
정보시스템 개발·운영자를 위한

홈페이지 취약점 진단제거 가이드

2013년 12월

< 이 페이지는 빈 페이지입니다. >

목차

제1장 개요	5
제1절 배경	5
제2절 가이드 목적 및 구성	6
제3절 웹 표준 점검 항목	7
 제2장 웹 어플리케이션 취약점	9
1. 운영체제 명령 실행	9
2. SQL 인젝션	13
3. XPath 인젝션	16
4. 정보누출	19
5. 악성콘텐츠	22
6. 크로스 사이트 스크립트(XSS)	24
7. 약한 문자열 강도	29
8. 불충분한 인증 및 인가	33
9. 취약한 비밀번호 복구	36
10. 불충분한 세션 관리	38
11. 크로스 사이트 리퀘스트 변조(CSRF)	42
12. 자동화 공격	46
13. 파일 업로드	49

14. 경로추적 및 파일 다운로드	52
15. 데이터 평문전송	55
16. 쿠키 변조	58
17. URL/파라미터 변조	61

제3장 웹 서버 취약점 63

1. 디렉터리 인덱싱	63
2. 관리자페이지 노출	66
3. 위치공개	68
4. 웹 서비스 메소드 설정 공격	70

부 록 72

제1절 웹 시스템 구동 개요	72
제2절 웹 소스코드 보안약점	75
제3절 웹 서버 보안 관리	80
제4절 용어정리	81
제5절 참고문헌	84

제1장 개요

제1절 배경

최근 사이버 공격은 주로 웹 사이트를 공격하는 경우가 대부분인데, 개인정보가 유출된 트위터, 비자·마스터 카드 등이 대표적인 사례이다. 이 외에도 미국 CIA나 국무성 등 주요 정부기관의 홈페이지도 해킹 공격을 당한 사례가 있어, 웹 사이트(홈페이지)에 대한 정보보호 중요성이 커지고 있다.

우리나라에서는 IT 신기술을 활용해 대부분의 서비스를 인터넷을 통해 제공하고 있으며 정보시스템 개발 시 필요한 기능들을 미리 구현해 둔 소프트웨어인 표준틀(프레임워크)이 민간기업으로 빠르게 확산되고 있는 것으로 보아 앞으로도 인터넷을 활용한 서비스가 확대될 것은 자명한 일이다. 그러나 인터넷을 활용한 서비스를 확대하는데 있어서 보안취약점 또는 해킹 공격은 커다란 걸림돌이며, 취약점을 잘 아는 해커들로부터 우리나라 정보시스템이 자유로울 수 없다.

본 가이드는 기존에 운영 중인 웹 사이트(홈페이지) 관리자 또는 보안담당자가 직접 따라 할 수 있도록 쉽게 작성된 홈페이지 취약점 진단·제거 가이드로, 웹 어플리케이션 및 웹 서버에 숨어있는 대표적인 보안취약점에 대해 사전 점검을 수행할 수 있는 가이드를 말한다. 본 가이드를 통해서 정보시스템 관리자 또는 보안담당자의 업무 능력을 향상시키고 지속적으로 발생하는 보안취약점을 미리 제거해 웹 서비스의 안전성 및 신뢰성을 확보하는데 기여하기를 바란다.

제2절 가이드 목적 및 구성

국가의 사회기반 전반에 ICT(정보통신기술) 의존도가 확산되면서 사이버 피해가 급증하는 추세이다. 이에 따라 사이버 위협은 사회 안정과 국가 안위로 직결될 우려 또한 증대되고 있으며, 문제점의 해결 및 지속적인 정보보호 수준 제고가 필요한 실정이다.

본 가이드는 21개의 웹 취약점 항목에 대한 개요 및 파급효과, 취약점 대응방안을 주로 다루고 있으며 웹 어플리케이션의 보안 취약점을 파악하고 지속적인 보안 체계 마련을 위한 방향성을 제시하여 안전한 웹 서비스를 운영할 수 있도록 함에 그 목적을 둔다.

취약점 항목의 경우 '홈페이지 SW(웹) 개발보안 가이드'와 '주요정보통신기반시설 취약점 분석·평가 기준' 항목을 바탕으로 구성하였으며 6개의 항목이 3개의 항목으로 통합하여 새롭게 재구성 하였으며, 통합된 항목들은 아래와 같다.

- ① 다운로드 항목과 경로추적 항목의 경우 진단 방법이 유사하며 높은 연관성을 가지고 있으므로 항목을 통합
- ② 세션 예측 항목과 불충분한 세션관리 항목의 경우 진단의 매개체가 세션으로 동일하므로 대응방안 측면에서 하나로 관리할 수 있으므로 항목 통합
- ③ 불충분한 인증 항목과 불충분한 인가 항목의 경우 두 항목 모두 사용자에게 대한 접근 미비로 인해 발생하는 취약점으로 대응방안 측면에서 하나로 관리할 수 있으므로 항목 통합

URL/파라미터 변조 항목의 경우 민간, 공공 대상 파라미터 변조를 통한 침해사고 사례가 다수 발생되고 있는 추세이며, 권한 검증 외 사용자 입력 값에 대해 검증 누락이 발생하는 모든 상황을 포함하는 취약점으로 심각한 경우 홈페이지 장악이 가능할 수 있기 때문에 새로운 항목으로 추가하여 조치할 수 있도록 구성 하였다.

제3절 웹 표준 점검 항목

단계	코드	취약점 항목	공격 피해	대응 방안
웹 어플리케이션	OC	운영체제 명령 실행	시스템 장악	p11
	SI	SQL 인젝션	DB 정보 유출	p14
	XI	XPath 인젝션	사용자 인증 우회	p18
	IL	정보누출	서버 정보 노출	p21
	CS	악성콘텐츠	악성코드 감염	p23
	XS	크로스 사이트 스크립트(XSS)	세션하이재킹 악성코드 전파	p26
	BF	약한 문자열 강도	사용자 계정 탈취	p31
	IN	불충분한 인증 및 인가	관리자 권한 탈취	p34
	PR	취약한 비밀번호 복구	사용자 계정 탈취	p37
	SM	불충분한 세션 관리	사용자 권한 탈취	p40
	CF	크로스 사이트 리퀘스트 변조(CSRF)	사용자 권한 탈취	p44
	AU	자동화 공격	시스템 과부하	p47
	FU	파일 업로드	시스템 장악	p50
	FD	경로추적 및 파일 다운로드	웹 서버 정보 노출	p54
	SN	데이터 평문전송	중요 정보 노출	p56
	CC	쿠키 변조	사용자 권한 탈취	p59
	UP	URL/파라미터 변조	사용자 권한 탈취	p62

웹 서 버	DI	디렉터리 인덱싱	시스템 파일 노출	p65
	AE	관리자페이지 노출	웹 사이트 정보 노출	p67
	PL	위치공개	웹 사이트 정보 노출	p69
	MS	웹 서비스 메소드 설정 공격	시스템 장악	p71

- ※ 가이드 점검 항목 구성은 '홈페이지 SW(웹) 개발보안 가이드'와 '주요정보통신기반시설 취약점 분석·평가 기준' 항목을 기반으로 통합 진단/관리를 위한 항목 통·폐합 및 신규 항목 추가를 통해 21개 항목으로 재구성
- ※ 세부적인 구성은 21개 취약점 항목별로 1. 개요, 2. 판단기준, 3. 대응방안으로 구성되어 있으며, 특히 대응방안의 경우 웹 어플리케이션, 웹 서버, 보안장비 등 3가지 부문으로 나누어 해당 취약점의 조치방법을 다양한 측면에서 제시하여 2-3중 보안체계를 구성할 수 있도록 구성
- ※ 대응방안의 관점에서 웹 어플리케이션(소스코드) 17개 항목과 웹 서버 4개 항목으로 세분화

제2장 웹 어플리케이션 취약점

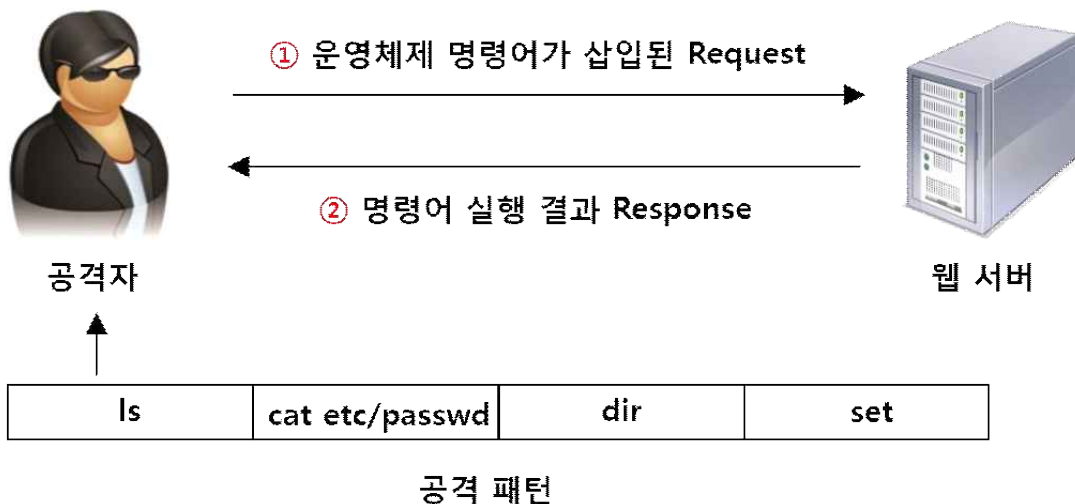
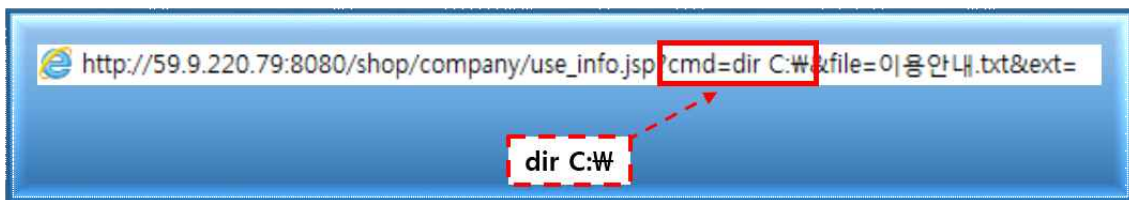
1. 운영체제 명령 실행

가. 개요

웹 어플리케이션에서 `system()`, `exec()`와 같은 시스템 명령어를 실행시킬 수 있는 함수를 제공하며 사용자 입력 값에 대한 필터링이 제대로 이루어지지 않을 경우 공격자가 운영체제 시스템 명령어를 호출하여 백도어 설치나 관리자 권한 탈취 등 시스템 보안에 심각한 영향을 미칠 수 있는 취약점

■ 파급효과

시스템 계정 정보 유출, 백도어 설치, 관리자 권한 탈취, 시스템 명령어 실행



[운영체제 명령 실행 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	URL	URL/파라미터에 ls, cat 등의 명령어 삽입(리눅스 계열)	운영체제 명령어 실행 결과 출력
		URL/파라미터에 dir, ipconfig 등의 명령어 삽입(윈도우 계열)	

※ URL(각종 파라미터) : 파라미터는 해당 사이트의 URL에서 아래와 같은 부분을 말하며 공격자는 여러 파라미터 중 임의로 몇 가지의 파라미터에 점검을 한 후 취약여부를 판단함

■ 예시

http://www.test.or.kr/board.php?search=content&login=yes&id=test

다. 대응방안

- ① 웹 방화벽 : 모든 사용자 입력 폼(로그인 폼, 검색 폼, URL 등)을 대상으로 특수문자, 특수구문 필터링 규칙 적용

필터링 대상				
	;	type	cat	dir
ls	ipconfig	ifconfig	set	netstat

- ② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용

■ 패턴 기법을 통한 명령어 실행 예 JAVA

```

1: // 허용 가능한 명령어
2: String allowCmd[] = {"type"};
3: // 서버로 전달된 파라미터를 변수에 저장
4: String cmdStr = request.getParameter("cmd");
5: // 허용 가능한 명령어가 맞는지 확인
6: for ( int n=0; n<allowCmd.length; n++ ) {
7:     // 허용 가능한 명령어가 존재할 경우
8:     if ( cmdStr.contentEquals(allowCmd[n]) ) {
9:         try {
10:            // 명령어 실행
11:            Process op = new ProcessBuilder("cmd", "/c", cmdStr).start();
12:            ... ..
13:        } catch(Exception e) { ... .. }
14:    }
15: }
```

■ switch 문을 이용한 명령어 실행 예 JAVA

```

1:  // 서버로 전달된 파라미터를 변수에 저장
2:  String in_code = request.getParameter("cmd");
3:  // 파라미터 값 확인 후 허용된 명령어만 실행 가능하도록 제어
4:  switch ( in_code ) {
5:      case "001":
6:          comm_exec = "dir";
7:          break;
8:      case "002":
9:          comm_exec = "netstat";
10:         break;
11:     default :
12:         comm_exec = "n/a";
13:         break;
14: }
15: // 허용 가능한 명령어가 존재할 경우
16: if (comm_exec != "n/a") {
17:     String cmd = new String("cmd.exe /c");
18:     // 명령어 실행
19:     Runtime.getRuntime().exec(cmd + " " + comm_exec);
20: }

```

웹 어플리케이션 운영 시 소스코드 상에 운영체제 명령어를 사용하지 않는 것이 바람직하나 부득이하게 운영체제 명령어를 사용해야 한다면 상기 예제들과 같이 허용 가능한 명령어를 지정하여 지정된 명령어만 실행할 수 있도록 설정해야 함

