(web.xml)

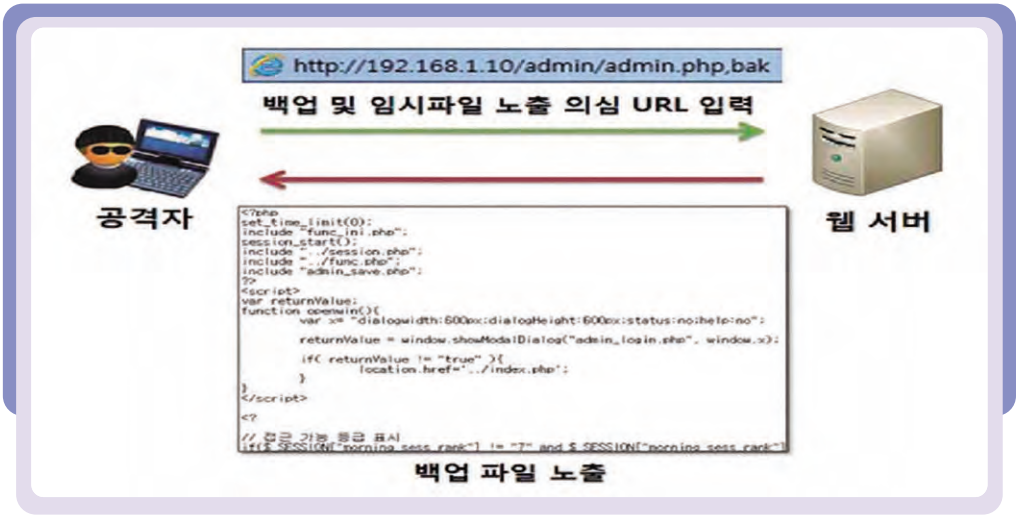
```
...
 <security-constraint>
  <web-resource-collection>
    <web-resource-name>Method Block</web-resource-name>
    <http-method>DELETE</http-method>
    <http-method>PUT</http-method>
  </web-resource-collection>
 <auth-constraint>
 <role-name></role-name>
</auth-constraint>
</security-constraint>
...
```

5. 취약한 파일 존재 취약점

가. 취약점 설명 및 사례

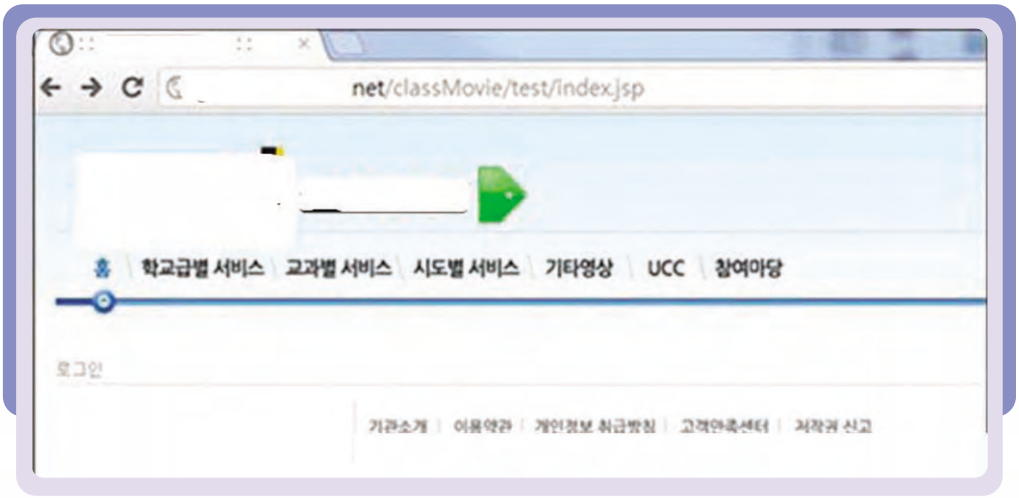
1) 취약점 설명

웹 루트 하위에 내부 문서나 백업파일, 로그파일, 압축파일과 같은 파일이 존재할 경우 파일명을 유추하여 파일명을 알아내고, 직접 요청하여 해킹에 필요한 서비스 정보를 획득할 수 있는 취약점



2) 사례

가) OO서비스는 웹서버 개발 시 개발·보수 등의 이유로 임시 페이지로 인해 시스템 정보가 노출



나) OO기관은 테스트용으로 사용했던 페이지가 존재

나. 점검방법

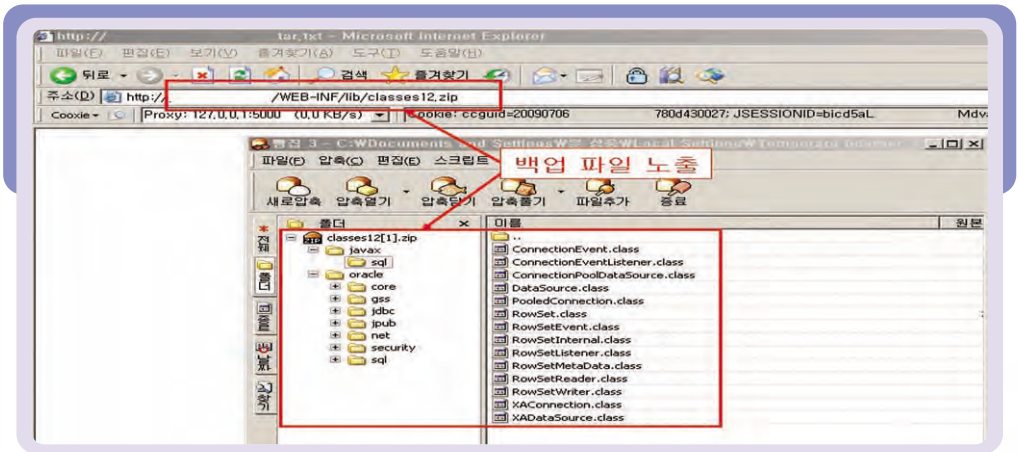
1) 웹서버의 가상 디렉터리로 이동하여 다음에 제시되는 확장자의 파일을 찾아 불필요한 정보가 포함 되었는지 여부를 판단

※ 문서파일일 경우 내용에 개인정보 등의 주요정보 존재여부 확인 필요

구 분	검색할 파일의 형식(확장자)
압축파일	.zip, .rar, .alz, .tar, .gz, .gzip 등의 압축파일
백업파일	.bak, .org 등
로그파일	.log, .txt 등
설정파일	.sql, .ini, .bat 등
문서파일	.hwp, .doc, .xls, .ppt, .pdf 등
기타	test.*, imsi.* .tmp 등

Windows의 검색 방법
예) dir /[웹서버디렉터리] /s *.bak

Unix, Linux의 검색 방법
예) find /[웹서버디렉터리] – name ".*.bak"



2) PHP 언어로 개발한 웹서버의 경우 아래와 같은 정보를 출력하는 웹페이지(phpinfo.php)가 존재하는지 점검

검색 방법
예) find /[웹서버디렉터리] –name "phpinfo.php"
또는 모든 PHP 파일에 아래의 문자열이 포함되어있는 파일을 조회함
(단, 파일내의 문자열 검색 시 시스템에 과부하가 발생할 수 있음)
예) grep "phpinfo()" *php

다. 대응방안

1) 웹 서버 내에서의 조치

가) 웹서버는 개발과 운영 환경을 분리하여 운영 환경에서 소스 코드 수정 또는 테스트 목적의 임시 파일을 생성하지 않도록 함

나) 웹 서버의 디렉터리에 존재하는 기본 설치 파일, 임시 및 백업 파일을 조사하여 웹 사용자가 접근하지 못하도록 조치

구 분	설치시 생성되는 기본 파일 위치
아파치(Apache)	ServerRoot/cgi-bin/
톰켓(Tomcat)	TOMCAT_HOME/examples
웹투비(WebToB)	ServerRoot/cgi-bin/

다) 정기적으로 불필요 파일을 검색하여 제거함

■ 일반적으로 존재하는 백업파일 유형

구분	내용
*.bak	Edit plus, Ultra Edit 등의 작업을 할 경우 기본 설정에 의해 백업파일이 생성된다.
ws_ftp.log	WS FTP를 사용하여 어플리케이션을 업로드 한 경우 로그파일에 디렉터리 구조, 숨겨진 파일들을 알아낼 수 있다.
*.tar.gz	주로 웹 어플리케이션을 압축한 형태로 존재 한다.
*.zip	주로 웹 어플리케이션을 압축한 형태로 존재 한다.
파일명.날짜	main.jsp.20121111 와 같은 형식의 백업파일을 사용한다.
*.html.old	기존 파일을 백업하는 개념으로 old를 사용한다.

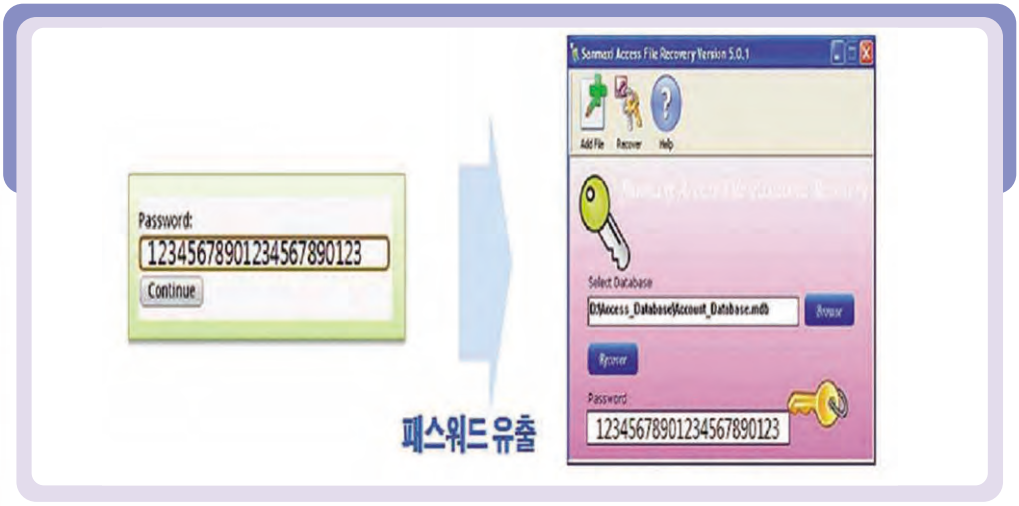
02 홈페이지 보안취약점 사례 및 대응

6. 계정 관리 취약점

가. 취약점 설명 및 사례

1) 취약점 설명

회원가입 시에 안전한 패스워드 규칙이 적용되지 않아서 취약한 패스워드로 회원 가입이 가능할 경우 무차별 대입공격을 통해 패스워드가 누출 될 수 있는 취약점



2) 사례

가) OO학교는 취약한 사용자 계정 생성으로 인해, 추측을 통한 사용자 계정의 이용이 가능

- 계정, 비밀번호 유추를 통해서 admin 계정으로 로그인 시도

– admin 계정의, 비밀번호 admin1234로 로그인에 성공

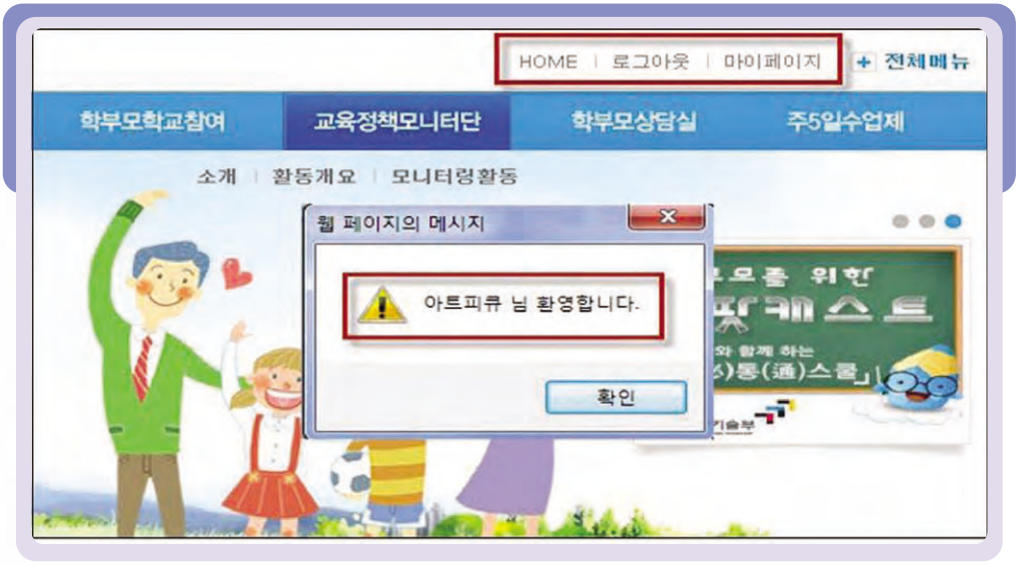
나. 점검방법

1) 로그인 페이지에서 추측 가능한 계정으로 로그인 시도

(EX : master, webmaster, admin, administrator, root, manager, test, masterweb)

가) 'test'계정 (ID/Password)으로 로그인 시도

나) 결과(로그인 성공)



다. 대응방안

1) 홈페이지 개발 보안 조치

가) 사용자가 취약한 패스워드를 사용할 수 없도록 패스워드 생성규칙을 강제 할 수 있는 로직을 적용

■ 패스워드 생성규칙

구 분	내 용
패스워드 생성규칙	• 세가지 종류 이상의 문자구성으로 8자리 이상의 길이 • 두가지 종류 이상의 문자구성으로 10자리 이상의 길이
패스워드 생성 금지규칙	• 간단한 문자(영어단어 포함)나 숫자의 연속사용은 금지 • 키보드 상에서 일련화 된 배열을 따르는 패스워드 선택 금지 • 사전에 있는 단어, 이를 거꾸로 철자화한 단어 사용 금지 • 생일, 전화번호, 개인정보 및 아이디와 비슷한 추측하기 쉬운 비밀번호 사용 금지 • 이전에 사용한 패스워드는 재사용 금지 • 계정 잠금 정책 설정 ex)로그인 5회 실패 시 30분 동안 사용중지

나) 제3장 홈페이지 개발 보안 방안의 12. 취약한 계정 생성 허용을 참조, 소스코드를 수정하여 보안취약점을 조치하시길 바람

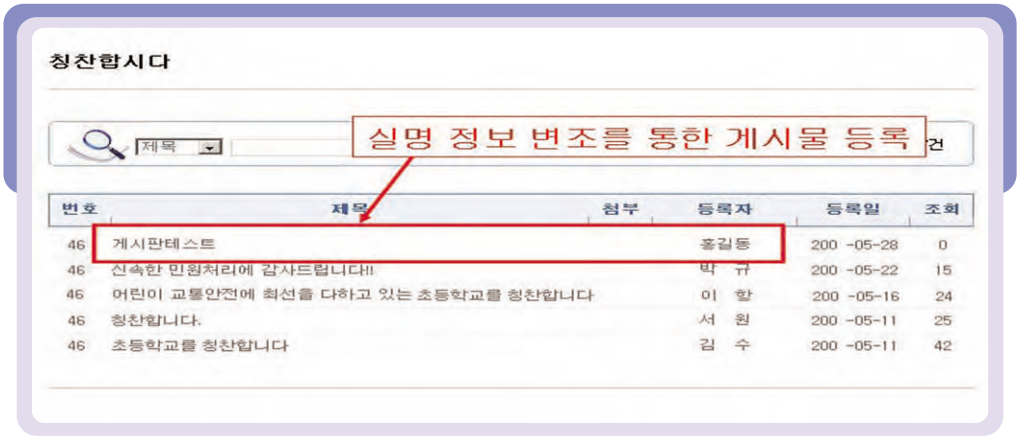
7. 실명인증 취약점

가. 취약점 설명 및 사례

1) 취약점 설명

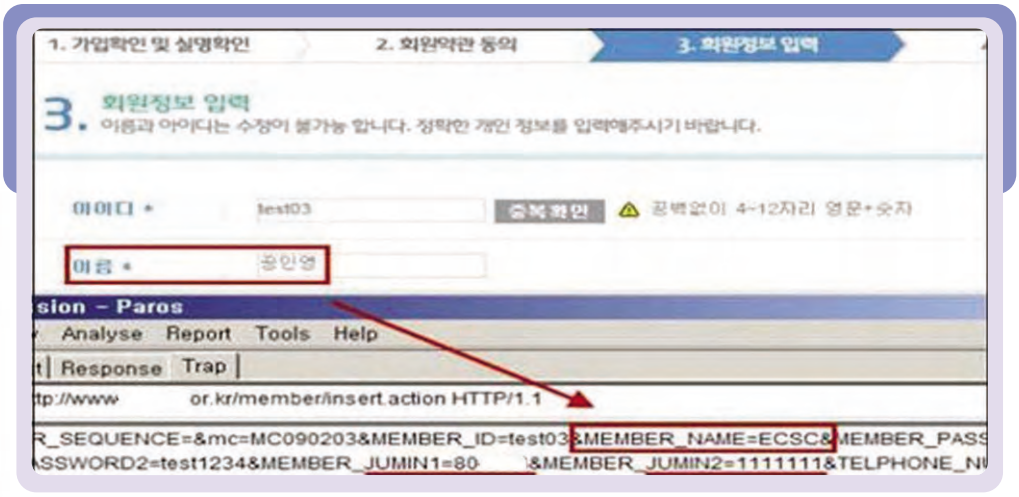
실명인증 우회 취약점은 사용자 본인 확인 과정상에서 취약한 프로그램을 악용하여 사용자 정보를 변조하는 것임. 실명인증 취약점을 통해 관리자로 위장하여 개인정보를 수집하거나 홈페이지 가입 시 제공하는 포인트 등을 악용하는 등의 공격이 발생 가능함

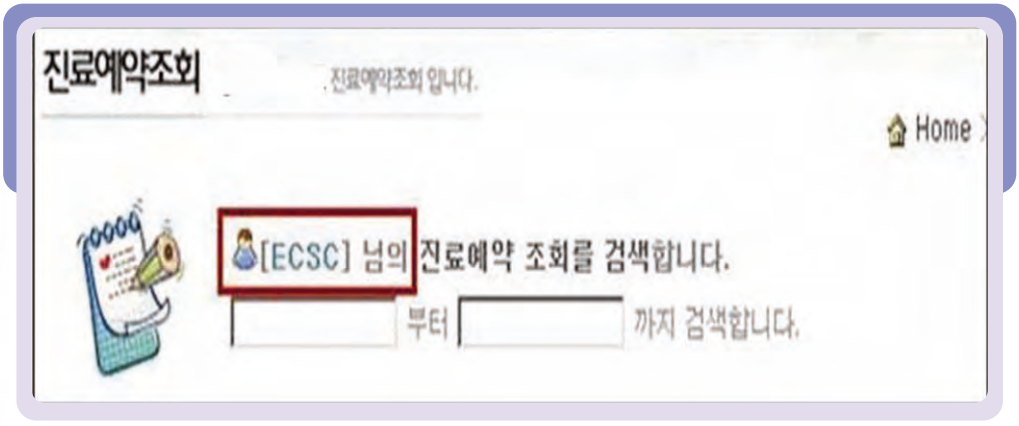
※ 『정보통신망 이용촉진 및 정보보호 등에 관한 법률』에 의거, 국가기관, 지방자치단체 및 공공단체 등에서 운영하는 게시판은 사용자 본인을 확인하기 위한 방법 및 절차가 마련되어야 함



2) 사례

가) OO기관은 정상적으로 실명인증을 받은 후 개인정보 입력 시 이름과 주민등록번호를 변조하여 전송하면 변조된 이름으로 회원가입이 가능





나. 점검방법

1) 웹 프록시(proxy) 프로그램을 이용하여 실명정보를 수정하는 방법으로 취약점 점검을 수행

가) 관리하고 있는 웹서버 내의 실명인증 페이지로 이동
나) 프록시(proxy) 프로그램을 이용하여 실명인증 과정 중에 발생하는 네트워크 트래픽을 모니터링

※ 대표적인 공개용 웹 프록시 프로그램은 아래와 같으며 사용 방법은 해당 프로그램 사용 설명서 참고

	구 분		비 고
1	Paros proxy	http://www.parosproxy.org	Java 기반의 프록시 서버
2	Burp proxy	http://portswigger.net	

다) 실명인증 성공 후의 결과정보를 조작

① 실명인증 우회 과정(아래 그림 참고)

㉠ 공격자는 취약점이 존재하는 웹서버에 정상적인 사용자의 개인정보로 접속하여 실명 인증 수행

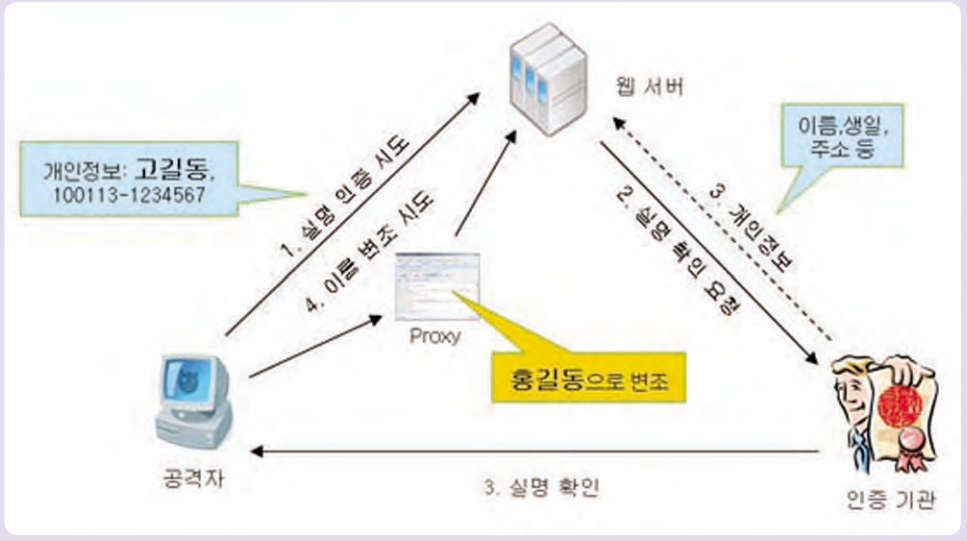
㉡ 웹서버(또는 개인)는 인증기관으로 실명정보 확인을 요청

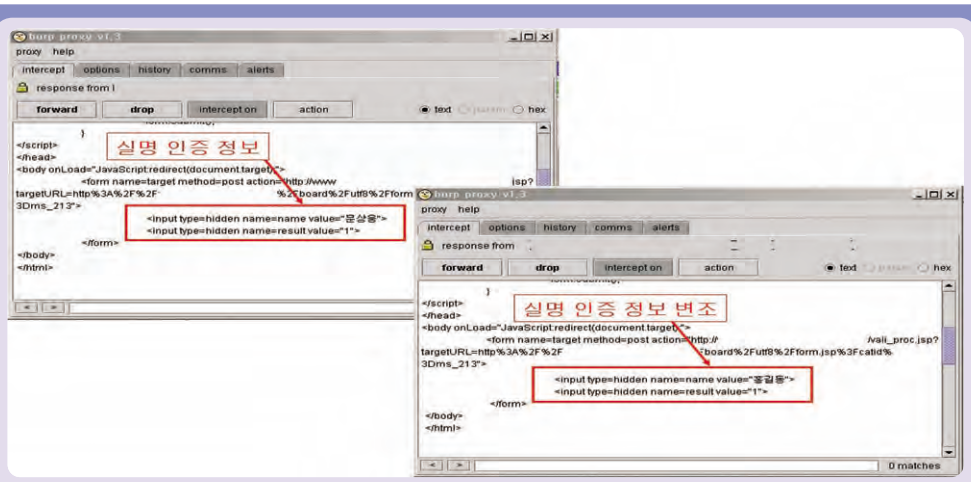
㉢ 실명정보를 확인한 인증기관은 웹서버에 사용자의 나이, 성별, 연락처 등의 개인정보를 전달하며 사용자에게는 "실명인증 성공" 메시지를 전달

㉣ 공격자는 수신한 실명인증 결과를 웹 프록시 툴을 이용하여 임의의 사용자로 변조 후 가입 또는 글 작성을 완료

㉤ 취약점이 존재하는 웹서버는 사용자가 요청한 정보를 검증과정 없이 신뢰하여 변조된 사용자의 가입(또는 글 작성)을 허용

실명 인증 우회 과정





라) 글쓰기(또는 회원 가입)를 완료한 후 글 작성자의 이름이 임의로 변조한 사용자 이름으로 등록 되었는지 확인

※ 사용자명이 변조되지 않고 실명인증을 받았던 정상적인 사용자 이름으로 게시되었을 경우 실명인증 우회 취약점이 존재하지 않음을 의미함

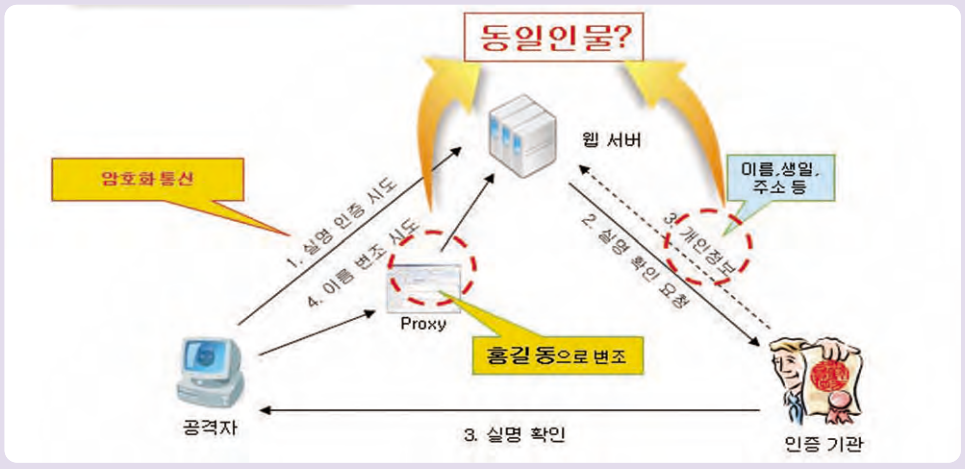
다. 대응방안

1) 홈페이지 개발 보안 조치

가) 중요한 정보가 있는 홈페이지(실명 등)은 재 인증 적용하고 안전하다고 확인된 라이브러리나 프레임워크(OpenSSL이나 ESAPI의 보안기능 등)를 사용

※ 제3장 홈페이지 개발 보안 방안의 11. URL / 파리미터 변조를 참조, 소스코드를 수정하여 보안취약점을 조치하시길 바람

실명 인증 우회 차단

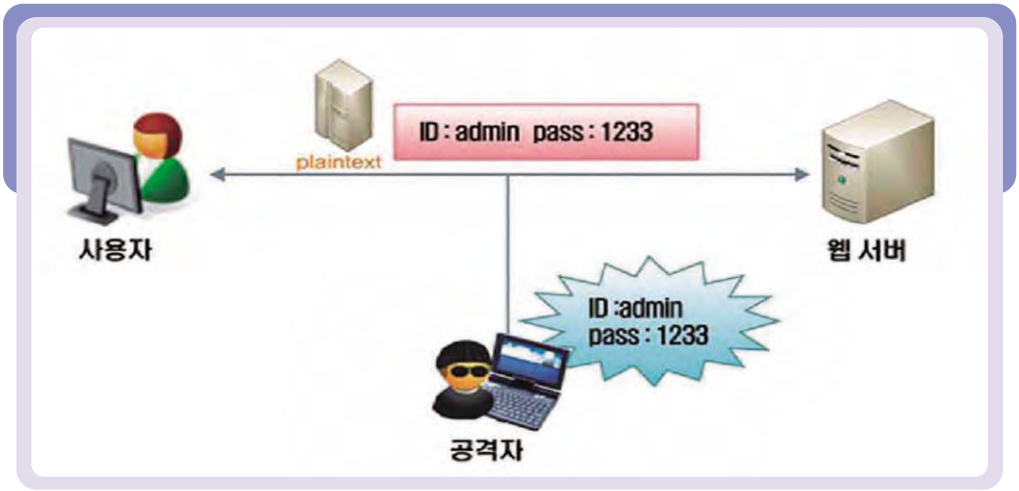


8. 전송 시 주요정보 노출 취약점

가. 취약점 설명 및 사례

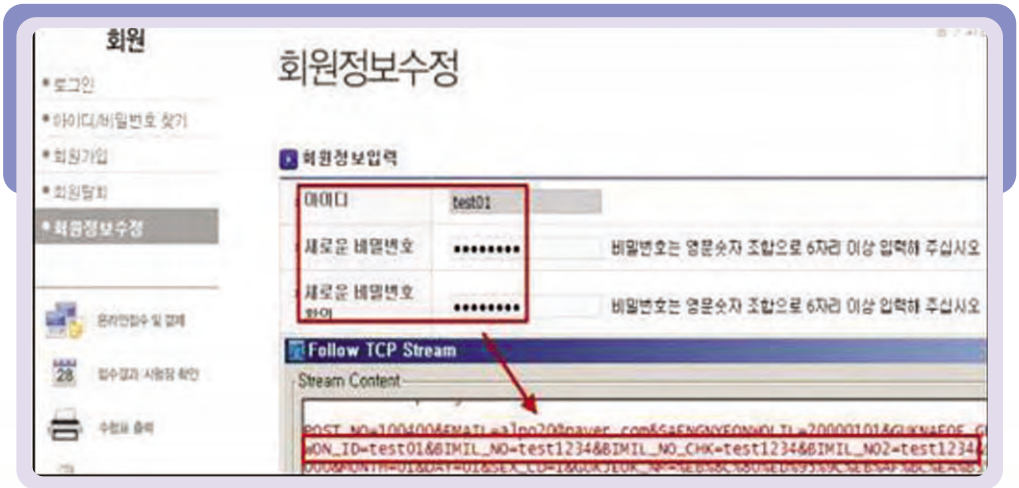
1) 취약점 설명

프로그램이 보안과 관련된 민감한 데이터를 평문으로 통신채널을 통해서 송수신 할 경우, 통신채널 스니핑을 통해 인가되지 않은 사용자에게 민감한 데이터가 노출될 수 있는 취약점



2) 사례

가) OO기관은 로그인 과정에서 사용자와 서버간의 통신 정보가 암호화되지 않아 감청을 통해 사용자 정보 획득이 가능



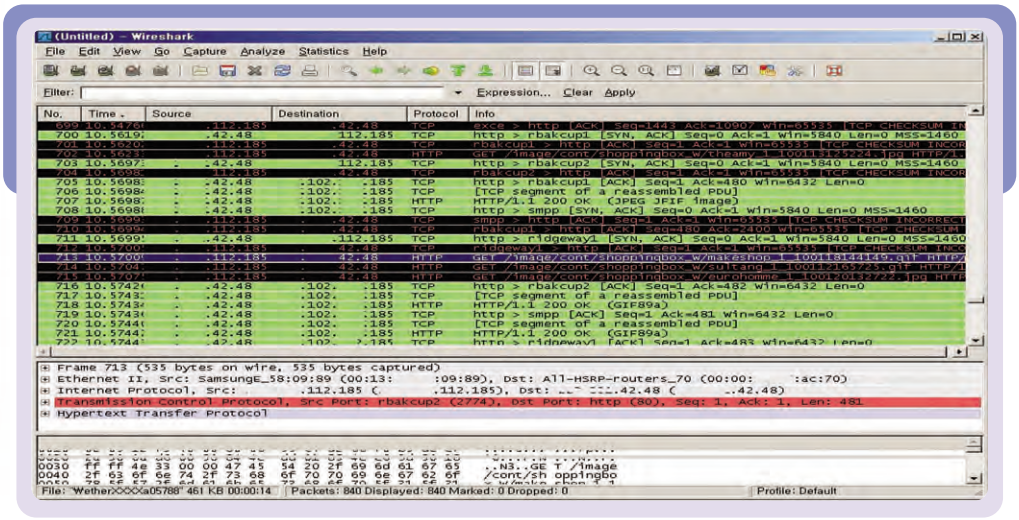
나. 점검방법

1) 점검 대상 웹서버의 로그인 페이지로 이동

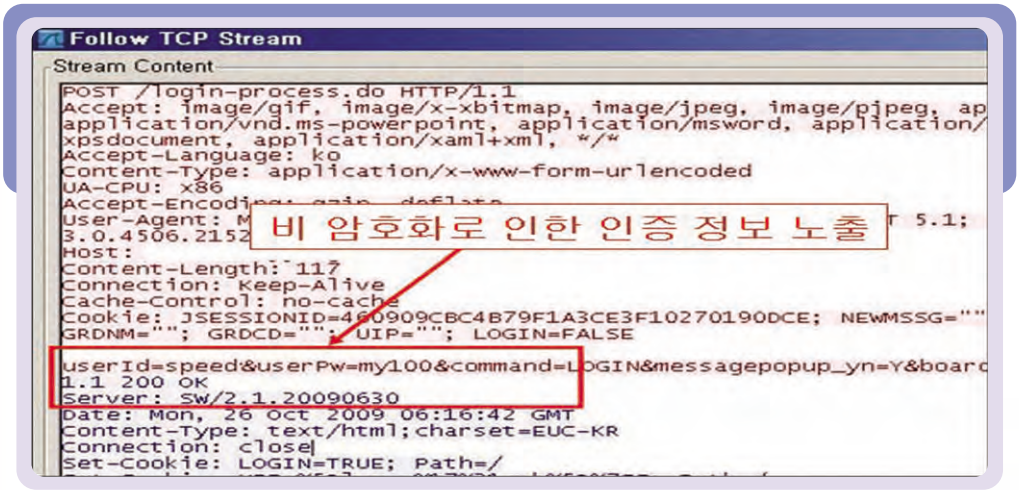
※ 로그인 과정상에서 I-PIN, 전자서명인증서를 사용할 경우 취약점이 존재하지 하지 않음
(단, ID, 비밀번호를 병행할 경우 취약할 수 있음)

2) 네트워크 패킷 모니터링 프로그램을 이용하여 로그인 과정상에서 발생하는 네트워크 트래픽을 저장

※ 공개용 패킷 모니터링 프로그램 : Wireshark(http://www.wireshark.org)



3) 로그인 후 네트워크 패킷 모니터링 프로그램을 통해 로그인 시 저장된 인증 정보를 찾아 암호화 여부를 확인



다. 대응방안

1) 웹 서버 내에서의 조치

가) 웹 서버는 전자서명인증서, SSL(Secure Socket Layer)을 이용하여 사용자 식별 및 DATA 전송 시 암호화 통신으로 데이터 전송의 안전성을 확보

나) 조치 완료 후 인증과정 등의 주요 정보 노출 여부를 재점검

2) 홈페이지 개발 보안 조치

가) 홈페이지는 중요정보와 관련된 민감한 데이터(개인정보, 비밀번호 등) 전송 시 통신채널(또는 전송데이터) 암호화적용

※ 제3장 홈페이지 개발 보안 방안의 14. 데이터 평문 전송을 참조, 소스코드를 수정하여 보안취약점을 조치하시길 바람

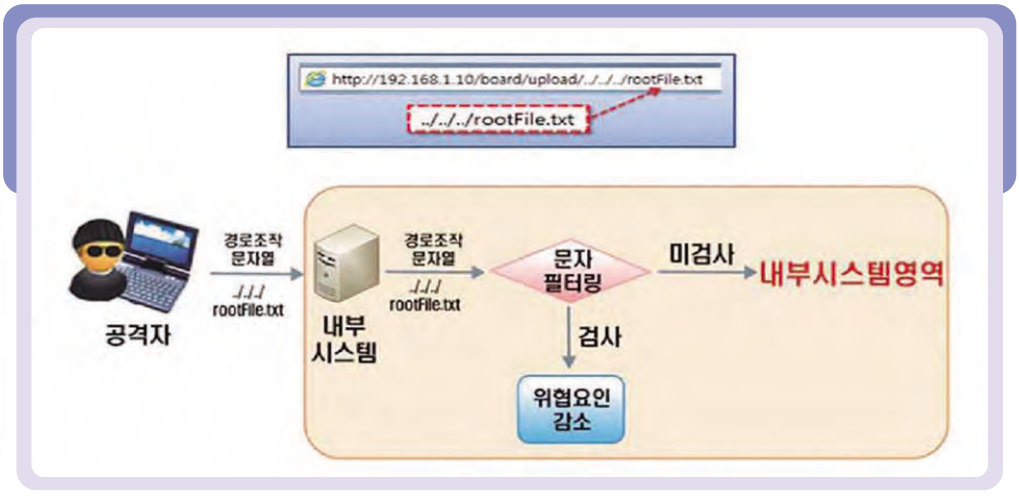
02 홈페이지 보안취약점 사례 및 대응

9. 파일 다운로드 취약점

가. 취약점 설명 및 사례

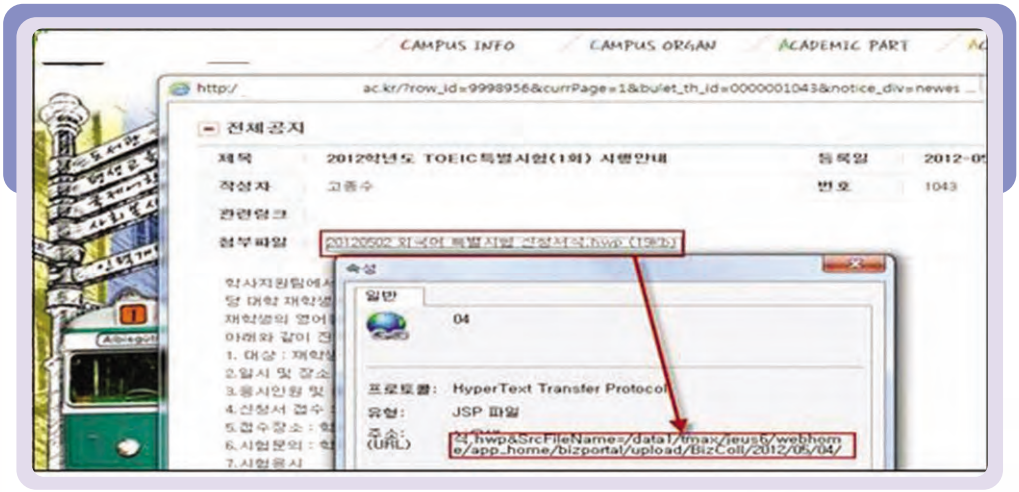
1) 취약점 설명

외부 입력 값에 대해 경로 조작에 사용될 수 있는 문자를 필터링하지 않으면, 예상 밖의 접근 제한 영역에 대한 경로 문자열 구성이 가능해져 시스템 정보누출, 서비스 장애 등을 유발 시킬 수 있는 취약점

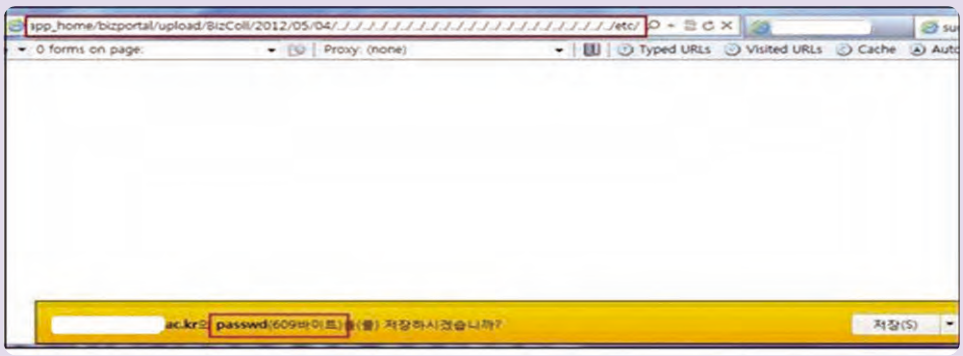


2) 사례

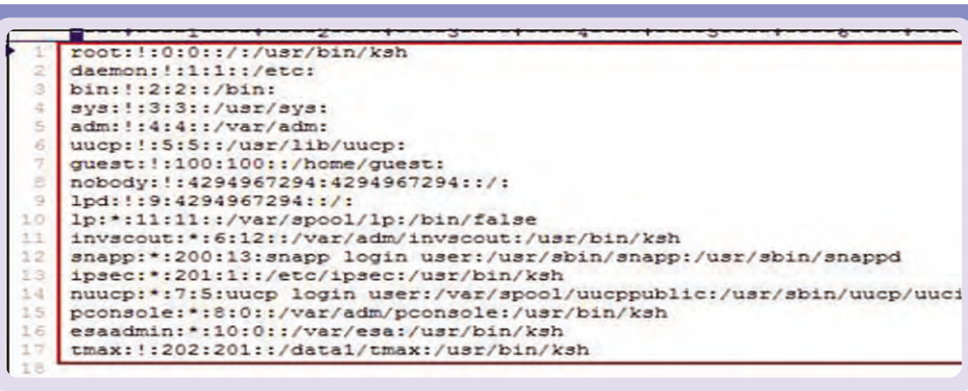
가) OO대학의 경우 홈페이지 다운로드 기능을 통해 시스템 파일의 다운로드가 가능
- 공지사항 게시판에서 첨부파일을 다운로드 받을 때의 URL을 파악