③ 웹 서버 보안 설정 : 해당사항 없음

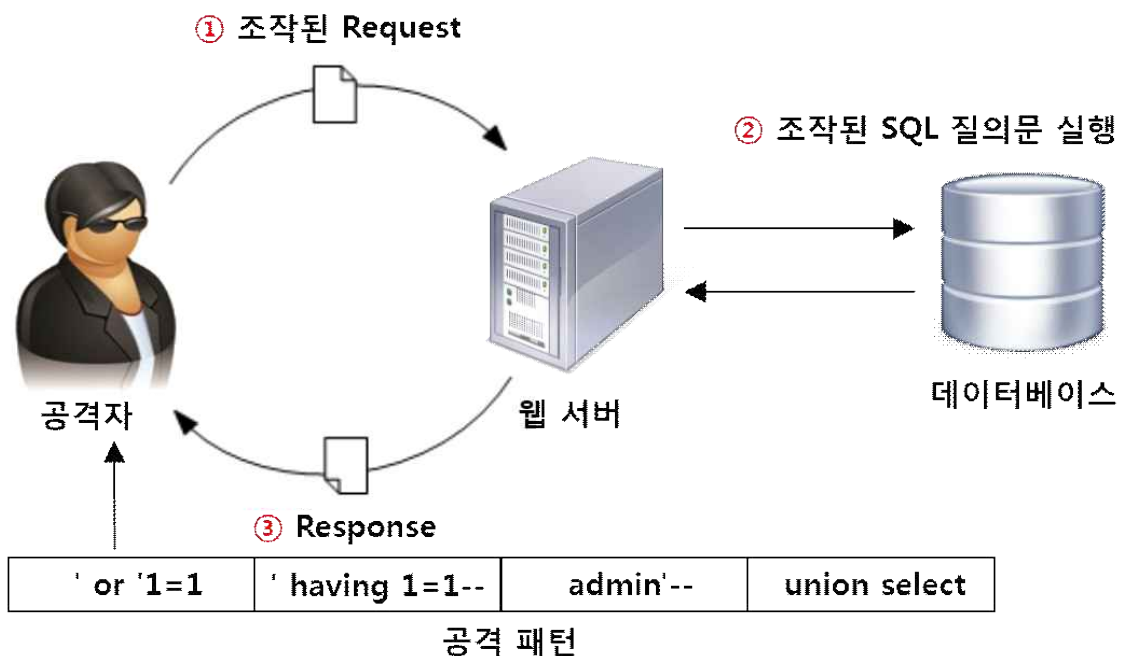
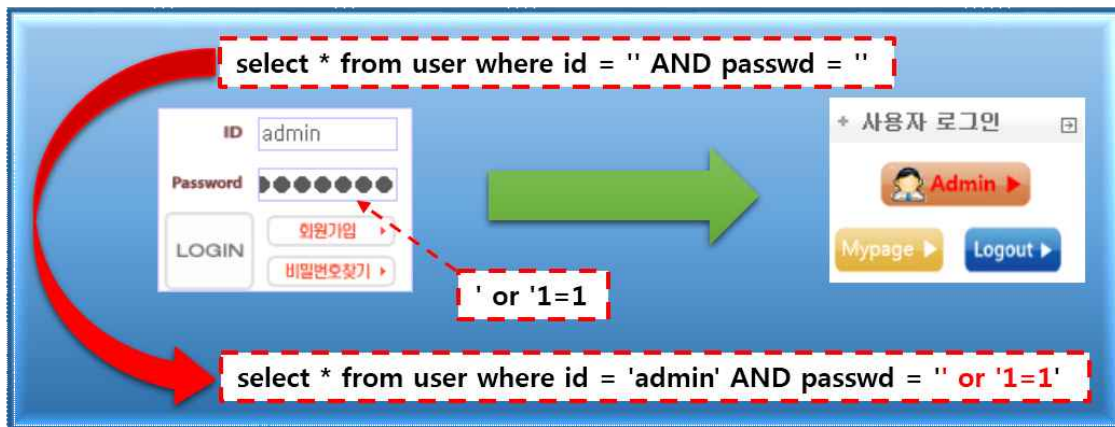
2. SQL 인젝션

가. 개요

데이터베이스와 연동된 웹 어플리케이션에서 SQL 질의문에 대한 필터링이 제대로 이루어지지 않을 경우 공격자가 입력이 가능한 폼(웹 브라우저 주소입력창 또는 로그인 폼 등)에 조작된 질의문을 삽입하여 웹 서버의 데이터베이스 정보를 열람 또는 조작을 할 수 있는 취약점

■ 파급효과

데이터베이스 정보노출, 데이터 삽입, 삭제 및 변경, 데이터베이스 서비스 중지, 사용자 인증 우회



[SQL 인젝션 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	URL 검색 폼	['], ["], [;], [#], [--] 입력 [' having 1=1 --] 입력	DB 에러 문구 발생
②	URL 검색 폼	[' and 'a'='a], [' and 'a'='b] 입력 후 결과 비교	두 입력에 대해 서로 다른 반응
③	로그인 폼	[' or '1=1] 입력	로그인 성공

※ URL(각종 파라미터) : 파라미터는 해당 사이트의 URL에서 아래와 같은 부분을 말하며 공격자는 여러 파라미터 중 임의로 몇 가지의 파라미터에 점검을 한 후 취약여부를 판단함

■ 예시

http://www.test.or.kr/board.php?search=content&login=yes&id=test

다. 대응방안

- ① 웹 방화벽 : 모든 사용자 입력 폼(로그인 폼, 검색 폼, URL 등)을 대상으로 특수문자, 특수구문 필터링 규칙 적용

필터링 대상				
'	"	--	#	=
()	*/	/*	+
<	>	&&		%
union	select	insert	from	where
update	drop	if	join	decalre
and	or	column_name	table_name	openrowset
substr	substring	xp_	sysobjects	syscolumns

② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용

■ 바인딩된 질의문 사용 예 JAVA

```

1: // JAVA PreparedStatement 객체 생성
2: PreparedStatement stmt = connection.prepareStatement("SELECT *
   FROM users WHERE idx=? AND userid=? AND password=?");
3: // setInt 메소드를 통한 바인딩질의 사용
4: stmt.setInt(1, user_idx);
5: // setString 메소드를 통한 바인딩질의 사용
6: stmt.setString(2, user_id);
7: stmt.setString(3, user_password);
8: ResultSet rs = stmt.executeQuery();

```

JAVA의 PreparedStatement 클래스에 존재하는 setString 혹은 setInt 메소드를 통해 바인딩된 질의문을 활용하여 질의문의 구조가 변경되지 않도록 해야 함
 질의문에 필요한 변수를 직접 삽입하지 않고 바인딩이라는 과정을 통해 실행할 경우 SQL 인젝션 공격을 예방할 수 있음

■ MyBatis 사용 시 안전한 질의문 예

```

1: // 바인딩된 질의문을 사용하기 위한 질의문 선언
2: insert into USER (id, name, email, phone) values (#{id}, #{name},
   #{email}, #{phone})
3: update USER set name = #{name}, email = #{email}, phone =
   #{phone} where id = #{id}
4: delete from USER where id = #{id}

```

#{} 구문을 사용하면 기본적으로 PreparedStatement 파라미터를 생성하고 그 값을 대입하므로 자동으로 바인딩된 질의문을 사용할 수 있음

부득이하게 **\${}** 구문을 사용해야 할 경우 질의문에 사용자 입력 값을 인자로 사용하지 않거나 적절한 검증 또는 예외 처리 과정이 필요함

③ 웹 서버 보안 설정 : 해당사항 없음

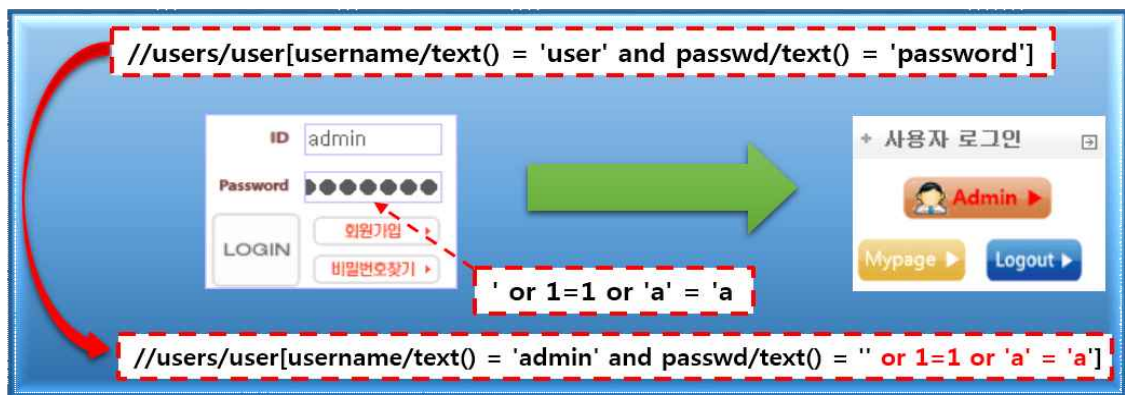
3. XPath 인젝션

가. 개요

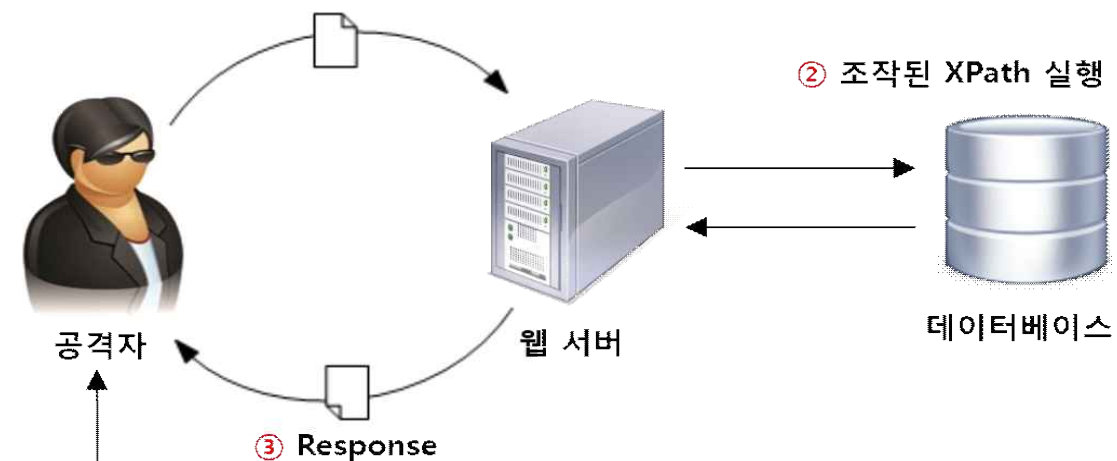
데이터베이스와 연동된 웹 어플리케이션에서 XPath 및 XQuery 질의문에 대한 필터링이 제대로 이루어지지 않을 경우 공격자가 입력이 가능한 폼(웹 브라우저 주소입력창 또는 로그인 폼 등)에 조작된 질의문을 삽입하여 인증 우회를 통해 XML 문서로부터 인가되지 않은 데이터를 열람 할 수 있는 취약점

■ 파급효과

사용자 인증 우회, 사용자 정보노출, 웹 사이트 로직 손상



① 조작된 Request



' or '1'='1	' or 1=1 or 'a' = ' a
] * user[@role='admin	something' or ' 1 ' = ' 1

공격 패턴

[XPath 및 XQuery 인젝션 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	로그인 폼	['] * user[@role='admin] 입력	계정 및 개인 정보 노출
②	로그인 폼	[something' or '1'='1] 입력	계정 및 개인 정보 노출

※ URL(각종 파라미터) : 파라미터는 해당 사이트의 URL에서 아래와 같은 부분을 말하며 공격자는 여러 파라미터 중 임의로 몇 가지의 파라미터에 점검을 한 후 취약여부를 판단함

■ 예시

http://www.test.or.kr/board.php?search=content&login=yes&id=test

다. 대응방안

- ① 웹 방화벽 : 모든 사용자 입력 폼(로그인 폼, 검색 폼, URL 등)을 대상으로 특수문자, 특수구문 필터링 규칙 적용

필터링 대상				
'	,	=	()
[]	/	:	*

- ② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용

■ 패턴 기법을 통한 인젝션 방어 예 JAVA

```

1: // XPath 인젝션 시 사용되는 문자 패턴
2: String CharList = "()= '[]:*/ ";
3: // 서버로 전달된 파라미터를 변수에 저장
4: String xStr = request.getParameter("query");
5: // 인코딩 우회를 방지하기 위한 디코딩
6: String dValue = URLDecoder.decode(xStr, Charset.defaultCharset().name());
7: // 인젝션 문자가 존재하는지 확인
8: for ( char c : dValue.toCharArray() ) {
9:     if ( CharList.indexOf(c) != -1 ) { // 인젝션 문자가 존재할 경우
10:         ... // 에러 메시지 출력
11:         break;
12:     }
13: }
```

서버로 전달되는 사용자 입력 값 검증 시 XPath 인젝션에 주로 사용되는 문자(' , " , [등)들을 리스트로 지정하여 해당하는 문자가 존재할 경우 질의문 실행이 불가능하도록 설정해야 함

또한 URLDecoder 클래스에 존재하는 decode 메소드를 통해 URL 인코딩이 적용된 사용자 입력 값을 디코딩함으로써 우회공격을 차단할 수 있음

- ③ 웹 서버 보안 설정 : 해당사항 없음

4. 정보누출

가. 개요

웹 어플리케이션의 민감한 정보가 개발자의 부주의로 인해 노출되는 것으로 중요 정보(관리자 계정 및 테스트 계정 등)를 주석구문에 포함시켜 의도하지 않게 정보가 노출되는 취약점

또한 디폴트로 설정된 에러 페이지를 그대로 사용할 경우 시스템 내부 문제점을 자세하게 출력해주기 때문에 절대경로, 상태코드, 데이터베이스 종류, 질의문 등이 노출될 수 있으며 이밖에도 공격자가 검색엔진을 통하여 각종 개인 정보 및 서버 정보 등 해킹에 필요한 정보를 획득할 수 있음

■ 파급효과

개발 이력 노출, 테스트 및 관리자 계정 노출, 데이터베이스 정보 노출



[정보누출 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	웹 페이지 전체	웹 페이지 소스보기	중요정보 노출
②	URL	URL에 웹 서버 디렉터리 명 입력	서버정보 노출

※ URL(각종 파라미터) : 파라미터는 해당 사이트의 URL에서 아래와 같은 부분을 말하며 공격자는 여러 파라미터 중 임의로 몇 가지의 파라미터에 점검을 한 후 취약여부를 판단함

■ 예시

http://www.test.or.kr/board.php?search=content&login=yes&id=test

다. 대응방안

- ① 웹 방화벽 : 해당사항 없음
- ② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용

모든 웹 페이지에 대해 개발단계에서 디버깅 및 테스트를 목적으로 작성한 주석구문에 서버 주요 정보가 포함되어 있을 경우 공격자가 해당 정보를 다른 취약점과 연계해 사용할 수 있으므로 제거해야 함

- ③ 웹 서버 보안 설정 : 미흡한 웹 서버 보안 설정 변경

■ 안전한 서버 설정 예 Tomcat

```

1: <web-app>
2:   <error-page>
3:     <error-code>404</error-code>
4:     <location>/error.jsp</location>
5:   </error-page>
6:   <error-page>
7:     <error-code>403</error-code>
8:     <location>/error.jsp</location>
9:   </error-page>
10:  <error-page>
11:    <error-code>500</error-code>
12:    <location>/error.jsp</location>
13:  </error-page>
14: </web-app>

```

전체적인 통합 에러 페이지를 작성한 후 모든 에러코드에 대해 통합 에러 페이지로 리다이렉트 되도록 설정하여 공격자가 서버정보 및 에러코드를 수집할 수 없도록 설정해야 함

5. 악성콘텐츠

가. 개요

웹 어플리케이션에서 사용자 입력 값에 대한 필터링이 제대로 이루어지지 않을 경우 공격자가 악성콘텐츠를 삽입할 수 있으며, 악성콘텐츠가 삽입된 페이지에 접속한 사용자는 악성코드 유포 사이트가 자동으로 호출되어 악성코드에 감염될 수 있는 취약점

악성콘텐츠는 SQL 인젝션, Cross Site Script, 파일 업로드를 통한 페이지 위변조 기법 등을 통해 삽입이 가능하므로 해당 취약점을 반드시 제거하여 악성콘텐츠 삽입이 불가능 하도록 조치가 필요함

■ 파급효과

악성코드 감염, 웹 페이지 변조



나. 판단기준

순번	점검위치	행위	취약반응
①	XSS 발생 지점	<script src="악성콘텐츠 주소"> </script> 입력 <iframe src="악성콘텐츠 주소"> </iframe> 입력	악성콘텐츠 삽입

다. 대응방안

- ① 웹 방화벽 : [제2장 6. 크로스 사이트 스크립트(XSS) p.39] 참고
- ② 웹 어플리케이션 : [제2장 6. 크로스 사이트 스크립트(XSS) p.39] 참고
- ③ 웹 서버 보안 설정 : 해당사항 없음

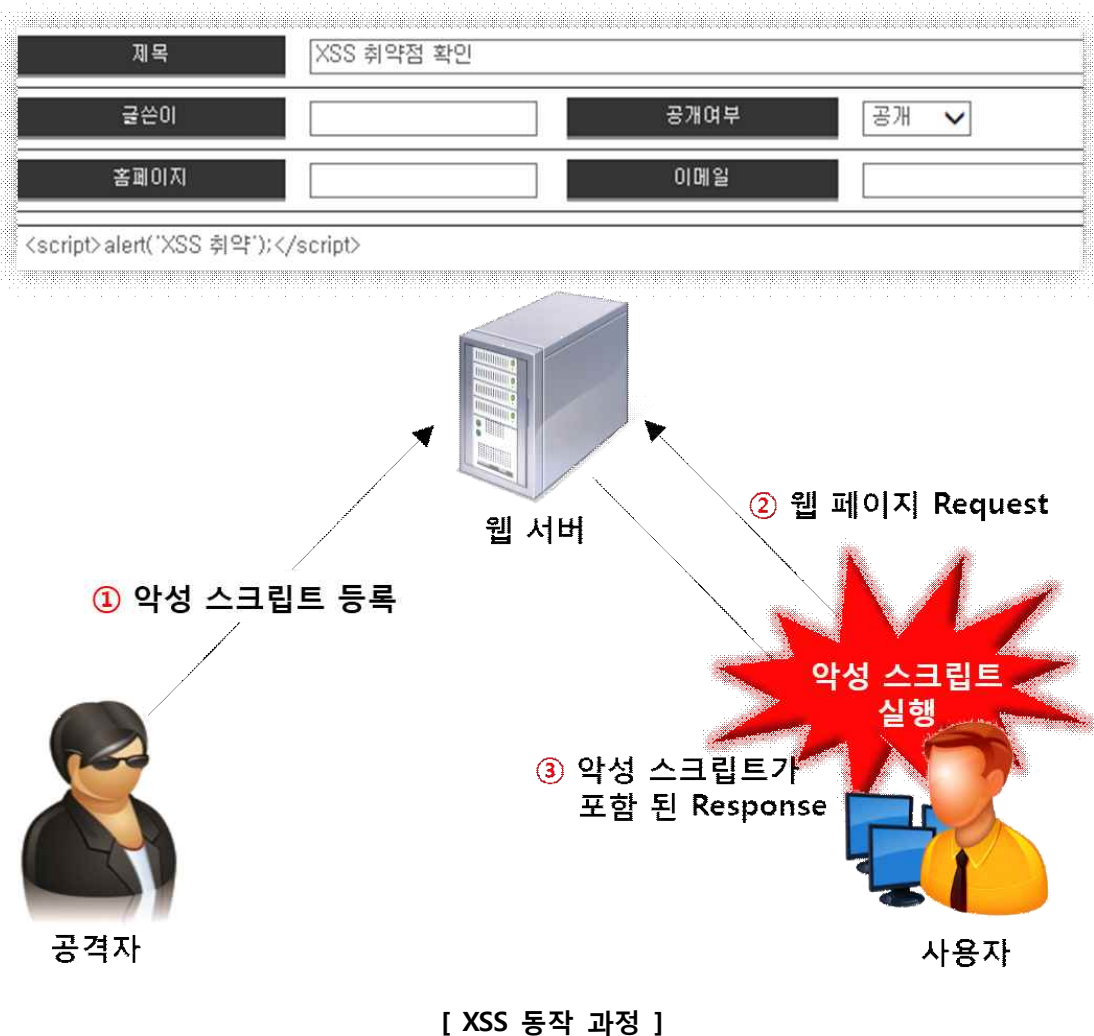
6. 크로스 사이트 스크립트(XSS)

가. 개요

웹 어플리케이션에서 사용자 입력 값에 대한 필터링이 제대로 이루어지지 않을 경우, 공격자가 입력이 가능한 품(웹 브라우저 주소입력 또는 게시판 등)에 악의적인 스크립트를 삽입하여 사용자 세션 도용, 악성코드를 유포할 수 있는 취약점

■ 파급효과

사용자의 개인정보 및 쿠키정보 탈취, 악성코드 감염, 웹 페이지 변조



나. 판단기준

순번	점검위치	행위	취약반응
①	사용자 입력 폼	<script>alert()</script> 입력	팝업창 발생
②	URL 검색 폼	<script>alert()</script> 입력	팝업창 발생

※ URL(각종 파라미터) : 파라미터는 해당 사이트의 URL에서 아래와 같은 부분을 말하며 공격자는 여러 파라미터 중 임의로 몇 가지의 파라미터에 점검을 한 후 취약여부를 판단함

■ 예시

http://www.test.or.kr/board.php?search=content&login=yes&id=test

다. 대응방안

- ① 웹 방화벽 : 모든 사용자 입력 폼(로그인 폼, 검색 폼, URL 등)을 대상으로 특수문자, 특수구문 필터링 규칙 적용

필터링 대상			
<	>	onstop	layer
javascript	eval	onactivae	onfocusin
applet	document	onclick	onkeydown
xml	create	onbeforecut	onkeyup
link	binding	ondeactivate	onload
script	msgbox	ondragend	onbounce
object	embed	ondragleave	onmovestart
frame	applet	ondragstart	onmouseout
ilayer	javascript	onerror	onmouseup
bgsound	href	embed	onabort
base	onstart	onfocus	onmovestart
onmove	onrowexit	onunload	onsubmit
innerHTML	onpaste	ondblclick	vbscript
charset	onresize	ondrag	expression
string	onselect	ondragenter	onchange
append	onscroll	ondragover	meta
alert	title	ondrop	void
refresh	iframe	oncopy	oncut
ilayer	blink	onfinish	frameset
cookie	style	onreset	onselectstart

② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩

■ 특수문자 치환 예 JAVA

```

1: // 서버로 전달된 게시판 내용(content)을 checkContent 변수에 저장
2: String checkContent = request.getParameter("content");
3: // 게시판 내용이 존재할 경우
4: if ( checkContent != null ) {
5:     // HTML 태그가 실행되지 않도록 문자열 치환
6:     checkContent = checkContent.replaceAll("<", "&lt;");
7:     checkContent = checkContent.replaceAll(">", "&gt;");
8:     ... ..
9:     // 허용할 HTML 태그만 실행 가능하도록 변경
10:    checkContent = checkContent.replaceAll("&lt;P&gt;", "<P>");
11: }
12: else { ... .. }

```

JAVA의 내장 함수인 replaceAll 메소드를 이용하여 "<", ">"와 같이 HTML 태그 실행에 사용되는 문자열을 치환하여 공격자가 삽입한 악의적인 스크립트가 실행되지 않도록 설정해야 함

부득이하게 사용해야 될 HTML 태그가 존재한다면 상기 예제와 같이 replaceAll 메소드를 사용하여 치환된 문자열을 다시 원래의 태그로 치환함으로써 HTML 태그가 실행 가능하도록 설정 하는 것을 권장함

변환 대상	변환 값	변환 대상	변환 값
<	<))
>	>	#	#
((&	&
"	"	'	'
/	/		

[치환이 필요한 특수문자]

■ 패턴 기법을 통한 XSS 방어 예 JAVA

```

1: // XSS에 사용되는 문자 패턴
2: String xssCharList = "<>";
3: // 서버로 전달된 파라미터를 변수에 저장
4: String xssStr = request.getParameter("contents");
5: // 인코딩 우회를 방지하기 위한 디코딩
6: String decodeValue = URLDecoder.decode(xssStr,
    Charset.defaultCharset().name());
7: // XSS에 사용되는 문자가 존재하는지 확인
8: for ( char c : decodeValue.toCharArray() ) {
9:     // 문자가 존재할 경우
10:    if ( xssCharList.indexOf(c) != -1 ) {
11:        ... .. // 에러 메시지 출력
12:        break;
13:    }
14: }

```

서버로 전달된 사용자 입력 값 검증 시 XSS에서 주로 사용되는 문자(<, > 등)들을 리스트로 지정하여 해당 문자가 존재할 시 삽입이 불가능하도록 설정해야 함

또한 URLDecoder 클래스에 존재하는 decode 메소드를 통해 URL 인코딩이 적용된 사용자 입력 값을 디코딩함으로써 우회공격을 차단할 수 있음

③ 웹 서버 보안 설정 : 해당사항 없음

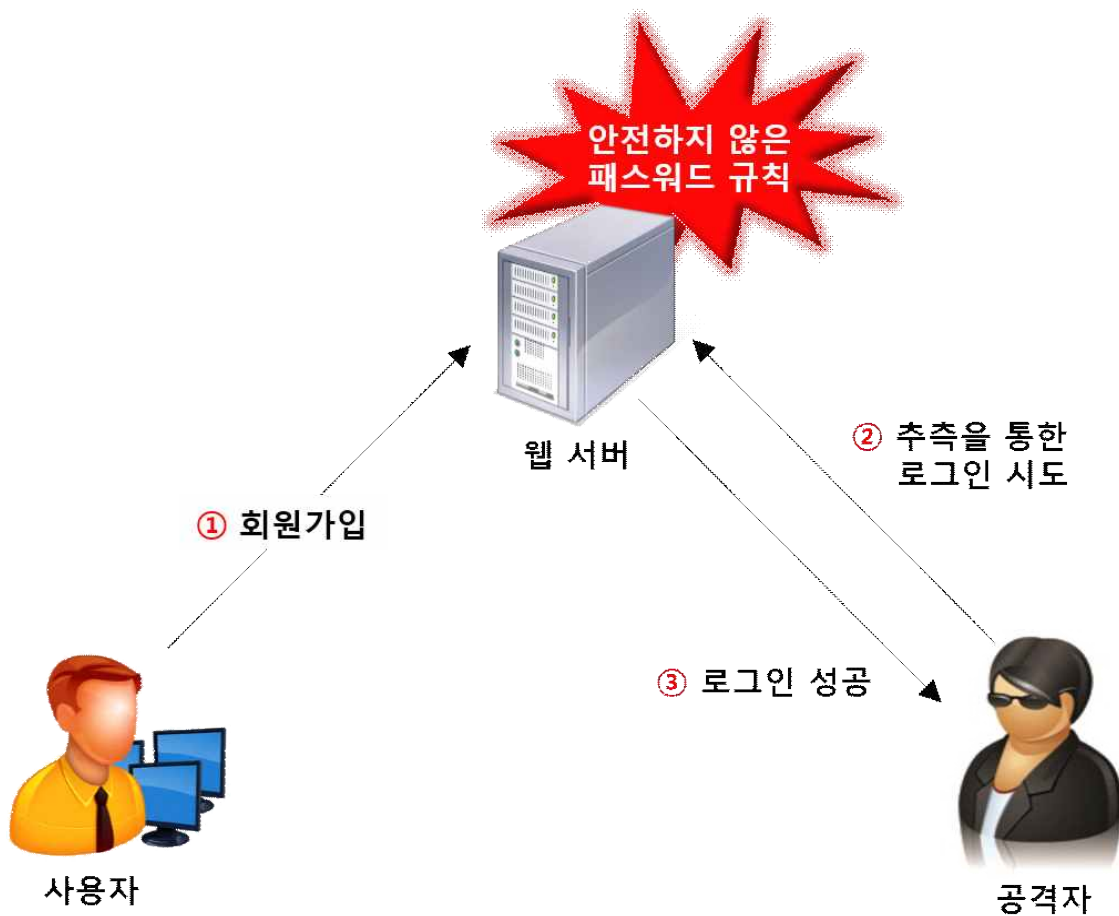
7. 약한 문자열 강도

가. 개요

웹 어플리케이션에서 회원가입 시 안전한 비밀번호 규칙이 적용되지 않아 취약한 비밀번호로 회원가입이 가능할 경우 공격자가 추측을 통한 대입 및 주변 정보를 수집하여 작성한 사전파일 통한 대입을 시도하여 사용자의 비밀번호를 추출할 수 있는 취약점

■ 파급효과

추측을 통한 사용자 권한 탈취



[약한 문자열 강도 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	로그인 폼	추측 가능한 계정 및 패스워드를 입력하여 로그인 시도	로그인 성공