– 파일 다운로드 기능 실행

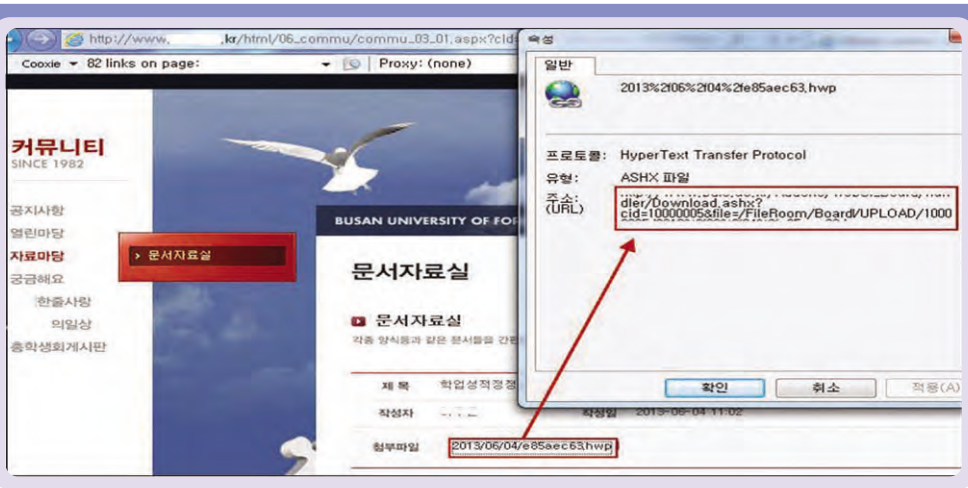


– 서버의 계정 파일이 다운로드됨



나) OO대학의 경우 홈페이지 다운로드 기능을 통해 시스템 파일의 다운로드가 가능

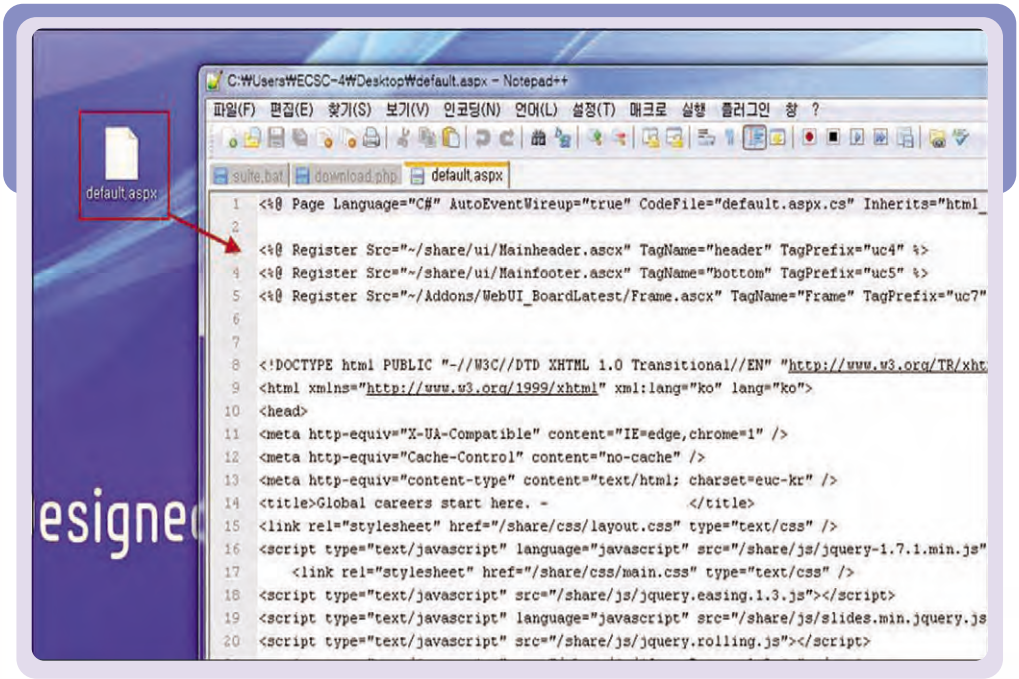
– 게시물 첨부파일의 속성으로 파일 다운로드 시의 경로를 파악



02 홈페이지 보안취약점 사례 및 대응

– 웹 어플리케이션 파일을 다운 받기 위해 경로 값 변경

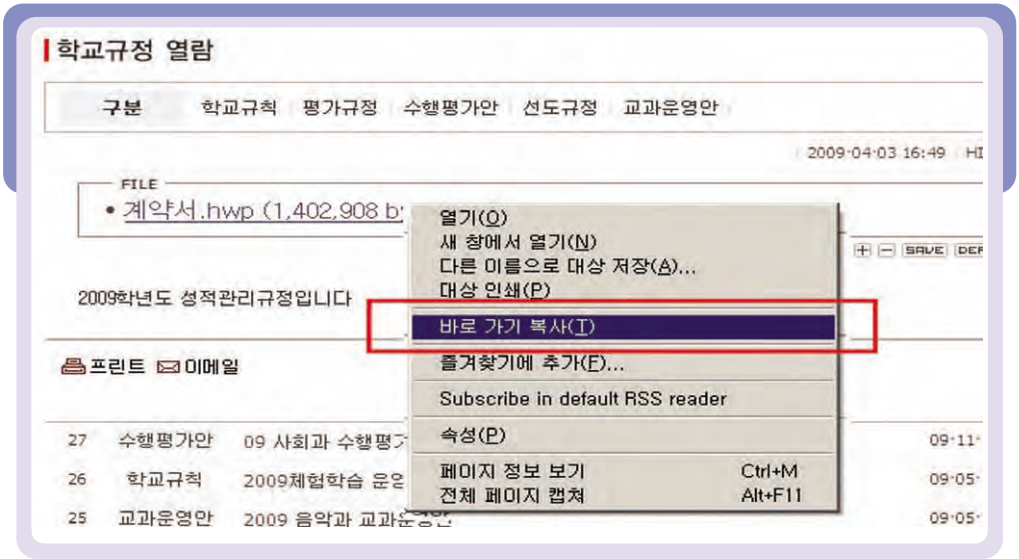
– 웹 어플리케이션 파일 다운로드 가능



나. 점검방법

1) 파일 다운로드 링크 확인

가) 마우스를 다운로드할 파일 링크에 가져간 후 마우스 오른쪽 클릭
[바로 가기 복사(T)] 웹 브라우저 주소입력란에 '붙여넣기'를 하여 링크가 가리키는 URL을 확인



2) 파일 다운로드가 가능할 경우 다운로드 파일의 URL을 확인하여 다운로드 방식을 확인

가) 파일 다운로드 방식 확인
- 파일 다운로드는 동적 방식과 정적 방식으로 나눌 수 있음. 동적방식은 URL 파라미터에 파일이름 혹은 파일번호를 할당하여 데이터를 처리하는 방식이며, 정적 방식은 특정 디렉터리에 존재하는 파일에 직접 링크를 설정하여 사용자에게 제공하는 방식으로 아래와 같은 형태임

[동적 방식의 URL 예]
[1] http://점검대상/bbs/Download.jsp?bbs=notice&no=11&filename=계약서.hwp&path=download
[2] http://점검대상/bbs/Download.jsp?bbs=notice&no=11&filename=/download/계약서.hwp
[3] http://점검대상/bbs/Download.asp?fn=c:\download\계약서.hwp

[정적 방식의 URL 예]
[3] http://점검대상/bbs/Download/계약서.hwp

– 정적 다운로드 방식은 첨부파일의 존재 위치와 파일의 이름을 공격자가 쉽게 획득할 수 있으므로 '파일 업로드' 공격 시 업로드 될 파일의 위치를 추정하는 등 웹서버의 내부 정보 수집에 활용될 수 있으므로 되도록 정적 방식을 지양(단, 정적 방식은 다운로드 취약점을 이용한 임의의 파일 다운로드가 불가능하여, 대부분의 경우 다운로드 취약점이 존재하지 않음)

나) 동적 파일다운로드 방식의 경우 파일명을 나타내는 변수와 파일의 위치를 나타내는 변수를 아래와 같이 수정하여 시스템 내부 파일의 다운로드를 시도

[1] http://점검대상/bbs/Download.jsp?bbs=notice&no=11&filename= **계약서.hwp**&path=**/download**

[2] http://점검대상/bbs/Download.jsp?bbs=notice&no=11&filename=passwd&path=../../../etc/

[3] http://점검대상/bbs/Download.jsp?bbs=notice&no=11&filename=boot.ini&path= ../../../

– [1]은 일반적인 다운로드 URL을 의미하며 『/download/계약서.hwp』를 다운로드 하겠다는 의미함
– [2]는 unix, linux 계열의 /etc/passwd 파일 다운로드 시도를 의미함
– [3]은 windows 계열의 boot.ini 파일 다운로드 시도를 의미함

※ **../ 는 상위 경로를 나타내며 최상위 경로(/)로 이동하기 위해 사용되며 최상위 폴더까지의 접근하기위해 ../ 문자열을 충분히 입력함**

다. 대응 방안

1) 홈페이지 개발 보안 조치

가) 홈페이지 상에 다운로드 파일의 이름을 데이터베이스에 저장하고 다운로드 수행 시 요청파일의 이름이 동일한지 여부를 검증토록 조치

나) 홈페이지 상에 다운로드 파일명 또는 경로에 '..', '/', '\' 값이 입력되지 않도록 조치

다) PHP언어로 개발된 서버의 경우 php.ini 내용 중 magic_quotes_gpc 항목의 값을 On으로 설정하여 '.\ 와 ./' 값 입력 시 치환되도록 설정

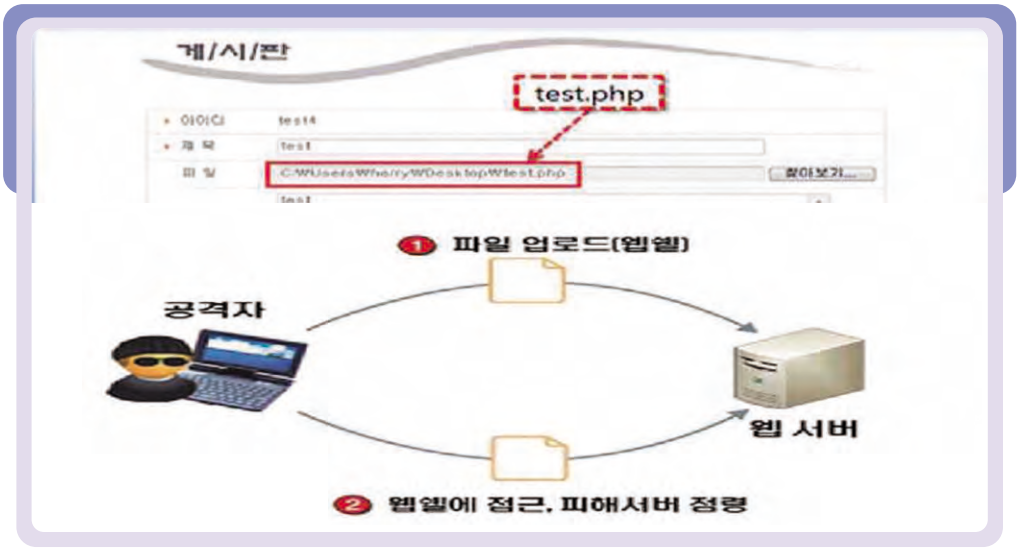
※ **제3장 홈페이지 개발 보안 방안의 7. 파일 다운로드를 참조, 소스코드를 수정하여 보안취약점을 조치하시길 바람**

10. 파일 업로드 취약점

가. 취약점 설명 및 사례

1) 취약점 설명

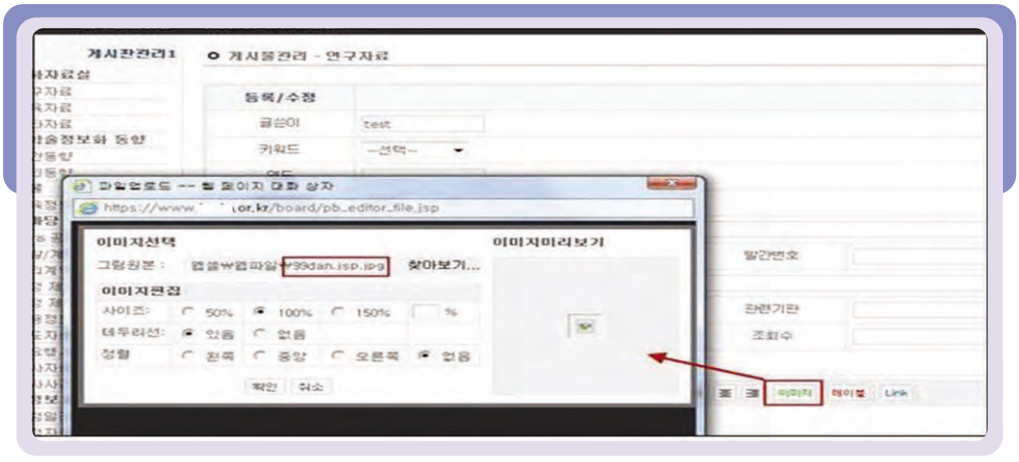
서버 측에서 실행될 수 있는 스크립트 파일(asp, jsp, php 파일 등)이 업로드 가능하고, 이 파일을 공격자가 웹을 통해 직접 실행시킬 수 있는 경우 시스템 내부명령어를 실행하거나 외부와 연결하여 시스템을 제어할 수 있는 취약점



2) 사례

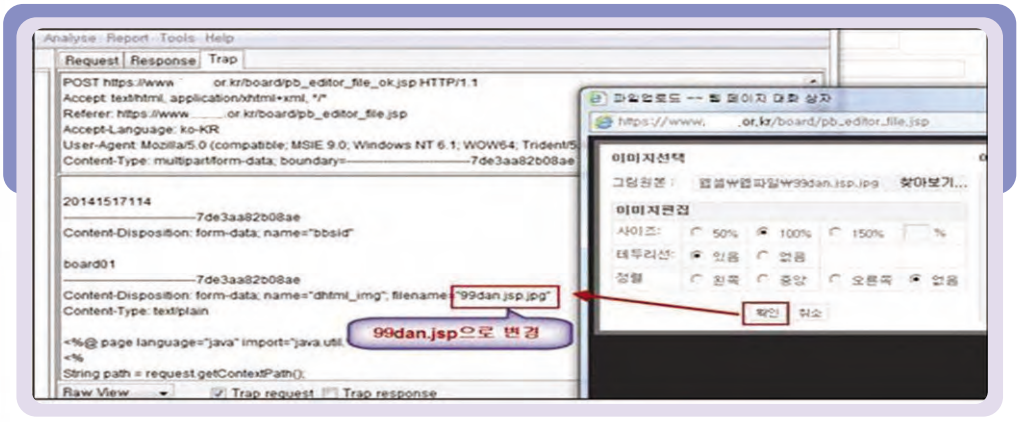
가) OO기관의 홈페이지 게시판에 jsp파일의 확장자를 변경 후 삽입시도

※ jpg, gif, bmp 확장자만 허용됨

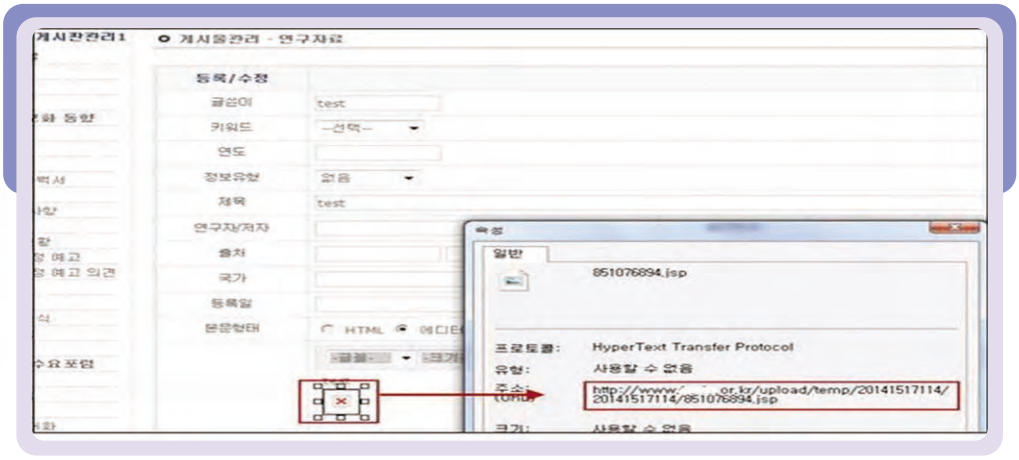


02 홈페이지 보안취약점 사례 및 대응

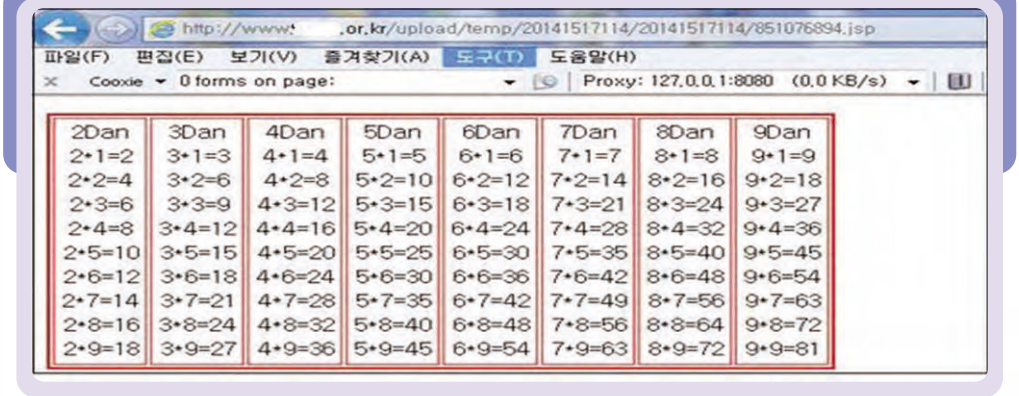
– 파일 삽입 요청 시 확장자 변경(.jpg) 후 서버 전송



– 삽입한 이미지 경로 확인

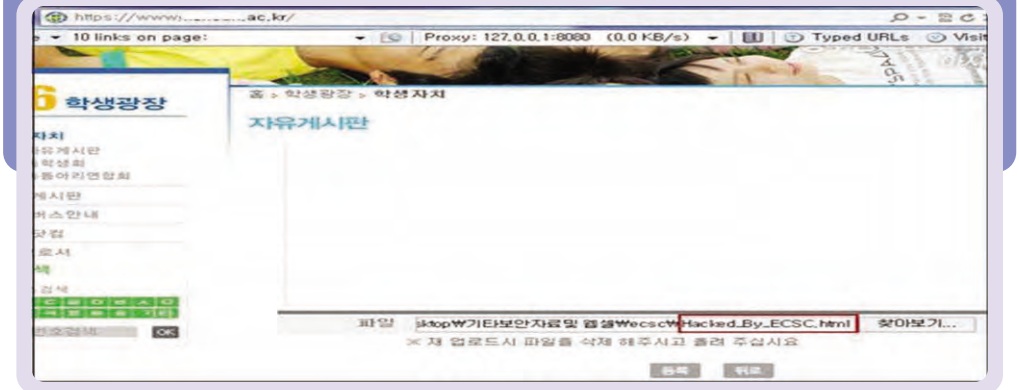


– 업로드된 URL 요청 시 정상동작 확인

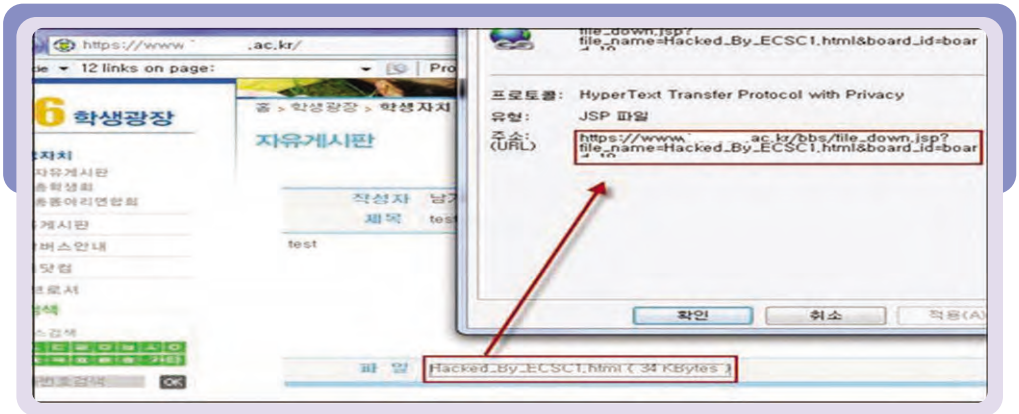


2Dan	3Dan	4Dan	5Dan	6Dan	7Dan	8Dan	9Dan
2*1=2	3*1=3	4*1=4	5*1=5	6*1=6	7*1=7	8*1=8	9*1=9
2*2=4	3*2=6	4*2=8	5*2=10	6*2=12	7*2=14	8*2=16	9*2=18
2*3=6	3*3=9	4*3=12	5*3=15	6*3=18	7*3=21	8*3=24	9*3=27
2*4=8	3*4=12	4*4=16	5*4=20	6*4=24	7*4=28	8*4=32	9*4=36
2*5=10	3*5=15	4*5=20	5*5=25	6*5=30	7*5=35	8*5=40	9*5=45
2*6=12	3*6=18	4*6=24	5*6=30	6*6=36	7*6=42	8*6=48	9*6=54
2*7=14	3*7=21	4*7=28	5*7=35	6*7=42	7*7=49	8*7=56	9*7=63
2*8=16	3*8=24	4*8=32	5*8=40	6*8=48	7*8=56	8*8=64	9*8=72
2*9=18	3*9=27	4*9=36	5*9=45	6*9=54	7*9=63	8*9=72	9*9=81

나) OO대학의 홈페이지 게시판에 html파일 업로드 시도

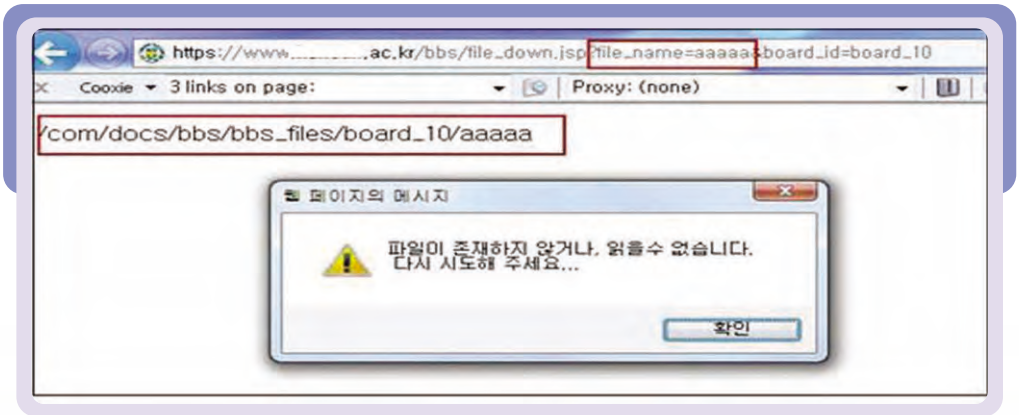


– 파일 업로드 후 속성에서 파일 다운로드 시의 URL 파악



– 특정 파라메터 값에 부적절한 값을 입력하면 업로드 파일의 경로를 알 수 있음

※ file_name 혹은 board_id 파라메터 값에 부적절한 값 입력



02 홈페이지 보안취약점 사례 및 대응

– URL을 수정하여 요청(Request)하면 업로드한 웹 파일이 실행됨

※ html파일에 스크립트 코드등을 삽입하여 업로드 하면 해당 사이트는 악성코드 유포지 혹은 경유지로 사용되는 등 다양하게 활용 가능

나. 점검 방법

1) 파일 다운로드 링크 확인

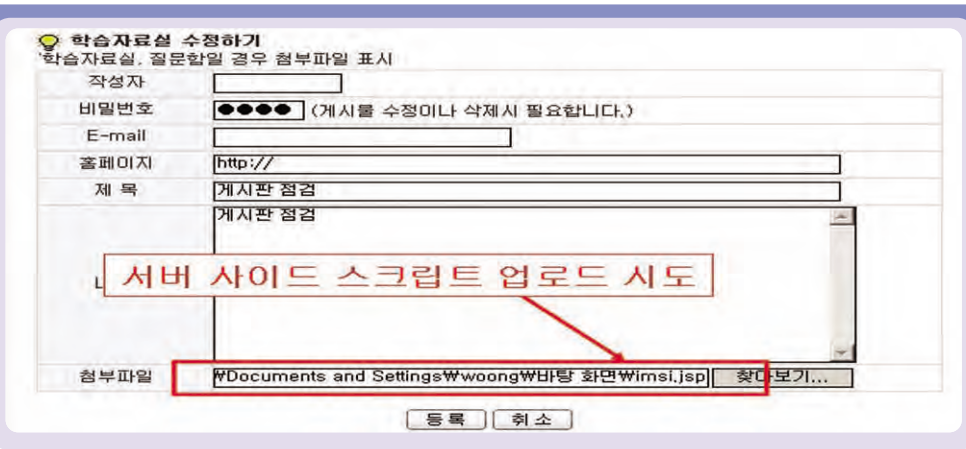
가) 파일 업로드(첨부)가 가능한 게시판 등에 서버 사이드 스크립트(ASP, PHP, PHP3, JSP, CGI 등) 및 html 등의 파일을 업로드

※운영 중인 서버의 개발언어에 맞는 서버 사이드 스크립트를 업로드 함

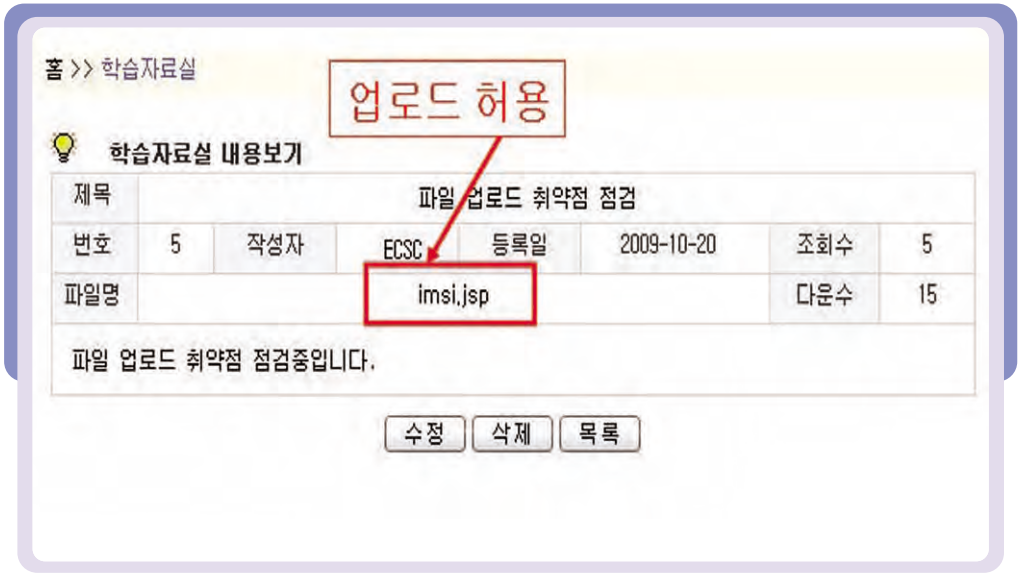
나) 서버 사이드 스크립트가 업로드 되었을 경우 '파일 업로드 취약점'의 존재를 추정할 수 있어 보다 상세한 분석이 필요함. 업로드가 되었을 경우 해당 파일이 웹서버에서 실행되는지 점검

다) 업로드 시도 시 다음 그림과 같이 업로드 실패 화면 또는 오류 메시지가 나타날 경우

※단, 업로드 차단기능이 javascript로 구현되어있는지 점검이 필요함

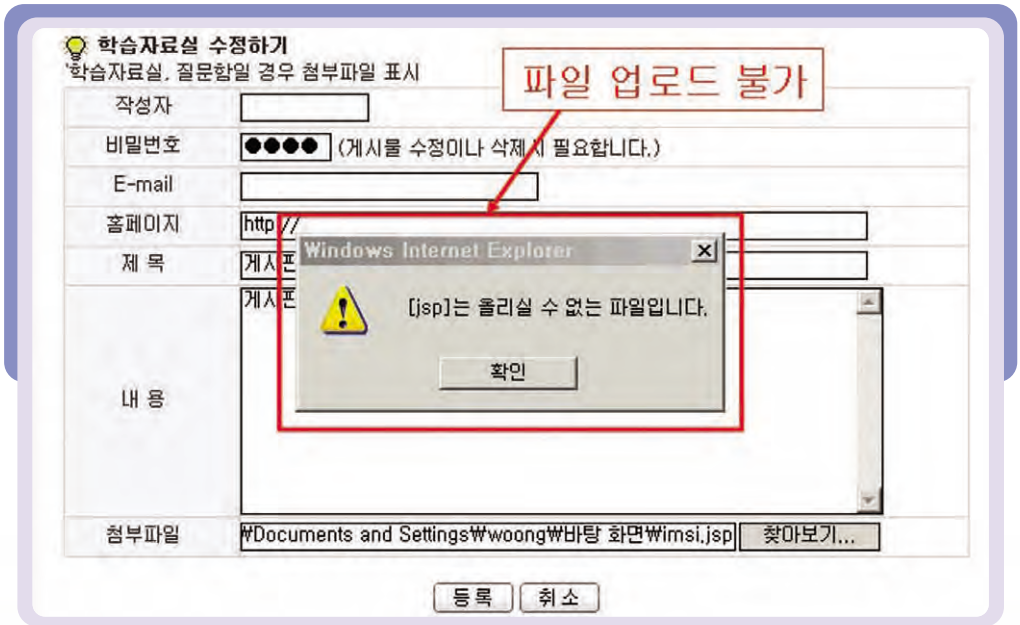


나) 서버 사이드 스크립트가 업로드 되었을 경우 '파일 업로드 취약점'의 존재를 추정할 수 있어 보다 상세한 분석이 필요함. 업로드가 되었을 경우 해당 파일이 웹서버에서 실행되는지 점검



다) 업로드 시도 시 다음 그림과 같이 업로드 실패 화면 또는 오류 메시지가 나타날 경우

※ 단, 업로드 차단기능이 javascript로 구현되어있는지 점검이 필요함



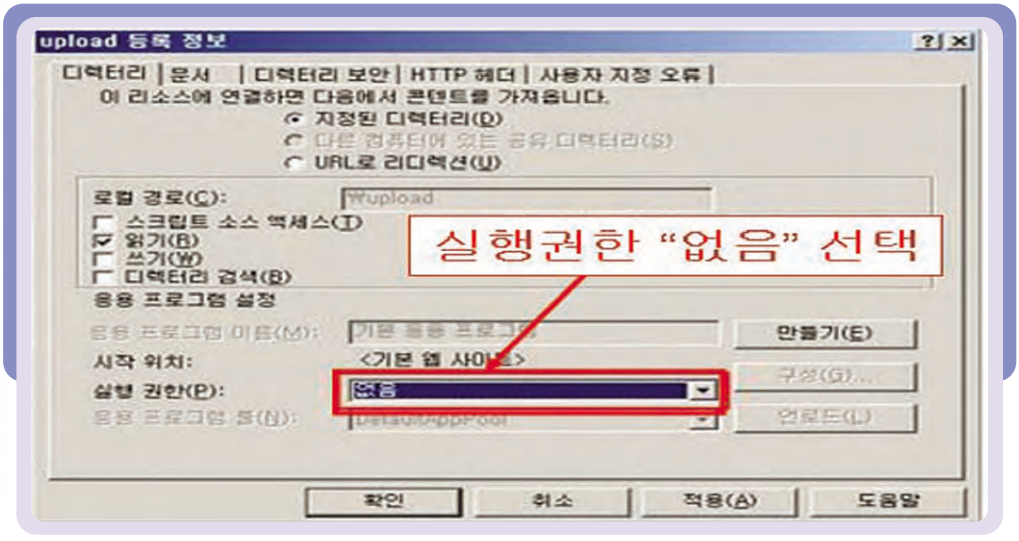
다. 대응 방안

1) 웹 서버 내에서의 조치

가) 웹 서버 설정을 변경하여 업로드 된 해당 파일의 실행권한을 차단

① IIS 웹서버 설정 방법

[제어판] → [관리도구] → [인터넷 서비스 관리자] → 업로드 폴더 선택 → [속성] 클릭 → 마우스 오른쪽 버튼 클릭 → [등록 정보] → [실행권한] → [없음] 선택



② Apache 웹서버 설정 방법

httpd.conf 파일에 다음과 같은 지시자를 추가한 후 apache 데몬을 재시작하여 해당 파일의 실행을 차단

PHP 언어로 개발한 사이트의 차단 예)

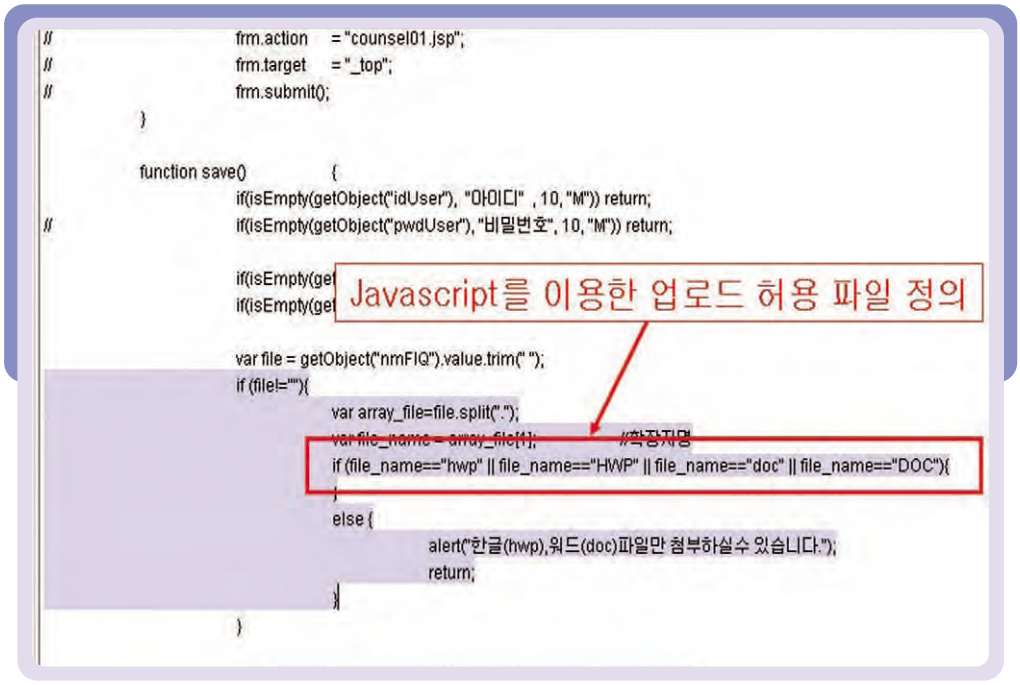
```
<Directory "업로드를 금지할 디렉터리">
AddType application/x-httpd-php3-source .php3 .php .phps
.ph .cgi .jsp .inc .htm .html shtml
Options IncludesNoExec
</Directory>
```

나) 웹 서버에는 업로드 대상 파일의 확장자를 검증하는 처리 프로그램을 통해 서버 사이드 스크립트(ASP, PHP, JSP, CGI 등) 파일의 업로드를 차단

다) 웹 서버에서는 우회 기법을 통한 악의적 파일 업로드를 차단

① 파일 업로드 가능 여부를 검증하는 기능을 서버 사이드 스크립트(ServerSide Script)로 구현하여 우회 기법을 통한 업로드 공격을 사전에 차단

② 다음 그림과 같이 javascript로 필터링 기능을 구현할 경우 사용자가 임의로 수정 및 삭제 할 수 있으므로 차단기능을 우회 가능




```
//                  frm.action    = "counsel01.jsp";
//                  frm.target    = "_top";
//                  frm.submit();
        }

        function save()          {
                  if(isEmpty(getObject("idUser"), "아이디" , 10, "M")) return;
//                if(isEmpty(getObject("pwdUser"), "비밀번호", 10, "M")) return;

                  if(isEmpty(get
                  if(isEmpty(get

                  var file = getObject("nmFIQ").value.trim(" ");
                  if (file!=""){
                            var array_file=file.split(".");
                            var file_name = array_file[1];          //확장자명
                            if (file_name=="hwp" || file_name=="HWP" || file_name=="doc" || file_name=="DOC"){
                            }
                            else {
                                      alert("한글(hwp),워드(doc)파일만 첨부하실수 있습니다.");
                                      return;
                            }
                  }
```

③ 파일 업로드 필터링 방식은 White-List 방식을 이용(업로드 가능한 확장자만 업로드 허용)하여 확장자 변경 등의 우회 기법을 차단

※ White-List 방식 : 업로드 할 파일(정상패턴)만 허용, 이외 모두 차단

라) 파일이 업로드 되는 디렉터리(위치 및 파일명)가 사용자에게 노출되지 않도록 조치

① 파일을 저장할 경우 파일명을 변경하여 저장하고 실제 파일명은 데이터베이스에 보관하는 등 정보를 이원화하여 운영

2) 홈페이지 개발 보안 조치

가) 홈페이지 소스코드는 미리 정의된 업로드 파일의 확장자만 허용하고 그 외 확장자는 업로드를 제한하고, 저장 시 외부에서 입력된 파일명을 그대로 저장되지 않도록 코드 수정

※ 제3장 홈페이지 개발 보안 방안의 6. 파일 업로드를 참조, 소스코드를 수정하여 보안취약점을 조치하시길 바람

02 홈페이지 보안취약점 사례 및 대응

11. 소스코드 내 중요정보 노출 취약점

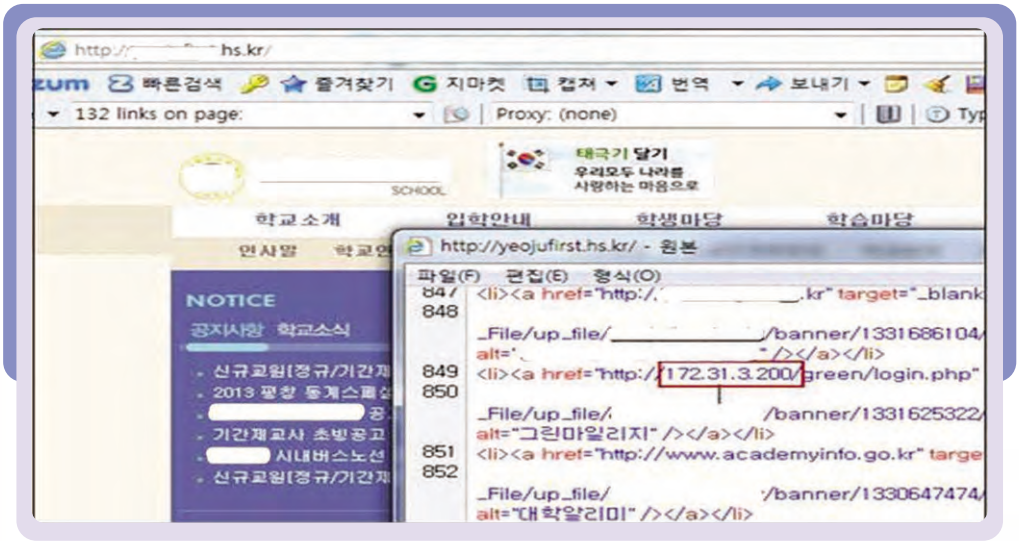
가. 취약점 설명 및 사례

1) 취약점 설명

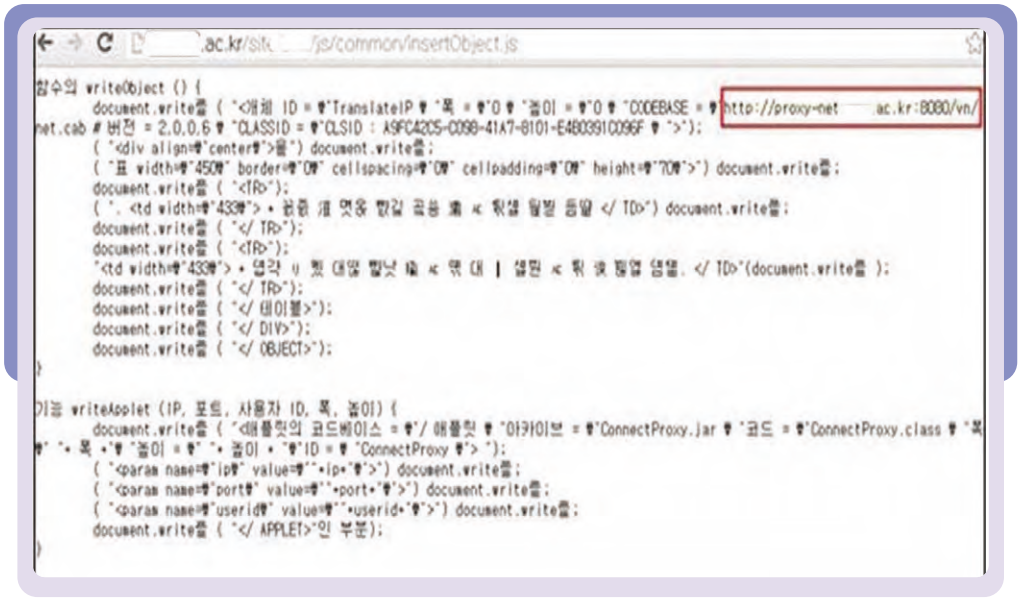
소스코드 주석문에 민감한 정보(개인 정보, 시스템 정보 등)이 포함되어 있는 경우, 외부공격자에 의해 패스워드 등 보안 관련정보가 노출될 수 있는 취약점