다. 대응방안

- ① 웹 방화벽 : 해당사항 없음
- ② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용

■ 안전한 패스워드 규칙 적용 예 JAVA

```

1:  // 비밀번호 복합도 상태 저장
2:  boolean isValidChar = false;
3:  boolean isValidNum = false;
4:  // 서버로 전달된 파라미터를 변수에 저장
5:  String pwStr = request.getParamet("passwd");
6:  // 허용 가능한 숫자 설정
7:  String validNum = "0123456789";
8:  // 허용 가능한 특수문자 설정
9:  String validChar = "!@%$&^*";
10: // 비밀번호에 숫자가 존재하는지 확인
11: for ( char c : pwStr.toCharArray() ) {
12:     if ( validNum.indexOf(c) != -1 ) {           // 숫자가 존재할 경우
13:         isValidNum = true;
14:         break;
15:     }
16: }
17: // 비밀번호에 특수문자가 존재하는지 확인
18: for ( char c : pwStr.toCharArray() ) {
19:     if ( validChar.indexOf(c) != -1 ) {         // 특수문자가 존재할 경우
20:         isValidChar = true;
21:         break;
22:     }
23: }
24: // 비밀번호에 특수문자와 숫자가 존재할 경우
25: if ( validChar && validNum ) {
26:     ... ..                                     // 정상적인 회원가입 처리
27: }

```

비밀번호 설정 시 영문자, 숫자, 특수문자가 반드시 하나씩 존재하도록 설정하는 것을 권장하며 아래와 같은 사항들을 주기적으로 점검하여 사용자가 취약한 패스워드를 사용할 수 없도록 패스워드 규칙을 선정하여 안내하는 것을 권장함

- 1) admin, master 등 유추하기 쉬운 계정 이름 사용 금지
- 2) 간단한 문자(영어단어 포함)나 숫자의 연속사용은 금함
- 3) 키보드 상에서 일련화 된 배열을 따르는 패스워드 선택 금지
- 4) 사전에 있는 단어, 이를 거꾸로 철자화한 단어 또는 숫자 하나를 더한 단어 사용 금지
- 5) 생일, 전화번호 개인정보 및 아이디와 비슷한 추측하기 쉬운 비밀번호는 사용하지 않도록 함
- 6) 패스워드를 주기적으로 변경해야 함
- 7) 패스워드의 최소 사용기간, 최대 사용기간을 설정함
- 8) 이전에 사용한 패스워드는 재사용 금지
- 9) 계정 잠금 정책 설정 ex) 로그인 5회 실패 시 30분 동안 사용 중지

[안전한 계정 설정 항목]

또한 로그인 시 아이디와 패스워드에 따라 다르게 출력되는 팝업창으로 인해 사용자의 계정 존재유무를 확인할 수 있으므로 "회원 정보가 올바르지 않습니다."와 같은 일관성 있는 팝업창을 출력해야 함

③ 웹 서버 보안 설정 : 해당사항 없음

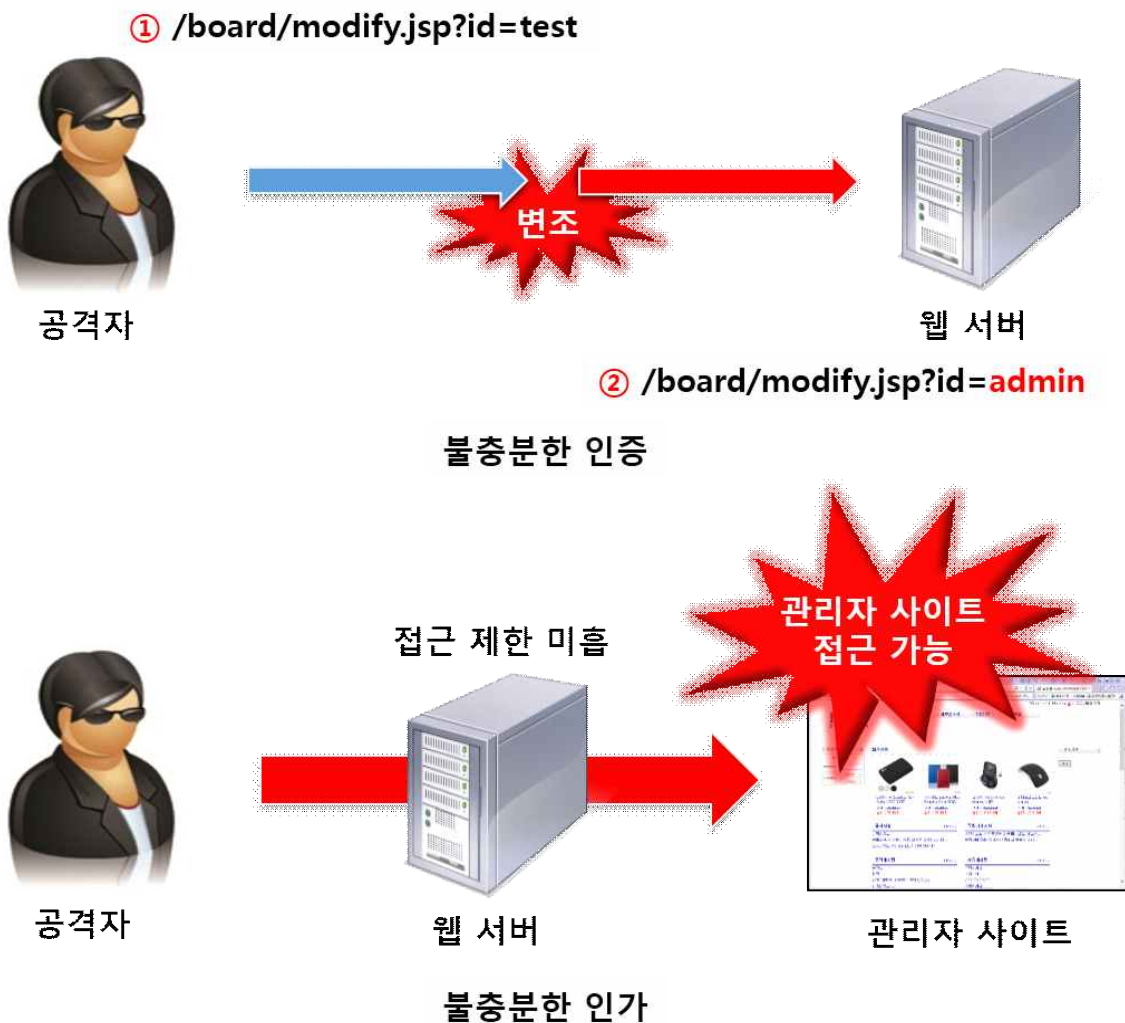
8. 불충분한 인증 및 인가

가. 개요

웹 어플리케이션에서 개인정보 수정 페이지나 통합 로그인(SSO)과 같은 곳에서 사용자 인증이 미흡(아이디로 인증)할 경우 공격자가 파라미터로 전달되는 값을 수정하여 사용자 도용 및 개인정보 노출 문제가 발생할 수 있는 취약점 또한 관리자 권한을 가지고 있는 페이지에 대해 접근 제한을 설정하지 않았을 경우 공격자가 해당 페이지로 접근하여 조작이 가능한 문제가 발생 할 수 있음

■ 파급효과

개인정보 노출, 사용자 인증 우회, 사용자 및 관리자 권한 도용, 관리자 권한 획득



[불충분한 인증 및 인가 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	URL	웹 프록시 도구를 사용하여 각종 파라미터 변조	인증 우회를 통한 사용자 페이지 접근
②	URL	페이지 명을 추측하여 URL 강제 브라우징	관리자 기능 페이지 접근

※ URL(각종 파라미터) : 파라미터는 해당 사이트의 URL에서 아래와 같은 부분을 말하며 공격자는 여러 파라미터 중 임의로 몇 가지의 파라미터에 점검을 한 후 취약여부를 판단함

■ 예시

http://www.test.or.kr/board.php?search=content&login=yes&id=test

다. 대응방안

- ① 웹 방화벽 : 해당사항 없음
- ② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용

■ 세션을 이용한 사용자 인증 예 JAVA

```

1: // 서버로 전달된 파라미터를 변수에 저장
2: String userID = request.getParameter("userid");
3: // 사용자의 세션 정보를 반환
4: HttpSession session = request.getSession(true);
5: // 세션에 존재하는 userID 정보를 반환
6: String sessionID = session.getAttribute("userID");
7: // 세션 정보와 파라미터 값을 비교 (사용자 검증)
8: if ( !sessionID.equals(userID) ) {
9:     ... .. // 에러 메시지 출력 (인증 실패)
10: }
```

공격자가 파라미터 변조를 통해 권한 없는 사용자의 인증을 요청할 경우 로그인 한 사용자들의 아이디 정보를 세션에 저장하여 세션에 저장된 아이디 값과 전달된 파라미터 값을 비교하여 사용자 검증(인증)을 할 수 있도록 설정해야 함

■ IP 접근 제한 예 JAVA

인가된 사용자만 접근할 수 있는 페이지 (관리자페이지)

```

1: // 사용자의 IP 정보를 반환
2: String userIP = request.getRemoteAddr();
3: // 사용자의 IP가 접근 가능한 IP인지 비교 (IP 접근 제한)
4: if ( !userIP.equals("192.168.1.2") ) {
5:     ... .. // 에러 메시지 출력 (비인가 사용자)
6: }
```

인가된 사용자만 접근이 가능한 페이지(관리자페이지)의 경우 상기 예제와 같이 허용된 IP만 접근할 수 있도록 설정하여 비인가 된 사용자는 접근할 수 없도록 관리해야 함

③ 웹 서버 보안 설정 : 해당사항 없음

9. 취약한 패스워드 복구

가. 개요

웹 어플리케이션에 존재하는 비밀번호 찾기 기능 또는 관리자에 의한 임시 비밀번호 발급 시 사용자 인증이 미흡하거나 비밀번호를 화면에 즉시 출력할 경우 공격자가 불법적으로 다른 사용자의 비밀번호를 획득, 변경, 복구할 수 있는 취약점

■ 파급효과

사용자 패스워드 노출



나. 판단기준

순번	점검위치	행위	취약반응
①	비밀번호 찾기 페이지	임의의 정보를 입력하여 비밀번호 찾기 시도	사용자 비밀번호 노출

다. 대응방안

- ① 웹 방화벽 : 해당사항 없음
- ② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용

■ DB 연동을 통한 사용자 인증 예 JAVA

```

1: // 서버로 전달된 파라미터를 변수에 저장
2: String userID = request.getParameter("userID");
3: String userEmail = request.getParameter("userEmail");
4: // JAVA PreparedStatement 객체 생성
5: PreparedStatement stmt = connection.prepareStatement("SELECT idx
   FROM users WHERE userid=? AND email=?");
6: // setString 메소드를 통한 바인딩질의 사용
7: stmt.setString(1, userID);
8: stmt.setString(2, userEmail);
9: // 질의문 실행
10: ResultSet rs = stmt.executeQuery();
11: int userIdx = 0;
12: // 사용자의 idx 값을 반환
13: int userIdx = rs.getInt("idx");
14: // 정상적인 사용자인지 확인 (사용자 인증)
15: if ( userIdx != 0 ) {
16:     ... .. // 정상적인 비밀번호 찾기 실행
17: }

```

사용자 비밀번호 찾기 시 데이터베이스에 저장되어 있는 사용자 정보와 입력한 사용자 정보가 일치하는지 비교하는 인증 구문을 적용해야 하며, 인증 후 발급되는 임시 비밀번호의 경우 웹 페이지에 노출 시키지 않고 사용자의 이메일 또는 SMS를 통해 전송되도록 설정해야 함

- ③ 웹 서버 보안 설정 : 해당사항 없음

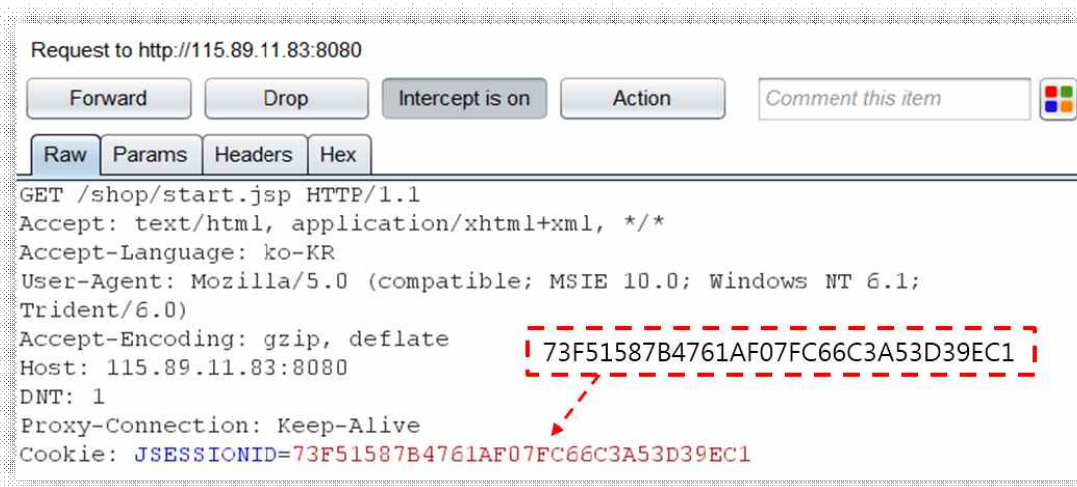
10. 불충분한 세션 관리

가. 개요

웹 어플리케이션에서 사용자가 로그인을 할 경우 매번 동일한 세션 ID(일정한 패턴이 존재)를 발급하거나 세션 타임아웃을 너무 길게 설정하였을 경우 공격자가 다른 사용자의 세션을 재사용 하여 해당 사용자의 권한을 탈취할 수 있는 취약점

■ 파급효과

사용자 권한 획득



[불충분한 세션 관리 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	로그인 후 세션 ID	세션 값을 변조하여 로그인 시도	다중 로그인 성공
②	로그인 후 세션 ID	로그인과 로그아웃을 반복하여 수집된 세션 ID의 패턴 분석	타 사용자 로그인 성공