2) 사례

가) OO학교의 웹 페이지의 소스를 통하여 내부 IP 획득 가능



나) OO학교는 소스코드를 통해 관리자 페이지의 존재를 확인 가능



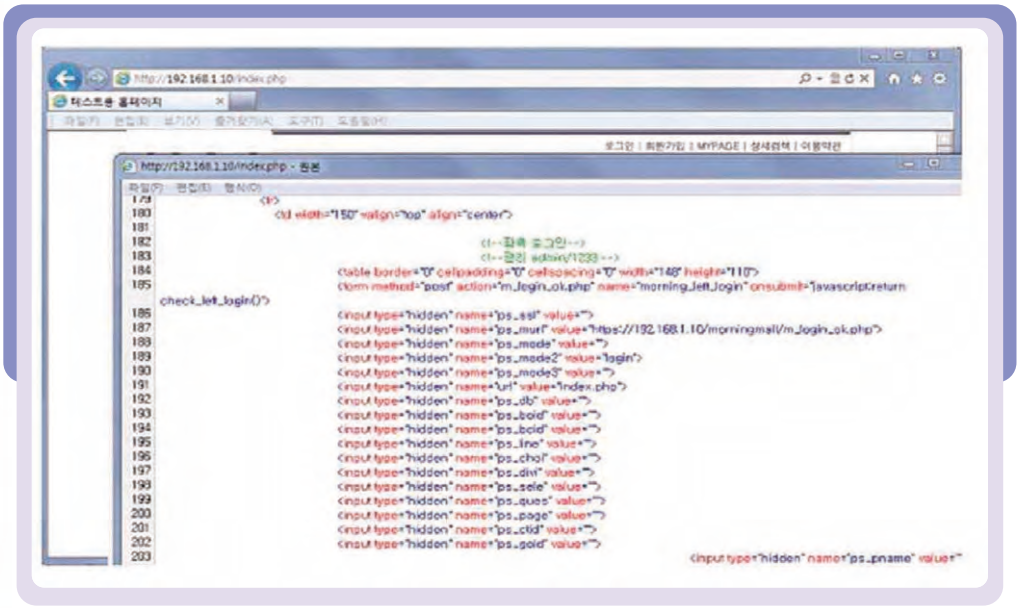
– 관리자 로그인 페이지 확인

02 홈페이지 보안취약점 사례 및 대응

나. 점검 방법

1) 웹 페이지에서 소스보기를 통해 민감한 정보 존재여부 확인

가) 로그인 등 중요정보가 포함될 것으로 의심이 되는 웹 페이지 소스보기 수행



다. 대응 방안

1) 홈페이지 개발 보안 조치

가) 홈페이지 소스코드에는 디버깅 목적으로 주석 ID, 패스워드, 시스템 관련정보 등 보안관련 정보가 남지 않도록 개발완료 후 제거 필요

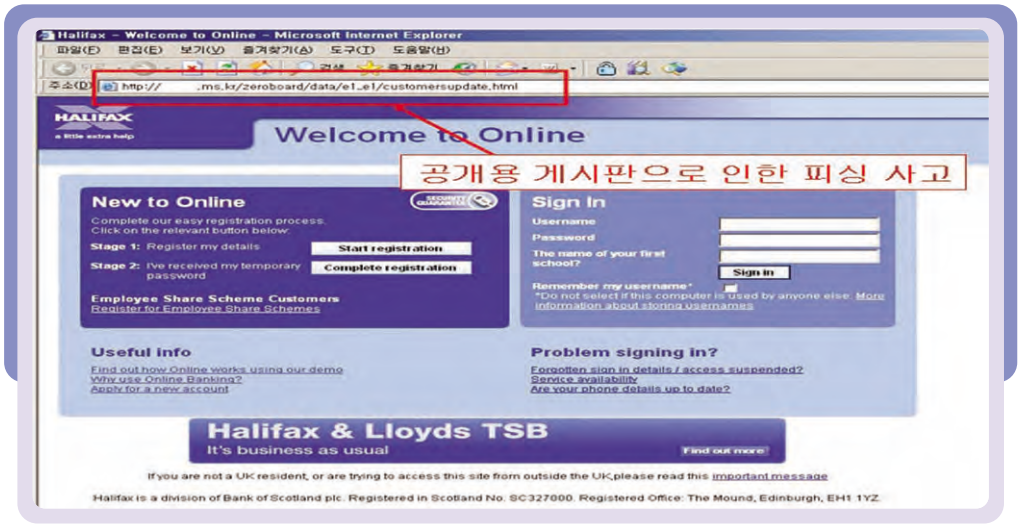
※ 제3장 홈페이지 개발 보안 방안의 18. 주석을 통한 정보 노출을 참조, 소스코드를 수정하여 보안취약점을 조치하시길 바람

12. 공개용 웹 게시판 취약점

가. 취약점 설명 및 사례

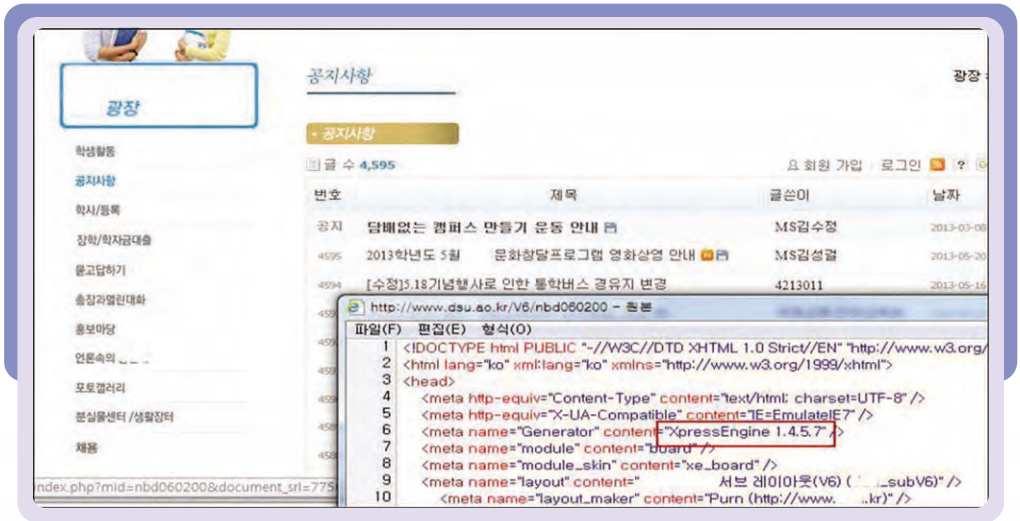
1) 취약점 설명

교육기관에서는 웹서버 구축 시 금전적, 시간적인 부담으로 인해 공개용게시판(제로보드, 테크노트 등)을 많이 이용하고 있으며, 공개용 게시판을 사용할 경우 인터넷에 공개된 각종 취약점 정보로 인해 홈페이지 변조 및 해킹 경유지로 사용될 수 있는 취약점



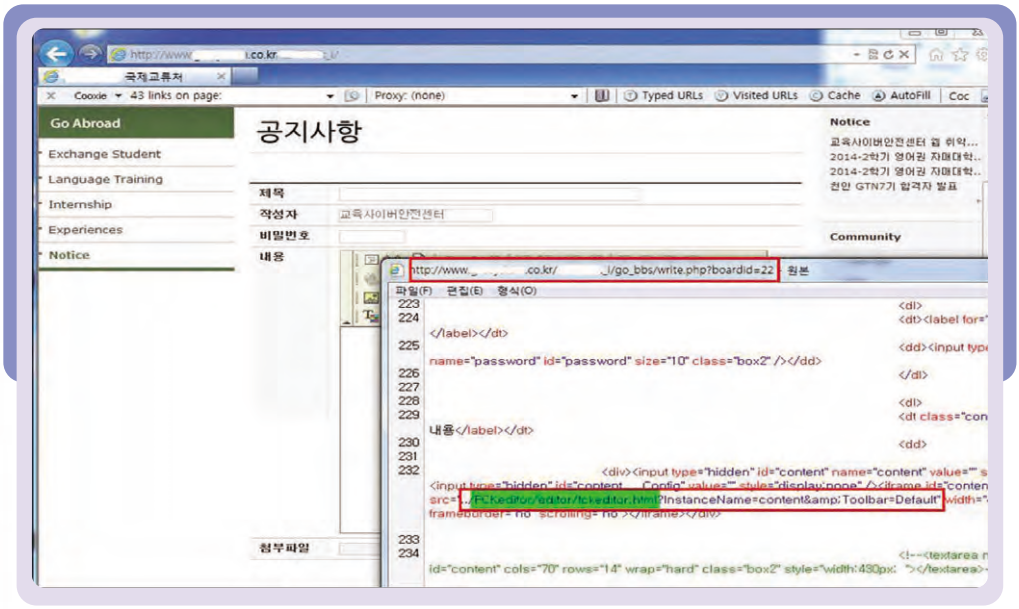
2) 사례

가) OO학교의 웹 페이지의 소스를 통하여 공개용 웹 게시판을 사용하고 있음을 확인

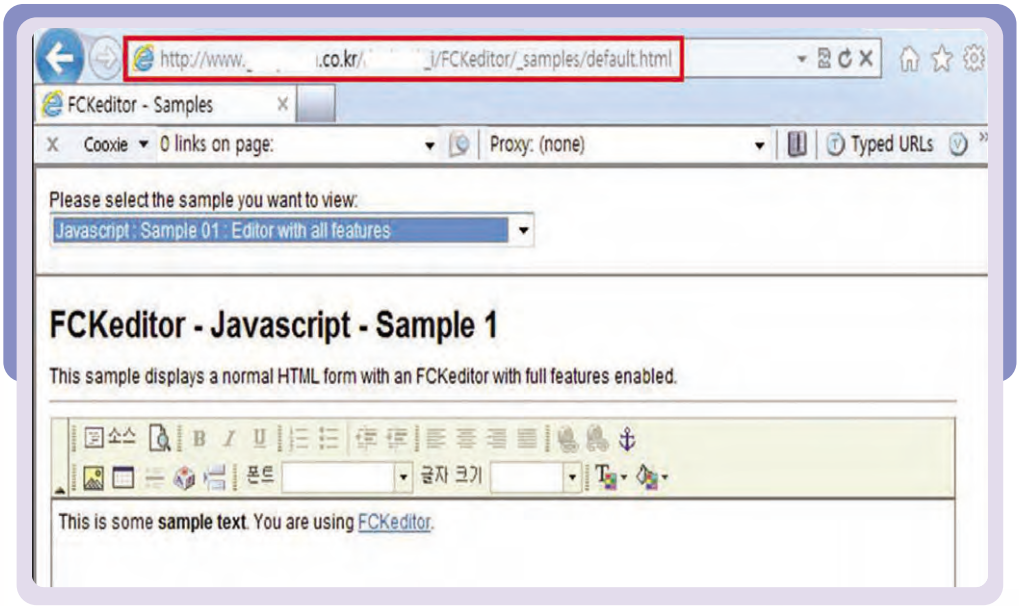


02 홈페이지 보안취약점 사례 및 대응

나) OO학교의 웹 페이지의 소스를 통하여 웹 게시판에 사용되는 공개용 프로그램 (FCKeditor) 경로 발견



– FCKeditor 샘플 페이지 접근 확인



나. 점검 방법

1) 웹서버에서 운영 중인 공개용 게시판을 검색

Unix, Linux 검색 예)

제로보드 검색)

find /[웹서버디렉터리] -name "license.txt" -exec ls -alt {} \; -exec grep "배포버전 :" {} \;

테크노트 검색)

find /[웹서버디렉터리] -name "config.cgi" -exec ls -alt {} \; -exec grep "# 최종수정 배포일" {} \;)

find /[웹서버디렉터리] -name "main.cgi" -exec ls -alt {} \; -exec grep "Update:" {} \;

브라우저를 통한 검색 예)

제로보드 버전 정보

http://홈페이지주소/bbs/license.txt
http://홈페이지주소/zb/license.txt
http://홈페이지주소/zeroboard/license.txt

테크노트 버전 정보

http://홈페이지주소/게시판 디렉터리/config.cgi

http://홈페이지주소/게시판 디렉터리/main.cgi

2) 공개용 게시판을 이용하지 않을 경우 운영 중인 게시판이 최신 버전임을 점검

다. 대응 방안

1) 웹 서버 내에서의 조치

가) 웹 서버에는 공개용 웹 게시판을 사용 지양

① 제로보드, 테크노트, 그누보드, 세팔보드 등

② 부득이 사용해야 할 경우 보안 취약점이 존재하지 않도록 보안 패치 또는 최신 버전의 제품으로 설치하며, 정기적으로 게시판 배포 사이트에 방문하여보안 취약점 정보를 확인

㉠ 게시판 재설치 시 기존 데이터가 삭제될 수 있으므로 백업 작업을 수행한 후 재설치

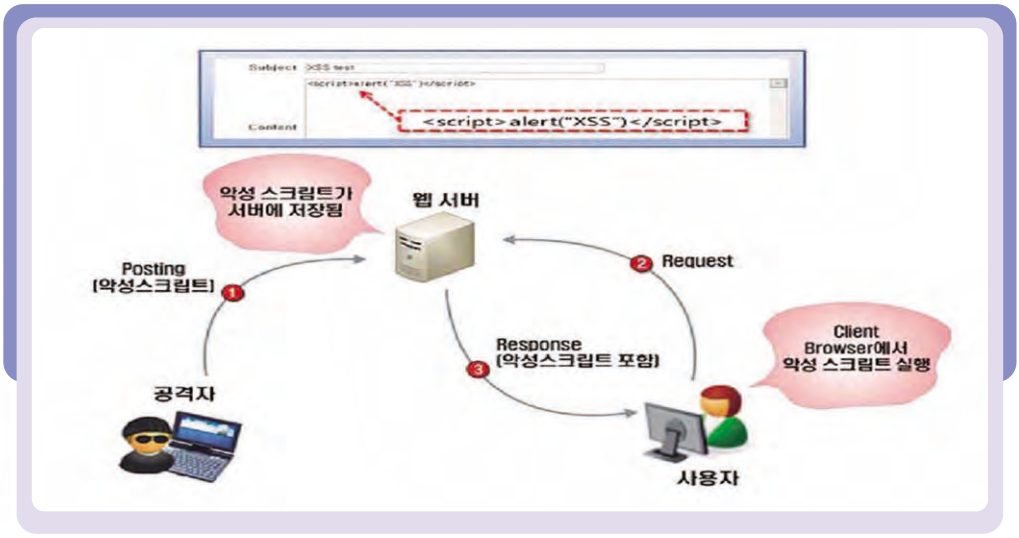
02 홈페이지 보안취약점 사례 및 대응

13. 크로스 사이트 스크립트(XSS) 취약점

가. 취약점 설명 및 사례

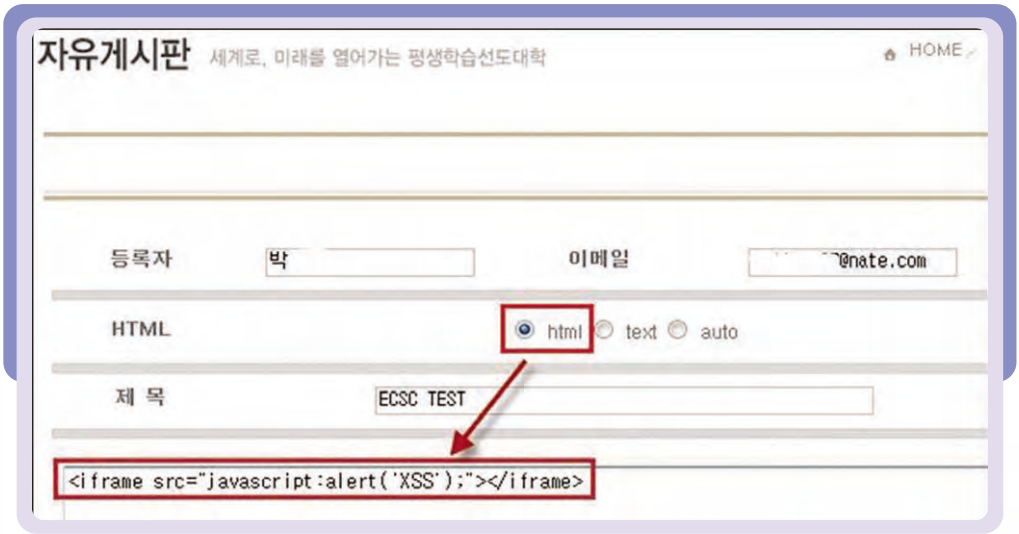
1) 취약점 설명

페이지에 악의적인 스크립트를 포함시켜 웹페이지를 열람하는 접속자의 권한으로 부적절한 스크립트가 수행되어 정보유출 등의 공격을 유발할 수 있는 취약점

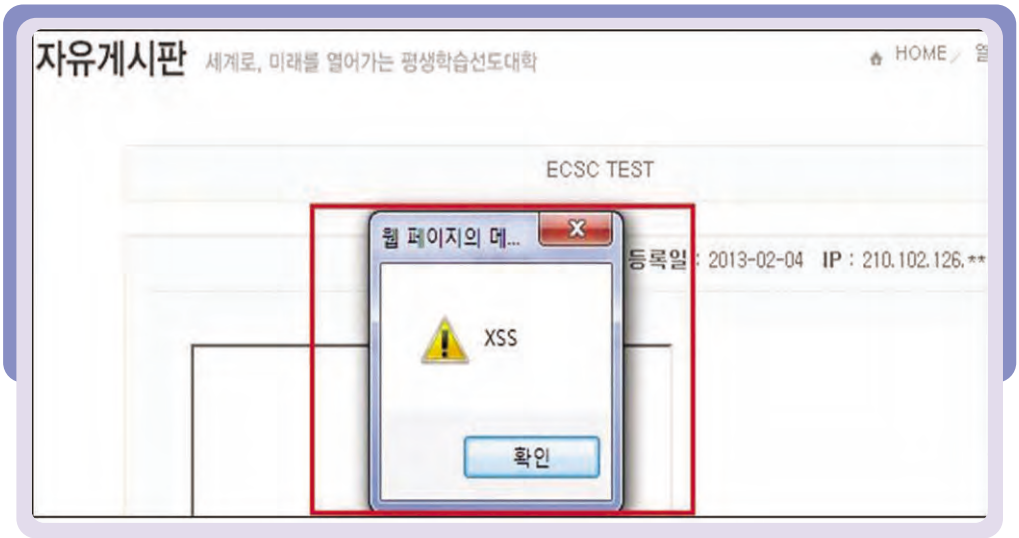


2) 사례

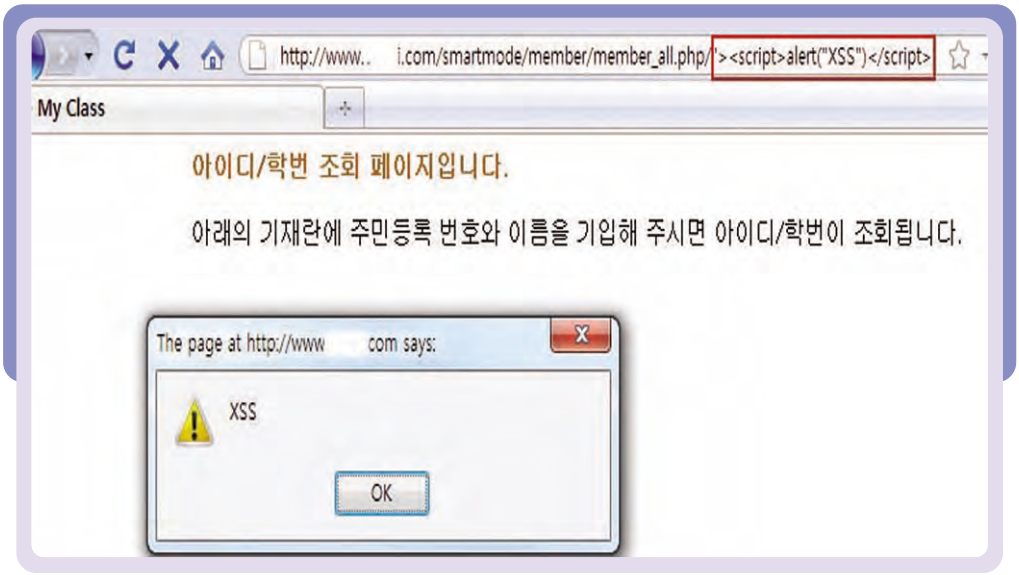
가) OO대학은 게시판의 댓글기능이 공개되어 있고 [HTML 편집기] 기능을이용하여 스크립트 코드의 삽입이 가능



– 삽입한 스크립트가 동작함



나) OO대학의 아이디/학번 조회 URL에 스크립트를 삽입하면 스크립트가 동작함



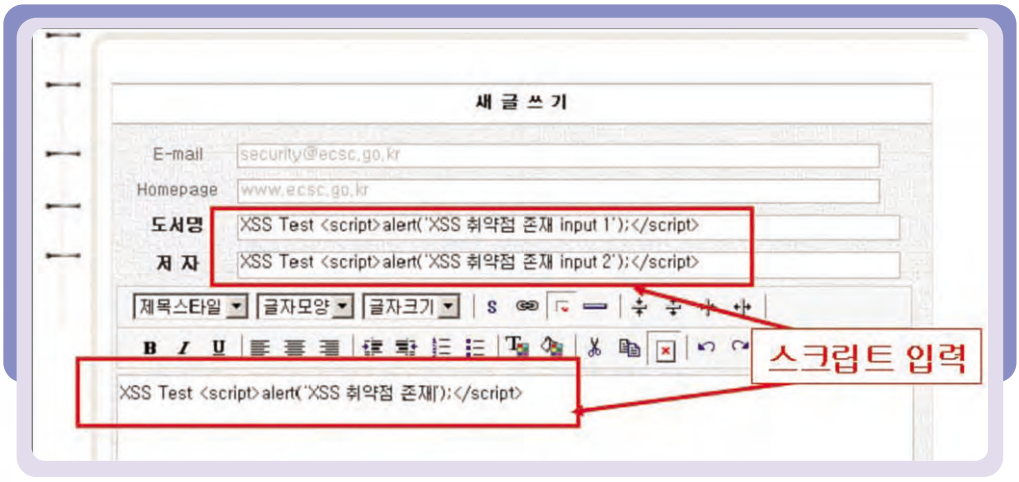
나. 점검 방법

1) 취약점 설명

글쓰기 기능이 있는 게시판 및 검색란에 다음과 같은 스크립트 문장을 각각 입력하고 글쓰기(또는 검색)를 시도

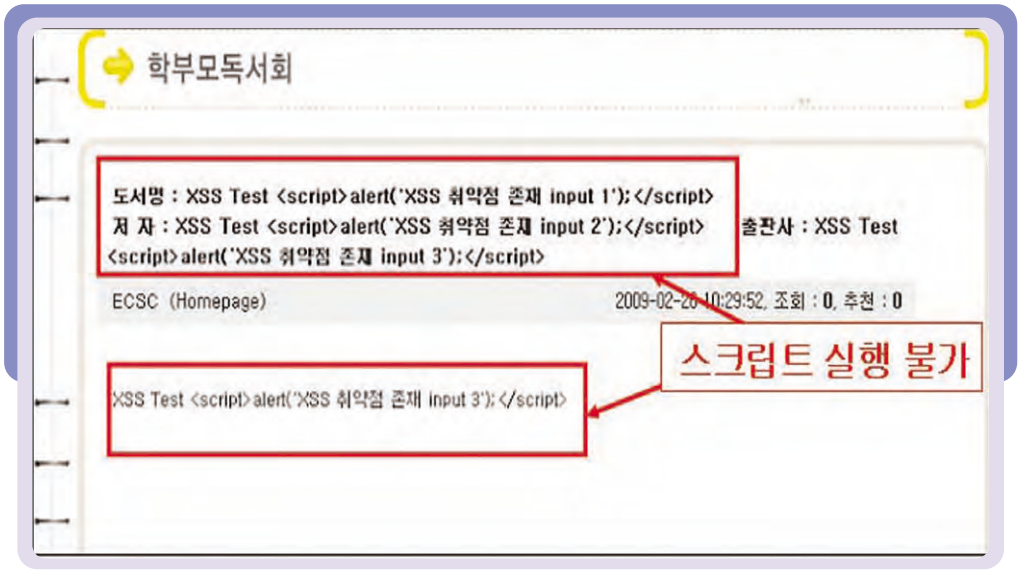
XSS 점검 예)

〈script〉alert('XSS 취약점 존재');〈/script〉
'〈embed src=http://cyber.ecsc.go.kr/xss_test.swf〉〈/embed〉

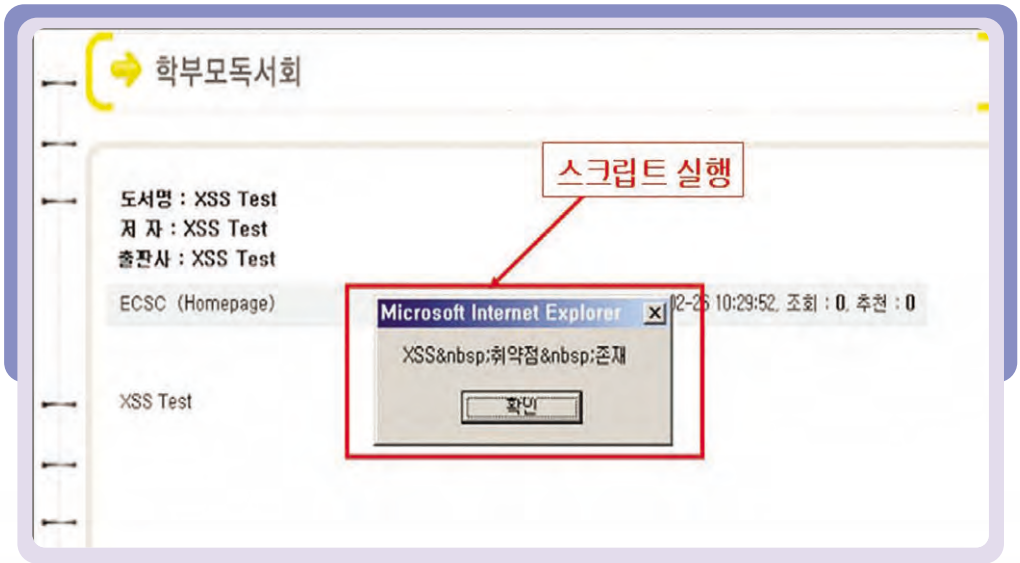


2) 글을 게시하는 중에 스크립트 태그 사용에 대한'오류'나 '경고' 메시지가 발생하며 입력한 정보가 등록되지 않는다면 취약점 없음

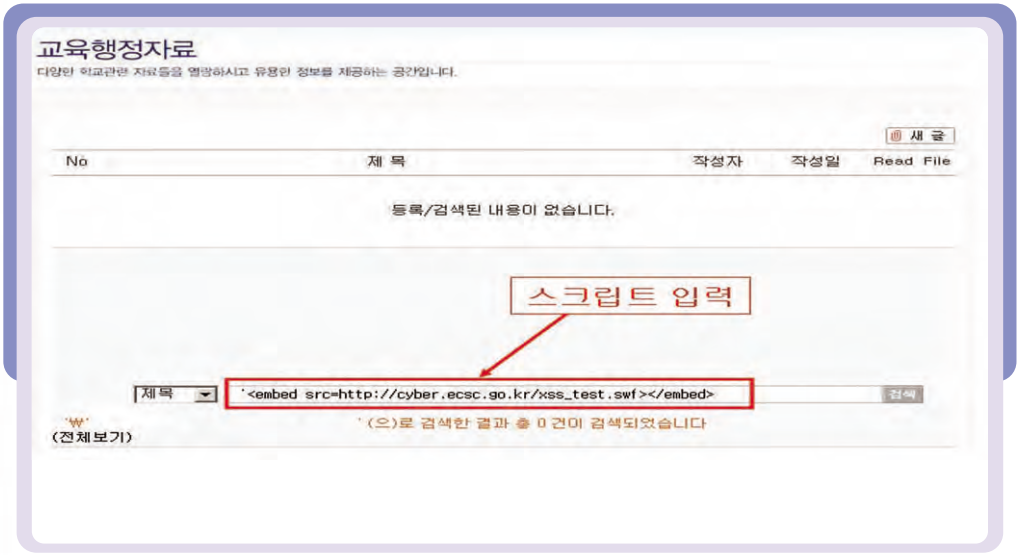
3) 윈도우 경고창이 나타나지 않고 아래 그림과 같이 스크립트 문장이 입력한 내용과 같이 나타나면 취약점 없음



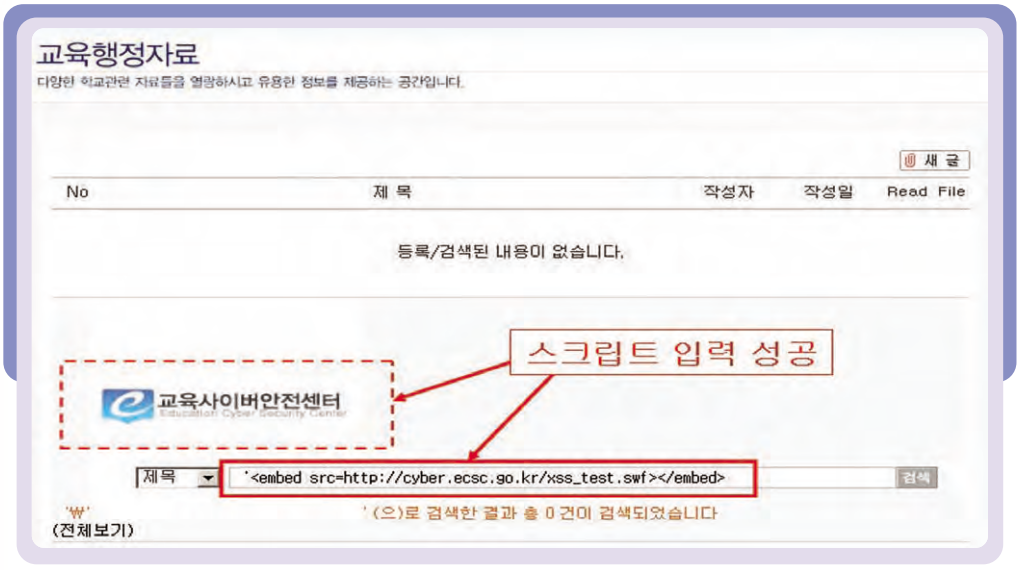
4) 윈도우 경고 창을 통해 1번 항목에서 입력한 문장인 'XSS 취약점 존재 또는 'XSS 취약점 존재'와 같은 형태의 팝업 경고창이 나타날 경우 XSS취약점이 존재



5) 또한 아래 그림과 같이 검색란에 스크립트를 입력



6) 아래와 같이 '교육사이버안전센터' 그림이 나타나면 취약점이 존재



다. 대응 방안

1) 웹 서버 내에서의 조치

가) 웹 서버에서 입력 값에 정의된 문자 길이를 검증하여 javascript 등의 명령이 삽입되지 않도록 수정

나) 웹 서버의 검증 치환 등의 과정은 서버 사이드 스크립트(Server Side Script)에서 구현하여 검증 치환기능의 우회를 차단
(검증 치환 등의 기능을 javascript로 구현할 경우 우회 가능)

다) 웹 서버에서 HTML 형식의 입력이 불가피할 경우만 XSS 공격에 주로 사용되는 Tag입력을 차단

라) 웹 서버는 사용자 입력 폼(로그인 폼, 검색 폼, URL 등)을 대상으로 특수문자, 특수구문 필터링 규칙 적용

필터링 대상				
〈	〉	<	&gt ;	
javascript	eval	onmousewheel	onactivae	onfocusin
vbscript	innerHTML	ondataavailable	oncopy	onfocusout
expression	charset	onafterprint	oncut	onhelp
applet	document	onafterupdate	onclick	onkeydown
meta	string	onmousedown	onchange	onkeypress
xml	create	onbeforeactivate	onbeforecut	onkeyup
blink	append	onbeforecopy	ondblclick	onrowsdelete
link	binding	ondatasetchanged	ondeactivate	onload
style	alert	onbeforedeactivate	ondrag	onlosecapture
script	msgbox	onbeforeeditfocus	ondragend	onbounce
embed	refresh	onbeforepaste	ondragenter	onmouseenter
object	embed	onbeforeprint	ondragleave	onmouseleave
iframe	ilayer	onbeforeunload	ondragover	onbefore
frame	applet	onbeforeupdate	ondragstart	onmouseout
frameset	cookie	onpropertychange	ondrop	onmouseover
ilayer	javascript	ondatasetcomplete	onerror	onmouseup
layer	void	oncellchange	onerrorupdate	onresizeend
bgsound	href	onlayoutcomplete	onfilterchange	onabort
title	on paste	onmousemove	onfinish	onmoveend
base	onstart	oncontextmenu	onfocus	onmovestart
onreset	onresize	oncontrolselect	onresizestart	onrowenter
onmove	onrowexit	onreadystatechange	onunload	onsubmit
onstop	onselect	onselectionchange	onselectstart	onblur
		onrowsinserted		

마) 웹 서버의 취약점 조치를 완료한 후 위 과정을 다시 수행하여 XSS 취약점의 추가 존재 여부를 재점검

2) 홈페이지 개발 보안 조치

가) 홈페이지 소스코드느 사용자가 입력한 문자열에서〈, 〉, &, ", "등을 replace등의 문자 변환함수(혹은 Method)를 사용하여 &lt, &gt, &amp, &quot로 치환

나) 홈페이지 게시판 등에서 HTML 태그 허용 시 HTML 태그의 리스트(White List)를 선정한 후, 해당 태그만 허용하는 방식 적용

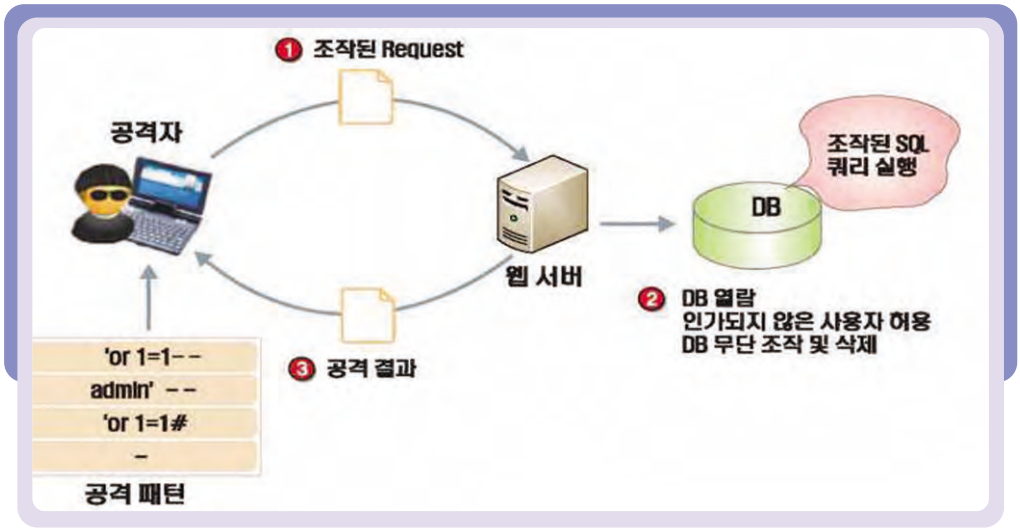
※ 제3장 홈페이지 개발 보안 방안의 5. 크로스사이트 스크립트를 참조, 소스코드를 수정하여 보안취약점을 조치하시길 바람

14. SQL Injection 취약점

가. 취약점 설명 및 사례

1) 취약점 설명

데이터베이스(DB)와 연동된 웹 어플리케이션에서 공격자가 입력 폼 및 URL입력란에 SQL문을 삽입하여 DB로부터 정보를 열람(또는 조작)할 수 있는 취약점

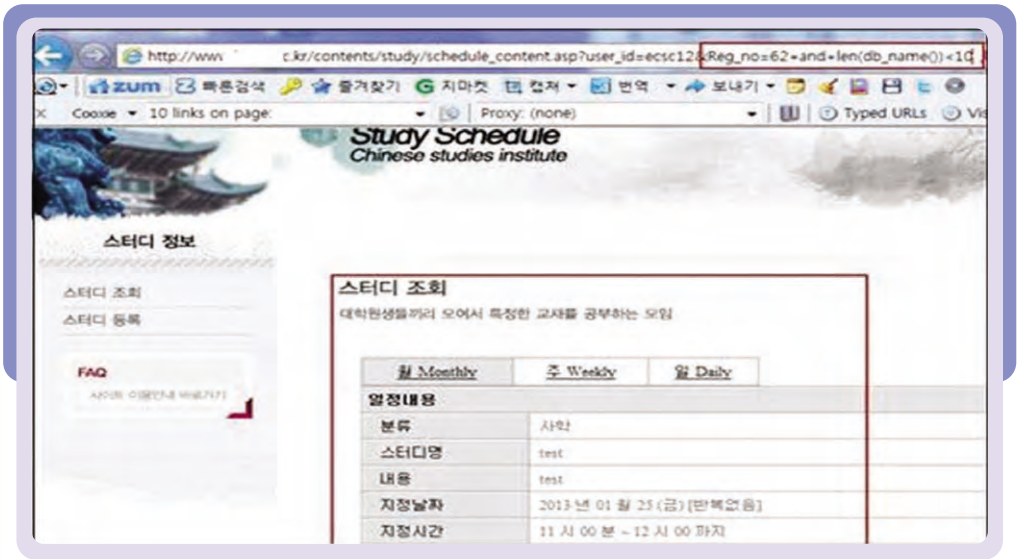


02 홈페이지 보안취약점 사례 및 대응

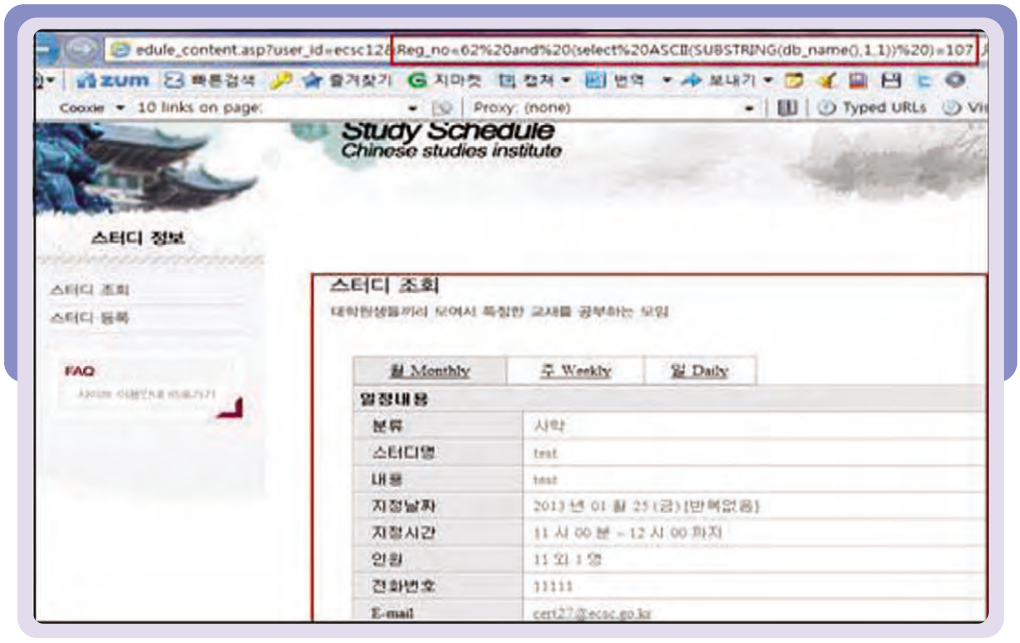
2) 사례

가) OO대학의 URL에 거짓 코드(Query)를 삽입하면 아래와 같이 비정상적인 화면이 노출

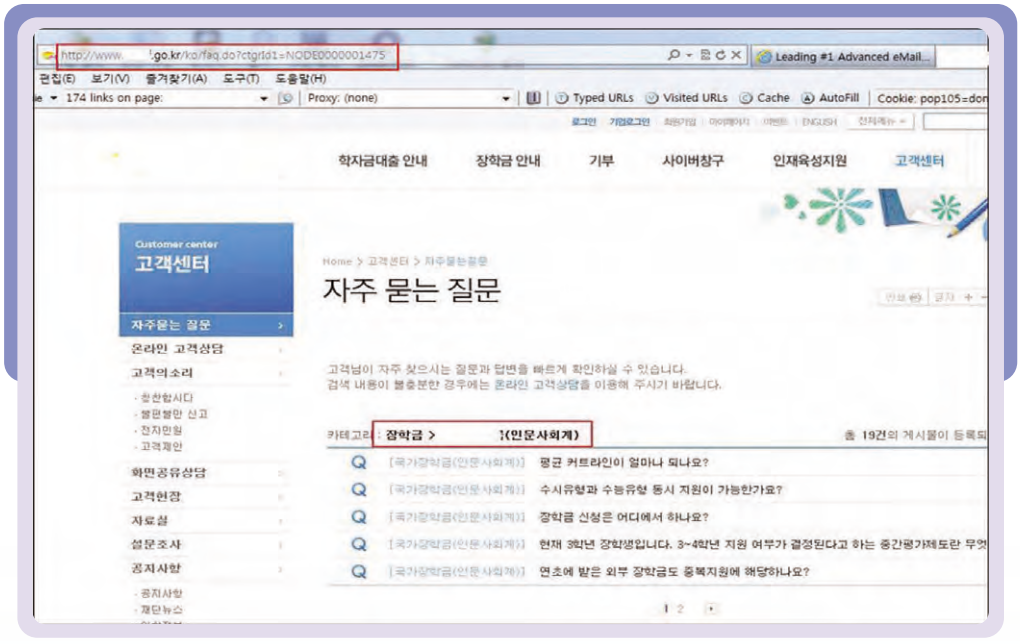
– DB이름의 길이를 알아내는 SQL코드를 삽입



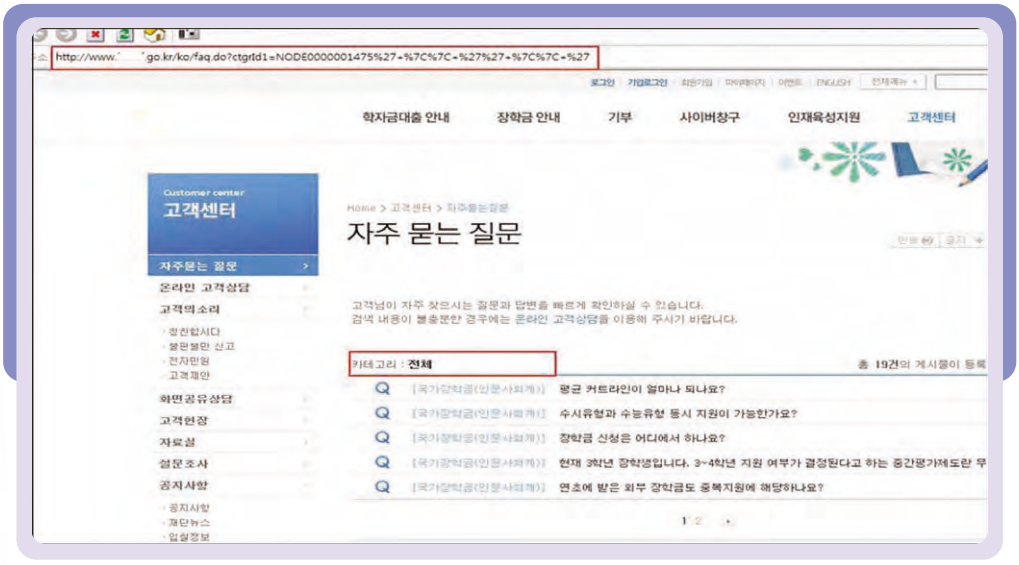
– 위와 같은 방식으로 범위를 좁혀나가고 단어를 하나씩 찾아내면 DB이름이"korea"로 시작함을 알 수 있음



나) OO기관의 매개변수 값에 오류를 발생시키는 코드('+||+"+||+')를 삽입하여 페이지를 요청하면 아래와 같이 카테고리 값이 변경되어 출력됨



02 홈페이지 보안취약점 사례 및 대응

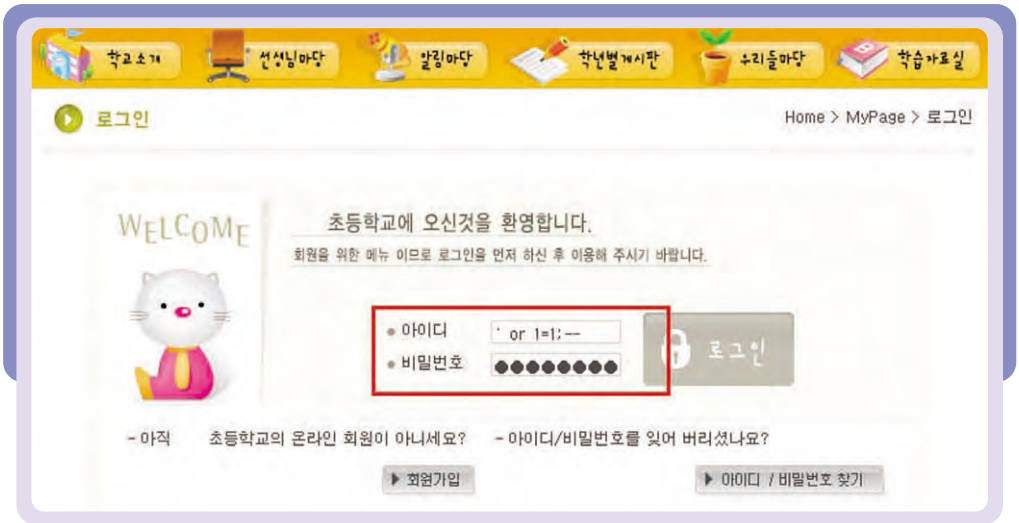


나. 점검 방법

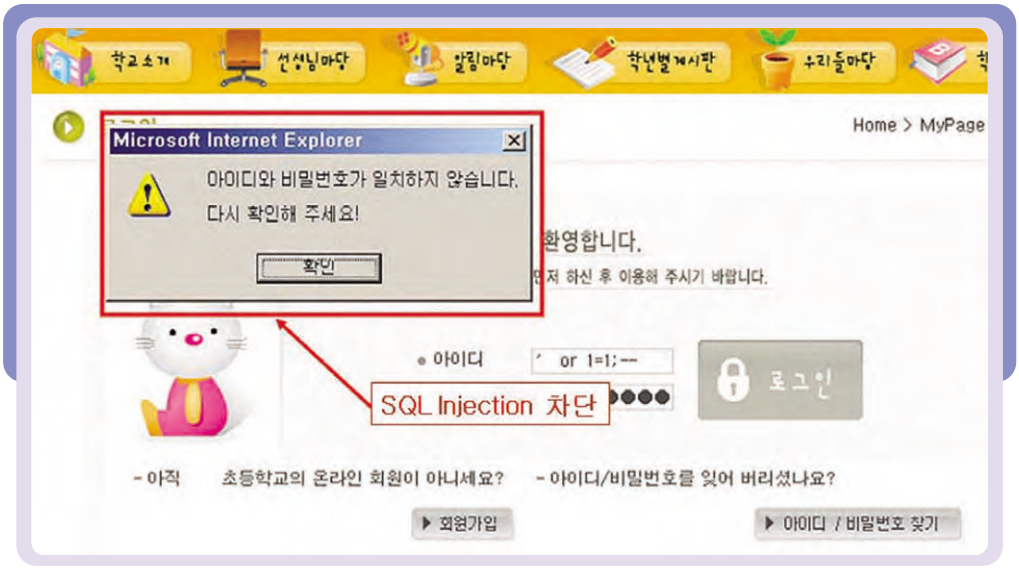
1) 로그인 페이지 점검

가) 관리하고 있는 웹서버의 로그인 페이지로 이동
나) 아이디와 패스워드 란에 아래 문자열을 입력하여 결과 확인

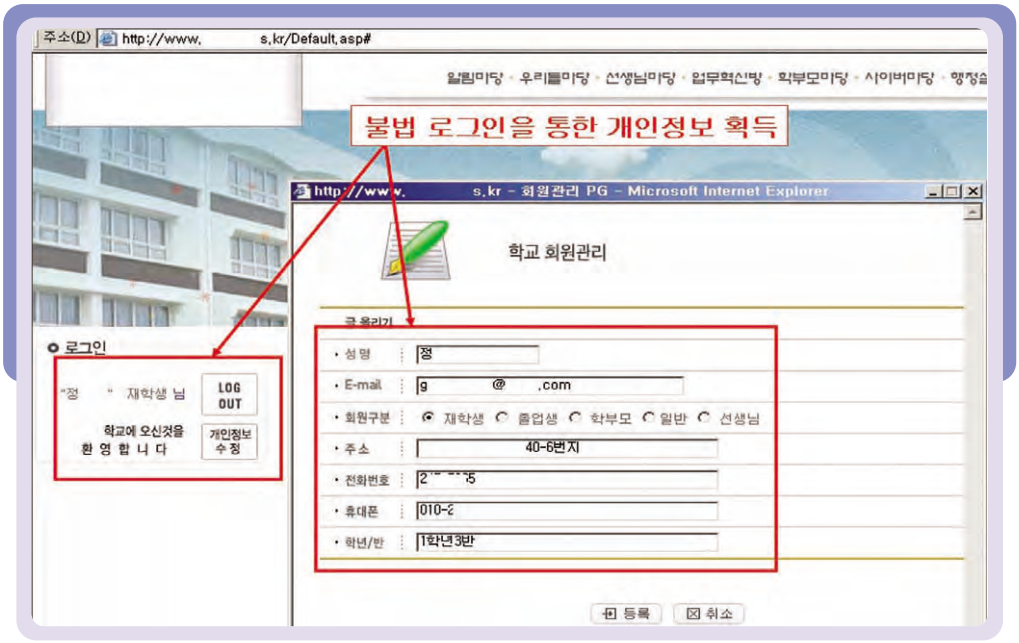
점검 예)
아이디 : ' or 1=1;--
비밀번호 : ' or 1=1;--



다) 인증 실패 메시지가 나타날 경우,'SQL Injection 취약점'은 존재하지 않는 것으로 추정 가능

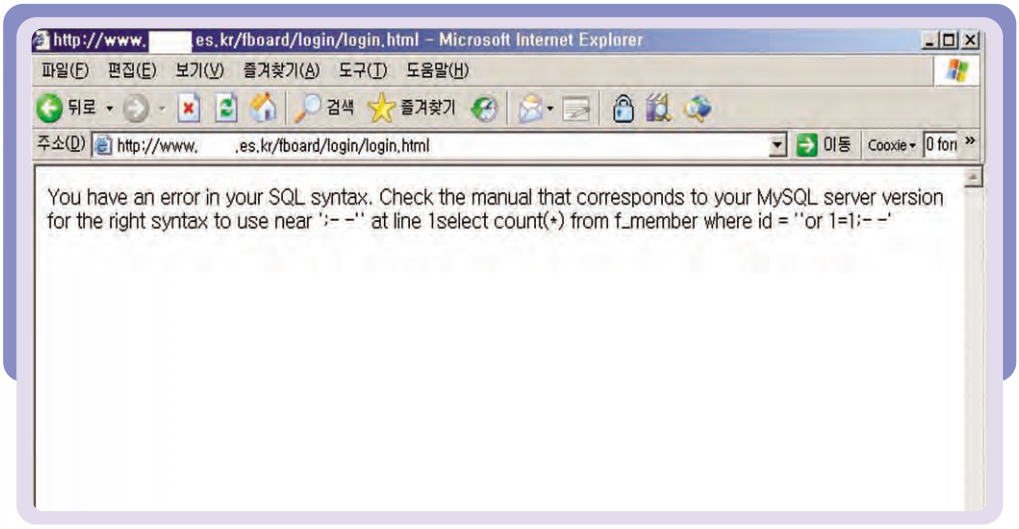


라) 로그인이 될 경우 SQL Injection 취약점이 존재



02 홈페이지 보안취약점 사례 및 대응

마) 웹서버 오류 메시지가 나타날 경우, SQL Injection 가능성이 있으므로 정보시스템, 홈페이지 보안가이드을 참고하여 세부적인 점검이 필요함



※ 다음과 같은 문자열을 추가적으로 점검해 볼 것.

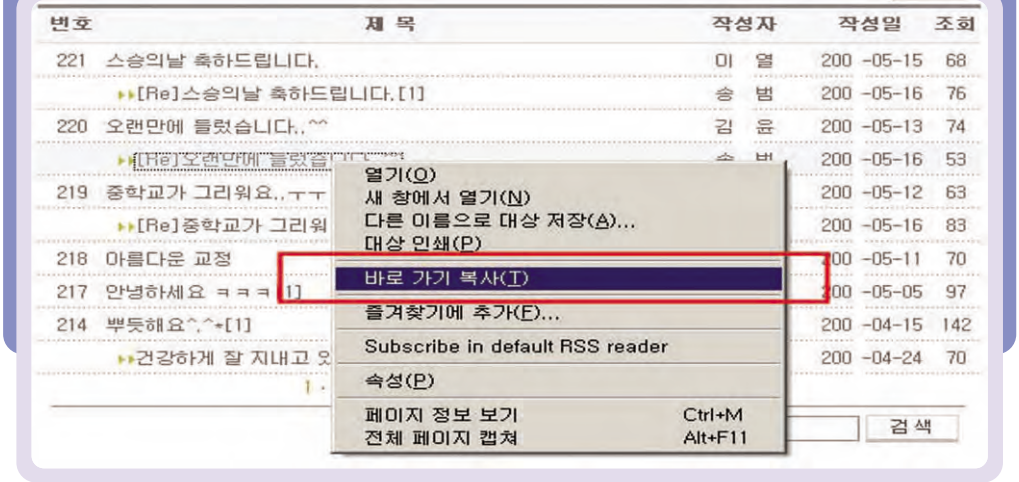
필터링 대상			
'or 1=1;-- -	or 1=1—	')or('a'='a	+ or 1=1-- -
' ' or 1=1-- -	'or 'a'='a	sql' or 1=1-- -	"
"or 1=1 —	" or "a"="a	sql" or 1=1—	,

2) 페이지 입력 값 점검

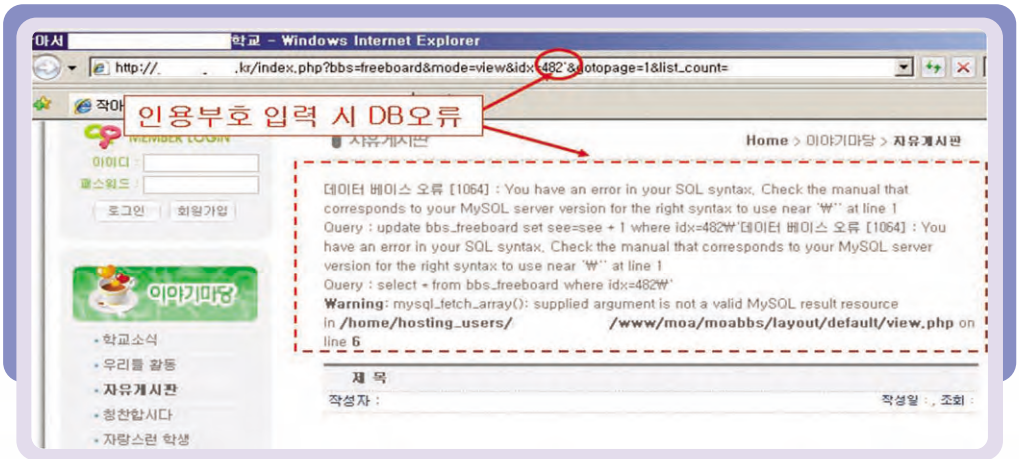
가) 관리하고 있는 웹서버의 게시판으로 이동
나) 게시판 등의 게시물 링크를 복사하여 브라우저의 주소표시줄에 입력
다) 주소표시줄에 입력한 값 중 게시판 번호(또는 글 번호) 등의 입력 값에 아래 예와 같이 인용부호(' 또는 ")를 입력하여 결과 확인

점검 예)

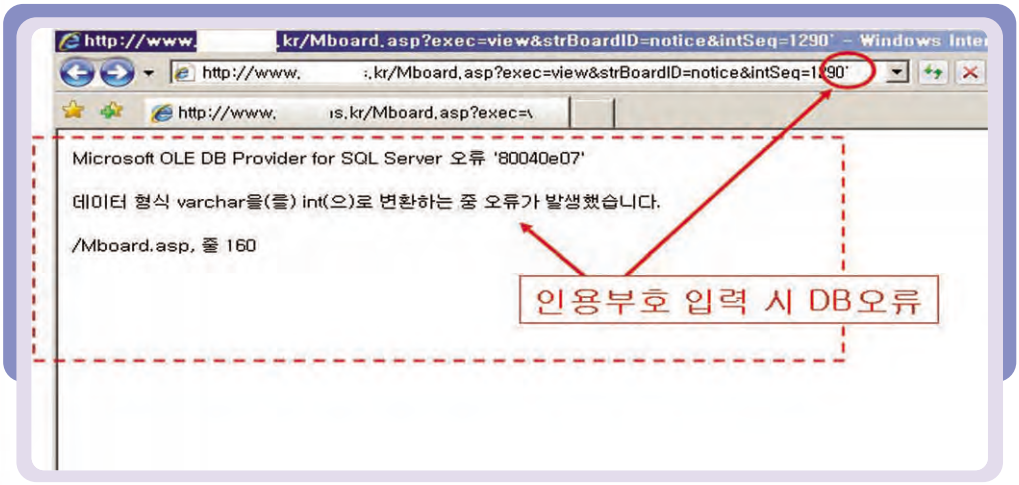
http://www.점검사이트.es.kr/bbs/view.asp?Name=Notice&bbs=09"
http://www.점검사이트.es.kr/bbs/view.asp?bbs=01"&page=09



라) 다음 같이 내용에 DB오류 또는 웹서버의 디렉터리가 노출될 경우, 입력값 검증 부재로 인해 추가 SQL Injection 공격을 차단(특수문자 치환 등)하는 장치가 마련되어 있지 않아 취약한 것을 의미



02 홈페이지 보안취약점 사례 및 대응



다. 대응 방안

1) 웹 서버 내에서의 조치

가) 웹 서버의 오류 정보가 사용자에게 노출되지 않도록 조치
나) 웹 애플리케이션과 연동되는 데이터베이스의 접근 권한을 최소화
다) 사용자 입력 폼(로그인 폼, 검색 폼, URL 등)을 대상으로 특수문자, 특수구분 필터링 규칙 적용

필터링 대상					
'	"	—	#	(	)
=	*/	/*	+	〈	〉
user_tables	user_table_columns		table_name	column_name	syscolumns
union	select	insert	drop	update	and
or	if	join	substrig	from	where
declare	substr	openrowset	xp_	'sysobjects	%

2) 홈페이지 개발 보안 조치

가) 홈페이지 소스코드는 사용자로부터 입력되는『입력 값』에 대한 검증과 예외처리
① ID, PASSWORD, 게시판 제목, 본문, 검색창, 주소검색창 등의 모든 입력란에 특수문자(등호, 부등호, 인용부호 등)를 직접 입력하지 못하도록 웹서버의 소스코드를 수정
② 입력 값에 정의된 문자 길이를 검증하여 SQL문이 추가 삽입되지 않도록 예외처리
③ 파라미터가 숫자인 경우 isnumeric과 같은 함수를 이용하여 검증하며, 문자인 경우 정규표현식을 이용하여 특수문자를 치환
특히 sql문에서 활용되는 문자(', ", ;, --, or 등)는 반드시 치환

※ 제3장 홈페이지 개발 보안 방안의 1. SQL 인젝션을 참조, 소스코드를 수정하여 보안취약점을 조치하시길 바람

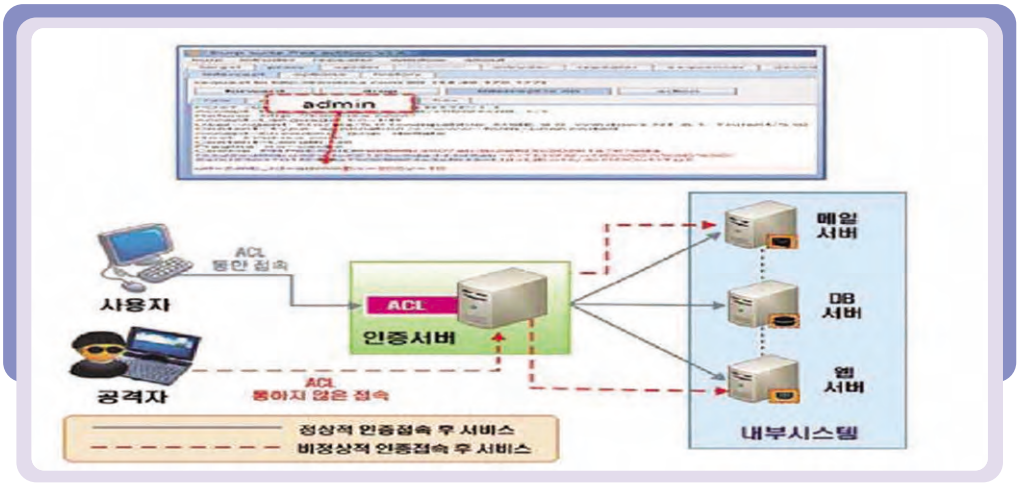
15. 권한인증 취약점

15. 1 파라미터/URL 변조

가. 취약점 설명 및 사례

1) 취약점 설명

웹 어플리케이션 상에서 모든 실행경로에 대해서 접근제어를 검사하지 않거나 불완전하게 검사하는 경우, 공격자가 접근 가능한 실행경로를 통해 정보를 유출할 수 있는 취약점



2) 사례

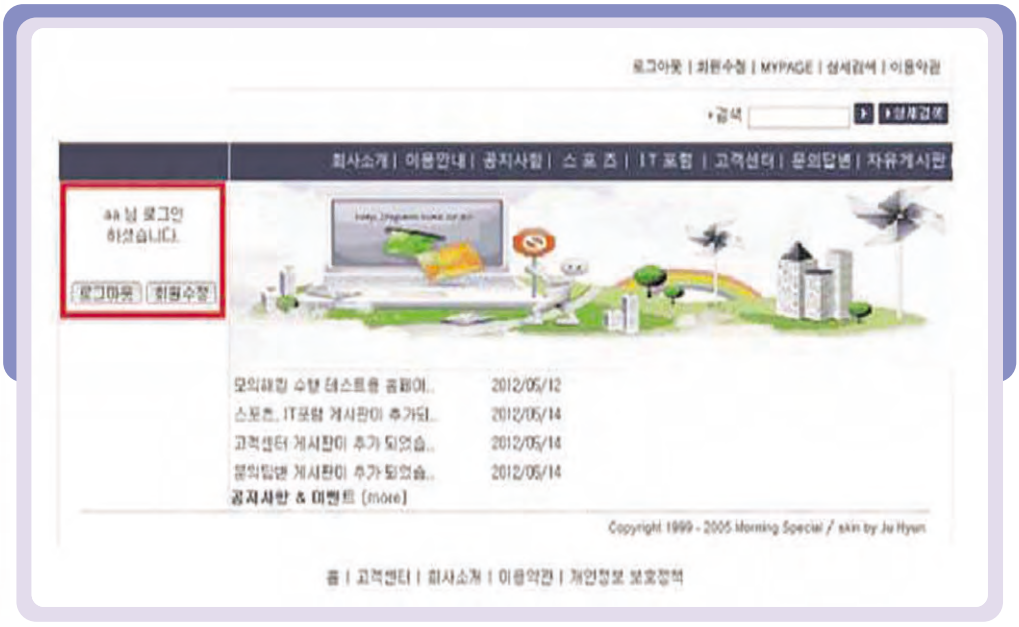
가) OO기관 홈페이지는 로그인 없이, 파라미터 값을 변경해서 관리자 게시물이나 타인의 게시물에 대한 수정 모드가 가능하고, 같은 방식으로 비밀 글 열람도 가능

02 홈페이지 보안취약점 사례 및 대응

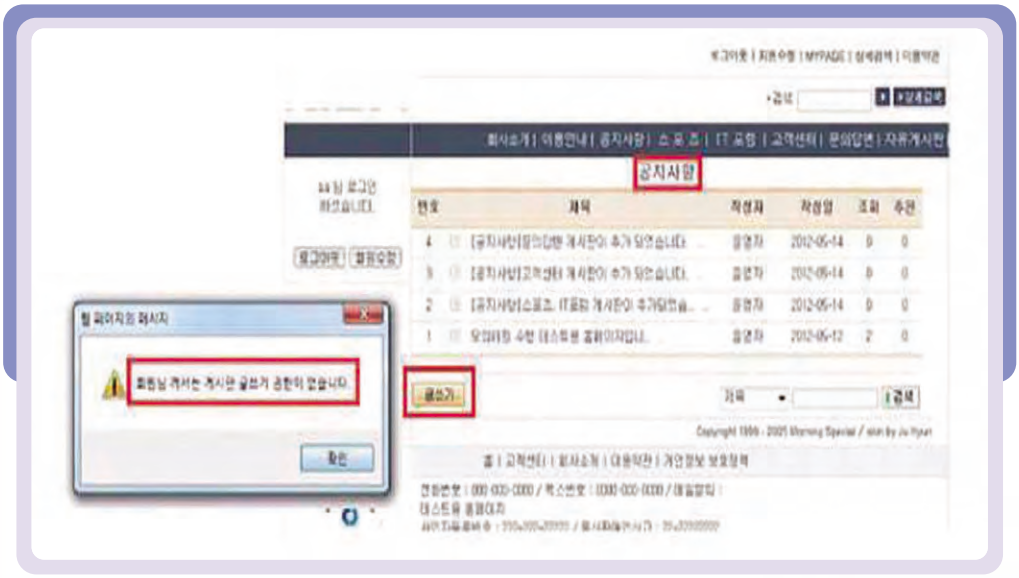
나. 점검 방법

1) 프록시 기능을 통해 파라미터 값을 조작하여 인증 우회 시도

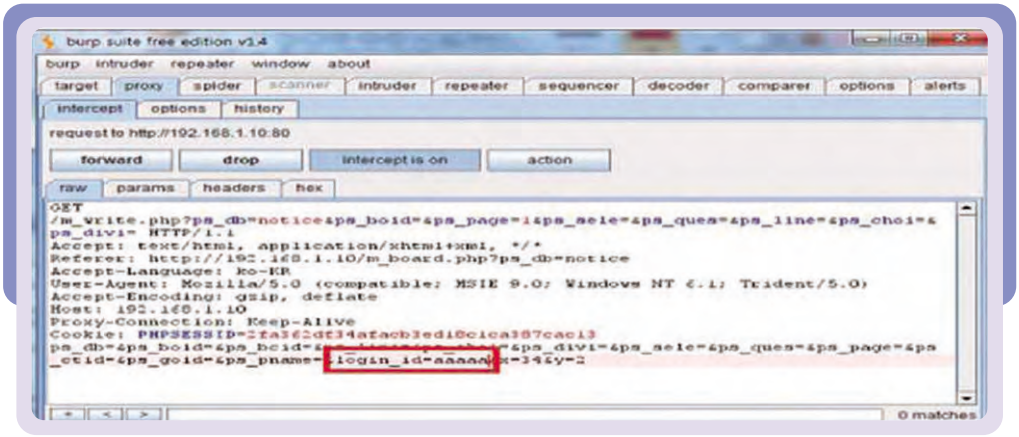
가) 일반 회원으로 로그인



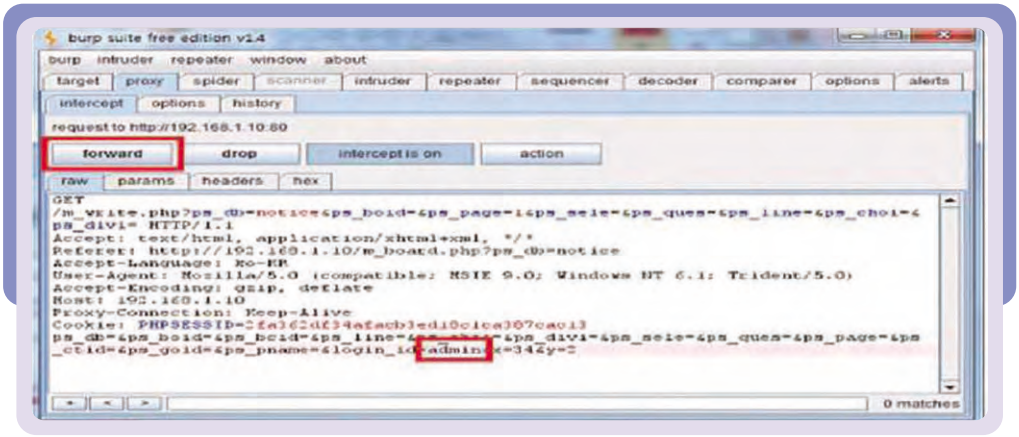
나) 공지사항(관리자 권한)글쓰기 시도



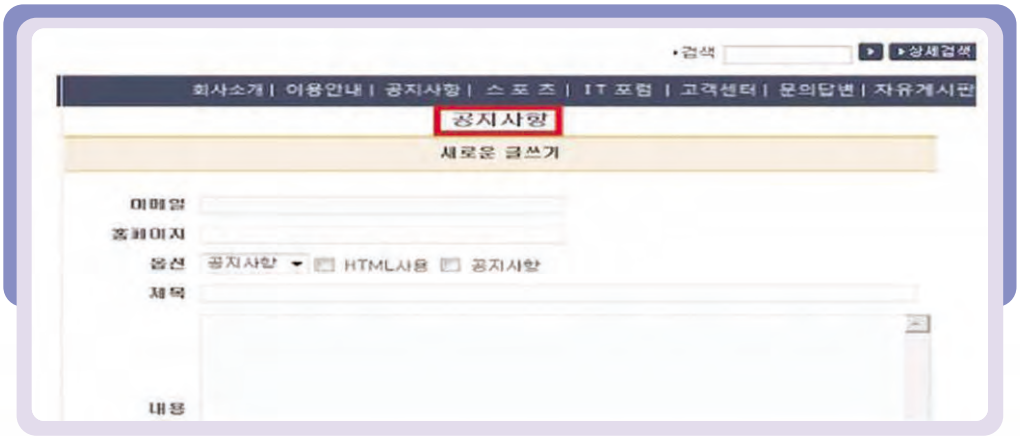
다) 글쓰기 시 프록시 기능을 통해[login_id]파라미터 확인



라) [login_id]의 파라미터 값을 [admin]으로 변경 후 전송



마) 운영자 권한으로 글쓰기 가능



02 홈페이지 보안취약점 사례 및 대응

2) URL을 조작하여 타 사용자의 게시글 수정

가) 일반 회원으로 로그인

나) 운영자의 글 수정 시도

다) 열람페이지 URL(m_view.php)를 수정페이지 URL(m_modify.php)로 변경

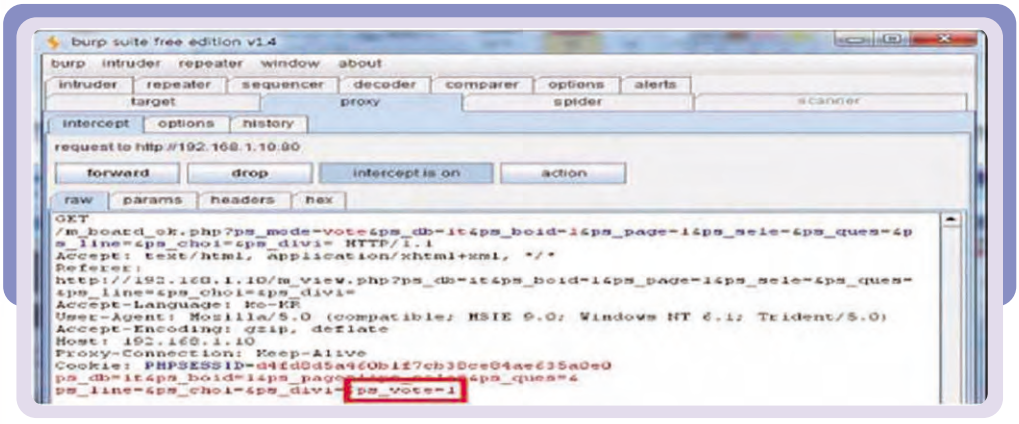
라) 권한 없이 수정페이지에 접근 및 성공

마) 결과(운영자 공지사항 수정 성공)

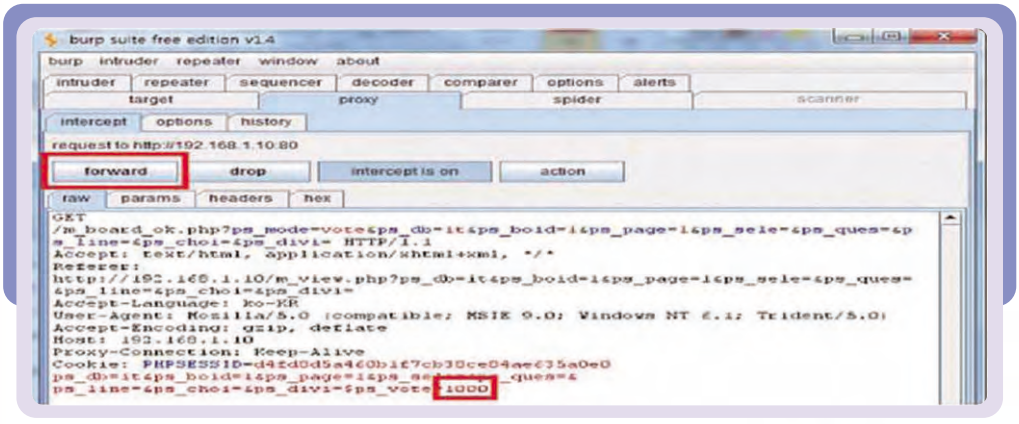
3. 프록시 기능을 통해 파라미터를 조작하여 게시판의 추천하기 기능 조작

가) 게시판의 [추천하기] 기능 확인

나) 추천 시 프록시 기능을 통해 [ps_vote] 파라미터 확인



다) [ps vote]의 파라미터 값 "1000"으로 변경 후 [forward] 클릭



라) 결과(추천 수 [1000]으로 조작 성공)

다. 대응 방안

1) 홈페이지 개발 보안 조치

가) 홈페이지 중 중요한 정보가 있는 페이지(계좌이체 등)는 재 인증 적용

나) 홈페이지 소스코드에는 안전하다고 확인된 라이브러리나 프레임워크 (OpenSSL이나 ESAPI의 보안 기능 등)를 사용

다) 응용프로그램이 제공하는 정보와 기능을 역할에 따라 배분함으로써 공격자에게 노출되는 공격노출면(attack surface) 최소화

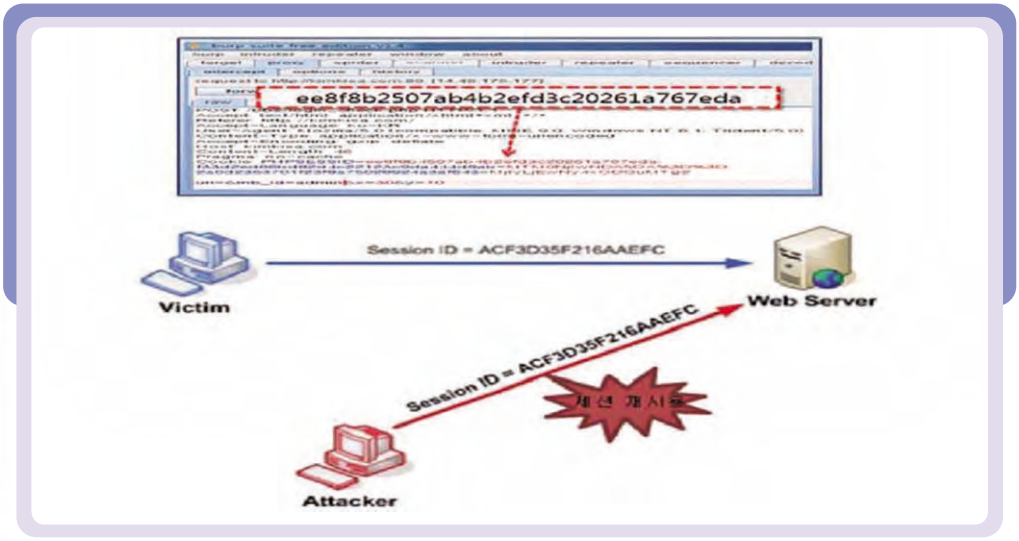
※ 제3장 홈페이지 개발 보안 방안의 11. URL/파리미터 변조를 참조, 소스코드를 수정하여 보안취약점을 조치하시길 바람

15. 2 세션 탈취

가. 취약점 설명 및 사례

1) 취약점 설명

인증 시 일정한 규칙이 존재하는 세션 ID가 발급되거나 세션 타임아웃을 너무 길게 설정된 경우 공격자에 의해 사용자 권한이 도용될 수 있는 취약점



2) 사례

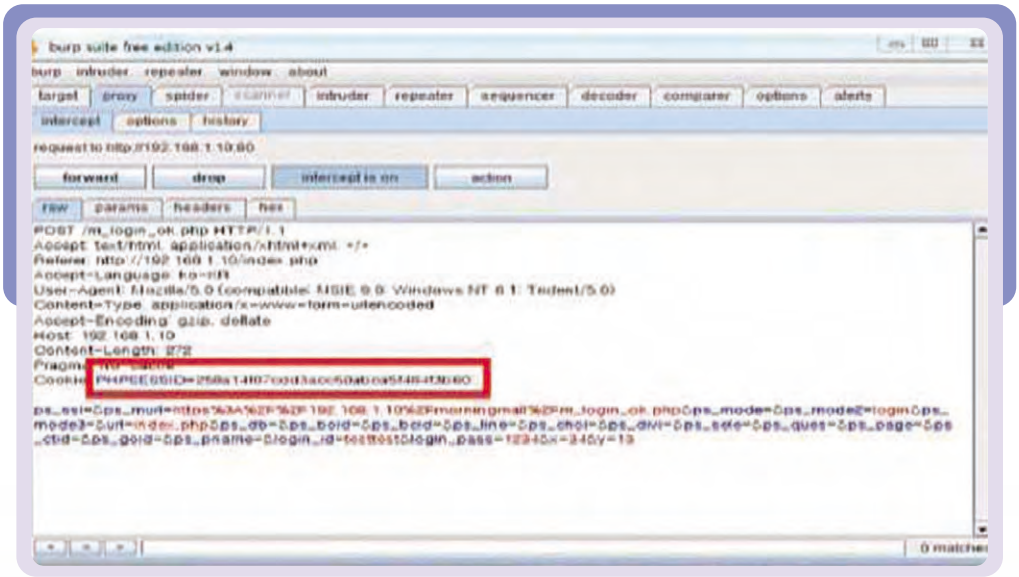
가) OO대학 홈페이지에 로그인 후 로그아웃을 하지 않고, 웹 브라우저를 종료 후 재접속하면 세션(Session)이 유지되어 로그인 상태로 접속됨

나. 점검 방법

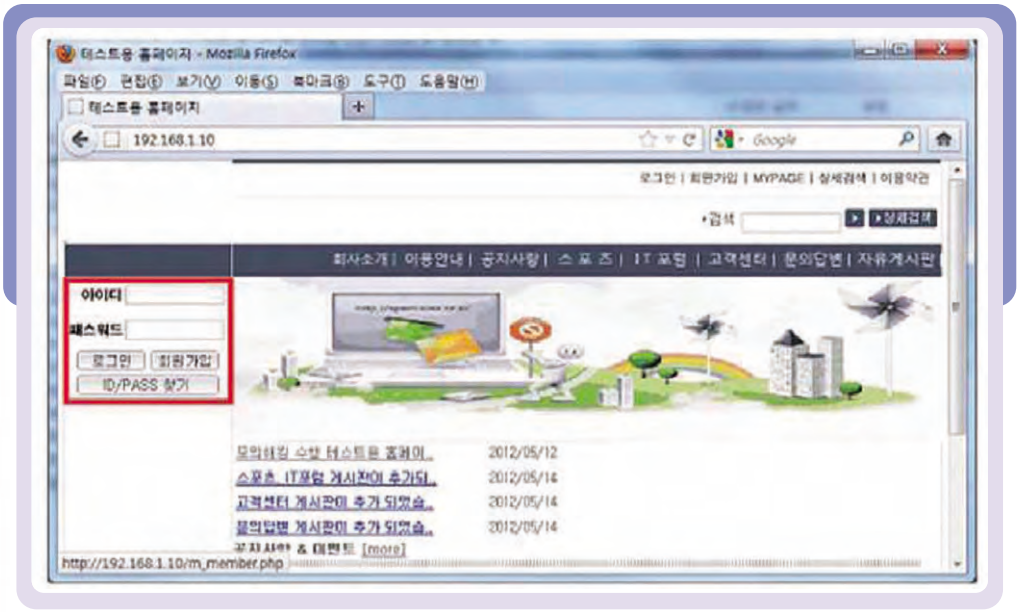
1) 자신의 아이디로 로그인 후 세션 값을 이용하여 다른 웹 브라우저에서 중복 로그인 가능 여부 확인

가) 일반 계정으로 로그인

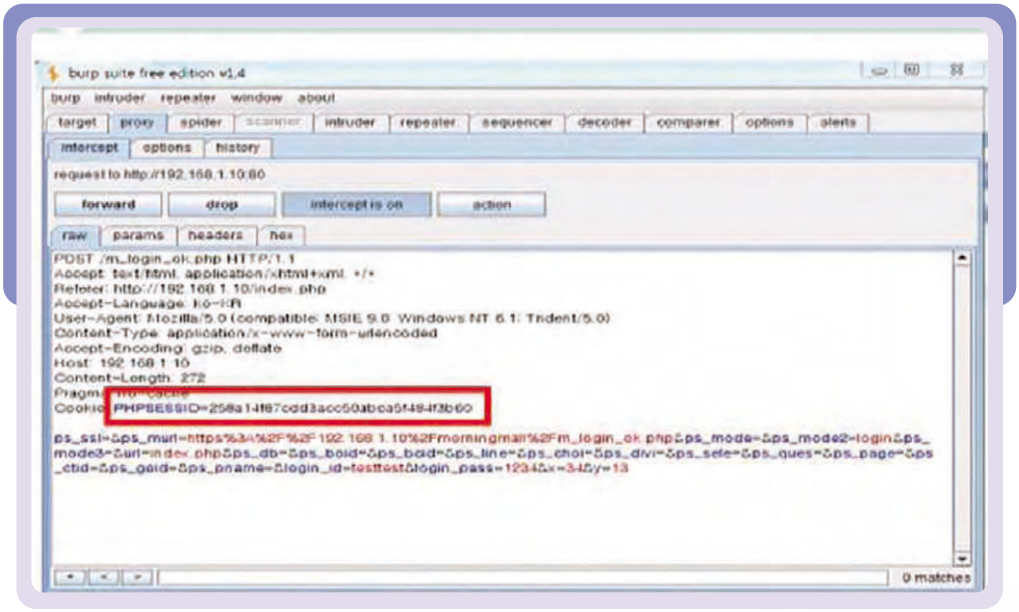
나) 프록시 프로그램을 활용하여 세션 값 확인



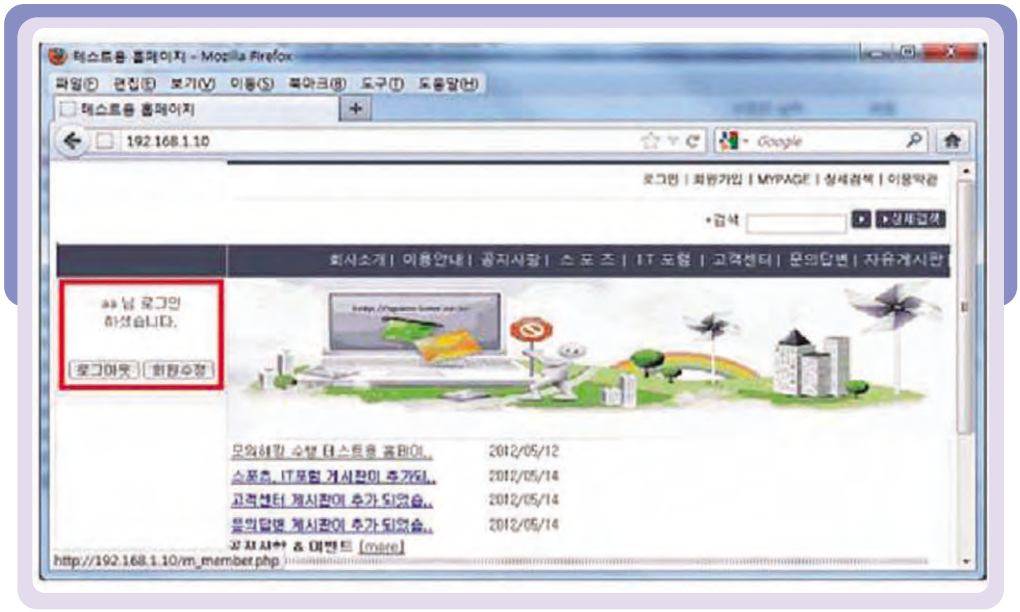
다) 새로운 웹 브라우저 실행(예:firefox)



라) 프록시 프로그램을 활용하여 기존의 세션 값으로 변조



마) 결과(변조된 세션 값으로 로그인 성공)