다. 대응방안

- ① 웹 방화벽 : 해당사항 없음
- ② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용

■ 중복 로그인 방지 예 JAVA

LoginManager.java

```

1:  // 로그인 한 사용자들을 저장하는 저장소
2:  static Hashtable loginusers = new Hashtable();
3:  // 중복 로그인을 막기 위해 사용자 아이디 검증 메소드
4:  public boolean CheckUsing(String userID) {
5:      // 로그인 중 인지 판단하는 변수
6:      boolean isUsing = false;
7:      // 저장소에 존재하는 사용자들을 불러옴
8:      Enumeration e = loginUsers.key();
9:      String key = "";
10:     // 사용자들을 모두 체크
11:     while ( e.hasMoreElements() ) {
12:         key = (String)e.nextElement();
13:         // 저장소에 존재하는 아이디와 같은 경우
14:         if ( userID.equals(loginUsers.get(key)) ) { isUsing = true; }
15:     }
16:     return isUsing;
17: }
```

Login_ok.jsp

```

1:  <%@ page import="LoginManager"%>
2:  <% LoginManager loginManager = new LoginManager();
3:      // 서버로 전달된 파라미터(로그인 아이디)를 변수에 저장
4:      String userID = request.getParameter("userID");
5:      // 중복 로그인 판단
6:      if ( !loginManager.CheckUsing(userID) ) {
7:          ... .. // 정상 로그인 처리
8:      } else { ... .. } // 에러 메시지 출력 (중복 로그인)
```


로그인 한 사용자들의 정보를 저장하는 별도의 변수를 지정하여 로그인 시 사용자의 아이디와 세션 값을 저장하여 추후에 발생하는 세션 요청에 대해 저장소의 값과 비교하여 저장소에 존재하는 사용자라면 중복 로그인으로 판단하여 로그인이 불가능 하도록 관리가 필요함

상기 예제는 로그인 페이지에서만 적용이 가능한 코딩 예제이며, 로그인 페이지 이외에 세션 변조를 통한 중복 로그인 방지를 위해서는 사용자의 IP주소 및 토큰 값을 별도로 저장하여 추후에 발생하는 요청에 대해 검증이 필요함

■ 세션 타임아웃 설정 예 JAVA

```
1: HttpSession session = request.getSession();
2: // 세션 타임아웃을 15분으로 설정
3: session.setMaxInactiveInterval(60*15);
```

세션 타임아웃 설정을 통해 일정 시간 동안 움직임이 없을 경우 자동으로 로그아웃 되도록 설정하는 것을 권장함

③ 웹 서버 보안 설정 : 해당사항 없음

11. 크로스 사이트 리퀘스트 변조(CSRF)

가. 개요

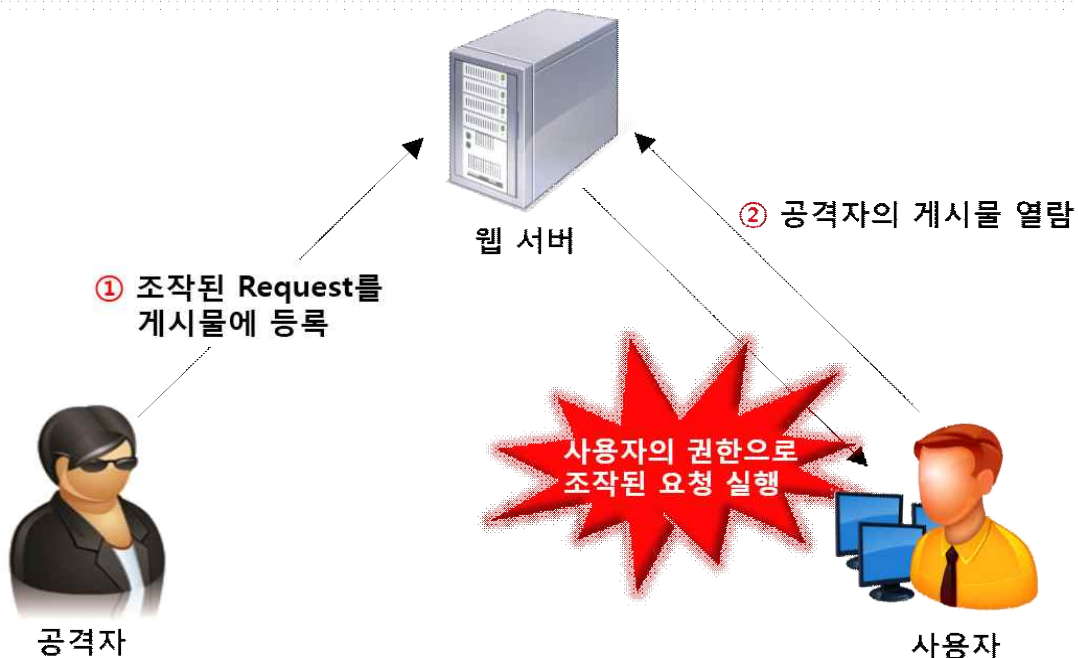
웹 어플리케이션에서 정상적인 경로를 통한 요청과 비정상적인 경로를 통한 요청을 서버가 구분하지 못할 경우 공격자가 스크립트 구문을 이용하여 정상적인 사용자로 하여금 조작된 요청을 전송 하도록 하여 게시판 설정 변경 및 자동 댓글, 회원 등급 변경 등의 문제가 발생할 수 있는 취약점

크로스 사이트 리퀘스트 변조의 경우 공격을 당한 사용자의 권한을 그대로 사용하게 되므로 사용자의 권한 수준에 따라 그 피해범위가 달라질 수 있음

■ 파급효과

사용자 권한 도용, 사용자 정보 변경

제목	CSRF 취약점 확인		
글쓴이	<input type="text"/>	공개여부	공개 <input type="checkbox"/>
홈페이지	<input type="text"/>	이메일	<input type="text"/>



[CSRF 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	사용자 입력 폼	입력	조작된 요청 실행

다. 대응방안

- ① 웹 방화벽 : [제1장 1절 6. 크로스 사이트 스크립트(XSS)] 참고
- ② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩

■ 토큰 사용 예 JAVA

#1 로그인 성공 시 로그인 처리와 함께 토큰을 생성하여 세션에 저장

```

1: // 무작위 문자열 생성을 위한 변수
2: char [] initRandomChar = {'A', 'B', 'C', 'D', 'F', 'G', 'H', 'I', 'J',
3:                             'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S',
4:                             'N', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', '0',
5:                             '1', '2', '3', '4', '5', '6', '7', '8', '9'};
6: // 랜덤 함수를 이용하여 무작위 문자열 생성
7: StringBuffer sb = new StringBuffer();
8: for ( int n=0; n<6; n++) {
9:     int index = (int) (initRandomChar.length * Math.random());
10:    sb.append(initRandomChar[index]);
11: }
12: // 토큰 생성
13: String CSRF_Token = sb.toString();
14: // 세션에 토큰 내용 저장
15: session.setAttribute("Token", CSRF_Token);

```

#2 웹 페이지 호출 시 URL을 통해서 토큰 전송

```

1: <input type="hidden" name="Token"
    value="<%=session.getAttribute("Token")%>">

```

#3 URL을 통해서 전송된 토큰과 세션에 저장된 토큰을 비교

```

1: // 서버로 전달된 파라미터를 변수에 저장
2: String token = request.getParamet("Token");
3: // 서버로 전달된 토큰과 세션에 저장된 토큰이 다를 경우
4: if ( !token.equals(session.getAttribute("Token")) ) {
5:     ... .. // 에러 메시지 출력 (비정상적인 요청)
6: }

```

크로스 사이트 리퀘스트 변조의 경우 웹 어플리케이션에 존재하는 모든 HTTP 요청 내에 예측할 수 없는 임의의 토큰을 추가하여 정상적인 요청과 비정상적인 요청을 판별하는 것을 권장하며 추가적으로 XSS 와 공격 방식이 유사하므로 “크로스 사이트 스크립트(XSS)” 대응방안을 참고하여 XSS 취약점을 제거하는 것을 권장함

③ 웹 서버 보안 설정 : 해당사항 없음

12. 자동화 공격

가. 개요

웹 어플리케이션 운영 시 특정 프로세스에 대한 접근시도 횟수 제한을 설정하지 않을 경우 공격자가 자동화 툴 및 봇을 활용하여 일분에 수백 번의 접근을 시도할 수 있으며 특정 프로세스를 반복 수행함으로써 자동으로 수많은 프로세스(DoS, 무차별 대입 기법 등)가 진행되어 시스템 성능에 영향을 미칠 수 있는 취약점

■ 파급효과

시스템 과부하, 웹 서비스 중지, 사용자 계정 탈취



[자동화 공격 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	로그인 폼	자동화 도구를 이용한 무차별 대입 공격 시도	계정 정보 획득

다. 대응방안

- ① 웹 방화벽 : 특정 시간 내 동일 프로세스가 반복 실행되지 않도록 시간제한을 설정해야 하며, 자동화 공격에 의한 시스템 과부하를 방지하기 위해 다량의 패킷이 유입될 경우 해당 접속을 차단하는 것을 권장함
- ② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용

■ 로그인 시도 횟수를 판단하는 예 JAVA

```

1: // JAVA PreparedStatement 객체 생성
2: PreparedStatement stmt = connection.prepareStatement("SELECT
   login_count FROM users WHERE userid=?");
3: // setString 메소드를 통한 바인딩질의 사용
4: stmt.setString(1, user_id);
5: ResultSet rs = stmt.executeQuery();
6: // 로그인 시도 횟수가 5회이상인지 확인.
7: if ( rs.getInt(1) > 5 ) {
8:     ... ..           // 로그인이 불가능 하도록 설정(계정 잠금)
9: }
10: else { ... .. }      // 정상적인 로그인 절차

```

웹 어플리케이션 로그인 관련 테이블에 로그인 시도 횟수를 저장하는 컬럼을 추가하여 로그인 시도가 있을 때마다 횟수를 증가시키고, 일정 횟수 이상이 되면 자동화 공격으로 인식하여 로그인을 할 수 없도록 차단해야 함

■ 캡차를 이용한 예 JAVA

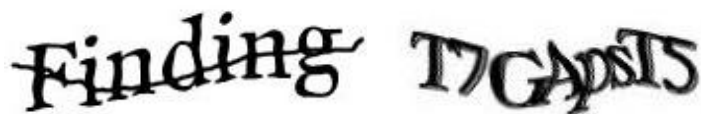
```

1: // 캡차 사용을 위한 라이브러리
2: <%@ page import = "nl.captcha.Captcha" %>
3: <%
4:     // 캡차 생성
5:     Captcha captcha = (Captcha)session.getAttribute(Captcha.NAME);
6:     request.setCharacterEncoding("UTF-8");
7:     String answer = request.getParameter("answer");
8:     // 캡차 내용이 맞는지 확인
9:     if ( captcha.isCorrect(answer) ) {
10:         ... .. // 정상적인 프로세스 처리
11:     }
12:     else { ... .. } // 에러 메시지 출력
13: %>

```

캡차를 사용할 경우 서버에 요청하는 사용자가 실제 사람인지, 컴퓨터 프로그램인지 구별 할 수 있기 때문에, 이를 회원 가입 및 로그인 등 프로세스에 적용 시 자동화 도구를 통한 공격을 방어할 수 있음

※ **캡차**란 실제 사람인지, 컴퓨터 프로그램인지를 구별하기 위한 방법으로 사람이 쉽게 인식할 수 있는 임의의 문자와 숫자를 그림으로 변환하여 제시하고 제시된 문자는 왜곡되거나 덧칠되어 기계가 자동으로 읽기 어려운 상태로 만드는 것이 특징임



[캡차 예]

③ 웹 서버 보안 설정 : 해당사항 없음

13. 파일 업로드

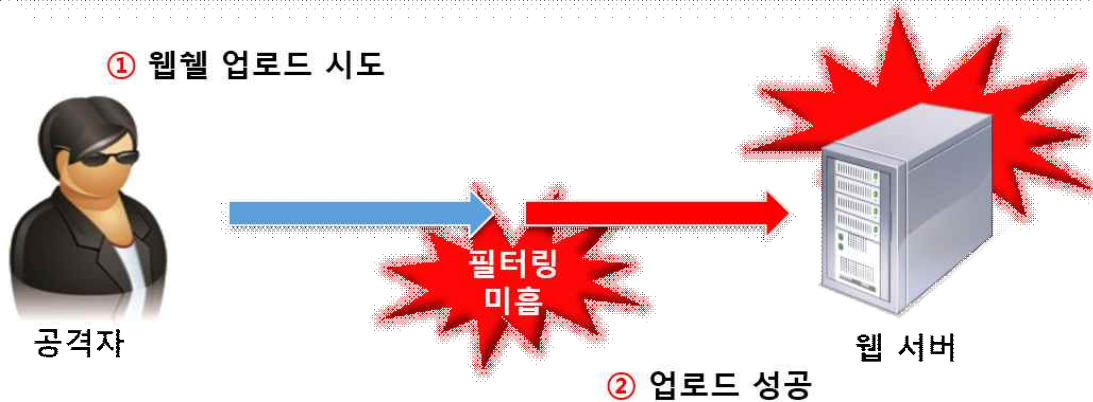
가. 개요

파일 업로드 기능이 존재하는 웹 어플리케이션에서 확장자 필터링이 제대로 이루어지지 않았을 경우 공격자가 악성 스크립트 파일(웹셸)을 업로드 하여 웹을 통해 해당 시스템을 제어할 수 있어 명령어 실행 및 디렉터리 열람이 가능하고 웹 페이지 또한 변조가 가능한 취약점

■ 파급효과

시스템 장악, 웹 페이지 변조

제목	파일 업로드 취약점 확인		
글쓴이		홈페이지	
이메일			
파일첨부	C:\WebShell.jsp		찾아보기...



[파일 업로드 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	게시판 (파일 업로드)	악성 스크립트 파일 업로드 시도	업로드 성공

다. 대응방안

- ① 웹 방화벽 : 파일 업로드 필드를 대상으로 특수문자 필터링 규칙 적용

필터링 대상		
%	;	..

- ② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용

■ 화이트리스트를 통한 파일 업로드 예 JAVA

```

1: // 허용 가능한 확장자
2: String allowUpload[] = {"png", "jpg", "gif"};
3: // 서버로 전달된 파일명을 fileName 변수에 저장
4: String fileName = request.getParameter("filename");
5: // 파일명 중 확장자 부분만 저장
6: String fileExt = fileName.substring(fileName.lastIndexOf('.') + 1).toLowerCase();
7: int point = 0;
8: // 허용 가능한 확장자가 맞는지 확인
9: for ( int n=0; n<allowUpload.length; n++ ) {
10: // 허용 가능한 확장자가 존재할 경우
11: if ( fileExt.equals(allowUpload[n]) ) {
12: point += 1;
13: }
14: }
15: if ( point > 0 ) {
16: // 시간에 따라 파일명 변경
17: String timemask = "KISA" + System.currentTimeMillis();
18: ... .. // 정상적인 업로드 실행
19: }
20: else { ... .. } // 차단 메시지 출력

```

상기 예제와 같이 화이트리스트 정책을 활용하여 허용 가능한 파일 확장자 목록을 만들고, 허용된 확장자가 아닌 경우 업로드가 불가능 하도록 설정해야 함

또한 파일 업로드 시 기존 파일명을 그대로 사용하는 것이 아니라 업로드 시간 또는 특수한 규칙(개발 시 적용)에 따라 파일명을 변경 하여 공격자로 하여금 경로 추적을 할 수 없도록 해야 함

③ 웹 서버 보안 설정 : 해당사항 없음

14. 경로추적 및 파일 다운로드

가. 개요

파일 다운로드 기능이 존재하는 웹 어플리케이션에서 파일 다운로드 시 파일의 경로 및 파일명을 파라미터로 받아 처리하는 경우 파일에 대한 접근 권한이 설정되어 있지 않다면 공격자가 파라미터를 조작하여 환경설정 파일, 웹 소스코드 파일, 데이터베이스 연동 파일 등을 다운 받을 수 있는 취약점 경로추적의 경우 인증되지 않은 사용자가 시스템에 접근하여 중요한 파일을 읽거나 권한 없는 기능 등을 수행할 수 있는 취약점

■ 파급효과

서버 주요파일 노출, 소스코드 직접 다운로드, 데이터베이스 정보노출



[경로추적 및 파일 다운로드 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	URL	파일 다운로드 시 파라미터 값을 조작하여 서버파일 다운로드 시도	파일 다운로드 성공
②	URL	웹 루트 상위로 접근 시도	주요 파일 노출

※ URL(각종 파라미터) : 파라미터는 해당 사이트의 URL에서 아래와 같은 부분을 말하며 공격자는 여러 파라미터 중 임의로 몇 가지의 파라미터에 점검을 한 후 취약여부를 판단함

■ 예시

http://www.test.or.kr/board.php?search=content&login=yes&id=test

다. 대응방안

- ① 웹 방화벽 : 다운로드 파라미터를 대상으로 특수문자 필터링 규칙 적용

필터링 대상				
../	./	..₩	.₩	%

- ② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용

■ 다운로드 시 특수문자 필터링 예 JAVA

```

1: // 서버로 전달된 파일명을 변수에 저장
2: String fileName= request.getParameter("filename");
3: // 전달된 파라미터가 null 인지 체크
4: if ( fileName == null || "".equals(fileName) ) {
5:     ... .. // 에러 메시지 출력
6: }
7: // 파일 경로 + 파일 이름
8: String filePath = UPLOAD_PATH + fileName;
9: // 인코딩 우회를 방지하기 위한 디코딩
10: String decodeFileName = URLDecoder.decode(fileName,
    Charset.defaultCharset().name());
11: // 파일 이름 체크 (특수문자 필터링)
12: if ( decodeFileName.equalsIgnoreCase("..") ||
13:     decodeFileName.equalsIgnoreCase("/") ) {
14:     ... .. // 에러 메시지 출력
15: }
```

파일 다운로드 시 우선적으로 웹 루트 상위 경로로 접근되지 않도록 권한 설정을 권장 하며 파일 경로 및 파일 이름을 처리하는 파라미터 변수에 대해 null 여부를 체크 및 상대경로를 설정할 수 있는 문자(/, ₩, &, . 등)가 존재할 경우 파일 다운로드가 불가능 하도록 구현하길 권장함

- ③ 웹 서버 보안 설정 : 해당사항 없음

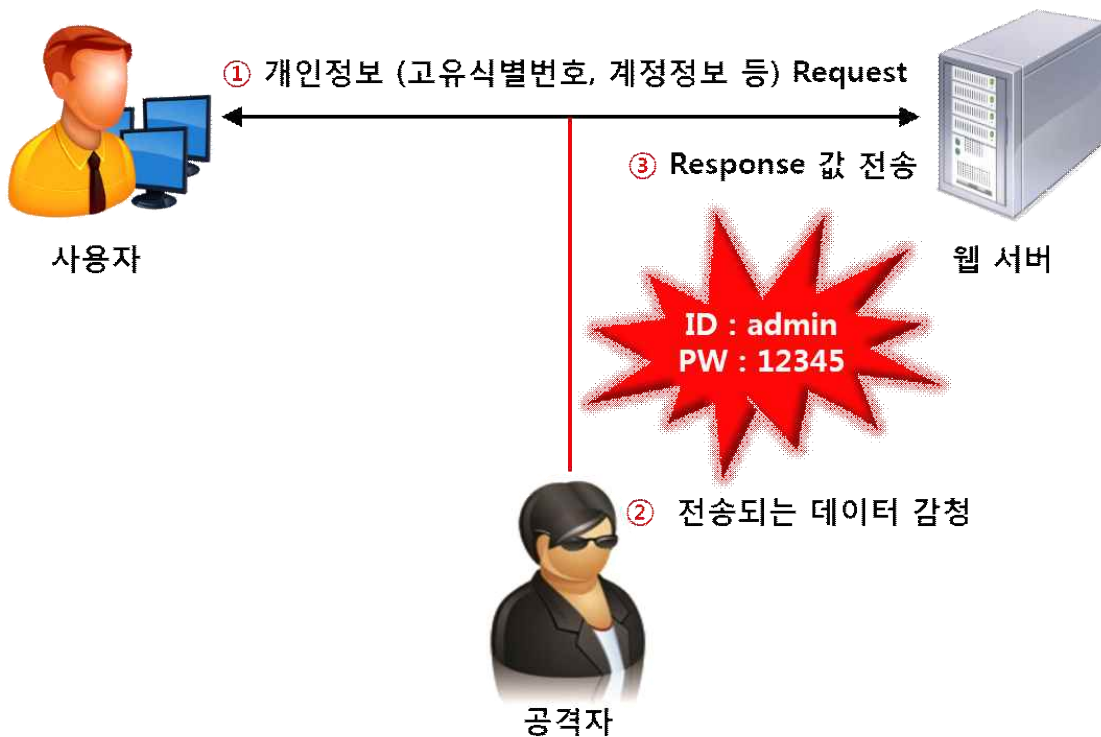
15. 데이터 평문전송

가. 개요

로그인 또는 실명인증 시 민감한 데이터(개인 식별번호, 계정정보 등)가 평문으로 통신채널을 통해 송수신 될 경우 공격자가 감청(스니핑)을 통해 다른 사용자의 민감한 데이터를 획득 할 수 있는 취약점

■ 파급효과

사용자 개인정보 노출



[데이터 평문전송 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	웹 어플리케이션	로그인 시 전송 데이터 확인	개인 정보 노출

다. 대응방안

- ① 웹 방화벽 : 해당사항 없음
- ② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용

■ 비밀번호 암호화 예 JAVA

```

1: // 비밀번호를 char 배열에 저장
2: char[] passWord = password.toCharArray();
3: String encrypted = "";
4: try {
5:     // 암호화 알고리즘 선택
6:     MessageDigest md = MessageDigest.getInstance("SHA-1");
7:     // UTF-8 인코딩
8:     md.update(new String(passWord).getBytes("UTF-8"));
9:     // 선택한 알고리즘으로 암호화
10:    byte[] digested = md.digest();
11:    // 암호화된 비밀번호를 base64로 인코딩
12:    encrypted = new String(Base64Coder.encode(digested));
13: } catch (Exception e) {
14:     ... .. // 예외 처리
15: }
```

사용자가 입력한 개인정보(또는 중요 데이터)에 대해 상기 예제와 같이 암호화를 적용하여 송수신 할 경우, 공격자가 중간에 데이터를 감청하더라도 암호화된 데이터를 복호화 할 수 없기 때문에 데이터의 기밀성을 유지할 수 있음

- ③ 보안 서버 적용 : 서버와 클라이언트 통신 시 중요정보가 사용되는 구간에 SSL 등 암호화 통신을 적용

※ **SSL(Secure Sockets Layer)**이란 웹 서버 인증, 서버 인증이라 불리는 프로토콜을 말하며, 클라이언트와 서버 간 통신에서 정보를 암호화하기 때문에 공격자의 감청으로 정보가 유출되더라도 정보의 내용을 보호할 수 있게 해 주는 보안 솔루션을 말함

■ 정보통신망 이용촉진 및 정보보호 등에 관한 법률 시행령

제 15조 (개인정보의 보호조치)

... 중략 ...

- ④ 법 제28조제1항제4호에 따라 정보통신서비스 제공자등은 개인정보가 안전하게 저장·전송될 수 있도록 다음 각 호의 보안조치를 하여야 한다.

... 중략 ...

3. 정보통신망을 통하여 이용자의 개인정보 및 인증정보를 송신·수신하는 경우 보안서버 구축 등의 조치

■ 개인정보의 기술적 관리적 보호조치 기준(방송통신위원회 고시)

제 6조 (개인정보의 암호화)

... 중략 ...

- ③ 정보통신서비스 제공자등은 정보통신망을 통해 이용자의 개인정보 및 인증정보를 송·수신할 때에는 안전한 보안서버 구축 등의 조치를 통해 이를 암호화해야 한다.

보안서버는 다음 각 호 중 하나의 기능을 갖추어야 한다.

1. 웹서버에 SSL(Secure Socket Layer) 인증서를 설치하여 전송하는 정보를 암호화하여 송·수신하는 기능
2. 웹서버에 암호화 응용프로그램을 설치하여 전송하는 정보를 암호화하여 송·수신하는 기능

[개인정보 관련 법령]

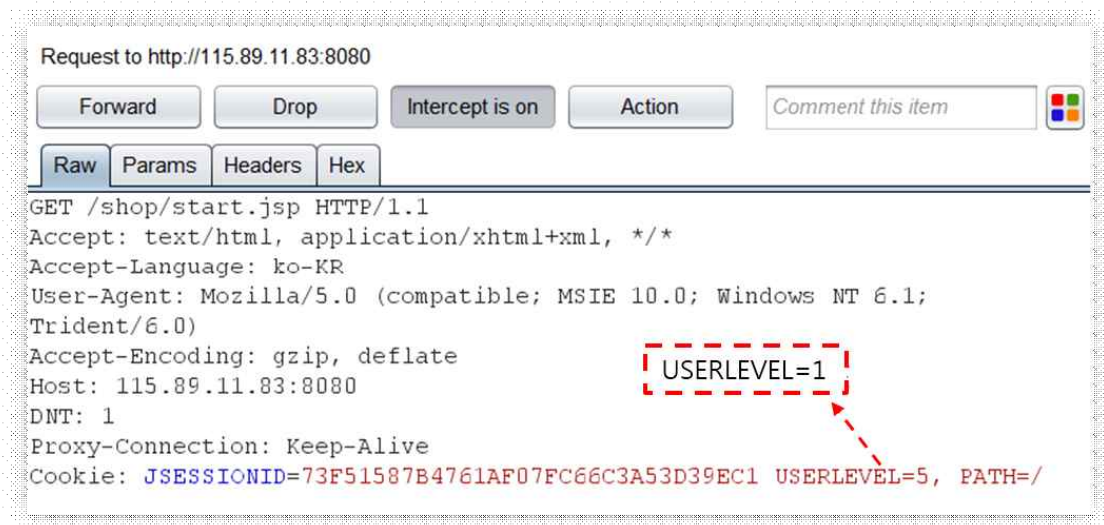
16. 쿠키 변조

가. 개요

웹 어플리케이션 운영 시 사용자 인증 방식 중 하나인 쿠키를 공격자가 변조하여 다른 사용자로 전환하거나 권한 상승이 가능한 취약점 클라이언트 측에 저장되는 쿠키는 그 특성상 변조 위험에 노출되어 있으나 활용 분야가 다양하고 구현이 쉽다는 이유로 인해 현재까지 많이 사용되고 있음

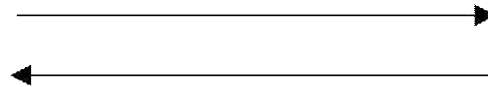
■ 파급효과

사용자 인증 우회, 권한 상승



공격자

① 조작된 쿠키



② 권한 상승



웹 사이트

[쿠키 변조 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	쿠키	쿠키 값 변조 시도	권한 상승

다. 대응방안

- ① 웹 방화벽 : 해당사항 없음
- ② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용

■ 바인딩된 질의문 사용 예 JAVA

```

1: // 암호화 시 사용될 키 설정
2: public static String key = "this_is_kisa_key";
3: // 암호화 비밀키 생성
4: SecretKeySpec skeySpec = new SecretKeySpec(key.getBytes(), "AES");
5: // 암호화 알고리즘 선택
6: Cipher cipher = Cipher.getInstance("AES");
7: // 초기 값 설정
8: cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
9: // 선택한 알고리즘으로 userLevel을 암호화
10: byte[] encrypted = cipher.doFinal(userLevel.getBytes());
11: // byte 배열을 16진수 문자열로 변경
12: StringBuffer sb = new StringBuffer(encrypted.length * 2);
13: String hexNumber;
14: for ( int n = 0; n < encrypted.length; n++ ) {
15:     hexNumber = "0" + Integer.toHexString(0xff & encrypted[n]);
16:     sb.append(hexNumber.substring(hexNumber.length() -2));
17: }
18: // 암호화된 값을 쿠키로 설정
19: Cookie cookie = new Cookie("USERLEVEL", sb.toString());

```

쿠키에 중요정보가 포함 되어있으면 공격자가 사용자의 중요한 정보를 변조하거나 획득할 수 있는 위험이 있으므로 웹 어플리케이션 운영 시 가급적 쿠키 방식 대신 세션 방식을 사용하는 것을 권장하며 부득이하게 쿠키를 사용해야 할 경우 SEED, 3DES, AES 등 공인된 암호화 알고리즘을 사용하여 암호화를 적용시켜야 함