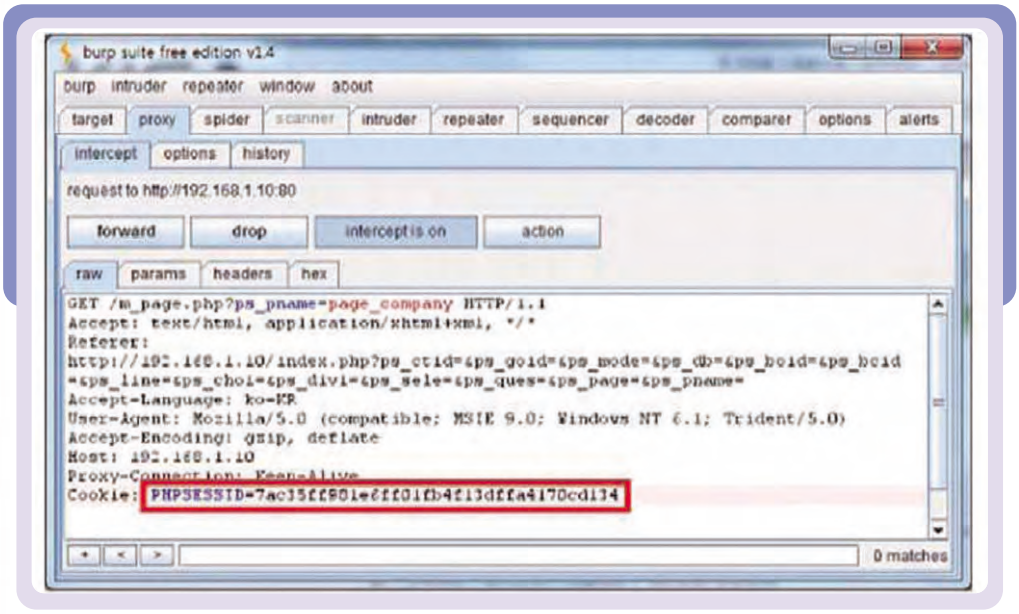


2) 여러 번 로그인을 시도하여 다음 로그인 대상의 세션 값이 특정한 규칙을 가지고 바뀌는지 확인

가) 일반 계정으로 로그인

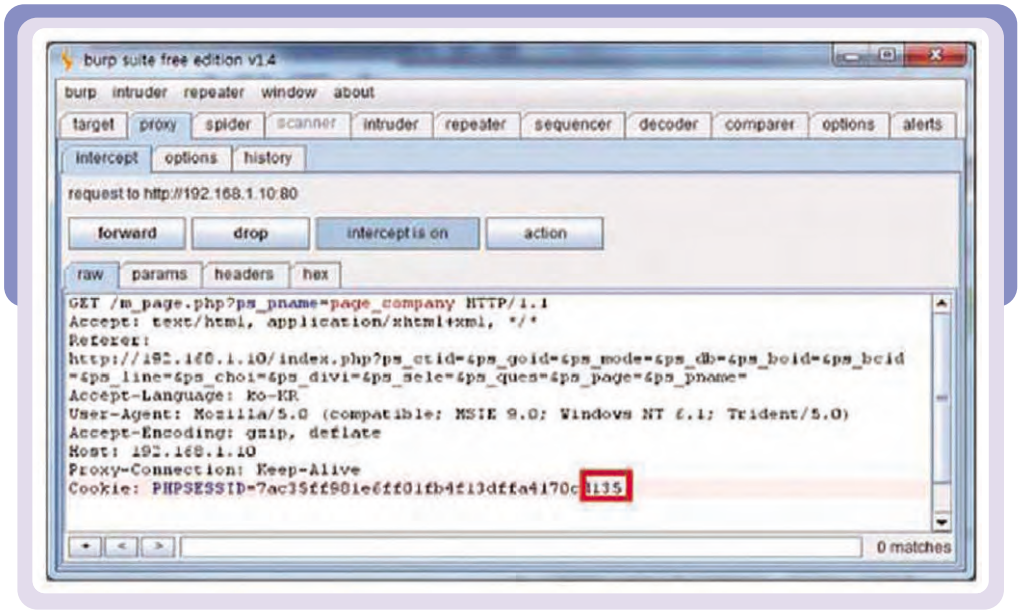
02 홈페이지 보안취약점 사례 및 대응

나) 프록시 프로그램을 활용하여 세션 값 확인

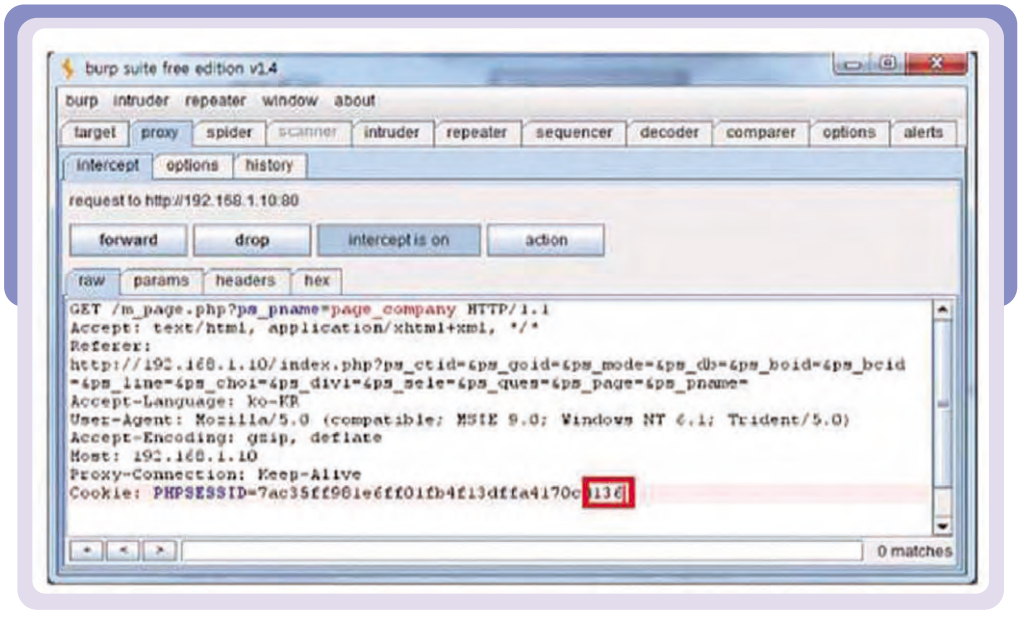


다) 로그인 후 같은 아이디로 로그인

라) 세션값이 1씩 커지는 것을 확인



마) 로그아웃 후 프록시 프로그램을 활용하여 기존 세션 값에 1을 더한 값으로 변조



바) 결과(로그인 성공)

다. 대응 방안

1) 홈페이지 개발 보안 조치

가) 홈페이지의 세션 ID는 로그인 시 마다 추측할 수 없는 새로운 세션 ID로 발급

나) 세션 타임아웃 설정을 통해 일정시간(최대 30분 이상) 동안 움직임이 없을 경우 자동 로그아웃 되도록 구현

※ 제3장 홈페이지 개발 보안 방안의 13. 불충분한 세션 관리를 참조, 소스코드를 수정하여 보안취약점을 조치

15.3 쿠키 변조

가. 취약점 설명 및 사례

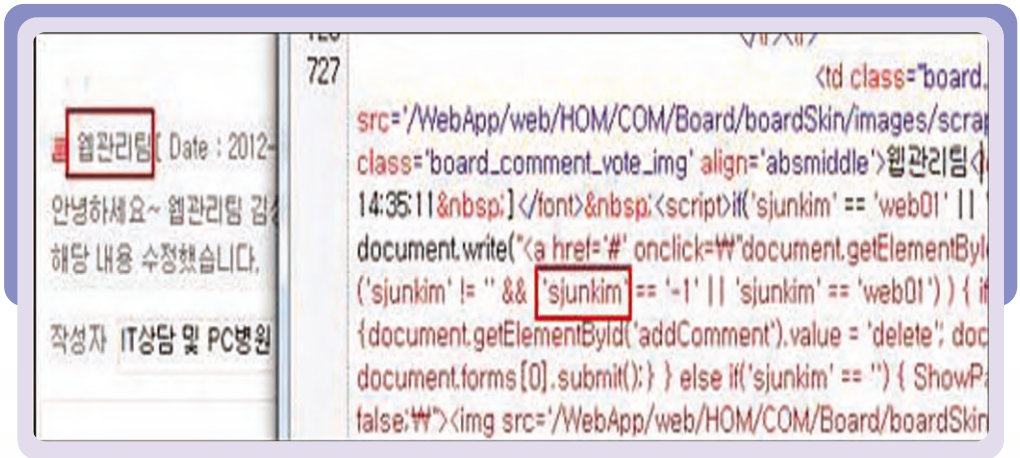
1) 취약점 설명

사용자 인증 방식중 하나인 쿠키를 변조하여 다른 사용자로 전환하거나 권한 상승이 가능한 취약점

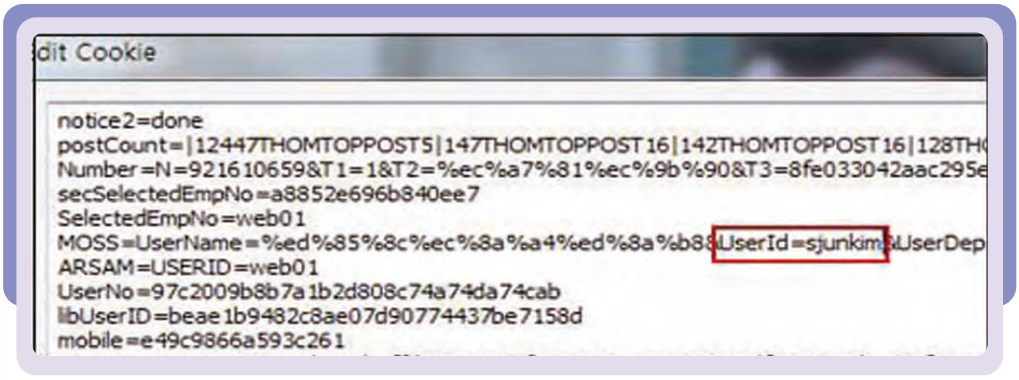
2) 사례

가) OO대학 사용자 세션(cookie 등)을 탈취, 인증 시스템을 우회하여 타사용자의 권한 획득이 가능

- 게시판에서 소스보기로 관리자 ID 파악



– 쿠키값 중 UserId의 값을 관리자 ID로 변경

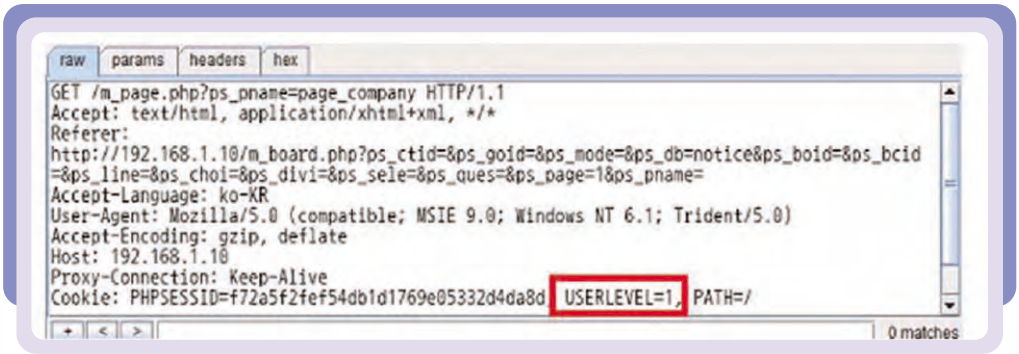


– 관리자로 로그인 됨. 관리자 페이지 접근가능

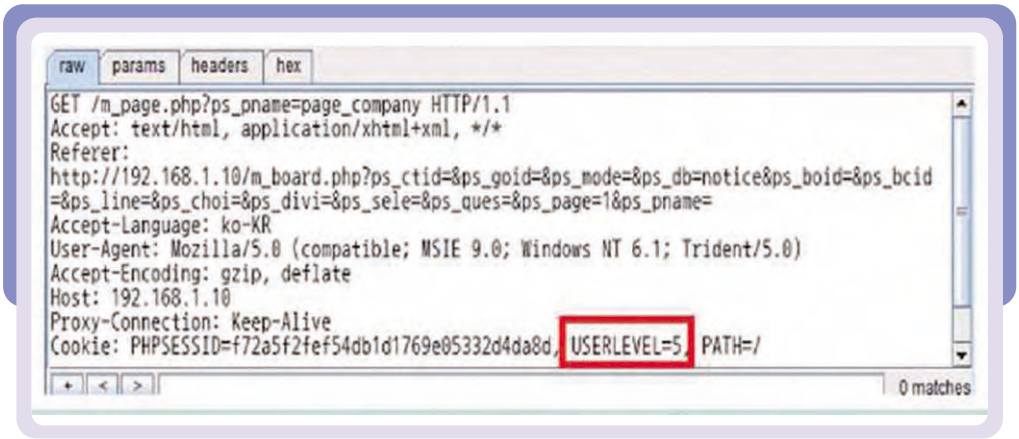
나. 점검 방법

1) 프록시 도구를 이용하여 쿠키 값 변조 후 로그인 결과 확인

가) 쿠키에서 사용자 레벨 확인



나) 쿠키 값 "(USERLEVEL=1 USERLEVEL=5)" 변조 후 시도



다) 일반 유저의 권한이 [5]로 상승

다. 대응 방안

1) 홈페이지 개발 보안 조치

가) 홈페이지는 사용자 인증 등 중요기능 구현 시 가급적이면 Cookie 대신 Session 방식 사용

나) 홈페이지의 사용자 인증 등 중요기능 구현 시 Cookie(또는 Session) 방식 활용 시 안전한 알고리즘(SEED, 3DES, AES 등)을 사용

※ 제3장 홈페이지 개발 보안 방안의 15. 쿠키변조 참조, 소스코드를 수정하여 보안취약점을 조치하시길 바람

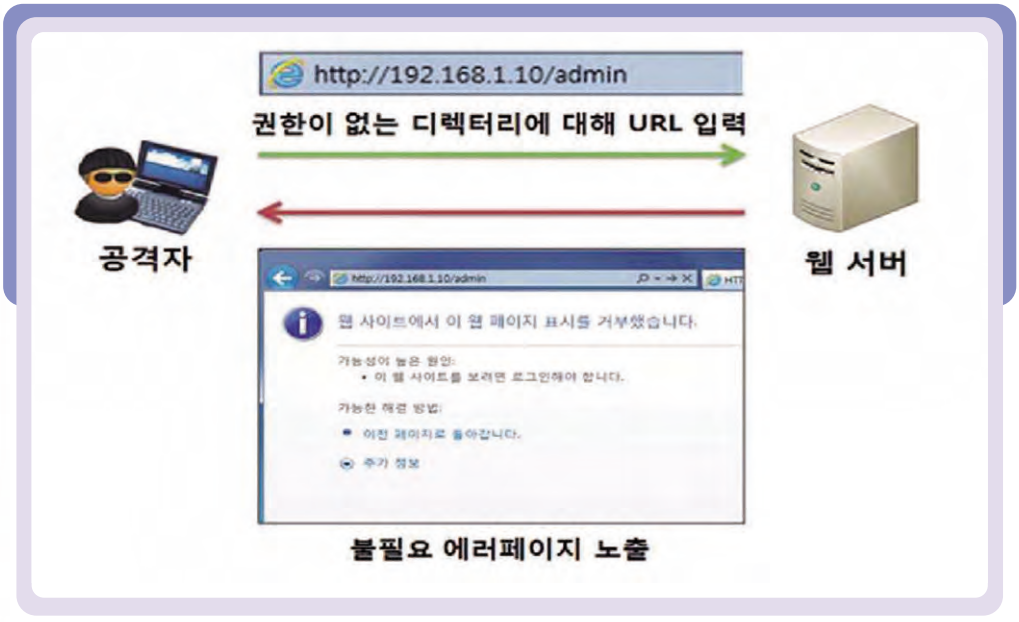
02 홈페이지 보안취약점 사례 및 대응

16. 에러처리 취약점

가. 취약점 설명 및 사례

1) 취약점 설명

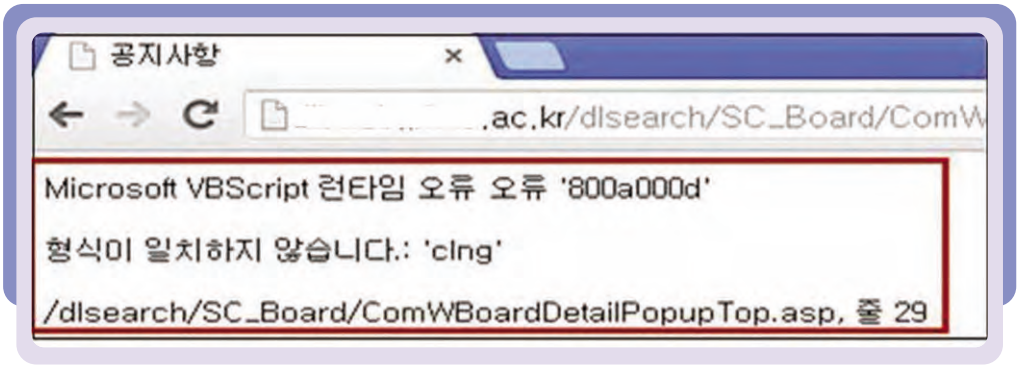
웹 서버에 별도의 에러페이지를 설정하지 않은 경우, 에러 메시지를 통해 서버 데이터 정보 등 공격에 필요한 정보가 노출되는 취약점



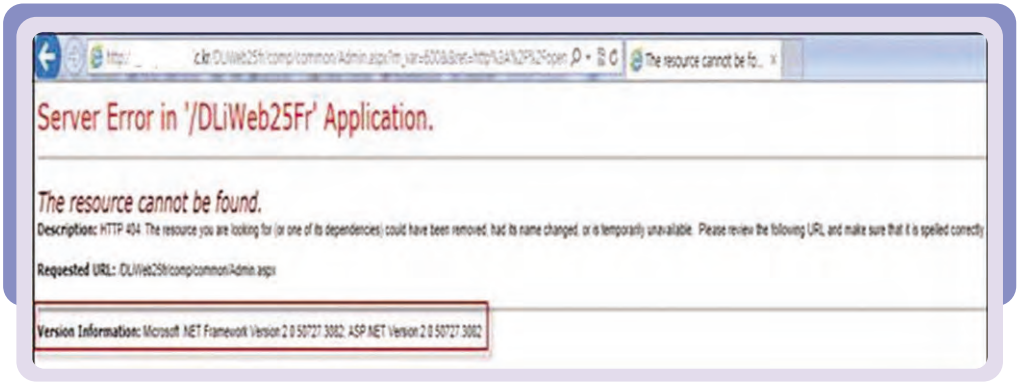
2) 사례

가) OO대학은 임의적인 정보 입력으로 웹 응용프로그램의 오류를 유도하여 응용프로그램의 정보 및 응용프로그램의 취약점을 파악 가능

– clng 함수 오류로, VBscript 에러가 발생함



나) OO대학은 ASP.NET 시스템 설정 오류로 시스템의 정보가 노출됨



나. 점검 방법

1) URL에 웹 서버 디렉터리명을 입력하여 에러페이지 확인

가) 디렉터리 경로가 포함된(예:admin/admin.php) URL 확인

나) admin.php 부분을 지운 후 접속하여 에러페이지 확인

다. 대응 방안

1) 홈페이지의 에러페이지는 별도의 에러페이지를 제작하여 에러발생 시 에러페이지로 Redirection

안전한 서버 설정

```
1:  <web-app>
2:  <error-page>
3:      <error-code>400</error-code>
4:      <location>/400error.jsp</location>
5:  </error-page>
6:  <error-page>
7:      <error-code>404</error-code>
8:      <location>/404error.jsp</location>
9:  </error-page>
10: <error-page>
11:     <error-code>403</error-code>
12:     <location>/403error.jsp</location>
13: </error-page>
14: <error-page>
15:     <error-code>500</error-code>
16:     <location>/500error.jsp</location>
17: </error-page>
18: <error-page>
19:     <exception-type>java.lang.Throwable</exception-type>
20:     <location>/exception_error.jsp</location>
21: </error-page>
22: </web-app>
```

웹 서버 및 홈페이지 취약점 점검 가이드

제 3장

홈페이지 개발 보안 방안

1. SQL Injection
2. 운영체제 명령 실행
3. XQuery 인젝션
4. XPath 인젝션
5. 크로스사이트 스크립트(XSS)
6. 파일 업로드
7. 파일 다운로드
8. 버퍼 오버플로우
9. LDAP 인젝션
10. HTTP 응답 분할
11. URL / 파라미터 변조
12. 취약한 계정 생성 허용
13. 불충분한 세션 관리
14. 데이터 평문전송
15. 쿠키 변조
16. 취약한 암호화 알고리즘 사용
17. 취약한 패스워드 복구
18. 주석을 통한 정보노출

03 홈페이지 개발 보안 방안

1. SQL 인젝션

가. 취약점 설명

공격자가 입력 폼 및 URL 입력란에 SQL 문을 삽입하여 DB로부터 정보를 열람하거나 조작할 수 있는 보안취약점

나. 보안대책

① Prepared Statement 객체 등을 이용하여 DB에 컴파일된 쿼리문(상수)을 전달하는 방법 사용
② Parameterized Statement를 사용하는 경우, 외부 입력데이터 에 대하여 특수문자 및 쿼리 예약어 필터링
③ Struts, Spring 등과 같은 프레임워크를 사용하는 경우 외부 입력 값 검증 모듈 사용

다. 코드예제

다음은 안전하지 않은 코드의 예를 나타낸 것으로, 외부로부터 tableName과 name의 값을 받아서 SQL 쿼리를 생성하고 있으며, name의 값으로 name' OR 'a'='a를 입력하면 조작된 쿼리문 전달이 가능한 형태의 코딩

안전하지 않은 코드의 예 JAVA

```
1:  try
2:  {
3:    String tableName = props.getProperty("jdbc.tableName");
4:    String name = props.getProperty("jdbc.name");
5:    String query = "SELECT * FROM " + tableName + " WHERE Name =" + name;
6:    stmt = con.prepareStatement(query);
7:    rs = stmt.executeQuery();
8:    ... ...
9:  }
10:  catch (SQLException sqle) {    }
11:  finally {    }
```

다음은 안전하지 않은 코드를 안전하게 고친 예를 나타낸 것으로, 외부로부터 인자를 받는 preparedStatement 객체를 상수 스트링으로 생성하고, 인자 부분을 setXXX Method로 설정하여, 외부의 입력이 쿼리문의 구조를 바꾸는 것을 방지하는 코딩

안전한 코드의 예 JAVA

```
1:  try
2:  {
3:    String tableName = props.getProperty("jdbc.tableName");
4:    String name = props.getProperty("jdbc.name");
5:    String query = "SELECT * FROM " + tableName + " WHERE Name =" + name;
6:    stmt = con.prepareStatement(query);
7:    rs = stmt.executeQuery();
8:    ... ...
9:  }
10:  catch (SQLException sqle) {    }
11:  finally {    }
```

2. 운영체제 명령 실행

가. 취약점 설명

적절한 검증절차를 거치지 않은 사용자 입력 값이 운영체제 명령어의 일부 또는 전부로 구성되어 실행되는 경우, 의도하지 않은 시스템 명령어가 실행되는 취약점

나. 보안대책

① 웹 인터페이스를 통해 서버내부로 시스템 명령어를 전달시키지 않도록 웹 어플리케이션 구성
② 외부 입력에 따라 명령어를 생성하거나 선택이 필요한 경우, 명령어 생성에 필요한 값들을 미리 지정해 놓고 외부 입력에 따라 선택하여 사용

다. 코드예제

다음의 예제는 cmd.exe 명령어를 사용하여 rmanDB.bat 배치 명령어를 수행하며, 외부에서 전달되는 dir_type 값이 manDB.bat의 인자값으로서 명령어 스트링의 생성에 사용될 수 있는 코딩

안전하지 않은 코드의 예 JAVA

```
1:  ……
2:    props.load(in);
3:    String version = props.getProperty("dir_type");
4:    String cmd = new String("cmd.exe /K   ₩"rmanDB.bat ₩"");
5:    Runtime.getRuntime().exec(cmd + " c:₩₩prog_cmd₩₩" + version);
6:  ……
```

다음의 예제와 같이 미리 정의된 인자 값의 배열을 만들어 놓고, 외부의 입력에 따라 적절한 인자 값을 선택하도록 하여, 외부의 부적절한 입력이 명령어로 사용될 가능성 배제한 코딩

안전한 코드의 예 JAVA

```
1:  ……
2:    props.load(in);
3:    String version[] = {"1.0", "1.1"};
4:    int versionSelection = Integer.parseInt(props.getProperty("version"));
5:    String cmd = new String("cmd.exe /K   ₩"rmanDB.bat ₩"");
6:    String vs = "";
7:    if (versionSelection == 0)
8:      vs = version[0];
9:    else if (versionSelection == 1)
10:     vs = version[1];
11:   else
12:     vs = version[1];
13:   Runtime.getRuntime().exec(cmd + "   c:₩₩prog_cmd₩₩" + vs);
14: ……
```

3. XQuery 인젝션

가. 취약점 설명

XML 데이터에 대한 동적 쿼리문(XQuery)을 생성할 때 외부 입력 값에 대해 적절한 검증절차가 존재하지 않으면, 공격자가 쿼리문의 구조를 임의로 변경 가능하게 하는 취약점

나. 보안대책

① XQuery에 사용되는 외부 입력데이터에 대하여 특수 문자및 쿼리 예약어 필터링
② XQuery를 사용한 쿼리문은 스트링을 연결하는 형태로 구성하지 않고 인자(파라메터)화된 쿼리문 사용

다. 코드예제

다음의 예제에서는 외부의 입력(name)값에 something' or '1'='1' 을 name의 값으로 변조한 후 executeQuery를 사용하여 쿼리문을 전달하면 파일 내의 모든 값이 출력될 수 있는 코딩

"doc('users.xml')/userlist/user[uname='something' or '1'='1']"

안전하지 않은 코드의 예 JAVA

```
1:  ……
2:  // 외부로 부터 입력을 받음
3:  String name = props.getProperty("name");
4:  Hashtable env = new Hashtable();
5:  env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
6:  env.put(Context.PROVIDER_URL, "ldap://localhost:389/o=rootDir");
7:  javax.naming.directory.DirContext ctx = new InitialDirContext(env);
8:  javax.xml.xquery.XQDataSource xqds = (javax.xml.xquery.XQDataSource)
    ctx.lookup("xqj/personnel");
9:  javax.xml.xquery.XQConnection conn = xqds.getConnection();
10: String es = "doc('users.xml')/userlist/user[uname='" + name + "']";
11: // 입력값이 Xquery의 인자로 사용
12: XQPreparedExpression expr = conn.prepareExpression(es);
13: XQResultSequence result = expr.executeQuery();
14: ……
```

다음의 예제에서는 외부 입력 값을 받고 해당 값 기반의 XQuery상의 쿼리 구조를 변경시키지 않는 bindXXX함수를 이용함으로써 외부의 입력으로 인하여 쿼리 구조가 바뀌는 것을 차단하도록 코딩

안전한 코드의 예 JAVA

```
1:  ……
2:  // 외부로 부터 입력을 받음
3:  String name = props.getProperty("name");
4:  Hashtable env = new Hashtable();
5:  env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
6:  env.put(Context.PROVIDER_URL, "ldap://localhost:389/o=rootDir");
7:  javax.naming.directory.DirContext ctx = new InitialDirContext(env);
8:  javax.xml.xquery.XQDataSource xqds = (javax.xml.xquery.XQDataSource)
    ctx.lookup("xqj/personnel");
9:  javax.xml.xquery.XQConnection conn = xqds.getConnection();
10:
11: String es = "doc('users.xml')/userlist/user[uname='$xpathname']";
12: // 입력값이 Xquery의 인자로 사용
13: XQPreparedExpression expr = conn.prepareExpression(es);
14: expr.bindString(new QName("xpathname"), name, null);
15: XQResultSequence result = expr.executeQuery();
16: while (result.next())
17: {
18:   String str = result.getAtomicValue();
19:   if (str.indexOf('>') < 0)
20:   {
21:     System.out.println(str);
22:   }
```

4. XPath 인젝션

가. 취약점 설명

외부 입력 값을 적절한 검사과정 없이 XPath 쿼리문 전달이 가능해 지면, 공격자가 쿼리문의 구조를 임의로 변경 가능하게 하는 취약점

나. 보안대책

① Xpath 쿼리에 사용되는 외부 입력데이터에 대하여 특수문자(', [,], /, =, @ 등) 및 쿼리 예약어 필터링 수행

② 인자화된 쿼리문을 지원하는 XQuery문 사용

다. 코드예제

다음 예제에서 name의 값으로 user1', passwd의 값으로 'or '='을 전달하면 다음과 같은 질의문이 생성되어 인증과정을 거치지 않고 로그인이 가능한 코딩
// users/user[login/text() = 'user1 and password/text() = '' or ''= '']/home_dir/text()

안전하지 않은 코드의 예 JAVA

```
1:  ……
2:  // 외부로 부터 입력을 받음
3:  String name = props.getProperty("name");
4:  String passwd = props.getProperty("password");
5:  ……
6:  XPathFactory factory = XPathFactory.newInstance();
7:  XPath xpath = factory.newXPath();
8:  ……
9:  // 외부 입력이 xpath의 인자로 사용
10: XPathExpression expr = xpath.compile("//users/user[login/text()='" + name + "' and
    password/text() = '" + passwd + "']/home_dir/text()");
11: Object result = expr.evaluate(doc, XPathConstants.NODESET);
12: NodeList nodes = (NodeList) result;
13: for (int i = 0; i < nodes.getLength(); i++)
14: {
15:   String value = nodes.item(i).getNodeValue();
16: ……
```

인자화된 쿼리문을 지원하는 XQuery를 사용하여 미리 쿼리 골격을 생성하고, 이에 인자값을 설정함으로써 외부입력으로 인해 쿼리 구조가 바뀌는 것을 차단하도록 코딩

안전한 코드의 예 JAVA

dologin.xp 파일

```
1:  declare variable $loginID as xs:string external;
2:  declare variable $password as xs:string external;
3:  //users/user[@loginID=$loginID and @password=$password]
```

XQuery를 이용한 XPath Injection 방지

```
1:  // 외부로 부터 입력을 받음
2:  String name = props.getProperty("name");
3:  String passwd = props.getProperty("password");
4:  Document doc = new Builder().build("users.xml");
5:  // XQuery를 위한 정보 로딩
6:  XQuery xquery = new XQueryFactory().createXQuery(new File("dologin.xq"));
7:  Map vars = new HashMap();
8:  vars.put("loginID", name);
9:  vars.put("password", passwd);
10: Nodes results = xquery.execute(doc, null, vars).toNodes();
11: for (int i=0; i < results.size(); i++)
12: {
13:     System.out.println(results.get(i).toXML());
14: }
```

5. 크로스 사이트 스크립트

가. 취약점 설명

외부 입력이 동적 웹페이지 생성에 사용될 경우, 전송된 동적 웹페이지를 열람하는 접속자의 권한으로 부적절한 스크립트가 수행 되는 취약점

나. 보안대책

① 사용자가 입력한 문자열에서〈, 〉, &, ", ' 등을 replace등의 문자 변환함수(혹은 Method)를 사용하여 &lt, &gt, &amp, &quot로 치환
② 게시판 등에서 HTML 태그 허용 시 HTML 태그의 리스트(White List)를 선정한 후, 해당 태그만 허용하는 방식 적용

다. 코드예제

파라미터(name)에 〈script〉alert(document.cookie);〈/script〉와 같은 스크립트 코드가 입력되고, 이 값이 그대로 사용되면 사용자의 쿠키정보가 공격자에게 전송될 수 있는 코딩

03 홈페이지 개발 보안 방안

안전하지 않은 코드의 예 JAVA

```
1:  <h1>XSS  Sample</h1>
2:  <%
3:    String name = request.getParameter("name");
4:  %>
5:  <p>NAME:<%=name%></p>
```

외부 입력 문자열에서 replaceAll() Method를 사용하여 〈, 〉, &, ” “ 같이 스크립트 생성에 사용되는 문자열을 &lt, &gt, &amp, &quot 등으로 변경하면, 파라미터 name에 악성코드가 포함되더라도 스크립트 실행 불가하도록 코딩

안전한 코드의 예 JAVA

```
1:  <%
2:    String name = request.getParameter("name");
3:  if ( name != null )
4:  {
5:    name = name.replaceAll("&", "&amp;");// &
6:    name = name.replaceAll("<", "&lt;"); // <
7:    name = name.replaceAll(">", "&gt;"); // >
8:    name = name.replaceAll("₩"", "&#34;"); // "
9:    name = name.replaceAll("₩'", "&#39;"); // '
10:   name = name.replaceAll("%00", null);   // null 문자
11:   name = name.replaceAll("%", "&#37;");  // %
12: }
13: else { return; }
14: %>
```

6. 파일 업로드

가. 취약점 설명

서버사이드에서 실행될 수 있는 스크립트 파일(asp, jsp, php 파일 등)이 업로드가 가능하고, 업로드 된 파일이 웹을 통해 실행될 수 있는 취약점

나. 보안대책

① 화이트리스트 방식으로 허용된 확장자만 업로드 허용
② 업로드 되는 파일을 저장할 때에는 파일명과 확장자를 외부시용자가 추측할 수 없는 문자열로 변경하여 저장
③ 저장 경로는 web document root' 밖에 위치시켜, 웹을 통한 직접 접근 차단

다. 코드예제

업로드할 파일에 대한 유효성을 검사하지 않아, 공격자에 의해 위험한 유형의 파일이 업로드가 가능한 형태의 코딩

안전하지 않은 코드의 예 JAVA

```
1:  .....
2:  public void upload(HttpServletRequest request) throws ServletException
3:  {
4:    MultipartHttpServletRequest mRequest = (MultipartHttpServletRequest) request;
5:    String next = (String) mRequest.getFileNames().next();
6:    MultipartFile file = mRequest.getFile(next);
7:
8:    // MultipartFile로부터 file을 얻음
9:    String fileName = file.getOriginalFilename();
10:
11:   // upload 파일에 대한 확장자 체크를 하지 않음
12:   File uploadDir = new File("/app/webapp/data/upload/notice");
13:   String uploadFilePath = uploadDir.getAbsolutePath()+"/" + fileName;
14:   /* 이하 file upload 루틴 */
```

미리 정의된 업로드 파일의 확장자만 허용하고 그 외 확장자는 업로드를 제한하고, 저장 시 외부에서 입력된 파일명을 그대로 저장되지 않도록 코딩

안전한 코드의 예 JAVA

```
1:  ……
2:  public void upload(HttpServletRequest request) throws ServletException
3:  {
4:    MultipartHttpServletRequest mRequest = (MultipartHttpServletRequest) request;
5:    String next = (String) mRequest.getFileNames().next();
6:    MultipartFile file = mRequest.getFile(next);
7:    if ( file == null ) return ;
8:    // 화이트리스트 방식으로 업로드 파일의 확장자를 체크한다.
9:    if ( fileName != null )
10:   {
11:     if ( fileName.endsWith(".doc") || fileName.endsWith(".hwp")  ||
       fileName.endsWith(".pdf") || fileName.endsWith(".xls") )
12:     {
13:       /* file 업로드 루틴 */
14:       // 저장 시 파일명을 외부 사용자가 추측할 수 없는 형태로 변경
15:       …
16:     }
17:   else throw new ServletExeption("에러");
18:   }
19: /* 이하 file upload 루틴 */
```

7. 파일 다운로드

가. 취약점 설명

외부 입력 값에 대해 경로 조작에 사용될 수 있는 문자를 필터링하지 않으면, 예상 밖의 접근 제한 영역에 대한 경로 문자열 구성이 가능해져 시스템 정보 등 중요정보가 누출이 되는 취약점

나. 보안대책

① 파일경로와 이름을 생성할 때 외부 입력 값을 사용하는 경우, 정해진 경로 이외의 디렉토리와 파일에 접근할 수 없도록 처리

② 외부 입력 값에 대해 replaceAll()등의 Method를 사용하여 예상 밖의 경로로의 접근을 허용하는 위험 문자열(",/,\,..)을 제거하는 필터링 적용

다. 코드예제

외부의 입력(name)이 삭제할 파일의 경로설정에 사용되고 있는 코드로, 공격자에 의해 name의 값으로 ../../..rootFile.txt와 같은 값을 전달하면 의도하지 않았던 파일이 삭제될 수 있는 코드

안전하지 않은 코드의 예 JAVA

```
1:  ……
2:  public void f(Properties request)
3:  {
4:    ……
5:    String name = request.getProperty("filename");
6:    if( name != null )
7:    {
8:      File file = new File("/usr/local/tmp/" + name);
9:      file.delete();
10:   }
11:  ……
12:  }
```

입력되는 값에 대하여 Null여부를 체크하고, 외부에서 입력되는 파일 이름(name)에서 상대경로(/,\\,&, . 등 특수문자)를 설정할 수 없도록 replaceAll을 이용하여 특수문자가 제거 될 수 있도록 코딩

안전한 코드의 예 JAVA

```
1: ……
2: public void f(Properties request)
3: {
4:   ……
5:   String name = request.getProperty("user");
6:  if ( name != null && !"".equals(name) )
7:   {
8:    name = name.replaceAll("&", "&amp;");// &
9:    name = name.replaceAll("<", "&lt;");// <
10:   name = name.replaceAll(">", "&gt;");// >
11:   name = name.replaceAll("₩"", "&#34;");// "
12:   name = name.replaceAll("₩'", "&#39;");// '
13:   name = name.replaceAll("%00", null);// null 문자
14:   name = name.replaceAll("%", "&#37;");// %
15:         name = name + "-report";
16:         File file = new File("/usr/local/tmp/" + name);
17.         if (file != null) file.delete();
18.  }
19. }
```

03 홈페이지 개발 보안 방안

8. 버퍼 오버플로우

가. 취약점 설명

정수형 변수의 오버플로우는 정수값이 증가하면서, Java에서 허용된 가장 큰 값보다 더 커져서 실제 저장되는 값은 의도하지 않게 아주 작은 수이거나 음수가 될 수 있는 취약점

나. 보안대책

① 언어/플래폼 별 정수타입의 범위를 확인하여 사용 정수형 변수를 연산에 사용하는 경우 결과 값의 범위 체크하는 모듈사용
② 외부 입력 값을 동적으로 할당하여 시용하는 경우 변수의 값 범위를 검사하여 적절한 범위 내 에 존재하는 값인지 확인

다. 코드예제

다음의 예제는 외부의 입력(args[0], args[1])을 이용하여 동적으로 계산한 값을 배열의 크기(size)를 결정하는데 사용되어, 외부 입력으로부터 계산된 값(size)이 오버플로우에 의해 음수값이 되면 배열의 크기가 음수가 되어 시스템 에 문제가 발생될 수 있는 코딩

안전하지 않은 코드의 예 JAVA

```
1:  ……
2:  public static void main(String[] args)
3:  {
4:    int size = new Integer(args[0]).intValue();
5:    size +=  new Integer(args[1]).intValue();
6:    MyClass[] data = new MyClass[size];
7:  ……
```

동적 메모리 할당을 위해 크기를 사용하는 경우 그 값이 음수가 아닌지 검사하여 버퍼오버플로우를 예방하는 형태로 코딩

안전한 코드의 예 JAVA

```
1:  ……
2:  public static void main(String[] args)
3:  {
4:    int size = new Integer(args[0]).intValue();
5:    size +=  new Integer(args[1]).intValue();
6:    // 배열의 크기 값이 음수값이 아닌지 검사한다.
7:    if (size < 0) return ;
8:    MyClass[ ] data = new MyClass[size];
9:  ……
```

9. LDAP 인젝션

가. 취약점 설명

외부 입력 값에 대해 특수문자(=, +, 〈, 〉, #, ;/등)를 필터링 하지 않으면 공격자에 의해 LDAP 명령어가 실행되는 취약점

나. 보안대책

① DN과 필터에 사용되는 사용자 입력 값에는 특수문자가 포함되지 않도록 특수문자 제거
② 특수문자를 사용해야 하는 경우 특수문자(DN에 사용되는 특수문자는 '\', 필터 에 사용되는 특수문자(=, +, 〈,〉,#, ; \ 등)에 대해서는 실행명령이 아닌 일반문자로 인식되도록 처리

다. 코드예제

name 변수의 값으로 "*"을 전달할 경우 필터 문자열은 "(name=*)"가 되어 항상 참이 되며 이는 의도하지 않은 동작을 유발시킬 수 있는 코딩

안전하지 않은 코드의 예 JAVA

```
1:  Properties props = new Properties();
2:  String fileName = "ldap.properties";
3:  FileInputStream in = new FileInputStream(fileName);
4:  props.load(in);
5:  String name = props.getProperty("name");
6:  String filter = "(name =" + name + ")";
7:  NamingEnumeration answer=ctx.search("ou=NewHires",filter,new
    SearchControls());
8:  printSearchEnumeration(answer);
9:  ctx.close();
```

검색 시 시용되는 필터 문자열(외부 입력 값)에 특수문자가 포함되어 있을 경우 공격 및 악의적인 목적으로 활용 가능하기 때문에 특수문자와 같은 위험한 문자열을 제거하도록 코딩

안전한 코드의 예 JAVA

```
1:  Properties props = new Properties();
2:  String fileName = "ldap.properties";
3:  FileInputStream in = new FileInputStream(fileName);
4:  if (in == null || in.available() <= 0) return;
5:  props.load(in);
6:  if (props == null || props.isEmpty()) return;
7:  String name = props.getProperty("name");
8:  if (name == null || "".equals(name)) return;
9:  String filter = "(name =" + name.replaceAll("₩₩*", "") + ")";
10: NamingEnumeration answer = ctx.search("ou=NewHires", filter, new SearchControls());
11: printSearchEnumeration(answer);
12: ctx.close();
```

03 홈페이지 개발 보안 방안

10. HTTP 응답분할

가. 취약점 설명

HTTP 요청에 들어 있는 인자값이 HTTP 응답헤더에 포함되어 사용자에게 다시 전달 될 때 입력값에 CR(Carriage Return)이나 LF(Line Feed)와 같은 개행문자가 존재하면 HTTP 응답이 2개 이상으로 분리되어, 공격자는 개행문자를 이용하여 첫 번째 응답을 종료시키고, 두번째 응답에 악의적인 코드를 주입하여 XSS 및 캐시 훼손(cache poisoning) 공격 등이 가능한 취약점

나. 보안대책

① 외부에서 입력된 인자값을 HTTP 응답헤더(Set Cookie 등)에 포함시킬 경우 CR, LF 등의 개행문자를 제거

다. 코드예제

공격자가 "Wiley Hacker\r\nHTTP/1.1 200 OK\r\n"를 authorName의 값으로 설정할 경우, 예와 같이 의도하지 않은 두개의 페이지가 전달되며, 두번째 응답페이지는 공격자가 값을 수정하여 공격이 가능한 코딩

안전하지 않은 코드의 예 JAVA

```
1:  throws IOException, ServletException
2:  {
3:    response.setContentType("text/html");
4:    String author = request.getParameter("authorName");
5:    Cookie cookie = new Cookie("replidedAuthor", author);
6:    cookie.setMaxAge(1000);
7:    response.addCookie(cookie);
8:    RequestDispatcher frd = request.getRequestDispatcher("cookieTest.jsp");
9:        frd.forward(request, response);
10: }
```