세션 방식은 접속자 별로 세션을 생성하여 사용자의 정보를 각각 지정할 수 있는 오브젝트로서, 페이지의 접근을 허가 또는 금지하거나 사용자 별 정보를 저장할 때 많이 사용함

따라서, 클라이언트의 자원을 사용하는 쿠키 보다 서버의 자원을 사용하므로 세션이 보안성 측면에서 우수하므로 웹 어플리케이션 개선·구축 시 세션 방식을 사용하길 권장함

③ 웹 서버 보안 설정 : 해당사항 없음

17. URL/파라미터 변조

가. 개요

웹 어플리케이션 상에 존재하는 모든 실행경로에 대해서 접근제어를 검사하지 않거나 미흡한 경우 공격자가 접근 가능한 실행경로를 통해 사용자의 정보를 유출 하거나 일시적인 권한 상승이 가능한 취약점

URL/파라미터 변조의 경우 불충분한 인증 및 인가 취약점과 유사하지만 프로세스 검증을 우회하는 것 이외에 사용자 입력 값에 대한 검증 누락이 발생하는 모든 상황을 포함하고 있으므로 SQL 인젝션, 불충분한 인증 및 인가, 크로스 사이트 스크립트(XSS) 공격에 활용될 수 있음

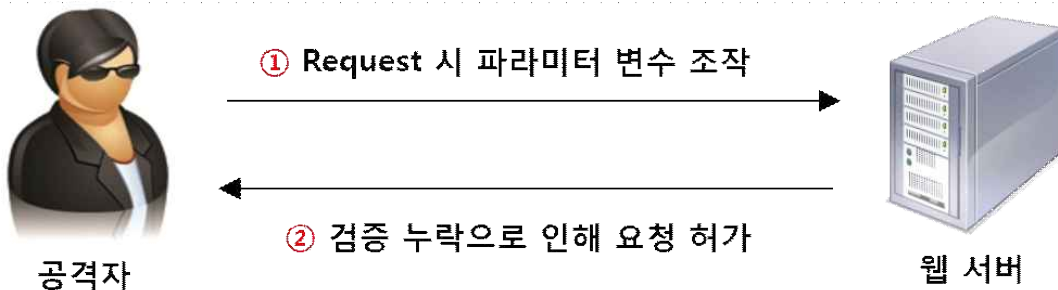
■ 파급효과

사용자 권한 획득, 인증 우회, 홈페이지 장악

```
Accept-Encoding: gzip, deflate
Content-Length: 75350
DNT: 1
Host: 59.9.220.79:8080
Pragma: no-cache
Cookie: name=sPy4rsDa; id="YWRtaW4="; email="YUBhLmE="; JSESSIONID=623577717E1762AF69A2CB
-----7dd286640063a
Content-Disposition: form-data; name="b_gubun"

02AA
-----7dd286640063a
Content-Disposition: form-data; name="notice_yn"

N
```



[URL/파라미터 변조 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	게시판 파라미터	게시판 페이지에 존재하는 파라미터 변수 값 조작	일시적인 권한 상승

다. 대응방안

- ① 웹 방화벽 : 해당사항 없음
- ② 웹 어플리케이션 : 웹 어플리케이션 소스코드 시큐어 코딩 적용

■ 세션을 이용한 사용자 인증 예 JAVA

```

1: // 사용자의 세션 정보를 반환
2: HttpSession session = request.getSession(true);
3: // 세션에 존재하는 userID 정보를 반환
4: String sessionID = session.getAttribute("userID");
5: // 서버로 전달된 파라미터를 변수에 저장
6: String userID = request.getParameter("userID");
7: // 사용자 입력 값과 세션 값을 비교
8: if ( sessionID.equals(userID) ) {
9:     ... .. // 정상적인 게시글 등록
10: }
11: else { ... .. } // 에러 메시지 출력 (파라미터 변조 탐지)

```

웹 어플리케이션에서 제공하는 정보와 기능을 역할에 따라 배분함으로써 사용자가 변경할 수 있는 데이터의 노출을 최소화하는 것이 필요하며, 서버로 전송된 사용자 입력 값의 경우 세션 및 데이터베이스와 비교를 하여 데이터가 변조되었는지 검증할 수 있는 과정을 구현해야 함

- ③ 웹 서버 보안 설정 : 해당사항 없음

제3장 웹 서버 취약점

1. 디렉터리 인덱싱

가. 개요

웹 어플리케이션을 사용하고 있는 서버의 미흡한 설정으로 인해 인덱싱 기능이 활성화가 되어있을 경우, 공격자가 강제 브라우징을 통해 서버내의 모든 디렉터리 및 파일에 대해 인덱싱이 가능하여 웹 어플리케이션 및 서버의 주요 정보가 노출될 수 있는 취약점

■ 파급효과

디렉터리 및 파일 목록 노출



[디렉터리 인덱싱 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	URL	웹 어플리케이션 경로 직접 접근	디렉터리 및 파일 목록 노출

※ URL(각종 파라미터) : 파라미터는 해당 사이트의 URL에서 아래와 같은 부분을 말하며 공격자는 여러 파라미터 중 임의로 몇 가지의 파라미터에 점검을 한 후 취약여부를 판단함

■ 예시

http://www.test.or.kr/board.php?search=content&login=yes&id=test

다. 대응방안

- ① 웹 방화벽 : 해당사항 없음
- ② 웹 어플리케이션 : 해당사항 없음
- ③ 웹 서버 보안 설정 : 미흡한 웹 서버 보안 설정 변경

■ 안전한 서버 설정 예 Tomcat

```

1: <servlet>
2:   <servlet-name>default</servlet-name>
3:   <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
4:   <init-param>
5:     <param-name>debug</param-name> se
6:     <param-value>0</param-value>
7:   </init-param>
8:   <init-param>
9:     <param-name>listings</param-name>
10:    <param-value>false</param-value>
11:  </init-param>
12:  <load-on-startup>1</load-on-startup>
13: </servlet>

```

\$CATALINA_HOME(톰캣 홈 디렉터리)/conf/web.xml 파일 내용 중 listings의 <param-value> 값을 false로 지정한 후 웹 서버를 재시작하면 URL 강제 브라우징 시 디렉터리 및 파일 목록이 노출되지 않음

디렉터리 인덱싱 제어 이후 노출되는 에러 페이지는 “정보누출” 대응방안의 보안 설정 적용을 통해, “인덱싱 제어 + 정보누출방지” 형태의 복합 대응이 필요함

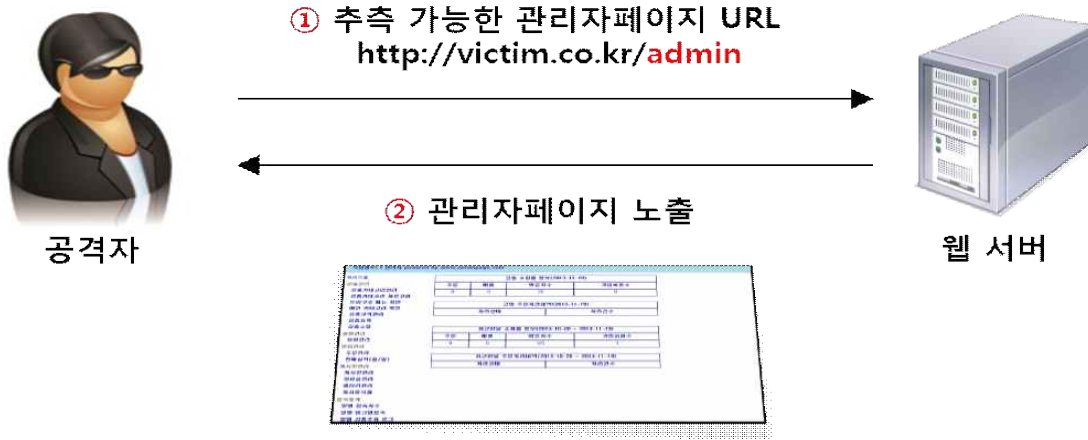
2. 관리자페이지 노출

가. 개요

웹 어플리케이션의 전반적인 기능 설정 및 회원 관리를 할 수 있는 관리자페이지가 추측 가능한 형태로 구성되어 있을 경우 공격자가 관리자페이지에 쉽게 접근을 할 수 있으며 무차별 대입 공격을 통하여 관리자 권한을 획득할 수 있는 취약점

■ 파급효과

회원 정보 조작 및 유출



[관리자페이지 노출 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	URL	추측 가능한 관리자페이지 (admin, manager, webmaster 등) 경로로 직접 접근	관리자페이지 접속

※ URL(각종 파라미터) : 파라미터는 해당 사이트의 URL에서 아래와 같은 부분을 말하며 공격자는 여러 파라미터 중 임의로 몇 가지의 파라미터에 점검을 한 후 취약여부를 판단함

■ 예시

`http://www.test.or.kr/board.php?search=content&login=yes&id=test`

다. 대응방안

- ① 웹 방화벽 : 관리자페이지에 대해 허용된 IP만 접근 할 수 있도록 IP 접근제한 규칙 적용
- ② 웹 어플리케이션 : 해당사항 없음
- ③ 웹 서버 보안 설정 : 미흡한 웹 서버 보안 설정 변경

■ 안전한 서버 설정 예 Tomcat

```

1: <Host ...>
2:   <Context path="/KISAadm" docBase="/tomcat/webapps/ROOT/KISAadm">
3:     <Valve className="org.apache.catalina.valves.RemoteHostValve"
         allow="192.168.1.2"/>
4:   </Context>
5: </Host>

```

\$CATALINA_HOME(톰캣 홈 디렉터리)/conf/server.xml 파일 내용 중 <Host> 필드 사이에 위와 같은 설정을 추가한 후 웹 서버를 재시작하면 관리자페이지에 대해 IP기반으로 접근제어가 가능

관리자페이지 접근제어 설정 이후 노출되는 에러 페이지는 "정보누출" 대응방안의 보안 설정 적용을 통해, "접근제어 + 정보누출방지" 형태의 복합 대응이 필요함

항목	설명
path	웹 어플리케이션 상의 관리자페이지 경로
docBase	웹 루트 디렉터리 상의 관리자페이지 경로
className	사용하고자 하는 모듈
deny	접근 제한 IP
allow	접근 허용 IP

[Context 파라미터 항목]

가. 개요

■ 파급효과

시스템 관련 정보노출, 소스코드 노출



순번	점검위치	행위	취약반응
①	웹 어플리케이션	추측 가능한 백업 파일명으로 직접 접근	백업 파일 노출
②	웹 어플리케이션	디렉터리 인덱싱을 통해 노출된 백업파일 확인	백업 파일 노출

다. 대응방안

- ① 웹 방화벽 : 해당사항 없음
- ② 웹 어플리케이션 : 해당사항 없음
- ③ 웹 서버 보안 설정 : 사용하고 있는 웹 서버의 디폴트 페이지 및 샘플 파일들을 삭제하는 것을 권장함

- 1) 설치된 웹 서버에 대한 기본 페이지와 배너를 삭제하여 배너 검색에 의한 시스템 기본 정보 유출을 사전에 차단
- 2) 톰캣, 아파치 등을 사용할 경우, 인스톨 혹은 기본 페이지 등을 삭제하고 샘플 파일 역시 삭제
- 3) 웹로직, 제우스, 톰캣 등 원격 관리 콘솔 기능을 제공하는 제품 사용 시, 관리 콘솔이 디폴트로 설치되어 있는지 확인하고 사용하지 않는 관리 콘솔을 중지하거나 삭제
- 4) 만일 관리 콘솔을 사용할 경우에는 반드시 접근 통제를 실시하여 임의의 위치 및 임의의 사용자가 접근할 수 없도록 하고, 관리자 계정이 유출되지 않도록 철저하게 관리
- 5) 관리자는 웹 디렉터리를 조사하여 *.jsp.bak 와 같은 백업파일을 모두 삭제
- 6) *.txt 같은 웹 페이지의 작업 도중 생성된 일반 텍스트 파일이나 그 밖에 이미지 파일 등도 본래 파일 이외에는 제거하여야 하며 *.jsp.bak 파일 이외의 *.bak 파일도 불필요한 경우는 삭제
- 7) 백업 파일 같은 경우 언제 생성될 지 관리자의 측면에서는 알기가 매우 어려우므로 주기적으로 검사 및 삭제가 필요하며, 가급적이면 웹 서버를 실행하는데 필요한 파일을 제외하고는 모두 삭제할 것을 권장

[백업 및 임시파일 조치방안]

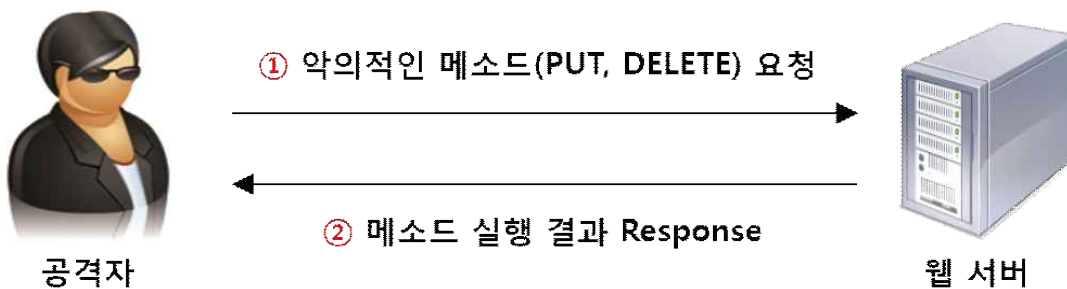
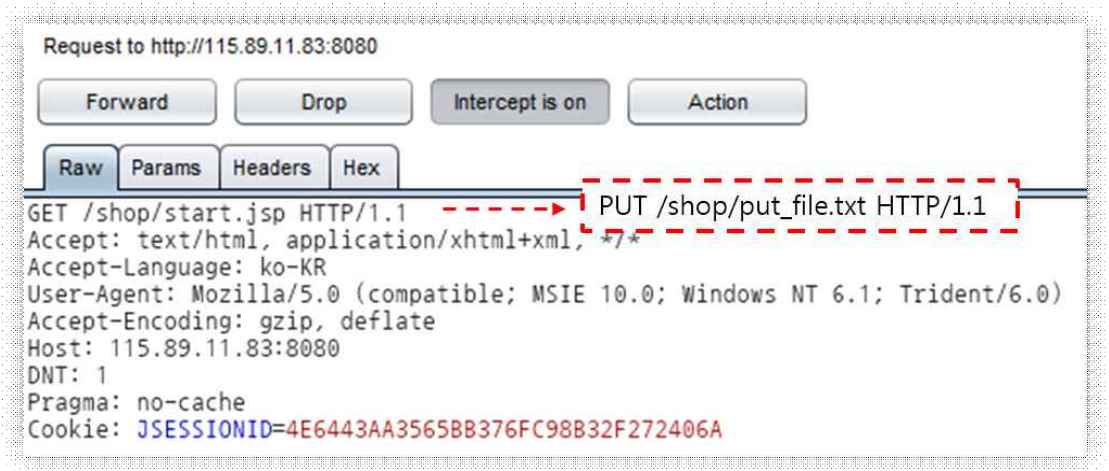
4. 웹 서비스 메소드 설정 공격