입력되는 값에 대하여 Null여부를 체크하고, 헤더값이 두 개로 나누어지는 것을 방지하기 위해 replaceAll을 이용하여 개행문자(\r, \n)를 제거하도록

안전한 코드의 예 JAVA

```
1:  throws IOException, ServletException
2:  {
3:    response.setContentType("text/html");
4:    String author = request.getParameter("authorName");
5:    if (author == null || "".equals(author)) return;
6:    String filtered_author = author.replaceAll("₩r", "").replaceAll("₩n", "");
7:    Cookie cookie = new Cookie("replidedAuthor", filtered_author);
8:    cookie.setMaxAge(1000);
9:    cookie.setSecure(true);
10:   response.addCookie(cookie);
11:   RequestDispatcher frd = request.getRequestDispatcher("cookieTest.jsp");
12:   frd.forward(request, response);
13: }
```

11. URL/파라미터 변조

가. 취약점 설명

실행경로에 대해서 접근제어를 검사하지 않거나 불완전하게 구현하여 공격자로 하여금 값을 변조하여 중요정보에 접근 가능해지는 취약점

나. 보안대책

① 중요한 정보가 있는 페이지(계좌이체 등)는 재 인증 적용
② 안전하다고 확인된 라이브러리나 프레임워크(OpenSSL이나 ESAPI의 보안 기능 등)를 사용
③ 응용프로그램이 제공하는 정보와 기능을 역할에 따라 배분함으로써 공격자에게 노출되는 공격노출면(attack surface) 최소화
④ 사용자의 권한에 따른 ACL(Access Control List) 관리

다. 코드예제

계좌이체 시 사용자 인증을 위한 별도의 접근제어 방법이 사용되지 않고 있으며, 이는 임의 사용자의 정보를 외부에서 접근할 수 있는 코딩

안전하지 않은 코드의 예 JAVA

```
1:  public void sendBankAccount(String accountNumber,double balance)
2:  {
3:      BankAccount account = new BankAccount();
4:      account.setAccountNumber(accountNumber);
5:      account.setToPerson(toPerson);
6:      account.setBalance(balance);
7:      AccountManager.send(account);
8:  }
```

계좌이체 시 username, password를 입력 받아 세션으로부터 credential을 읽어들여 사용자 재인증 수행 거치도록 코딩

안전한 코드의 예 JAVA

```
1:  public void sendBankAccount(HttpServletRequest request, HttpSession session, String
    accountNumber,double balance)
2:  {
3:      String newUserName = request.getParameter("username");
4:      String newPassword = request.getParameter("password");
5:      if ( newUserName == null || newPassword == null )
6:      {
7:          throw new MyEception("데이터 오류:);
8:      }
9:      String password = session.getValue("password");
10:     String userName = session.getValue("username");
11:     if (isAuthenticatedUser() && newUserName.equal(userName) &&
        newPassword.equal(password) )
12:     ...
13: }
```

12. 취약한 계정 생성 허용

가. 취약점 설명

회원가입 시 안전한 패스워드규칙이 적용되지 않아, 무차별 대입법 공격 등에 의해 공격자에게 패스워드가 노출될 수 있는 취약점

나. 보안대책

사용자가 취약한 패스워드를 사용할 수 없도록 패스워드 생성규칙을 강제 할 수 있는 로직 적용

〈참고〉 패스워드 생성 보안로직 권고사항

분 류	내 용
패스워드 생성규칙	• 세가지 종류 이상의 문자구성으로 8자리 이상의 길이 • 두가지 종류 이상의 문자구성으로 10자리 이상의 길이
패스워드 생성 금지규칙	• 간단한 문자(영어단어 포함)나 숫자의 연속사용은 금지 • 키보드 상에서 일련화 된 배열을 따르는 패스워드 선택 금지 • 사전에 있는 단어, 이를 거꾸로 철자화한 단어 사용 금지 • 생일, 전화번호, 개인정보 및 아이디와 비슷한 추측하기 쉬운 비밀번호 사용 금지 • 이전에 사용한 패스워드는 재사용 금지 • 계정 잠금 정책 설정 ex)로그인 5회 실패 시 30분 동안 사용중지

다. 코드예제

회원가입 시 사용자가 입력한 패스워드를 확인하는 함수에서 단순히 패스워드 길이만을 확인하는 형태의 코딩

안전하지 않은 코드의 예 JAVA

```
1:  public boolean pwchk(String pw)
2:  {
3:    int pwlen = strlen(pw);
4:    if( pwlen > 6 ) return true;
5:    else return false;
6:  }
```

자주 사용하거나 유추하기 쉬운 패스워드를 별도의 파일로 관리하여 패스워드 생성 시 파일(취약한 패스워드 리스트)과 비교한 뒤 해당사항이 없을 경우에만 패스워드가 생성될 수 있도록 코딩

안전하지 않은 코드의 예 JAVA

```
1:  throws IOException, ServletException
2:  {
3:    response.setContentType("text/html");
4:    String author = request.getParameter("authorName");
5:    Cookie cookie = new Cookie("replidedAuthor", author);
6:    cookie.setMaxAge(1000);
7:    response.addCookie(cookie);
8:    RequestDispatcher frd = request.getRequestDispatcher("cookieTest.jsp");
9:        frd.forward(request, response);
10: }
```

안전한 코드의 예 JAVA

```
1:  public boolean pwchk(String pw)
2:  {
3:    BufferedReader reader = new BufferedReader(new FileReader("weakpwList"));
4:    String weakpw = null;
5:    while ((weakpw = reader.readLine()) != null) {
6:      if( weakpw.equalsIgnoreCase(pw) == true ){
7:        return false;
8:      }
9:    }
10:   reader.close();
11:   int pwlen = strlen(pw);
12:   if( pwlen > 8 ) return true;
13:   else return false;
14: }
```

13. 불충분한 세션관리

가. 취약점 설명

인증 시 마다 동일한 세션 ID가 발급되거나 세션 타임아웃을 너무 길게 설정하여 공격자가 세션을 재사용 할 수 있는 취약점

나. 보안대책

① 세션 ID는 로그인 시 마다 추측할 수 없는 새로운 세션 ID로 발급
② 세션 타임아웃 설정을 통해 일정시간(최대 30분 이상) 동안 움직임이 없을 경우 자동 로그아웃 되도록 구현

다. 코드예제

세션 유지 시간을 10000분으로 설정하여 세션 타임아웃 시간 설정이 의미가 없게 코딩되어 있으며, 중복 로그인을 허용하는 형태의 코딩

안전하지 않은 코드의 예 JAVA

```
1:  public void setKeepTime(){
2:    // Session의 유지 시간을 Setting
3:    String strTime = Param.getPropertyFromXML("SessionPersistenceTime");
4:    if(strTime == null) {
5:      session.setMaxInactiveInterval(60*10000);
6:    } else {
7:      session.setMaxInactiveInterval(new Integer(strTime)).intValue());
8:    }
9:  }
10: ... 중략
```

세션 유지 시간을 20분이내로 설정하고, 중복 로그인을 허용하지 않도록 코딩

안전한 코드의 예 JAVA

```
1:  public void setKeepTime(){
2:    // Session의 유지 시간을 Setting
3:    String strTime = Param.getPropertyFromXML("SessionPersistenceTime");
4:    if(strTime == null) {
5:      session.setMaxInactiveInterval(60*20);
6:    } else {
7:      session.setMaxInactiveInterval(new Integer(strTime)).intValue());
8:    }
9:  }
10:  public boolean isUsing(String userID)
11:  {
12:    boolean isUsing = false;
13:    Enumeration e = loginUsers.keys();
14:    String key = "";
15:    while(e.hasMoreElements()) {
16:      key = (String)e.nextElement();
17:      if(userID.equals(loginUsers.get(key))) {
18:        isUsing = true;
19:      }
20:    }
21:    return isUsing;
22:  ... 중략
```

14. 데이터 평문전송

가. 취약점 설명

중요정보와 관련된 민감한 데이터 (개인정보, 비밀번호 등)를 평문으로 송수신 할 경우, 통신채널 스니핑을 통해 인가되지 않은 사용자에게 민감한 데이터가 노출될 수 있는 취약점

나. 보안대책

① 중요정보와 관련된 민감한 데이터(개인정보, 비밀번호 등) 전송 시 통신채널(또는 전송데이터) 암호화적용

다. 코드예제

패스워드가 암호화 없이 네트워크를 통하여 서버로 전송되도록 코딩되어 공격자로 부터 패킷 스니핑시 노출 될 수 있는 취약한 코딩

안전하지 않은 코드의 예 JAVA

```
1:  void foo()
2:  {
3:    try
4:    {
5:    Socket socket = new Socket("taranis", 4444);
6:    PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
7:    String password = getPassword();
8:    out.write(password);
9:    }
10:   catch (FileNotFoundException e)
11:   {
12:     ...
```

패스워드를 네트워크를 통하여 서버에 전송하기 전에 암호화를 수행하도록 코딩

안전하지 않은 코드의 예 JAVA

```
1:  void foo()
2:  {
3:    try
4:    {
5:      Socket socket = new Socket("taranis", 4444);
6:      PrintStream out = new PrintStream(socket.getOutputStream(), true);
7:      Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding");
8:      String password = getPassword();
9:      encryptedStr= c.update(password.getBytes());
10:     out.write(encryptedStr,0,encryptedStr.length);
11:   }
12:   catch (FileNotFoundException e) {
```

15. 쿠키변조

가. 취약점 설명

웹 서비스에서 사용자 인증 등 중요기능 구현 시 쿠키를 활용할 경우 공격자의 패킷스니핑을 통해 해당 쿠키가 탈취되어 타 사용자로 로그인이 가능해지는 취약점

나. 보안대책

① 사용자 인증 등 중요기능 구현 시 가급적이면 Cookie 대신 Session 방식 사용
② 사용자 인증 등 중요기능 구현 시 Cookie(또는 Session) 방식 활용 시 안전한 알고리즘(SEED, 3DES, AES 등)을 사용

다. 코드예제

쿠키를 생성하여 사용하고 있지만 암호화 하지 않고 평문으로 값을 사용하여 공격자가 해당 값을 조작하여 권한 상승이 가능한 형태의 코딩

안전하지 않은 코드의 예 JAVA

```
1:  public void createCookie(){
2:    Cookie cookie = new Cookie("cookie_key", "1234");
3:    cookie.setMaxAge(60*60*24);
4:    cookie.setPath("/");
5:    response.addCookie(cookie);
6:  }
```

아래의 코드는 쿠키를 생성 시 value 값을 받아들여 getEncrypt(AES 암호화) 함수를 통해 AES 암호화 후 쿠키를 생성하게 하여 쿠키 변조 공격에 안전하도록 코딩

안전한 코드의 예 JAVA

```
1:  public void createCookie(){
2:    String value = "1234";
3:    value = getEncrypt(value);
4:    Cookie cookie = new Cookie("cookie_key", value);
5:    cookie.setMaxAge(60*60*24);
6:    cookie.setPath("/");
7:    response.addCookie(cookie);
8:  }
```

16. 취약한 암호화 알고리즘 사용

가. 취약점 설명

중요정보를 암호화하기 위해 사용되는 방법이 인코딩(또는 암호화)을 하는데 취약한 인 코딩방법(예: base64) 및 암호화 알고리즘(예:RC2, RC4, RC5, MD4, MD5, SHA1, DES 등)을 사용할 경우, 공격자가 해독이 가능하여 중요정보가 노출될 수 있는 취약점

나. 보안대책

① 개인정보 등 중요정보를 보호하기 위해 사용하는 암호알고리즘[참고] 적용시, IT보안인증 사무국이 안전성을 확인한 검증필 암호모듈 사용

분 류		내 용
최소 안전성 수준		112 비트
블록암호		ARIA(키 길이 : 128/192/256),
블록암호 운영모드	기밀성	SEED(키 길이 : 128)
	기밀성/인증	ECD, CBC, CFB, OFB, CTR
해쉬함수		CCM, GCM
메시지 인증코드	해쉬함수기반	SHA-224/256/384/512
	블록기반	HMAC
난수발생기	해쉬함수 / HMAC 기반	CMAC, GMAC
		HASH_DRBG, HMAC_DRBG
	블록기반	CTR_DRBG
공개키 암호		- RSAES (공개키 길이) 2048, 3072 - RSA-OAEP에서 사용되는 해쉬함수 : SHA-224/256
기밀성		RSA-PSS, KCDSA, ECDSA, EC-KCDSA
기밀성/인증		DH, ECDH

〈참고〉 패스워드 생성 보안로직 권고사항

보호함수		보호함수 파라미터
시스템 파라미터	RSA-PSS	(공개키 길이) 2048, 3072
	KCDSA, DH	(공개키 길이, 개인키 길이) (2048, 224), (2048, 256)
	ECDSA, EC-KCDSA, ECDH	(FIPS) B-233, B-283 (FIPS) K-233, K-283 (FIPS) P-224, P-256

※ 출처 : 암호알고리즘 검증기준 Ver 2.0(2012.3), 암호모듈시험기관(IT보안인증사무국)

다. 코드예제

메세지(msg)를 취약한 DES 알고리즘으로 암호화 진행

안전하지 않은 코드의 예 JAVA

```
1:  try
2:  {
3:    Cipher c = Cipher.getInstance("DES");
4:    c.init(Cipher.ENCRYPT_MODE, k);
5:    rslt = c.update(msg);
6:  }
7:  catch (InvalidKeyException e) {
```

안전하다고 알려진 암호화 알고리즘(SEED)를 사용하여 암호화 진행

안전한 코드의 예 JAVA

```
1:  try
2:  {
3:    Cipher c = Cipher.getInstance("SEED/CBC/PKCS5Padding");
4:    c.init(Cipher.ENCRYPT_MODE, k);
5:    rslt = c.update(msg);
6:  }
7:  catch (InvalidKeyException e) {
```

17. 취약한 패스워드 복구

가. 취약점 설명

패스워드 복구 메커니즘(아이디/비밀번호 찾기 등)이 취약한 경우 불법적으로 다른 사용자의 패스워드를 획득, 변경, 복구 되는 취약점

나. 보안대책

① 사용자를 식별하기 위한 수단 활용 시 그 사용자의 유일한 값 사용
② 사용자 본인인증 메커니즘 구현 시 추측이 불가하게 구현
③ 임시 패스워드 발급 시 안전한 난수 값 사용

다. 코드예제

사용자 본인확인 인증 시 아이디로만 인증하는 형태의 코딩

안전하지 않은 코드의 예 JAVA

```
1:  try
2:  {
3:  String tableName = props.getProperty("jdbc.tableName");
4:  String userid = props.getProperty("jdbc.id");
5:  String query = "SELECT  * FROM  ? WHERE Name = ? ";
6:  stmt = con.prepareStatement(query);
7:  stmt.setString(1, tableName);
8:  stmt.setString(2, userid);
9:  rs = stmt.executeQuery();
10: ... ... 11:  }
12: catch (SQLException sqle) {  }
13: finally {  }
```

사용자 본인확인 인증 시 아이디와 이메일을 활용하여 검증

안전한 코드의 예 JAVA

```
1:  try
2:  {
3:  String tableName = props.getProperty("jdbc.tableName");
4:  String userid = props.getProperty("jdbc.id");
5:  String useremail = props.getProperty("jdbc.email");
6:  String query = "SELECT  * FROM  ? WHERE Name = ? AND Email = ? ";
7:  stmt = con.prepareStatement(query);
8:  stmt.setString(1, tableName);
9:  stmt.setString(2, userid);
10: stmt.setString(3, useremail);
11: rs = stmt.executeQuery();
12: ... ...
13: }
14: catch (SQLException sqle) {  }
15: finally {  }
```

18. 주석을 통한 정보 노출

가. 취약점 설명

소스코드 주석문에 민감한 정보(개인정보, 시스템 정보 등)이 포함되어 있는 경우, 외부 공격자에 의해 패스워드 등 보안 관련정보가 노출될 수 있는 취약점

나. 보안대책

① 디버깅 목적으로 주석 ID, 패스워드, 시스템 관련정보 등 보안관련 정보는 개발완료 후 제거 필요

다. 코드예제

디버깅 등의 목적으로 주석문에 관리자아이디, 패스워드 기술한 형태의 코딩

안전하지 않은 코드의 예 JAVA

```
1:  ……
2:  // ID : tiger, Password : "tiger."
3:   public boolean DBConnect()
4:   {
5:   String url = "DBServer";
6:   String password = "tiger";
7:   Connection con = null;
8:
9:   try
10:  {
11:       con = DriverManager.getConnection(url, "scott", password);
12:    }
13:    catch (SQLException e)
14:    }
15:          ……
```

프로그램 개발 시 주석문 등에 남겨놓은 사용자 계정이나 패스워드 등의 정보는 개발 완료 후 삭제조치 하도록 코딩

안전한 코드의 예 JAVA

```
1:  ……
2:  //디버깅 등의 용도로 소스 주석에 적어놓은 패스워드 삭제 조치 필요
3:   public Connection DBConnect(String id, String password)
4:   {
5:   String url = "DBServer";
6:   Connection conn = null;
7:   try
8:     {
9:   String CONNECT_STRING = url + ":" + id + ":" + password;
10:    InitialContext ctx = new InitialContext();
11:   DataSource datasource = (DataSource) ctx.lookup(CONNECT_STRING);
12:   conn = datasource.getConnection();
13:    }
14:  catch (SQLException e)  { ……
15:       return conn;
16:  }
```

웹 서버 및 홈페이지 취약점 점검 가이드

부 록

1. 원격 자가점검 시스템 사용 매뉴얼

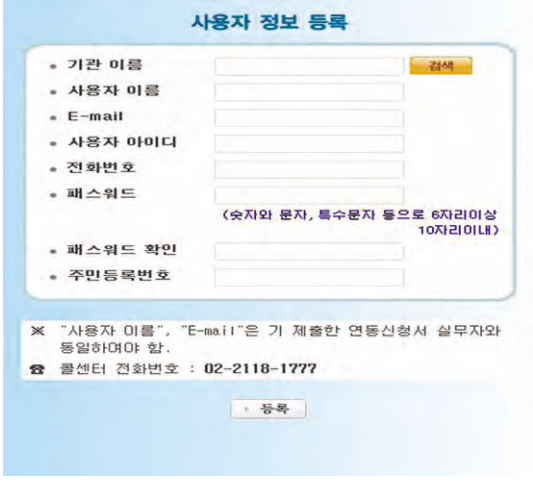
2 로그인

⦿ 홈페이지 주소 : http://cyber.ecsc.go.kr

⦿ 사이버 안전 지원 시스템 접속

[사용자등록] 선택

[사용자 정보 등록 화면] 관련정보 입력



※ 기관에서 제출하신 연동신청서의 사용자 정보를 입력

아이디는 임의 부여된 ID가 생성되어 있으나 변경 가능
[기관이름]은 검색버튼 선택 후 기관을 검색
[사용자이름],[E_mail],[전화번호]는 연동신청서와 동일하게 입력

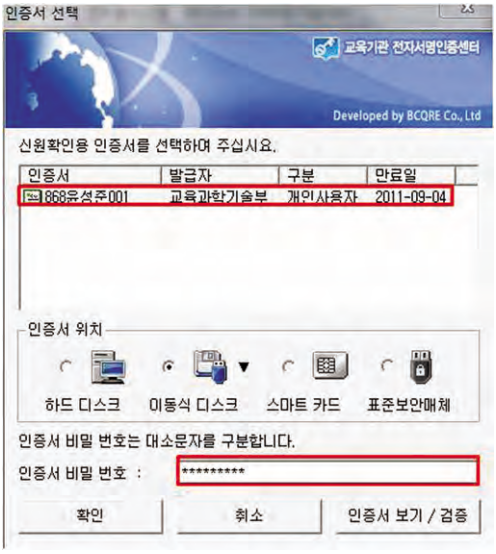
※ 주민등록번호 입력 후 등록버튼 선택
주민등록번호는 개인용인증서 등록 시 확인용도로 사용하고, 저장 및 보관하지 않음

※ 연동신청서를 공문으로 제출한 후 담당자가 변경되었을 경우, [붙임]자료(6페이지)를 교육사이버 안전 지원 시스템 담당자에게 제출 시 가입 가능

⦿ 인증서 선택

[사용자등록]버튼 클릭할 경우 후, 인증서 화면 확인 가능
사용자는 교육과학기술부에서 발급한 개인용 인증서를 선택

※ 인증서 발급관련 문의는 각 기관의 LRA담당자나 전사서명인증센터 (www.epki.go.kr, 02-2118-1755)로 문의



[사용자 등록]버튼을 클릭한 후 다음과 같은 인증서 선택 화면 생성
교육과학기술부에서 발급한 개인용 인증서 선택

※ 인증서 발급관련 문의는 각 기관LRA 담당자나 전자서명인증센터 (www.epki.go.kr, 02-2118-1755)로 문의

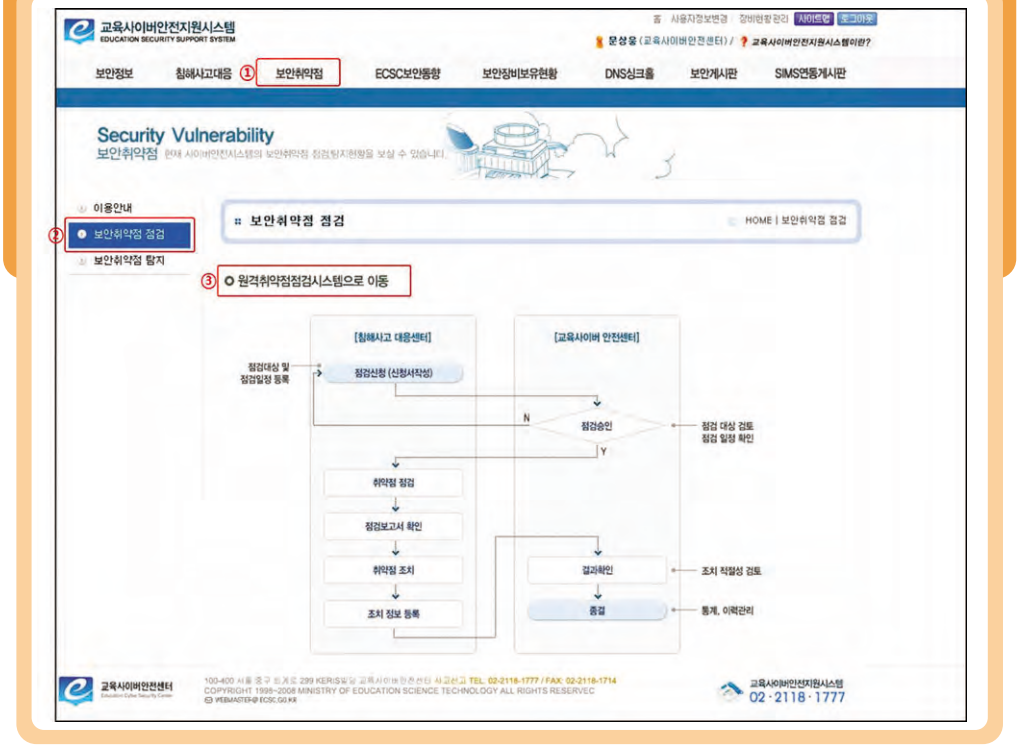
⊙ 인증서 미등록 사용자 등록

발급한 아이디와 패스워드를 입력하고 개인용인증서 확인을 위해 주민번호를 입력하면 인증서 등록화면이 활성화됨

1. 회원가입

⦿ 접속 방법

원격 보안 취약점 자가 점검 시스템 접속
[교육사이버안전지원시스템(cyber.ecsc.go.kr)] 〉 [보안취약점] 〉 [보안취약점 점검] 〉 [원격보안 취약점 자가 점검 신청]



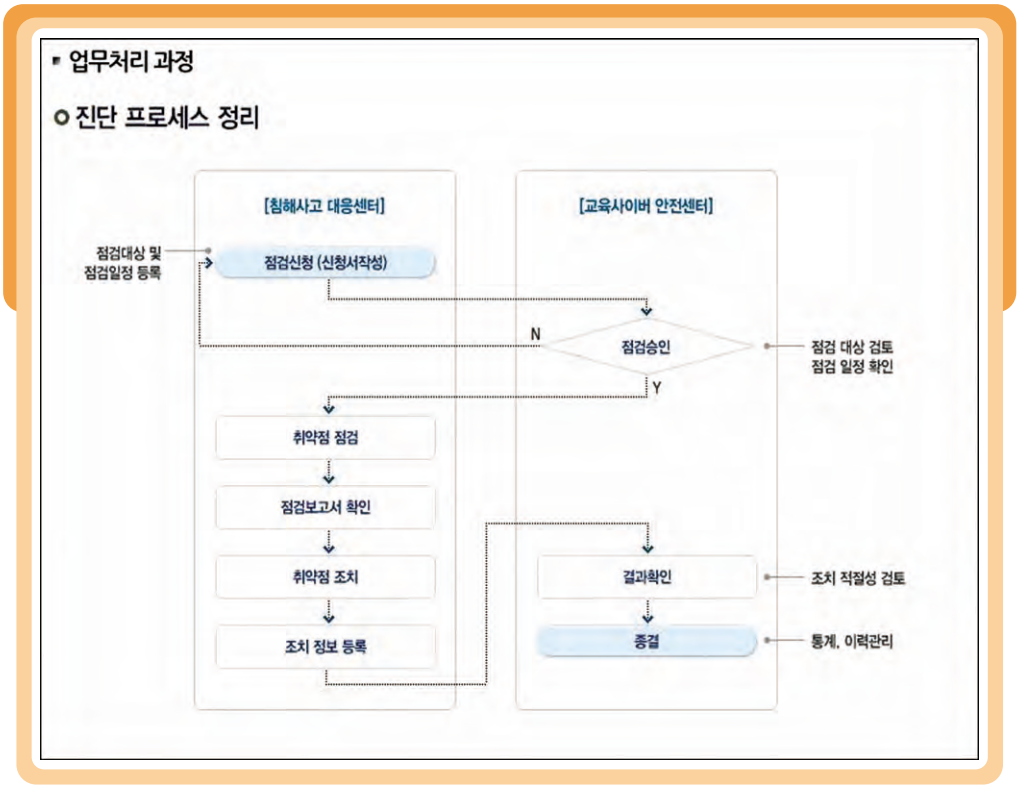
[그림 1] 취약점 점검 신청

3. 업무처리 과정

⦿ 보안 취약점 점검 절차

– 취약점 점검은 위와 같은 과정으로 진행됨
[대상등록] 〉 [점검신청] 〉 [점검승인] 〉 [점검] 〉 [결과확인 및 조치] 〉 [조치 정보 등록] 〉 [종결]

– 절차별 세부 내용
- 대상 등록 : 점검 대상 사이트 및 담당자 정보 입력
- 점검 신청 : 등록한 사이트에 대해 점검 신청
- 점검 : 취약점을 점검하는 과정으로 신청 일정에 따라 자동 점검
- 조치정보 등록 : 발견된 취약점에 대해 조치결과 등록



[그림 2] 취약점 점검 절차

4. 취약점 점검 대상 등록

⦿ 취약점 점검에 앞서 점검 대상 시스템의 정보를 등록하며, 다음과 같이 진행함

- 취약점 점검이 필요한 "점검대상" 사이트를 등록
  [취약점 점검] 〉 [점검 대상 등록 및 결과] 〉 [등록하기] 선택

⊙ 담당자 및 대상 사이트 정보 입력

– 다음과 같은 항목에 대상 사이트 및 담당자 정보를 입력 한다.

① [서비스] 선택(또는 임의 서비스 이름 생성)
예) 업무포털, 대표홈페이지, 웹 메일 등

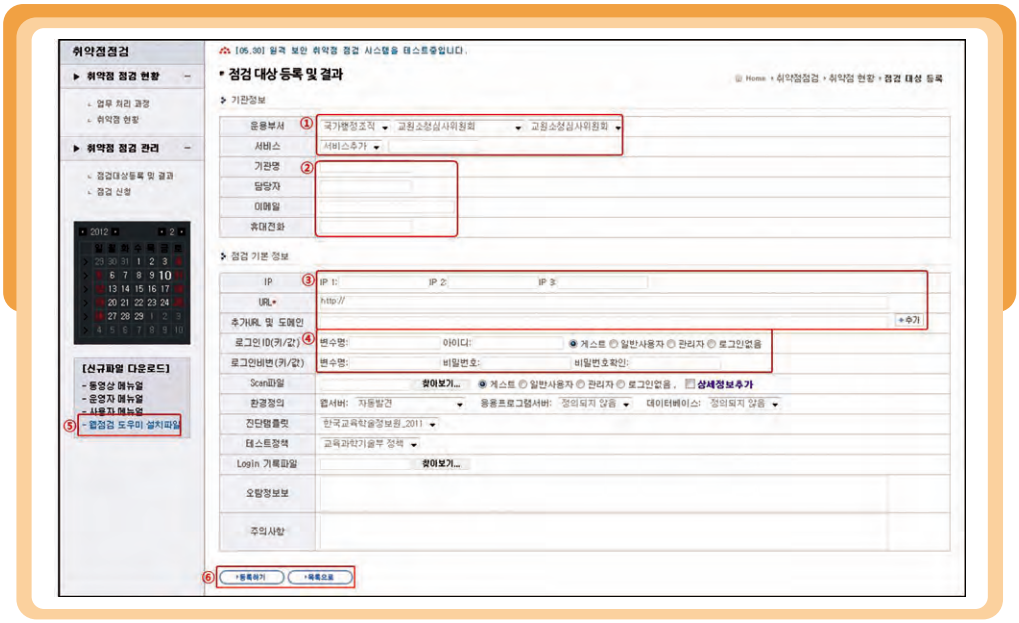
② [기관명], [담당자], [이메일], 연락처 입력

③ [IP], [URL]란에 점검 대상서버의 IP와 URL을 입력

④ [로그인ID, 비밀번호 변수 명] 웹점검 도우미 이용 입력

⑤ [웹점검 도우미 설치] 파일을 다운로드 후 설치하여, 설치 폴더의 Keris_finder.exe 파일을 클릭하여 실행 (p.5 참조)

⑥ 정보를 입력한 후 [등록하기] 선택



[그림 4] 점검대상 정보 등록

⦿ 웹 점검 도우미는 사용자 인증과정 중에 필요한 매개변수 정보를 검증 하는 도구로, 로그인이 필요한 홈페이지를 점검할 경우 활용함

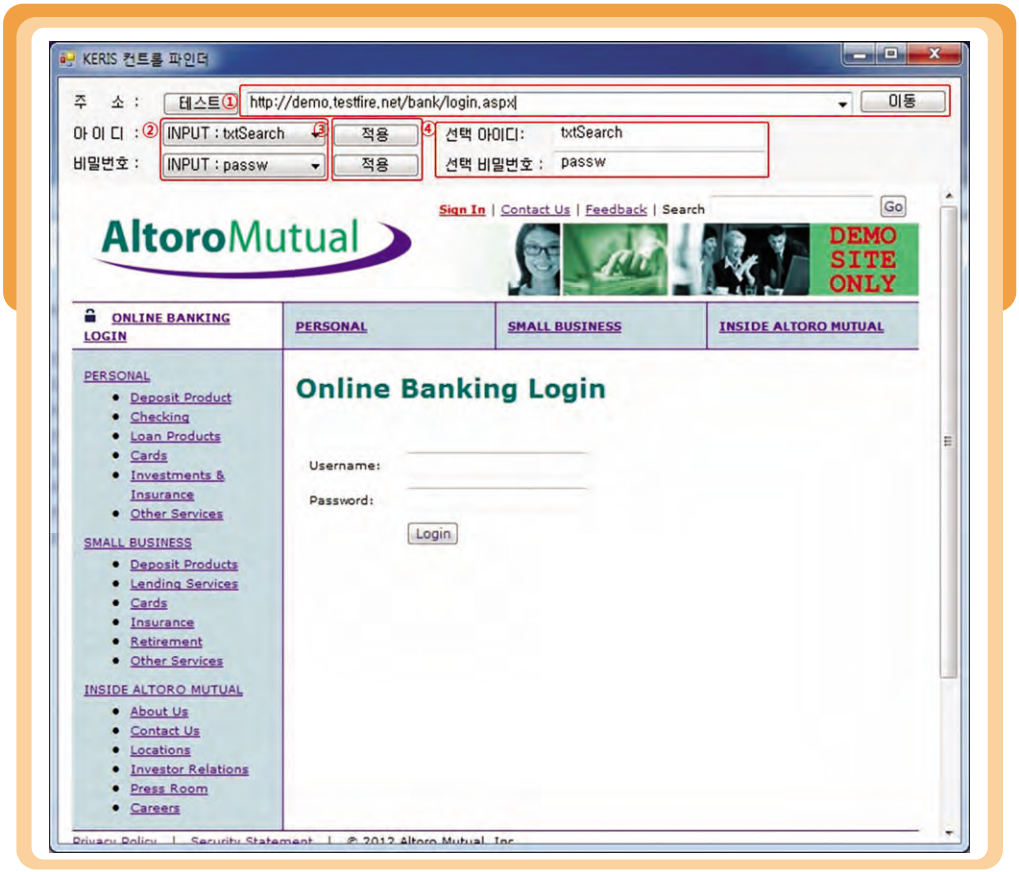
– 웹 점검 도우미 사용 방법

① 점검 대상의 로그인 페이지 URL을 입력 후 이동 버튼 클릭

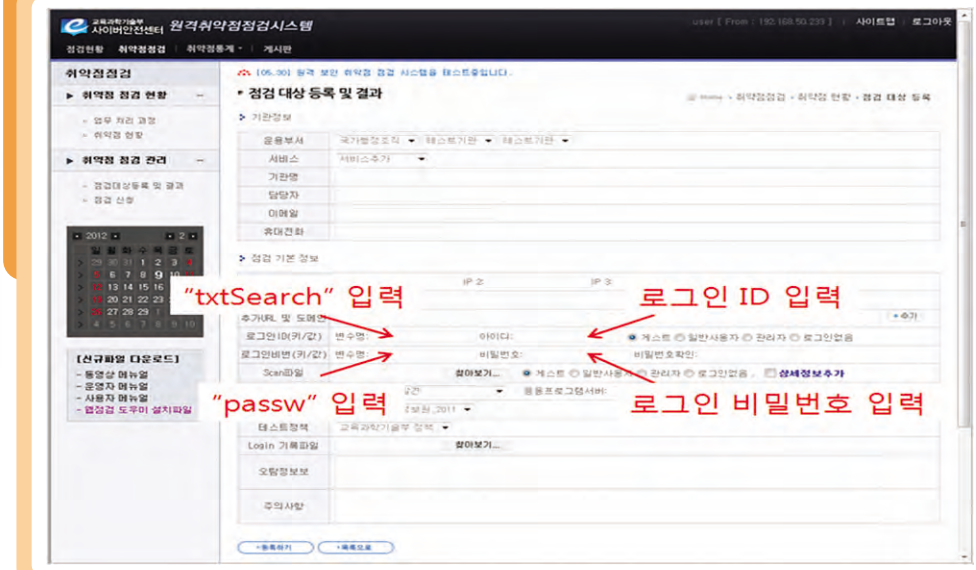
② 아이디, 비밀번호 스크롤을 클릭하여 INPUT 아이디가 ID,PW와 같은 컨트롤 명을 찾아 선택

③ 적용 버튼 클릭

④ 선택 된 아이디 및 비밀번호를 복사하여, 점검대상 등록 페이지의 변수명에 각각 입력

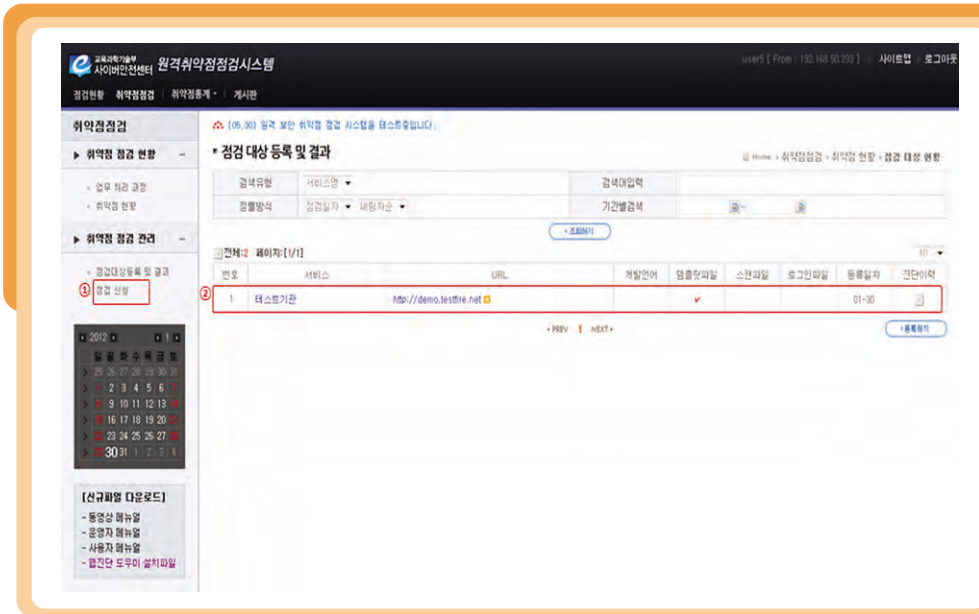


[그림 5] 웹 점검 도우미 화면



⊙ 등록 정보 확인

– 정상적으로 등록되었는지 확인한다.

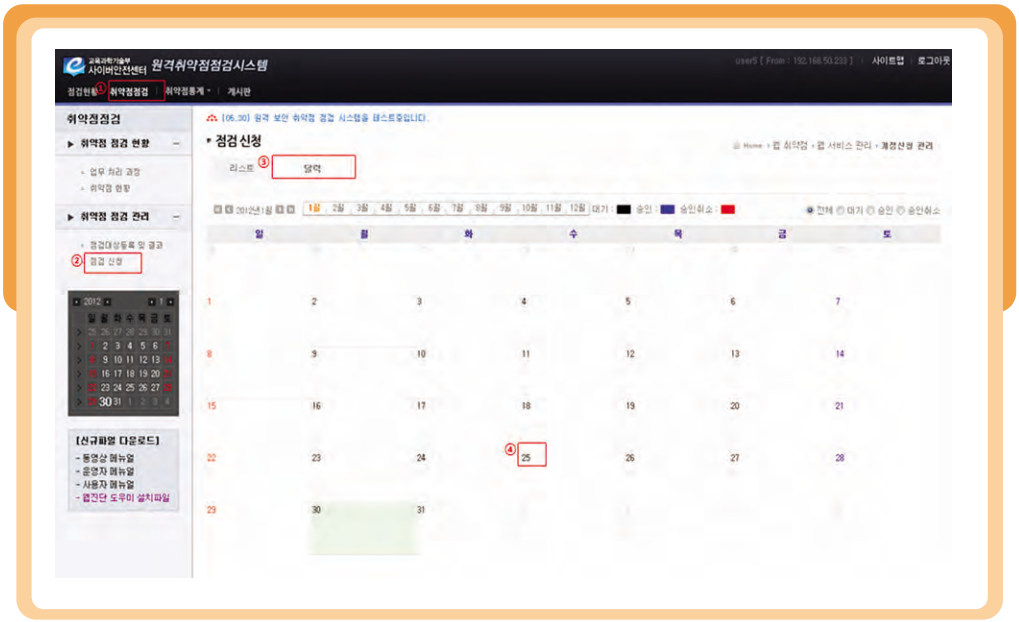


[그림 6] 점검대상 등록결과 확인

5. 취약점 점검 신청 및 승인 확인

⊙ 정보 등록을 완료하면, 등록된 정보를 점검 신청하며, 다음과 같이 수행함

- 진단대상 점검신청을 등록
  [취약점 점검] 〉 [점검 신청] 〉 [달력]탭 선택
- 날짜 선택 시 [서비스] 선택, 점검 시간 선택 후 [신청]

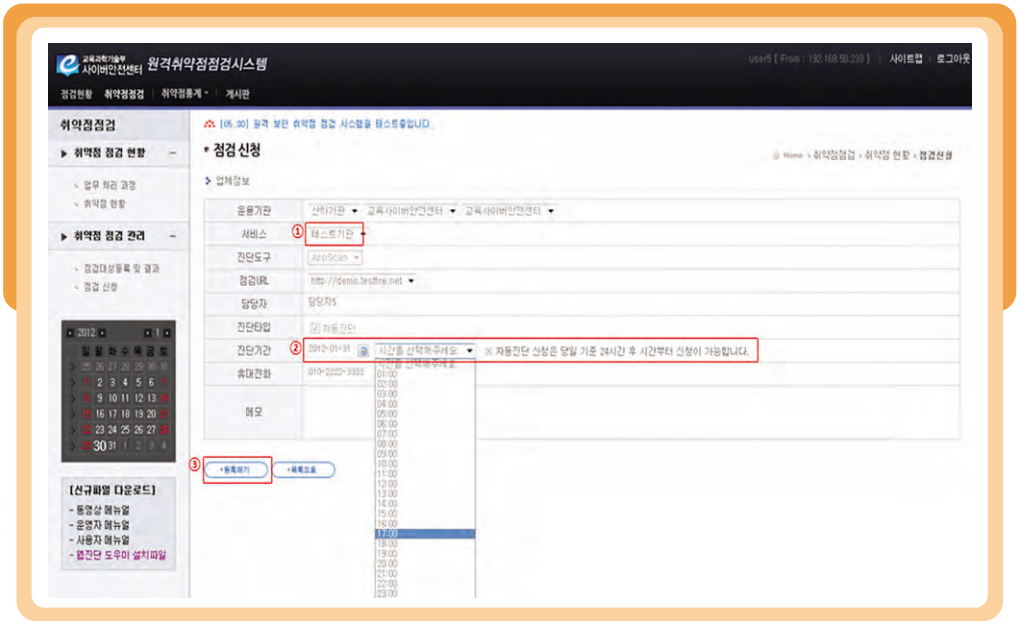


[그림 7] 점검 희망일 선택

※ 점검은 희망하는 날로부터 24시간 이전부터 신청 가능

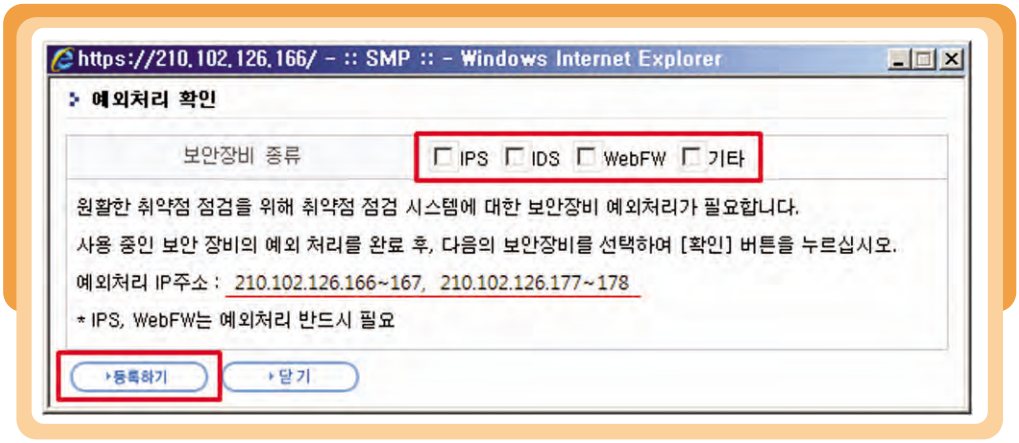
⊙ 점검 신청 정보 입력

① [서비스] 선택
② [진단기간] 점검요청시간 선택
③ 점검 신청 정보 확인 후 [등록하기]



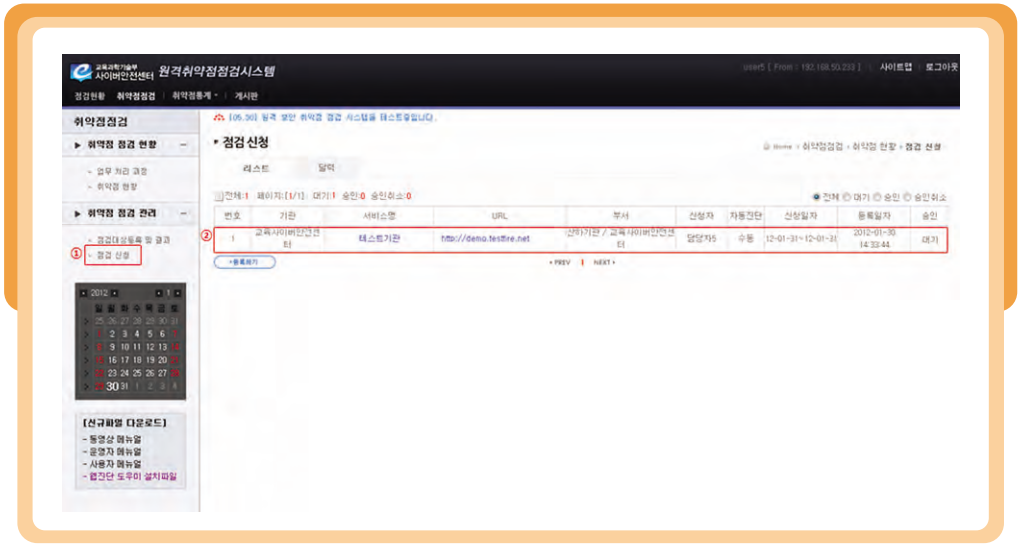
– 예외처리 확인
원격에서 점검이 이루어지므로 각종 보안 장비에 아래의 IP주소를 예외처리(또는 화이트리스트 등록)가 반드시 선행되어야함

※ 예외 IP : 210.102.126.166~167, 210.102.126.177~177 / 4개 IP

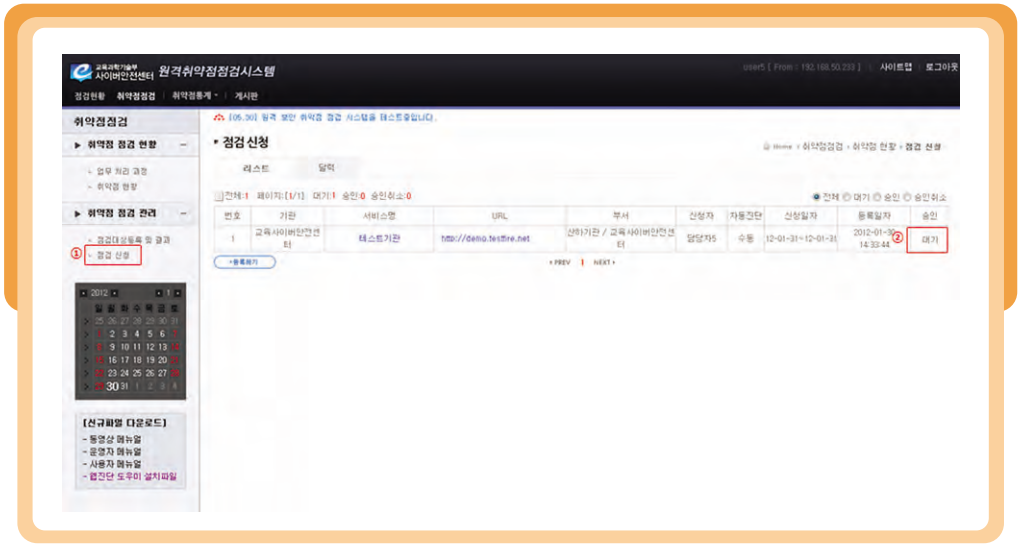


⊙ 점검 신청 완료 및 승인

– 신청을 완료하면 ECSC 업무 담당자의 승인 후 점검 실시됨
승인 과정은 매시간 ECSC 담당자 승인하며, 승인 과정에서 점검 대상 사이트의 적절성을 확인함



[그림 9] 신청 완료



[그림 10] 신청 승인 확인

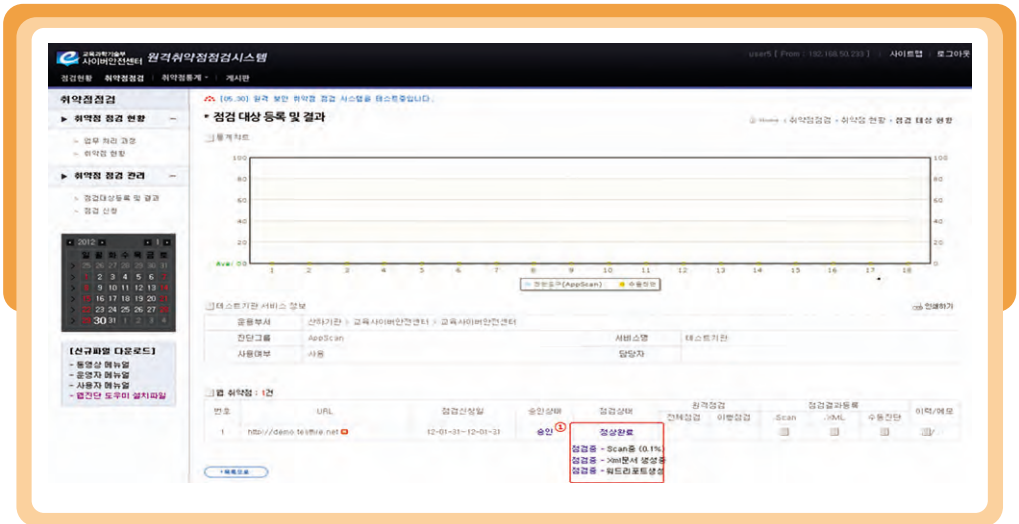
6. 취약점 점검

⊙ 신청 승인이 완료되면 취약점 점검은 신청한 날짜의 시간에 자동으로 시작하며 진행 상태는 다음과 같이 확인이 가능함

– 취약점 점검 상태는 [취약점점검] 〉 [점검대상등록 및 결과] 〉 등록한 "서비스" 이름을 선택하여 확인 가능

[그림 11] 취약점 점검 상태 조회

– 아래와 같이 4가지 점검 상태를 확인할 수 있음
- 점검중 – Scan : 점검 대상 기관의 취약점 진단 중(1~6시간 소요)
- 점검중 – XML : 진단 완료 후 XML 파일 생성(10~20분 소요)
- 점검중 – 워드리포트생성 : DOC 확장자의 보고서 생성(20분 소요)
- 점검완료 : 점검이 최종 완료됨

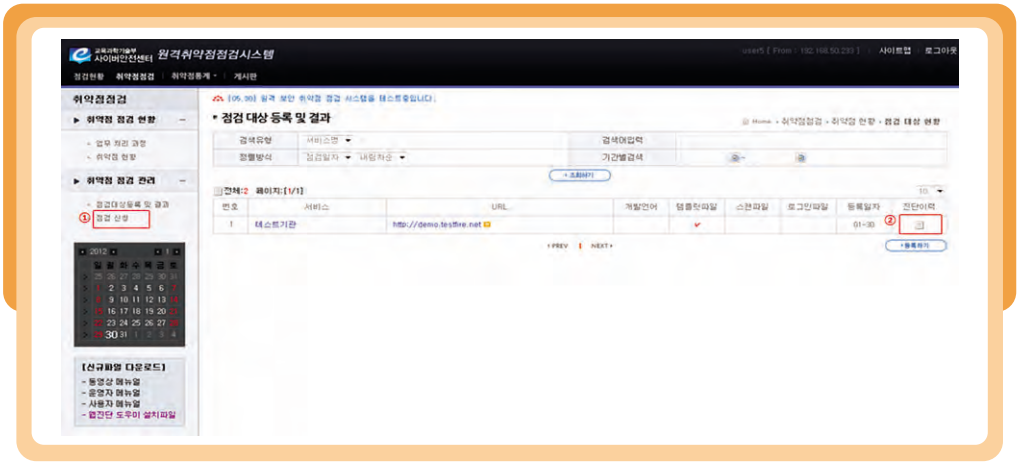


[그림 12] 취약점 점검 상태 확인

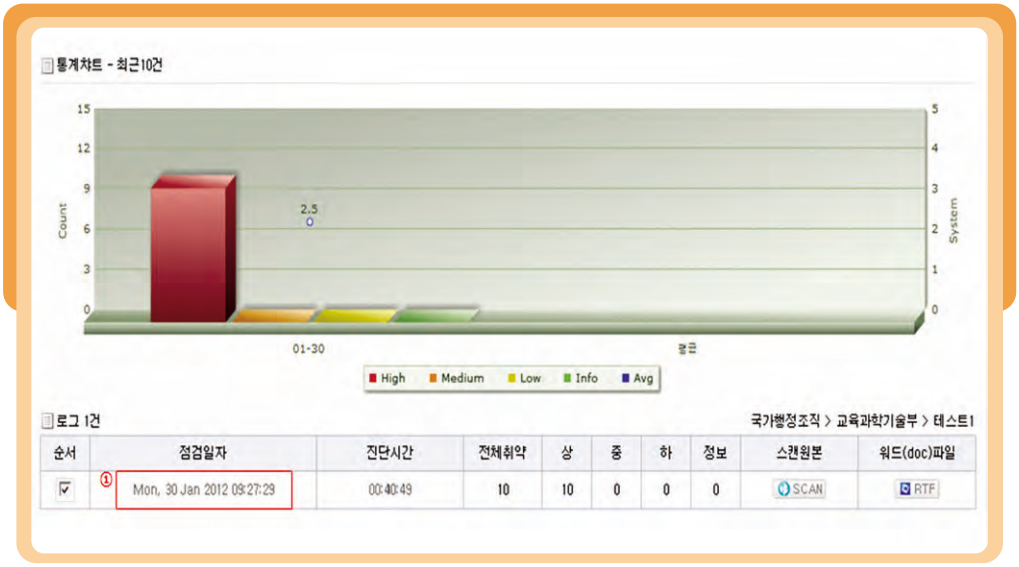
7. 점검 결과 확인 및 조치 등록

⦿ 점검이 완료 되면 점검결과를 확인하여 취약점에대한 조치 및 미조치 사유를 등록 관리할 수 있으며 다음과 같이 확인이 가능함

- 진단 이력을 확인하여 점검 결과를 조회[취약점점검] 〉 [점검대상등록 및 결과] 〉 [진단이력] 선택



[그림 13] 취약점 점검 이력 조회



[그림 14] 취약점 점검 이력 상세 조회

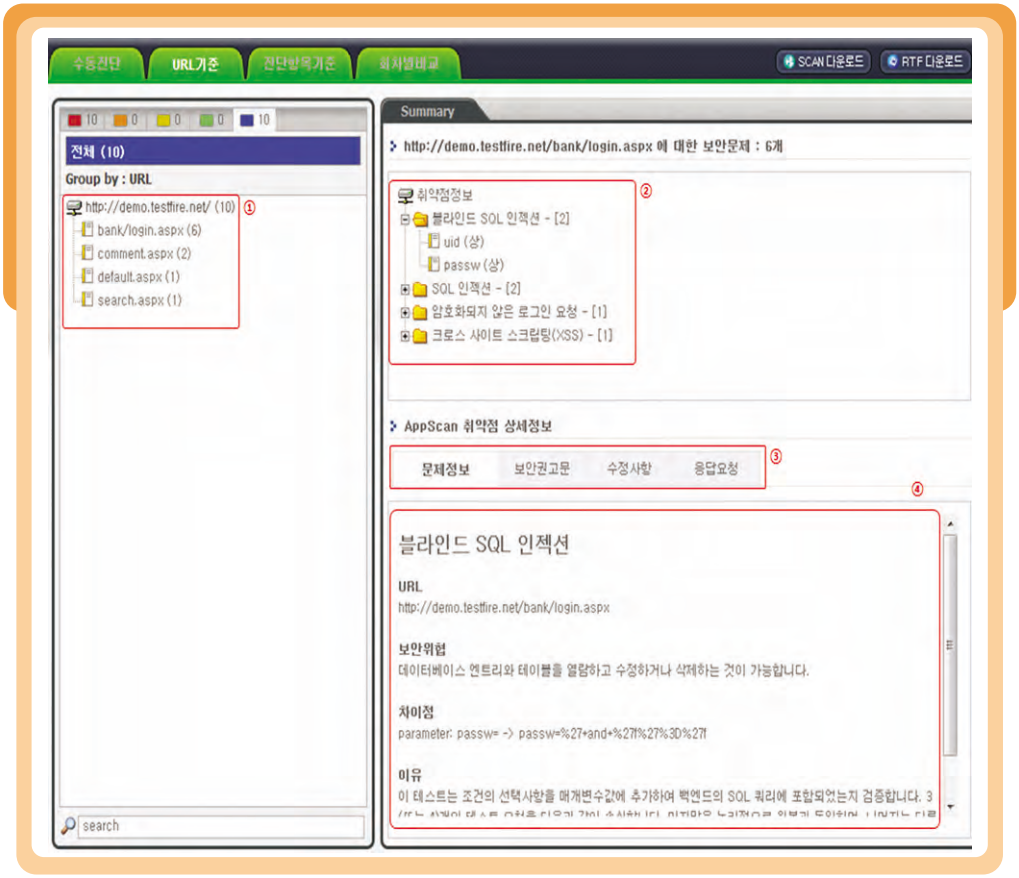
⦿ 세부 점검 결과 확인

– 취약점 점검 결과는 아래와 같이 4가지 항목으로 분류함

① 수동진단 : 분석가 매뉴얼점검에 대한 결과정보를 확인 하는 페이지
② URL기준 : URL을 기준으로 점검 결과를 출력
③ 진단항목기준 : 교과부의 17개 취약점 항목별로 취약점을 출력
④ 회차별비교 : 2회 이상 점검 시 과거 점검 결과를 비교 출력

[그림 15] 취약점 상세 조회

– 발견된 취약점을 URL기준으로 확인할 경우,
① 발견된 취약점을 URL별로 화면 출력
② 취약점 URL, 매개변수 단위로 화면 출력
③ 매개변수 단위로 “문제정보/권고문/수정사항/통신정보” 확인
④ 매개변수 단위의 세부 정보를 화면에 출력

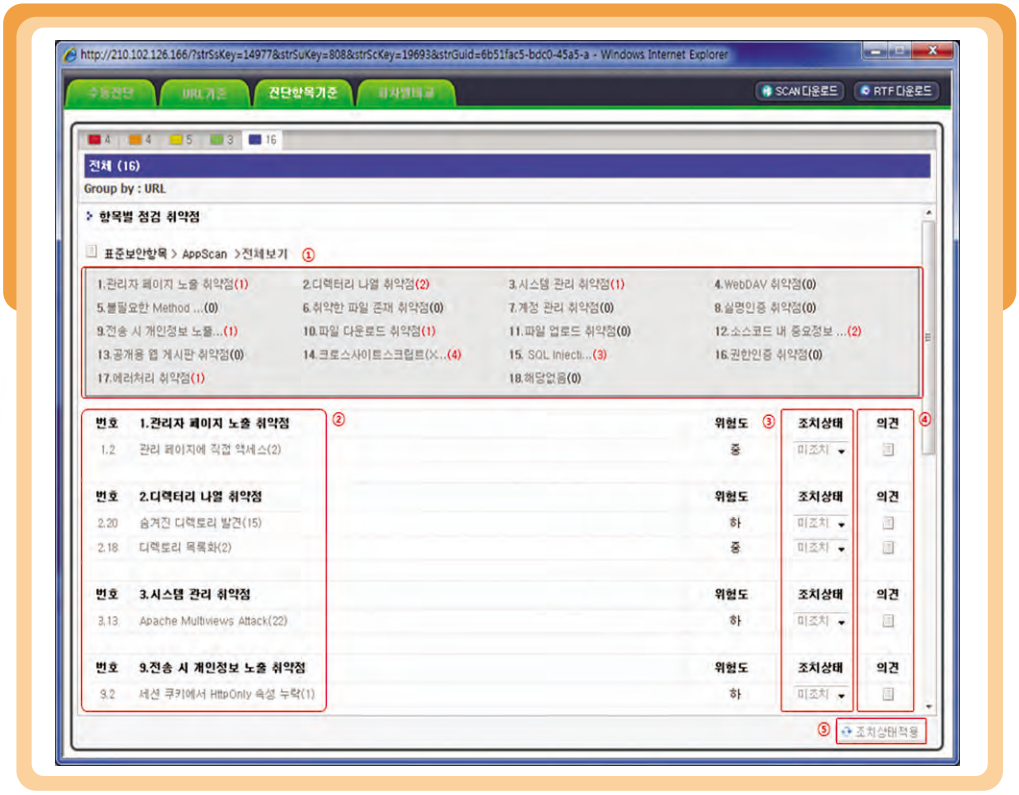


[그림 15] 취약점 상세 조회

- 발견된 취약점을 교과부 17개 항목으로 구분하여 조회
  ① 교과부 17개 점검 항목기준으로 화면 출력
  ② 17개 항목으로 분류된 세부 점검 결과 정보 출력
  ③ 조치 상태를 입력할 수 있는 버튼

※ 취약점 조치를 완료하면 '조치완료'를 필히 선택하여 저장

  ④ 취약점 및 조치 의견을 입력할 수 있는 버튼
  ⑤ 조치 내용을 저장하는 버튼



[그림 17] 교과부 점검항목(17개) 결과 출력

※ 조치 결과 정보는 반드시 등록하여 관리한다.

부록

2. 개인정보 암호화 조치 안내서

제1장 개인정보 암호화 방식

제1절 전송시 암호화

1.1 웹서버와 클라이언트 간 암호화

- 웹서버와 클라이언트 간 개인정보 전송시 암호화를 위하여 공인인증기관이 발급한 서버 인증서를 설치한 보안서버를 사용하는 방식으로 웹브라우저에 기본적으로 내장된 SSL/TLS 프로토콜로 접속하는 SSL 방식과 웹브라우저에 보안 프로그램을 설치하여 접속하는 응용 프로그램 방식으로 구분할 수 있다.

- SSL 방식은 웹페이지 전체를 암호화(웹페이지내 이미지 포함)하며 응용프로그램 방식은 특정 데이터만을 선택적으로 암호화할 수 있지만, 보안서버와 웹브라우저에 부가적인 프로그램을 설치해야 한다.

- 공공기관에서는 국가정보원이 안전성을 확인한 암호모듈 또는 제품을 우선 적용해야 한다.

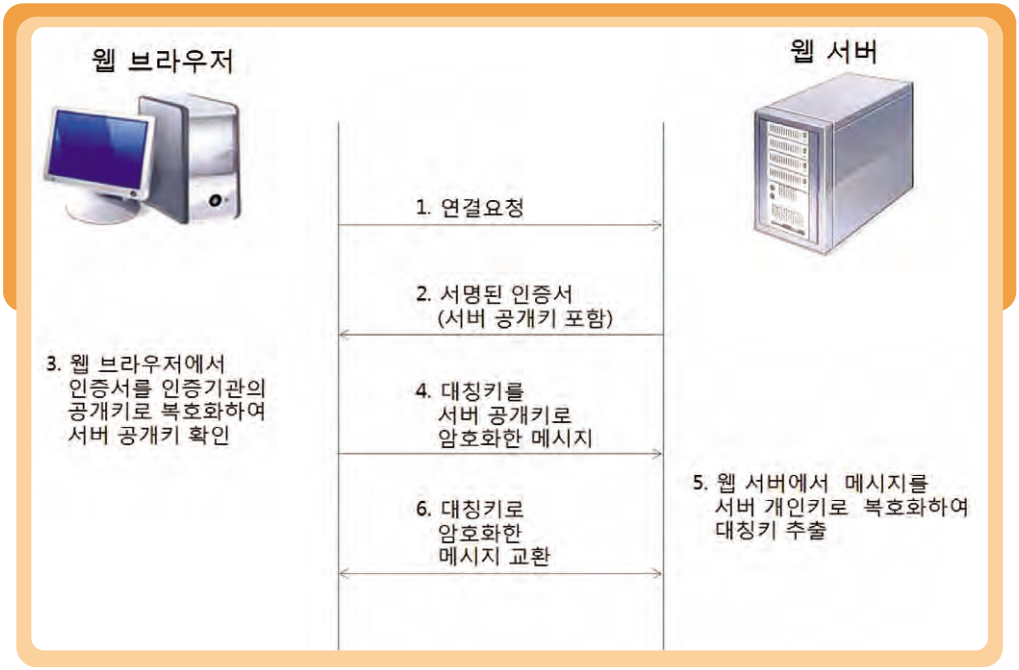
1.1.1 S S L 방식

- SSL 방식은 전송 계층(Transport Layer)을 기반으로 한 응용 계층(Application Layer)에서 암호화를 수행한다. 암호키교환은 비대칭키 암호 알고리즘을 이용하고, 기밀성을 위한암호화는 대칭키 암호 알고리즘을 이용하며 메시지의 무결성은 메시지 인증 코드(해쉬함수)를 이용하여 보장한다.

- 인터넷 쇼핑이나 인터넷 뱅킹 시 계좌정보 및 주민등록번호 등과 같은 중요한 정보를 입력할 때, 거래당사자의 신원 및 거래내용의 위·변조 여부를 확인하고 중요 정보가 제3자에게 유출되는 것을 막기 위해 SSL/TLS와 같은 통신 암호기술을 이용할 수 있다.

- 〈그림 3〉은 인증기관으로부터 인증서를 발급받은 웹서버와 사용자의 웹브라우저 간 SSL/TLS를 이용한 보안 통신의 개념을 간단하게 소개하고 있다. 사용자가 웹서버에 처음 접속하면 인증서 및 통신 암호화에 이용할 암호키를 생성하기 위한 정보를 공유하고, 이후 공유된 정보를 통해 생성된 암호키를 이용하여 데이터를 암호화하여 전송한다.

• SSL/TLS 통신을 하는 경우에는 로그인 페이지 등 보안이 필요한 웹페이지에 접속하면 웹브라우저 하단 상태 표시줄에 자물쇠 모양의 표시를 확인할 수 있다.



[그림 3] 웹서버와 웹르라우저 간의 SSL/TLS 통신구조

1.1.2 응용프로그램 방식

• 응용프로그램 방식은 별도의 모듈을 서버와 클라이언트에 설치해야 하며 필요한 데이터만 암호화하여 전달할 수 있다. 이를 위해 웹서버 프로그램에 대한 수정작업이 필요하며, 응용프로그램 방식을 제공하는 솔루션에 따라 수정작업의 범위가 달라질 수 있다.

• 보안서버를 구현한 웹서버에 사용자가 접속하면 사용자 컴퓨터에 자동으로 보안 프로그램이 설치되고 이를 통해 개인정보를 암호화하여 통신이 이루어진다. 웹브라우저의 확장기능인 플러그인 형태로 구현되며 웹사이트 접속 시 초기화면이나 로그인 후 윈도우 화면 오른쪽 하단 작업표시줄 알림영역을 확인하여 프로그램이 실행되고 있음을 알 수 있다.

1.2 개인정보처리시스템 간 암호화

- 개인정보처리시스템 간에 개인정보를 전송할 때 암호화를 지원하기 위하여 공중망을 이용한 VPN(가상사설망)을 구축할 수 있다.

- VPN은 기반이 되는 보안 프로토콜의 종류에 따라 IPsec VPN 방식, SSL VPN 방식, SSH VPN 방식 등으로 구분할 수 있으며, 개인정보처리시스템 간의 통신에서 사용할 수 있는 VPN 전송 방식의 특징을 간단히 비교하면 [표 2]와 같다.

[표2] 개인정보처리 시스템 간 정송시 암호화 방식 비교

방 식	VPN 서버 부하	NAT 통과
IPsec VPN	낮음	어려움
SSL VPN	다소 높음	쉬움
SSH VPN	다소 높음	쉬움

※NAT(Network Address Translation) : 사설 IP 주소를 공인 IP 주소로 바꿔주는데 사용하는 통신망의 주소 변환기

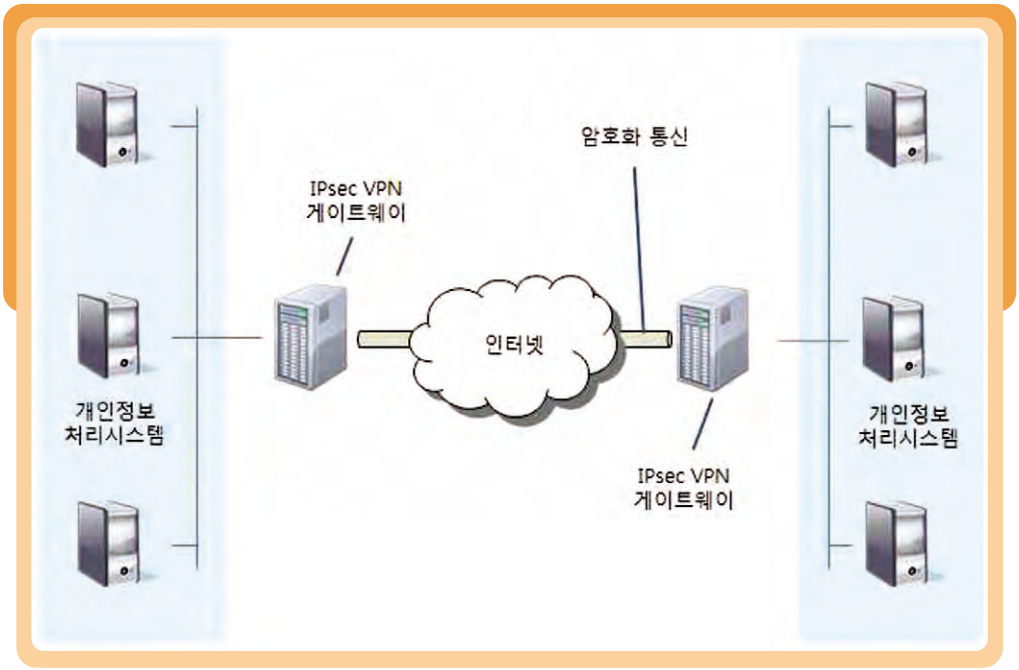
- VPN은 공중망을 통해 데이터를 송신하기 전에 데이터를 암호화하고 수신측에서 이를 복호화 하는 방식으로 송·수신 정보에 대한 기밀성 및 무결성을 보장하며, 그 외에도 데이터 출처 인증, 재전송 방지, 접근제어 등 다양한 보안 기능을 제공한다.

1.2.1 IPsec VPN 방식

- IPsec VPN 방식은 응용프로그램을 수정할 필요가 없으나 IPsec 패킷의 IP 주소를 변경해야 하는 NAT와 같이 사용하기 어려운 점이 있다.
  사용자 인증이 필요 없으므로 VPN 장비 간 서로 인증이 된 경우, 사용자는 다른 인증절차를 거치지 않아도 된다.

- IPsec VPN 방식의 구조는 게이트웨이 대 게이트웨이, 호스트 대 게이트웨이, 호스트 대 호스트로 구분할 수 있다. 게이트웨이 대 게이트웨이는 네트워크 간의 암호화 통신, 호스트 대 게이트웨이는 개인정보처리시스템과 네트워크 간의 암호화 통신, 호스트 대 호스트는 개인정보처리시스템 간의 암호화 통신을 설정할 수 있다.

〈그림 4〉는 게이트웨이 대 게이트웨이 IPsec VPN 방식을 이용하여 인터넷을 통과하는 암호화 통신을 보여준다.

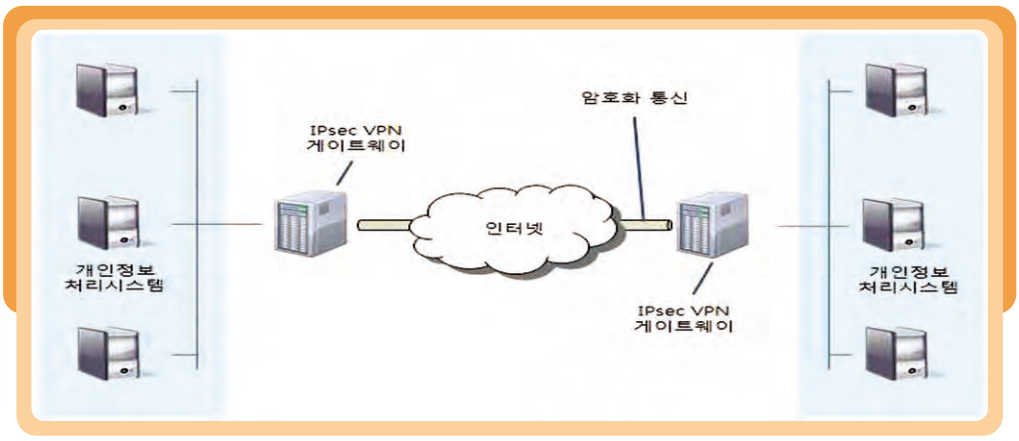


[그림 4] IPsec VPN 방식(게이트웨이 대 게이트웨이)의 개념도

1.2.2 SSL VPN 방식

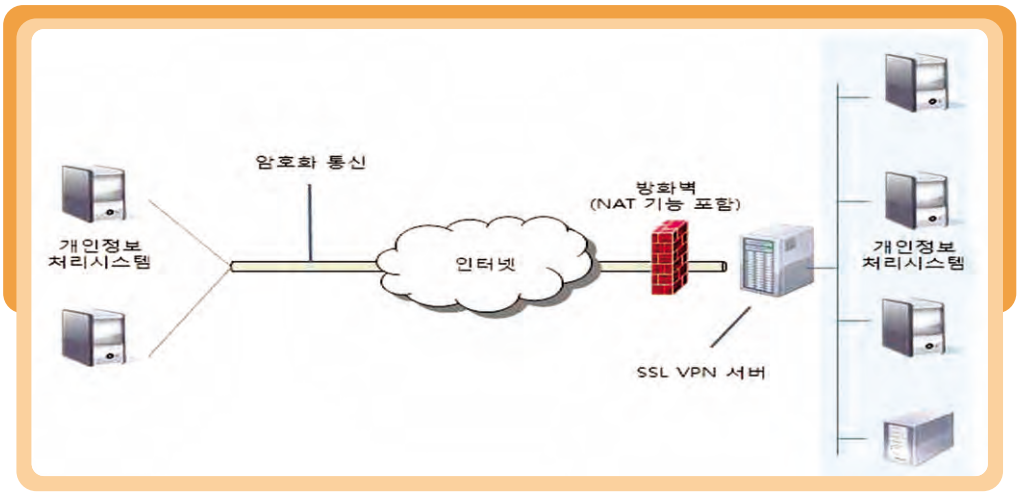
- SSL VPN 방식은 응용프로그램 수준에서 SSL/TLS을 구현하는 것이 일반적이며 NAT를 사용할 수 있다. SSL/TLS는 메모리 소비가 많으므로 동시 접속이 많은 대용량 처리에서 성능 저하가 발생할 수 있다. 하지만 개별 사용자 인증이 필요한 경우 SSL VPN 방식이 좋은 선택이 될 수 있다.

- 〈그림 5〉는 SSL VPN 방식에서 SSL VPN 서버를 거친 개인정보처리시스템 간의 암호화 통신을 보여준다. 이러한 구조는 방화벽 외부에 위치한 개인정보처리시스템과 SSL VPN 서버가 설치된 LAN에 위치한 개인정보처리시스템 간의 통신에 이용이 가능하다.



[그림 5] SSL VPN 방식의 개념도

1.2.3 SSH VPN 방식



- SSH VPN 방식은 응용계층의 VPN 기술로서 원격 단말기에서 접속하는 경우에 주로 이용되며 SSH를 이용한 파일 전송 및 파일 복사 프로토콜(예: SFTP, SCP)을 이용할 수 있다. 오픈소스 SSH의 일종인 OpenSSH의 경우 프락시 방식의 VPN 서버로 구성할 수도 있다.

- 〈그림 6〉은 SSH VPN 방식에서 개인정보처리시스템 간의 암호화 통신을 보여준다. 각 개인정보처리시스템에 설치된 SSH 기능을 사용하여 VPN을 구성할 수 있다.

TIP 개인정보처리시스템 간 전송시 공중망과 분리된 전용선을 사용하면 암호화에 상응하는 보안성을 제공할 수 있다.

1.3 개인정보취급자 간 암호화

개인정보취급자 간에 개인정보를 전송할 때 주로 이메일을 이용하게 된다. 이메일은 네트워크를 통해 전송되는 과정에서 공격자에 의해 유출되거나 위조될 가능성이 있다. 이러한 위협으로부터 이메일로 전송되는 메시지를 보호하기 위해서 PGP 또는 S/MIME을 이용하는 이메일 암호화 방식과 암호화된 파일을 이메일에 첨부하여 전송하는 이메일 첨부문서 암호화 방식이 있다.

개인정보취급자 간에 이메일을 사용하지 않고 직접 파일을 전송하고자 하는 경우는 개인정보처리시스템 간 전송시 암호화 방식의 VPN 기능을 적용할 수 있다.

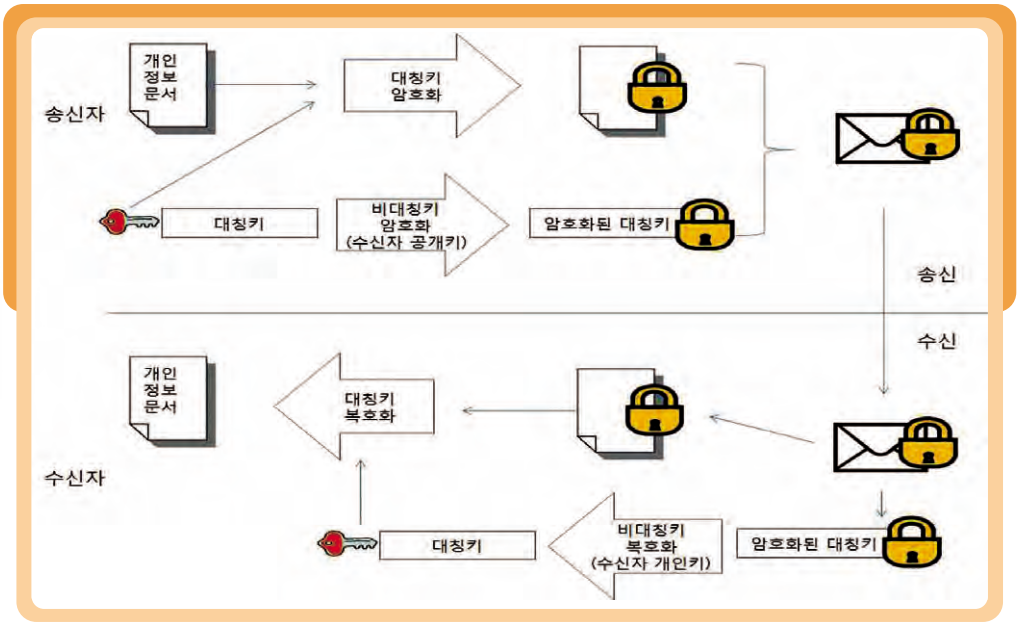
[표2] 개인정보처리 시스템 간 정송시 암호화 방식 비교

방 식		공인인증서 필요 여부	표준형식
이메일 암호화	PGP	필요하지 않음	PGP 자체정의
	S/MIME	필요함	X.509, PKCS#7
이메일 첨부문서 암호화		필요하지 않음	없음

- S/MIME은 공개키를 포함한 공인인증서를 발급받고 등록해야 하는 번거로움이 있다. 이에 비해 PGP의 경우 개인 간의 신뢰를 바탕으로 공개키를 등록하거나 안전한 채널로 미리 확보하는 방법을 사용할 수 있다.

1.3.1 이메일 암호화 방식

이메일 암호화 방식은 송·수신되는 이메일의 내용을 암호화함으로써 메일 내 중요 개인정보의 유출을 방지하는 것이며, 대표적인 이메일 보안 프로토콜로는 PGP와 S/MIME이 있다. 〈그림 7〉은 이메일 암호화 방식의 처리 과정을 보여준다.



[그림7] 이메일 화호화 방식의 개념도

- PGP는 다양한 응용프로그램에 적용하여 문서, 이메일, 파일, 파일시스템, 디스크 등을 암호화할 수 있다.

- S/MIME은 인증, 메시지 무결성, 부인방지, 메시지 암호화 등에 사용되며 대부분의 이메일 클라이언트에서 기본적으로 지원한다. S/MIME을 사용하기 위해서는 공인인증기관이 발행한 공인인증서가 있어야 한다.

1.3.2 이메일 첨부문서 암호화 방식

- 업무용 컴퓨터에서 주로 사용하는 문서 도구(예 : 한글, MS 워드 등)의자체 암호화 방식, 암호 유틸리티를 이용한 암호화 방식 등을 통해 암호화한 파일을 이메일의 첨부문서로 송 · 수신할 수 있다. [1)]

- 이메일을 송 · 수신할 개인정보취급자 간에는 암호키(또는 비밀번호) 안전하게 공유하여야 한다.

1)파일 암호화 방식은 '1장 2.2 업무용 컴퓨터 암호화'를 참고

제 2 절 저장시 암호화

2.1 개인정보처리시스템 암호화

2.1.1 개요

- 개인정보를 처리하고 관리하는 개인정보처리시스템은 DB에 저장된 개인 정보를 암호화하여 저장함으로써 개인정보의 변경, 파괴 및 유출을 방지 해야 한다.

- 개인정보처리시스템의 DB를 암호화할 수 있는 방식은 암·복호화 모듈의 위치와 암·복호화 모듈의 요청 위치의 조합에 따라 [표 4]와 같이 구분할 수 있다.

[표4] 개인정보처리시스템 암호화 방식의 구분

방 식	암·복호화 모듈 위치	암·복호화 요청위치	주요내용
응용 프로그램 자체 암호화	어플리케이션 서버	응용 프로그램	• 암·복호화 모듈이 API 라이브러리 형태로 각 어플리케이션 서버에 설치되고, 응용프로그램에서 해당 암·복호화 모듈을 호출하는 방식 • DB 서버에 영향을 주지 않아 DB 서버의 성능 저하가 적은 편이지만 구축 시 응용프로그램 전체 또는 일부 수정 필요 • 기존 API 방식과 유사
DB 서버 암호화	DB 서버	DB 서버	• 암·복호화 모듈이 DB 서버에 설치되고 DB 서버에서 암·복호화 모듈을 호출하는 방식 • 구축 시 응용프로그램의 수정을 최소화 할 수 있으나 DB 서버에 부하가 발생하며 DB 스키마의 추가 필요 • 기존 Plug-In 방식과 유사
DBMS 자체 암호화	DB 서버	DBMS 엔진	• DB 서버의 DBMS 커널이 자체적으로 암·복호화 기능을 수행하는 방식 • 구축 시 응용프로그램 수정이 거의 없으나, DBMS에서 DB 스키마의 지정 • 기존 커널 방식(TDE)과 유사

부록 2. 개인정보 암호화 조치 안내서

방 식	암·복호화 모듈 위치	암·복호화 요청위치	주요내용
DBMS 암호화 기능 호출	DB 서버	응용 프로그램	• 응용프로그램에서 DB 서버의 DBMS 커널이 제공하는 암·복호화 API를 호출하는 방식 • 구축 시 암·복호화 API를 사용하는응용프로그램의 수정이 필요 • 기존 커널 방식(DBMS 함수 호출)과 유사
운영체제 암호화	파일서버	DB 서버	• OS에서 발생하는 물리적인 입출력(I/O)을 이용한 암·복호화 방식으로 DBMS의 데이터파일 암호화 • DB 서버의 성능 저하가 상대적으로 적으나 OS, DBMS, 저장장치와의 호환성 검토 필요 • 기존 DB 파일암호화 방식과 유사

TIP 각 방식의 단점을 보완하기 위하여 두 가지 이상의 방식을 혼합하여 구현하기도 한다. 이 경우, 구축 시 많은 비용이 소요되지만 어플리케이션 서버 및 DB 서버의 성능과 보안성을 높일 수 있다.

• 개인정보처리시스템 암호화 방식마다 성능에 미치는 영향이 다르므로구축 환경에 따라 암호화 방식의 특성, 장단점 및 제약사항 등을 고려하여 DB 암호화 방식을 선택해야 한다. [표 5]는 개인정보처리시스템 암호화 방식의 선택 시 고려해야 할 사항이다.

[표 5] 개인정보처리 암호화 방식의 구분

방 식	공인인증서 필요 여부
일반적 고려사항	구현 용이성, 구축 비용, 기술지원 및 유지보수 여부
	암호화 성능 및 안전성
	공공기관의 경우, 국가정보원 인증 또는 검증 여부
기술적 고려사항	암 복호화 위치(어플리케이션 서버, DB 서버, 파일서버 등)
	색인검색 가능 유무, 배치처리 가능 여부

TIP 성능이 매우 중요한 요소가 되는 환경에서 DB 서버 암호화 방식을 고려하는 경우에는 반드시 벤치마킹 테스트(BMT) 등을 수행하여, 최적의 솔루션을 선택하는 것이 바람직하다.

- 공공기관에서는 국가정보원이 안전성을 확인한 암호모듈 또는 제품을 우선 적용해야 한다.[2)]

- 암·복호화 모듈의 위치와 암·복호화 요청 위치에 따라 어플리케이션 서버 또는 DB 서버의 성능에 영향을 미칠 수 있다. 예를 들어, DB 서버 암호화 방식은 암·복호화시 DB 서버의 자원을 추가적으로 사용하므로 대량의 트랜잭션 작업에서 DB 서버의 성능 저하가 발생할 수 있다.

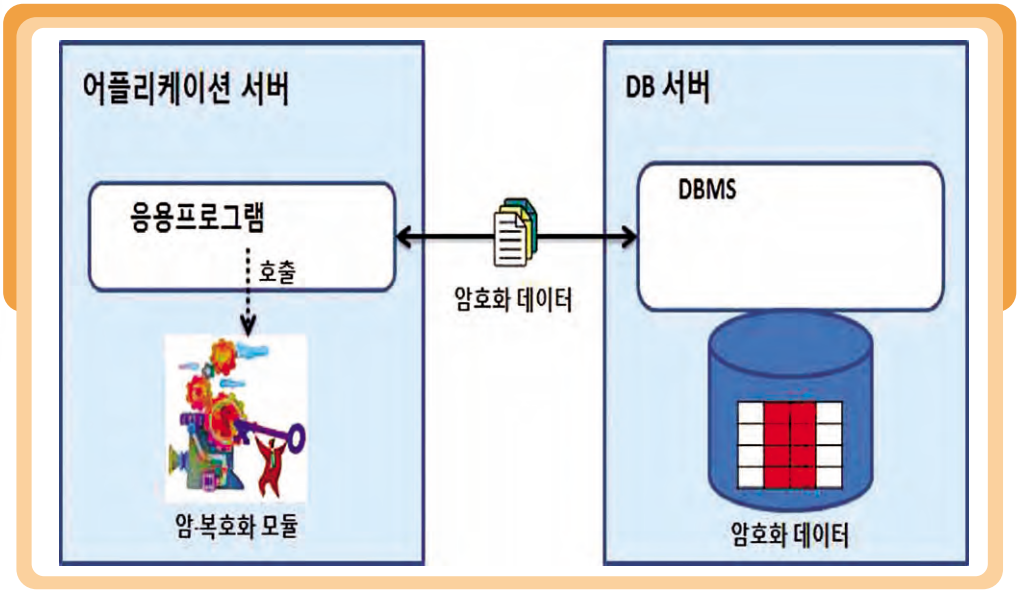
- 현재 운영 중이거나 향후 개발 예정인 개인정보처리시스템의 목적 및 환경에 맞게 쉽게 구현이 가능한 암호화 방식을 선택해야 한다. 응용프로그램 및 DB 스키마 수정 등을 최소화하고 개발 환경에 맞게 성능을 최대화할 수 있도록 해야 한다.

- DB 암호화의 안전성을 확보하기 위해서는 안전한 암호키의 관리가 필요하다. 암호화된 개인정보가 유출되더라도 복호화 할 수 없도록 암호키에 대한 추가적인 보안과 제한된 관리자만 허용하도록 하는 기술의 적용을 권고한다.

2) IT보안인증사무국(http://service1.nis.go.kr)의 검증필 암호 모듈 또는 제품 확인이 가능함

2.1.2 응용프로그램 자체 암호화 방식

- 응용프로그램 자체 암호화 방식은 〈그림 8〉과 같이 암·복호화 모듈이 API 라이브러리 형태로 각 어플리케이션 서버에 설치되고 응용프로그램에서 암·복호화 모듈을 호출하는 방식이다.
- DB 서버에는 영향을 주지 않지만 어플리케이션 서버에 암·복호화를 위한 추가적인 부하가 발생하며, 구축 시 응용프로그램 전체 또는 일부 수정이 필요하다.
- 추가적으로 어플리케이션 서버와 DB 서버 간의 통신에서 암호화된 개인정보의 전송을 보장할 수 있다.



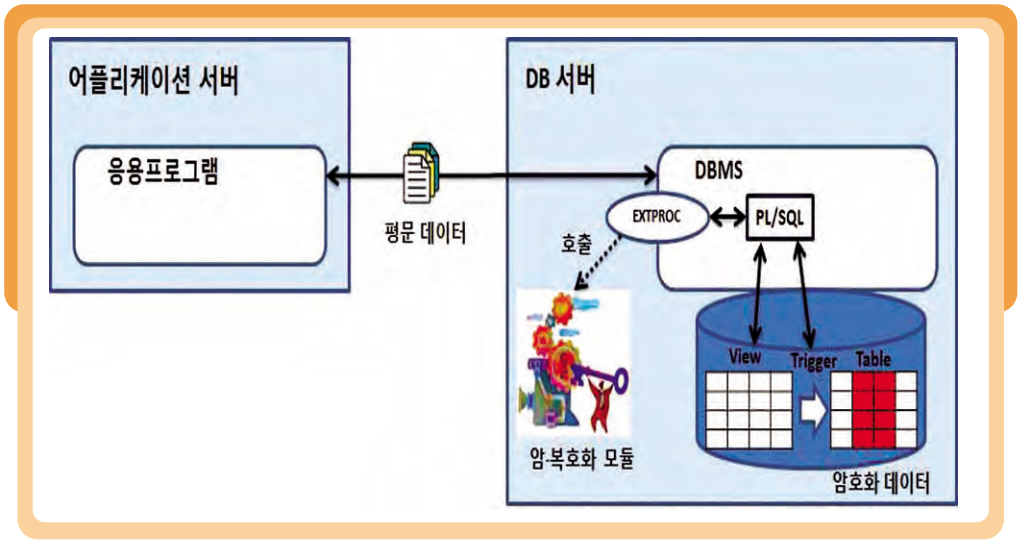
- 응용프로그램 자체 암호화 방식의 주요 특성은 [표 6]과 같다.

[표 5] 개인정보처리 암호화 방식의 구분

항 목	주 요 내 용
암·복호화 모듈	어플리케이션 서버
암·복호화 요청	응용프로그램
DB 서버의 부하	없음(어플리케이션 서버에 부하 발생)
색인 검색	일치검색 가능 별도 색인 테이블 생성을 통해 가능(추가 작업 필요)
배치 처리	가능
응용프로그램 수정	필요함
DB 스키마 수정	거의 필요하지 않음(암호화에 따른 속성 타입이나 길이의 변경이 필요할 수 있음)

2.1.3 DB 서버 암호화 방식

- DB 서버 암호화 방식은 〈그림 9〉와 같이 암·복호화 모듈이 DB서버에 설치되고 DBMS에서 플러그인(plug-in)으로 연결된 암·복호화 모듈을 호출하는 방식이다.
- 응용프로그램의 수정이 거의 필요하지 않아 구현 용이성이 뛰어나지만,기존 DB 스키마와 대응하는 뷰(view)를 생성하고 암호화할 테이블을 추가하는 작업이 필요하다.
- 어플리케이션 서버의 성능에는 영향을 주지 않지만 DBMS에서 DB 서버의 암·복호화 모듈을 플러그인으로 호출할 때 추가적인 부하가 발생하여 성능이 저하될 수 있다.



[그림9] DB 서버 암호화 방식의 개념도

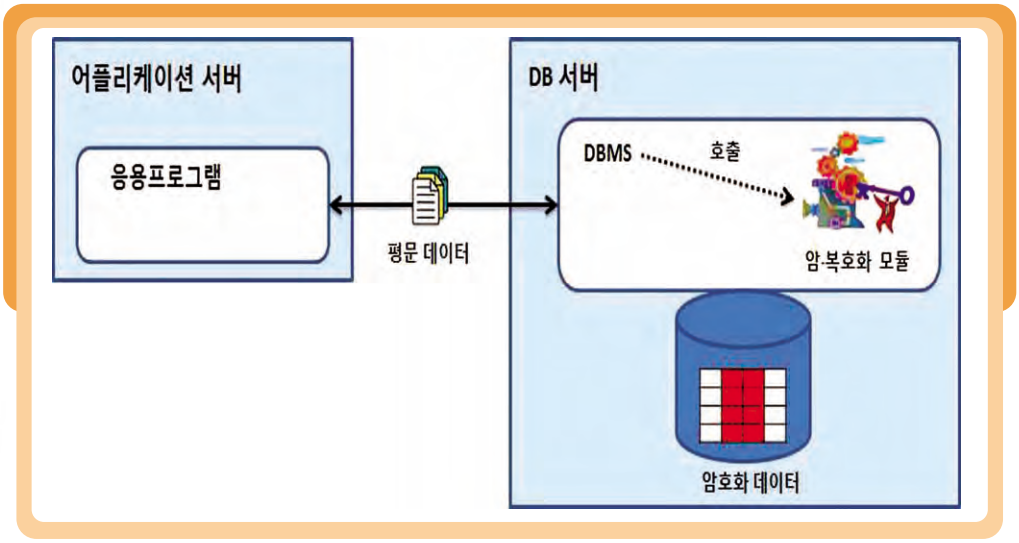
- DB 서버 암호화 방식의 주요 특성은 [표 7]과 같다.

[표 7] DB 서버 암호화 방식의 주요 특성

항 목	주 요 내 용
암·복호화 모듈	DB 서버
암·복호화 요청	DB 서버
DB 서버의 부하	있음
색인 검색	가능
배치 처리	가능(대량의 배치 트랜젝션 처리는 많이 느릴 수 있음)
응용프로그램 수정	기본적으로 수정 없이 적용할 수 있으나, 제약사항 또는 성능 문제가 있는 경우 수정이 필요함
DB 스키마 수정	필요함

2.1.4 DBMS 자체 암호화 방식

- DBMS 자체 암호화 방식은 〈그림 10〉과 같이 DBMS에 내장되어 있는암호화 기능(TDE : Transparent Data Encryption)을 이용하여 암·복호화 처리를 수행하는 방식이다.
- DBMS 커널 수준에서 처리되므로 기존 응용프로그램의 수정이나 DB스키마의 변경이 거의 필요하지 않고 DBMS 엔진에 최적화된 성능을제공할 수 있다.



[그림10] DBMS 자체 암호화 방식의 개념도

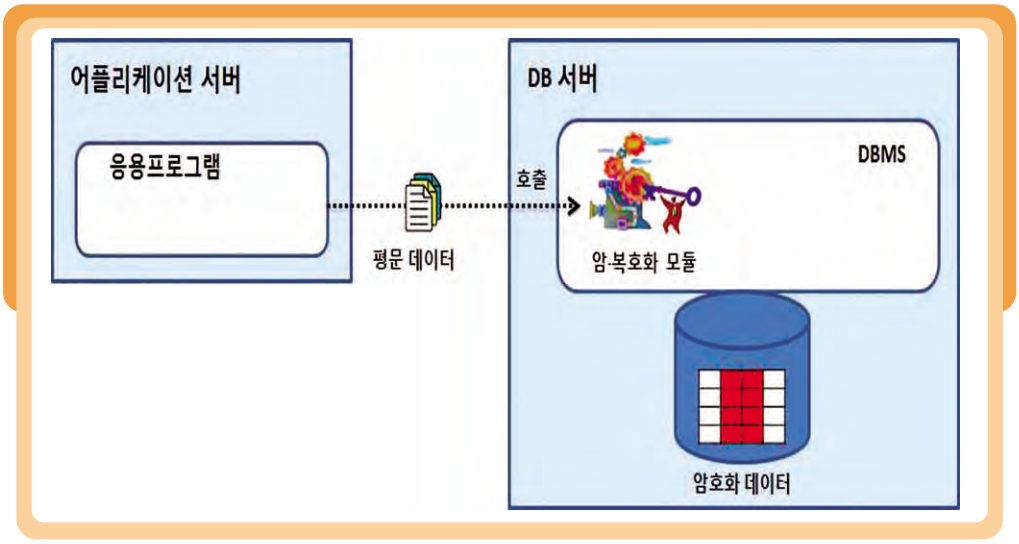
- DBMS 자체 암호화 방식의 주요 특성은 [표 8]과 같다.

[표 8] DBMS 자체 암호화 방식의 주요 특성

항 목	주 요 내 용
암·복호화 모듈	DB 서버
암·복호화 요청	DBMS 엔진
DB 서버의 부하	있음
색인 검색	가능
배치 처리	가능
응용프로그램 수정	필요하지 않음
DB 스키마 수정	거의 필요하지 않음(암호화할 DB 스키마 지정 필요)

2.1.5 DBMS 암호화 기능 호출 방식

- DBMS 암호화 기능 호출 방식은 〈그림 11〉과 같이 DBMS가 자체적으로 암·복호화 기능을 수행하는 API를 제공하고 해당 함수를 사용하기 위해 응용프로그램에서 호출하는 방식이다.
- 암·복호화 API를 사용하는 응용프로그램의 수정이 필요하고, DB 서버에 추가적인 부하가 발생할 수 있다.



[그림11] DBMS 암호화 기능 호출 방식의 개념도

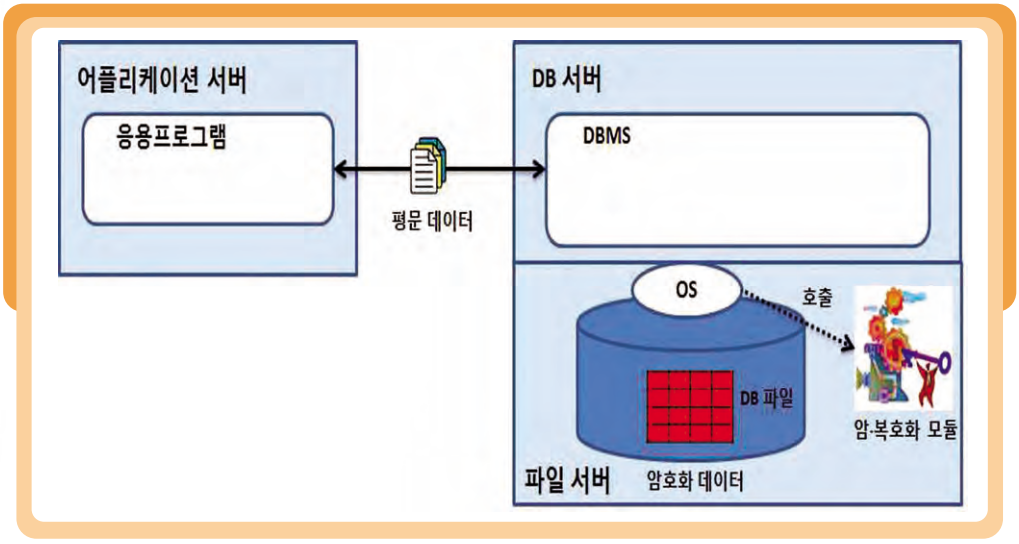
- DBMS 암호화 기능 호출 방식의 주요 특성은 [표 9]와 같다.

[표 9] DBMS 암호화 기능 호출 방식의 주요 특성

항 목	주 요 내 용
암·복호화 모듈	DB 서버
암·복호화 요청	응용프로그램
DB 서버의 부하	있음
색인 검색	불가능
배치 처리	가능(대량의 배치 트랜잭션 처리는 많이 느릴 수 있음)
응용프로그램 수정	수정 필요
DB 스키마 수정	일부 수정 필요

2.1.6 운영체제 암호화 방식

- 운영체제 암호화 방식은 〈그림 12〉와 같이 OS에서 발생하는 입출력 시스템 호출을 이용한 암·복호화 방식으로서 DB 파일 자체를 암호화한다.
- 응용프로그램이나 DB 스키마의 수정이 필요하지 않지만 DB 파일 전체를 암호화하는데 따른 파일 서버 및 DB 서버에 추가적인 부하가 발생할 수 있다.



[그림12] 운영제제 암호화 방식의 개념도

- 운영체제 암호화 방식의 주요 특성은 [표 10]과 같다.

[표 10] 운영체제 암호화 방식의 주요 특성

항 목	주 요 내 용
암·복호화 모듈	파일 서버(또는 DB 서버)
암·복호화 요청	운영체제
DB 서버의 부하	있음
색인 검색	가능
배치 처리	가능
응용프로그램 수정	필요하지 않음
DB 스키마 수정	필요하지 않음

2.2 업무용 컴퓨터 암호화

2.2.1 개요

- 업무용 컴퓨터에서는 보조저장매체에 저장된 개인정보의 보호를 위하여 개별 문서 파일 단위로 암호화(파일 암호화) 또는 디렉터리 단위로 암호화(디스크 암호화)를 수행해야 한다.
- 파일 암호화는 업무용 컴퓨터에 저장된 개인정보에 대한 보호뿐만 아니라 개인정보취급자 간에 네트워크상으로 파일을 안전하게 전송하기 위한 방식으로도 사용할 수 있다.
- 업무용 컴퓨터에서 가능한 암호화 방식은 [표 11]과 같이 구분할 수 있다.

[표 11] 업무용 컴퓨터 암호화 방식의 구분

방식	주 요 내 용
문서 도구 자체 암호화	• 업무용 컴퓨터에서 사용하는 문서도구의 자체 암호화 • 기능을 통하여 개인정보 파일 암호화
암호 유틸리티를 이용한 암호화	• 업무용 컴퓨터의 OS에서 제공하는 파일 암호 • 유틸리티또는 파일 암호 전용 유틸리티를 이용한 개인정보 파일의 암호화
DRM (Digital Right Management)	• DRM을 이용하여 다양한 종류의 파일 및 개인정보파일의 암호화
디스크 암호화	• 디스크에 데이터를 기록할 때 자동으로 암호화하고, 읽을 때 자동으로 복호화하는 기능을 제공함 • 디스크 전체 또는 일부 디렉터리를 인가되지 않은 사용자에게 보이지 않게 설정하여 암호화 여부와 관계없이 특정 디렉터리 보호 가능

- 업무용 컴퓨터 암호화 방식의 특징을 간단히 비교하면 [표 12]와 같다.

[표 12] 업무용 컴퓨터 암호화 방식의 비교

방 식	주 요 내 용	
	특 정 문 서 *	일 반 파 일 * *
문서 도구 자체 암호화	지원함	지원하지 않음
암호 유틸리티를 이용한 암호화	지원함	지원함
DRM	지원함	지원함
디스트 암호화	지원함	지원함

*특정문서 : 흔히 사용하는 문서 도구(예 : 한글, MS 워드 등)로 작성한 파일
**일반문서 : 특정 문서 이외의 문서(예 : 텍스트 파일, 이미지 파일 등)

2.2.2 문서 도구 자체 암호화 방식

- 업무용 컴퓨터에서 주로 사용하는 문서 도구(예를 들어, 글, MS 워드 등)에서는 자체 암호화 기능을 통하여 개인정보 파일을 암호화할 수 있다.

2.2.3 암호 유틸리티를 이용한 암호화 방식

- 업무용 컴퓨터에서는 해당 컴퓨터의 OS에서 제공하는 파일암호 유틸리티 또는 파일암호 전용 유틸리티를 이용하여 개인정보 파일 또는 디렉터리를 암호화할 수 있다.

2.2.4 DRM 방식

- DRM은 조직 내부에서 생성되는 전자문서를 암호화하고 해당 문서를 접근 및 사용할 수 있는 권한을 지정함으로써 허가된 사용자만 중요 문서(개인정보 문서, 기밀문서 등)를 사용하게 하는 기술이다.

- DRM은 중요 문서 외에 다양한 종류의 멀티미디어 콘텐츠(음악, 사진,동영상, 이미지 등)에 대한 보안 기능을 제공할 수 있다.

- DRM으로 암호화된 문서는 DRM 클라이언트가 없는 PC에서는 열람이 불가능하며, 열람 중에도 파일이 복호화 되지 않고 암호화 상태를 유지한다.

2.2.5 디스크 암호화 방식

- 디스크 암호화는 디스크에 데이터를 기록할 때 자동으로 암호화하고, 주기억장치로 읽을 때 자동으로 복호화하는 방식이다.

- 휴대용 보조기억매체는 개방된 장소에 놓일 수 있기 때문에 적절한 물리적 보안을 제공하기 어려움이 있다. 따라서 휴대용 보조기억매체는 저장된 개인정보의 기밀성을 위해 디스크 암호화 솔루션을 이용하여 암호화하기를 권고한다.

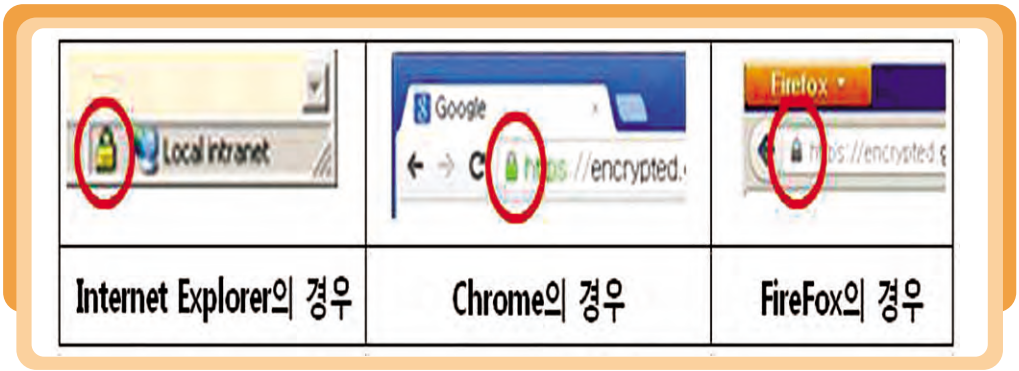
제 2장 개인정보 암호화 적용 사례

제1절 전송시 암호화

1.1 웹서버와 클라이언트 간 암호화 사례

• 1.1.1 아파치(Apache) 웹서버를 이용한 SSL 방식의 설정
대표적인 오픈소스 웹서버 소프트웨어인 아파치에서 설정파일인 'httpd.conf'를 변경하여 SSL/TLS를 설정할 수 있다. 이 설정파일에는 공인인증서의 위치, 서버용 인증서 위치, 공개키와 개인키의 위치 등이 들어가며 SSL/TLS에서 사용하는 암호 알고리즘을 정해준다.

• 웹브라우저가 SSL 방식으로 웹서버에 연결된 경우, 〈그림 13〉과 같이 웹브라우저 주소창 또는 하단의 상태표시줄에 자물쇠 표시가 나타나게 된다.



[그림13] SSL 방식에서 나타나는 웹브라우저 자물쇠 표시

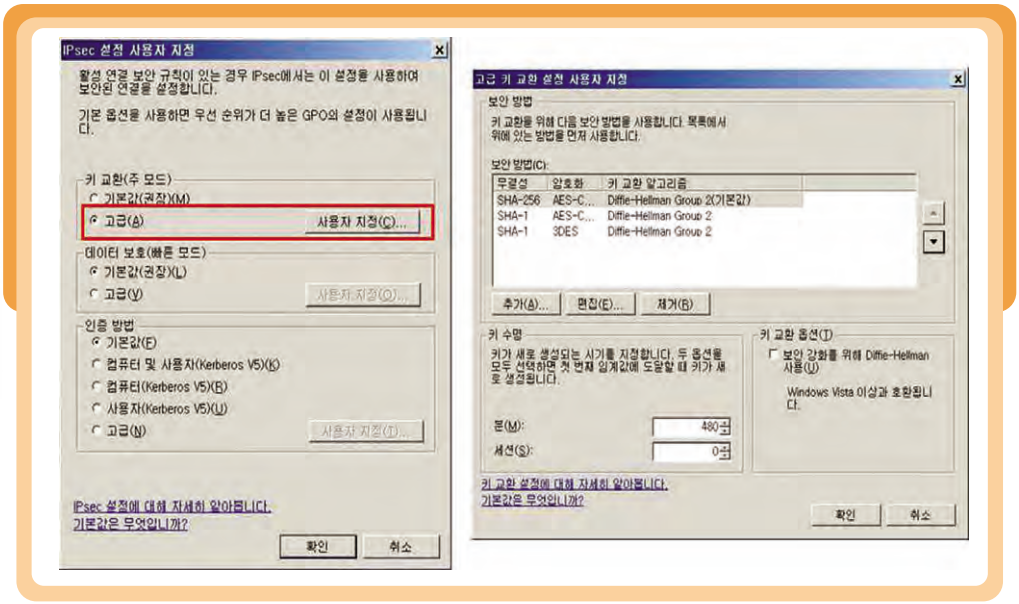
1.2 개인정보처리시스템 간 암호화 사례

1.2.1 윈도우(Windows)에서 IPsec VPN 방식의 설정

• 윈도우를 호스트로 사용하여 IPsec VPN에 접속할 경우, 안전한 암호 알고리즘의 선택을 위해 추가 설정이 필요할 수 있다.

• Windows 7의 제어판 메뉴에서 [윈도우 방화벽] → [고급설정] → [로컬 컴퓨터 고급 보안이 포함된 윈도우 방화벽] → [속성] → [IPsec 설정] → [사용자 지정]을 선택한다.

• 〈그림 14〉의 [IPsec 설정 사용자 지정]과 같은 대화창이 나타나면, [키교환] → [사용자 지정]을 선택하여 [고급 키 교환 설정 사용자 지정]에서 IPsec VPN 방식에 사용할 암호 알고리즘을 변경할 수 있다.

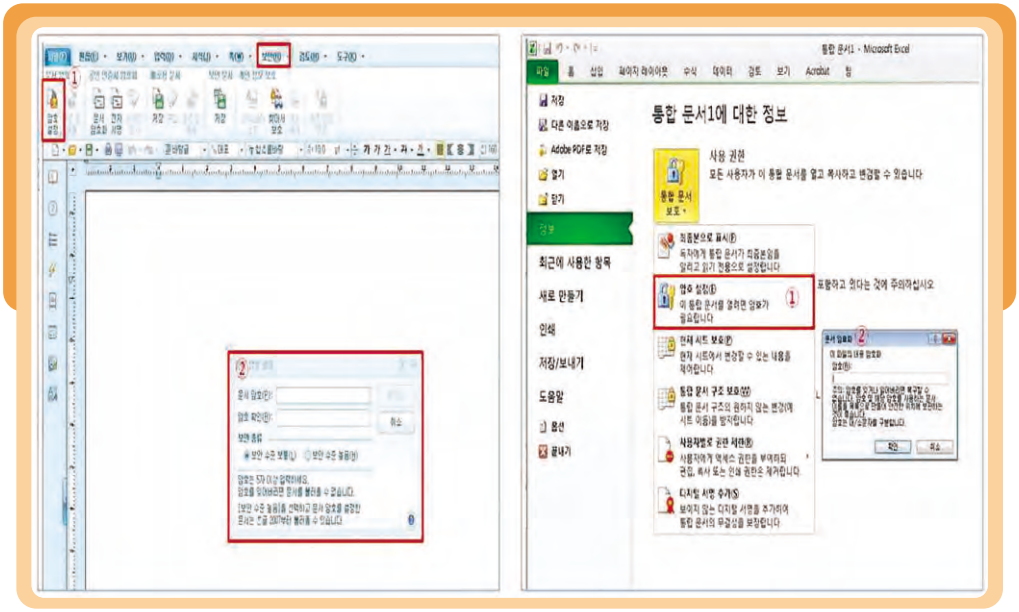


[그림 14]Windows 7에서 IPsec VPN방식을 위한 암호 알고리즘 설정

1.3 개인정보취급자 간 암호화 사례

1.3.1 첨부문서 암호화 후, 이메일로 전송

- 먼저, 응용프로그램의 암호화 기능을 사용하여 암호를 설정한 후, 문서를 저장한다.
- 한글 2010의 경우, 상단 메뉴의 [보안] → [문서 암호 설정]을 이용하여 문서의 암호를 설정 한 후, [파일] → [저장하기] 메뉴를 이용하여 문서 내용을 저장한다.
- MS 엑셀 2010의 경우 상단 메뉴의 [파일] → [정보] → [통합 문서 보호] → [암호 설정]을 이용하여 문서의 암호를 설정 한 후, [파일] → [저장하기] 메뉴를 이용하여 문서의 내용을 저장한다.



[그림15]한글 2010과 MS 엑셀 2010에서 문서 암호화 설정

• 암호화된 문서를 이메일에 첨부한 후, 수신자에게 이메일을 전송한다.

제2절 저장시 암호화

2.1 개인정보처리시스템 암호화 사례

2.1.1 응용프로그램 자체 암호화 방식

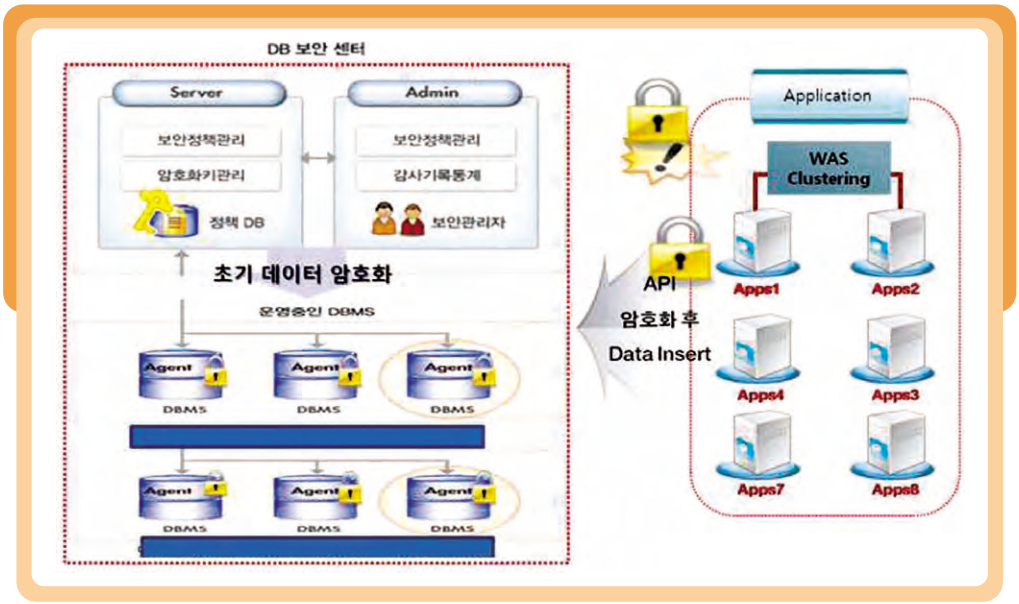
■ 적용 환경

• 적용분야 : 공공기관
• 업무종류 : OO기관 대국민서비스
• 개인정보보유량 : 약 9천3백만 건

■ 적용 사유

• 차세대 시스템으로 새로운 응용프로그램 개발이 필요함
• 기존 DBMS에서 플러그인을 제공하지 않음

■ 적용 구성도



[그림 16] 응용프로그램 자체 암호화 방식의 적용 구성도

■ 주요 특징

- 암·복호화 작업이 다수의 어플리케이션 서버로 부하 분산
- 암호화 컬럼 크기 증가에 따라 관련 응용프로그램 인터페이스의 변경이 필요
- 암호화 후 DB 서버의 성능 저하는 적으나, 일부 질의에서는 색인 처리불가로 응용프로그램 코드의 변경이 요구
- 암호화 컬럼 크기 증가에 따른 DB 서버 디스크 및 주기억장치 증설 필요
- 암·복호화 작업 부하에 따라 자원 여유율이 매우 작은 응용 서버(예 :WAS)는 자원 증설이 요구

2.1.2 DB 서버 암호화 방식

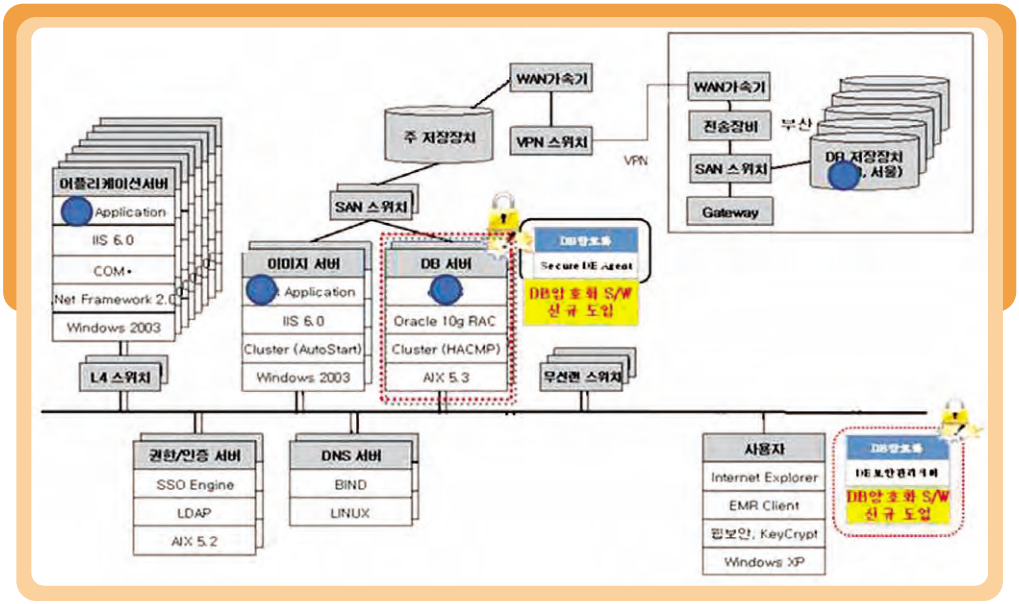
■ 적용 환경

- 적용분야 : 공공기관
- 업무종류 : OO기관 통합정보시스템
- 개인정보보유량: 약 1억 건

■ 적용 사유

- 운영 중인 응용프로그램 및 패키지 응용프로그램 수정을 최소화하여 단기간에 개발이 필요함
- 기존 DBMS에서 플러그인 기능을 제공함
- DB 서버의 성능 저하가 발생할 만한 복잡한 트랜잭션이나 배치 업무가 적음

■ 적용 구성도



[그림 17]DB 서버 암호화 방식의 적용 구성도

■ 주요 특징

- 암·복호화 작업이 DB 서버에 집중됨으로써 해당 서버의 자원 (CPU 및 주기억장치) 사용률 증가
- 암·복호화 뷰(암호화 이전 테이블명과 동일)와 트리거 구조를 이용하여 응용프로그램 변경 최소화
- 암호화 컬럼 크기 증가에 따라 관련 응용프로그램 인터페이스의 변경필요
- 암호화로 인한 DB 성능 저하를 최소화하기 위하여 DB 질의와 응용 프로그램의 튜닝 필요
- 암호화 컬럼 크기 증가와 암·복호화 작업 부하로 인해 DB 서버에 CPU 및 주기억장치, 디스크 증설 필요

2.1.3 DBMS 자체 암호화 방식

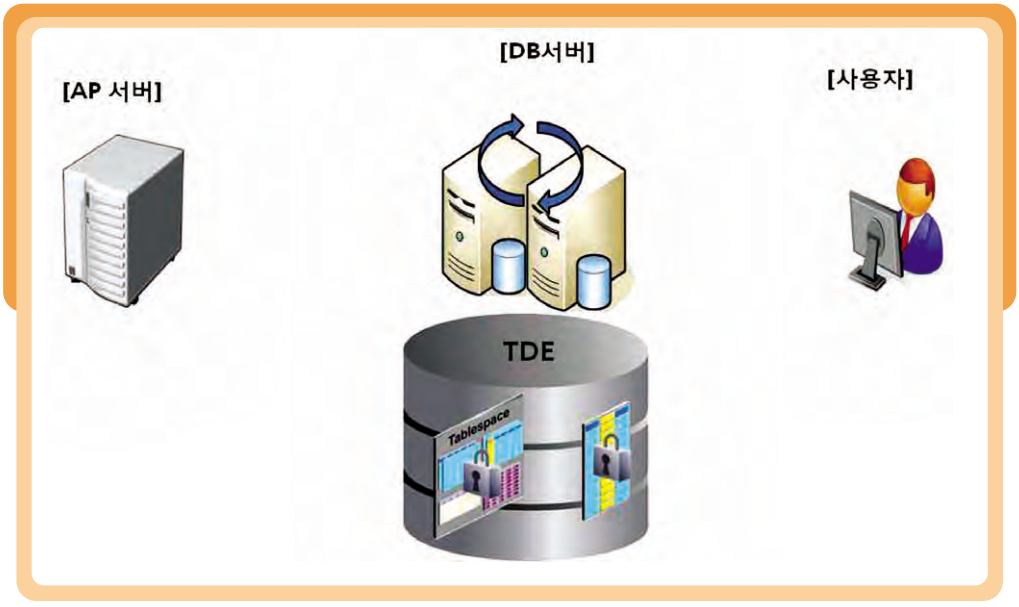
■ 적용 환경

- 적용분야: 민간기관
- 업무종류: OO병원 수납시스템
- 개인정보보유량: 약 3억8천만 건

■ 적용 사유

- 개발 인력의 부족으로 기존 응용프로그램의 수정을 최소화해야 함
- 대량 개인정보의 안정적인 처리가 필요함

■ 적용 구성도



[그림 18]DBMS자체 암호화 방식의 적용 구성도

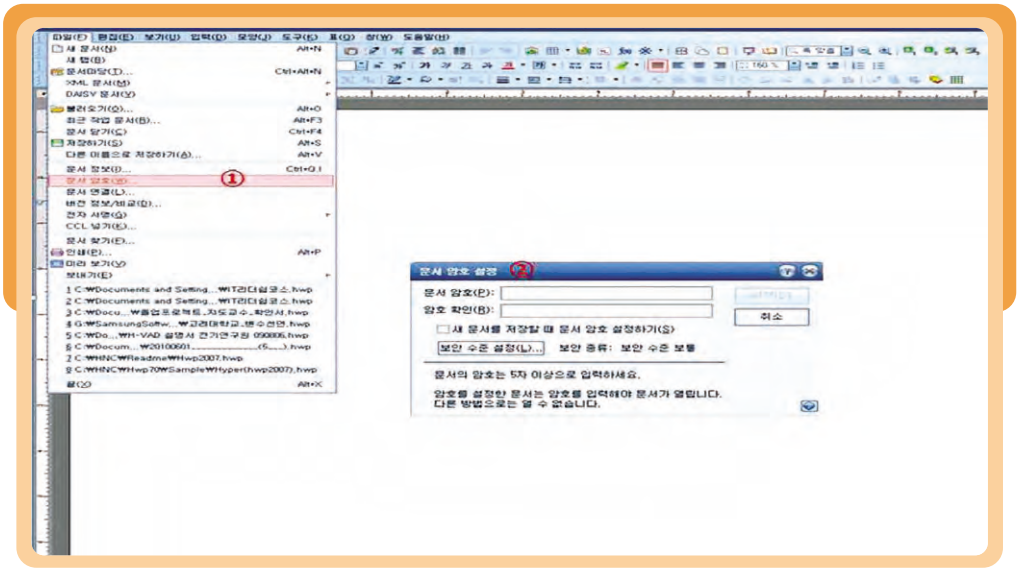
■ 주요 특징

- DB 커널에서 암·복호화 수행하므로 DB 서버의 CPU, 주기억장치, 디스크 등의 추가적인 부하가 적음
- 응용프로그램의 변경이 없으며, ERP 등 패키지에 암호화 적용 가능
- 암호화 테이블과 기존 테이블의 관리 도구 일원화로 운영 편의성 제공
- 비밀번호 일방향 암호화를 위한 암·복호화 모듈의 추가 필요

2.2 업무용 컴퓨터 암호화 사례

■ 한글 2007 문서 암호화 예제

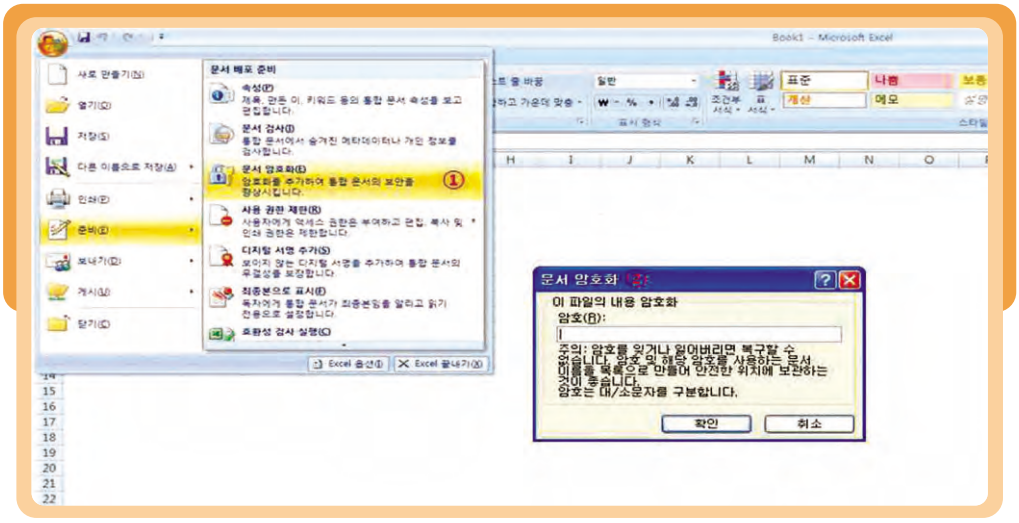
• [파일] → [문서암호] (한글 2010경우: [보안] → [문서암호 설정])



[그림 19]한글 2007을 이용한 문서 암호화 적용

■ MS 엑셀 2007 문서 암호화 예제

• [오피스 단추] → [준비] → [문서암호화] (MS 엑셀 2010의 경우 :
[파일] → [정보] → [통합 문서 보호] → [암호 설정])



[그림 20]MS 엑셀 2007을 이용한 문서 암호화 적용

참고문헌

[1] 정보시스템 개발 · 운영자를 위한 홈페이지 SW(웹) 개발보안 가이드, 안전행정부, 한국인터넷진흥원 / 2012

[2] 홈페이지취약점 점검매뉴얼, 국가사이버안전센터, 국가보안기술연구소 / 2012

[3] 개인정보 암호화 조치 안내서, 안정행정부, 한국인터넷진흥원 / 2012

[4] OWASP Top 10,http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

웹 서버 및 홈페이지 취약점 점검 가이드

2014년 8월 인쇄
2014년 8월 발행
발행처 : 교육부 · 한국교육학술정보원
세종특별자치시 갈매로 408
정부세종청사
대구광역시 동구 동내로 64
한국교육학술정보원
Tel: (053)714-0777

(비매품)