가. 개요

웹 어플리케이션에서 일반적으로 사용하는 GET, POST 메소드 이외의 PUT, DELETE, COPY, MOVE 등의 불필요한 메소드를 허용하였을 경우 공격자가 이를 이용하여 웹 서버에 파일을 생성하거나 삭제 및 수정이 가능한 취약점

■ 파급효과

임의의 파일 삭제 및 업로드, 시스템 장악



[웹 서비스 메소드 설정 공격 동작 과정]

나. 판단기준

순번	점검위치	행위	취약반응
①	웹 어플리케이션	악의적인 목적으로 사용할 수 있는 메소드 요청	메소드 허용

다. 대응방안

- ① 웹 방화벽 : 해당사항 없음
- ② 웹 어플리케이션 : 해당사항 없음
- ③ 웹 서버 보안 설정 : GET, POST를 제외한 불필요 메소드를 비활성화
시키는 것을 권장함

■ 안전한 서버 설정 예 Tomcat

```

1: <security-constraint>
2:   <display-name>Forbidden</display-name>
3:   <web-resource-collection>
4:     <web-resource-name>Protected Context</web-resource-name>
5:     <url-pattern>/*</url-pattern>
6:     <http-method>DELETE</http-method>
7:     <http-method>GET</http-method>
8:     <http-method>POST</http-method>
9:     <http-method>PUT</http-method>
10:   </web-resource-collection>
11:   <auth-constraint></auth-constraint>
12: </security-constraint>

```

\$CATALINA_HOME(톰캣 홈 디렉터리)/conf/web.xml 파일의 <web-app> 필드 내에 위와 같은 설정을 추가하여 악의적으로 이용할 수 있는 메소드를 차단해야 함

부 록

제1절 웹 시스템 구동 개요

웹 시스템 구동은 기본적으로 사용자가 웹 브라우저를 통해 입력한 값들이 서버로 전송이 되고 서버에서는 입력 값을 받아 내부 프로세스에 따라 처리한 후 응답해 주는 구조이다.

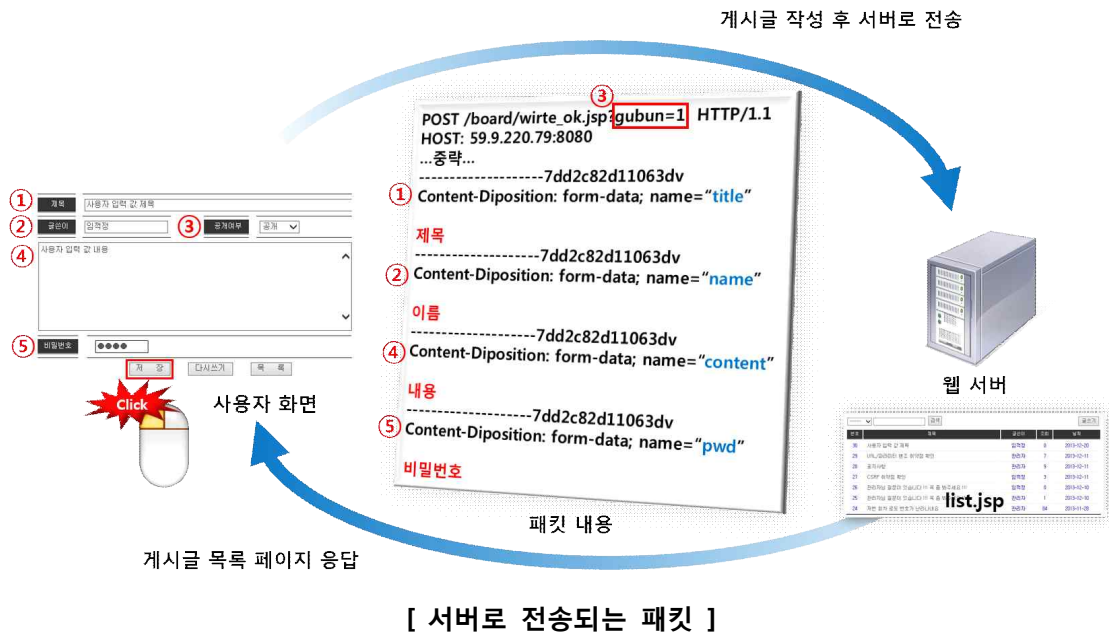
참고로 사용자가 입력할 수 있는 값들은 아래와 같다.



- ① 페이지 주소 ② 페이지 주소에 사용되는 파라미터 ③ 검색 폼
④ 로그인 폼 ⑤ 게시판 관련 입력 폼

[사용자 입력 폼 종류]

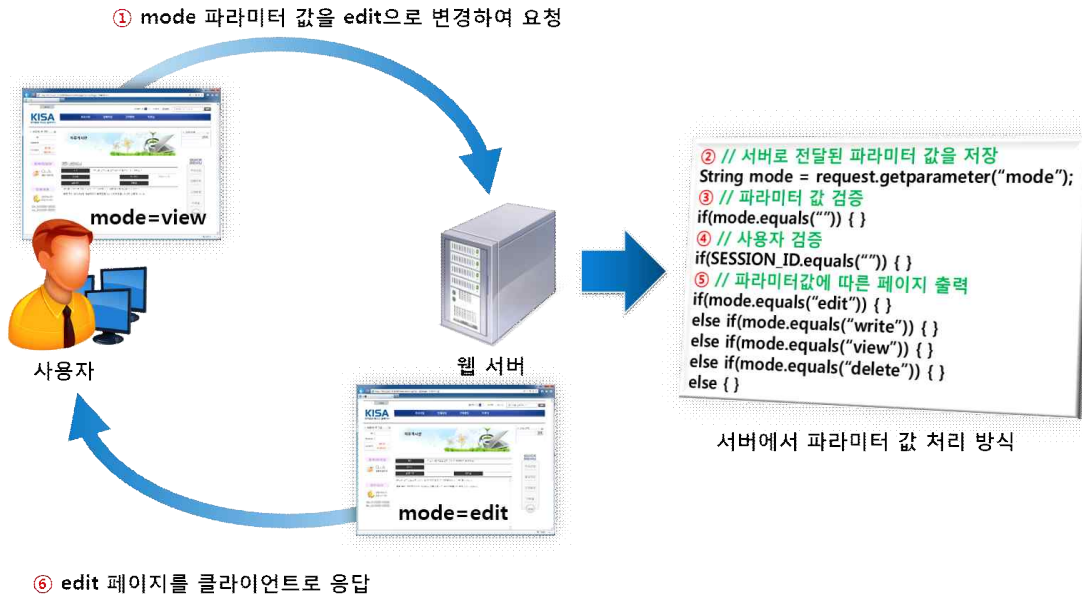
사용자가 이러한 입력 폼을 입력한 후 게시글을 등록하고자 할 때 웹 서버로 패킷이 전송되어지며 패킷의 내용을 아래와 같다.



- ① 게시글 제목 필드로 패킷의 POST 데이터 중 "title" 영역으로 전송
- ② 게시글 작성자 필드로 패킷의 POST 데이터 중 "name" 영역으로 전송
- ③ 게시글 공개여부 필드로 GET 방식으로 URL 파라미터(gubun)로 전송
- ④ 게시글 내용 필드로 패킷의 POST 데이터 중 "content" 영역으로 전송
- ⑤ 게시글 비밀번호 필드로 패킷의 POST 데이터 중 "pwd" 영역으로 전송

상기의 데이터를 포함한 패킷이 웹 서버로 전송되고 웹 서버에서는 내부적인 프로세스 따라 입력 값들을 검증하고 데이터베이스에 저장한 후 응답페이지를 사용자에게 전송한다.

이때, 웹 서버에서 사용자 입력 값을 처리하는 과정은



[파라미터 처리 과정]

- ① 사용자가 서버로 전달되는 파라미터 값 변경
- ② 전달된 파라미터 값(사용자 입력 값)을 변수로 저장
- ③ 저장된 변수를 이용하여 파라미터 값 검증(공백, 특수문자 필터링)
- ④ 사용자 세션 값을 이용하여 사용자 검증(사용자 인증)
- ⑤ 파라미터 값에 따른 페이지 출력
- ⑥ 사용자에게 웹 페이지 응답

이와 같은 순서로 이루어지며 만약 파라미터 검증과 사용자 검증 부분이 미흡한 경우 공격자가 파라미터 변조를 통해 운영체제 명령 실행, 사용자 인증우회와 같은 공격을 수행할 수 있으므로 본 가이드를 참고하여 검증 부분에 시큐어 코딩을 적용할 필요성이 있다.

제2절 웹 소스코드 보안약점

웹 보안 취약점은 웹 응용 프로그램 개발단계에서 보안에 대한 고려 없이 개발되어 발생하는 소스코드 관련 보안 취약점과 웹 서버 보안설정이 미흡해서 발생하는 웹 서버 관련 보안 취약점을 나눌 수 있다.

웹 소스코드 관련 보안 취약점은 기존 'SW 개발보안제도' 에서 정의한 소스코드 보안약점 중에서 웹에 관련된 보안약점을 정리한 것으로 한국인터넷진흥원에서 분석한 결과 웹 결과 보안약점이 모두 보안 취약점으로 나타날 수 있다.

참고로 'SW 개발보안제도' 에서 정의한 주요 보안약점은 다음과 같다.

- 1. 입력데이터 검증 및 표현** : 프로그램 입력 값에 대한 검증 누락 또는 부적절한 검증, 데이터의 잘못된 형식지정으로 인해 발생할 수 있는 보안약점

번호	보안약점	설명
①	SQL 삽입	사용자의 입력 값 등 외부 입력 값이 SQL 쿼리에 삽입되어 공격자가 쿼리를 조작해 공격할 수 있는 보안약점
②	자원 삽입	외부 입력 값에 대한 검증이 없거나 혹은 잘못된 검증을 거쳐서 시스템 자원에 접근하는 경로 등의 정보로 이용될 때 발생하는 보안약점
③	크로스 사이트 스크립트	검증되지 않은 외부 입력 값에 의해 브라우저에서 악의적인 코드가 실행되는 보안약점
④	운영체제 명령어 삽입	운영체제 명령어를 구성하는 외부 입력 값이 적절한 필터링을 거치지 않고 쓰여져서 공격자가 운영체제 명령어를 조작할 수 있는 보안약점
⑤	위험한 형식 파일 업로드	파일의 확장자 등 파일형식에 대한 검증없이 업로드를 허용하여 발생하는 보안약점

⑥	신뢰되지 않은 URL 주소로 자동 접속 연결	사용자의 입력 값 등 외부 입력 값이 링크 표현에 사용되고, 이 링크를 이용하여 악의적인 사이트로 리다이렉트(redirect)되는 보안약점
⑦	nXQuery 삽입	사용자의 입력 값 등 외부 입력 값이 XQuery 표현에 삽입되어 악의적인 쿼리가 실행되는 보안약점
⑧	XPath 삽입	사용자의 입력 값 등 외부 입력 값이 XPath 표현에 삽입되어 악의적인 쿼리가 실행되는 보안약점
⑨	LDAP 삽입	검증되지 않은 입력 값을 사용해서 동적으로 생성된 LDAP문에 의해 악의적인 LDAP 명령이 실행되는 보안약점
⑩	크로스사이트 요청 위조	검증되지 않은 외부 입력 값에 의해 브라우저에서 악의적인 코드가 실행되어 공격자가 원하는 요청(Request)이 다른 사용자(관리자 등)의 권한으로 서버로 전송되는 보안약점
⑪	디렉터리 경로 조작	지정된 경로 밖의 파일시스템 경로에 접근하게 되는 보안약점
⑫	HTTP 응답분할	사용자의 입력 값 등 외부 입력 값이 HTTP 응답헤더에 삽입되어 악의적인 코드가 실행되는 보안약점
⑬	정수 오버플로우	정수를 사용한 연산의 결과가 정수 값의 범위를 넘어서는 경우 프로그램이 예기치 않은 동작될 수 있는 보안약점
⑭	보호메커니즘을 우회할 수 있는 입력 값 변조	사용자에 의해 변경될 수 있는 값을 사용하여 보안결정 (인증/인가/권한부여 등)을 수행하여 보안 메커니즘이 우회될 수 있는 보안약점

2. 보안기능 : 보안기능(인증, 접근제어, 기밀성, 암호화, 권한 관리 등)을 적절하지 않게 구현 시 발생할 수 있는 보안약점

번호	보안약점	설명
①	적절한 인증 없는 중요기능 허용	적절한 인증없이 중요정보(계좌이체 정보, 개인정보 등)를 열람(또는 변경)할 수 있게 하는 보안약점
②	부적절한 인가	적절한 접근제어 없이 외부 입력 값을 포함한 문자열로 서버자원에 접근(혹은 서버 실행 인가)을 할 수 있게 하는 보안약점
③	중요한 자원에 대한 잘못된 권한설정	중요자원(프로그램 설정, 민감한 사용자 데이터 등)에 대한 적절한 접근권한을 부여하지 않아, 의도하지 않는 사용자에게 의해 중요정보가 노출·수정되는 보안약점

④	취약한 암호화 알고리즘 사용	중요정보(패스워드, 개인정보 등)의 기밀성을 보장할 수 없는 취약한 암호화 알고리즘을 사용하여 정보가 노출될 수 있는 보안약점
⑤	사용자 중요정보 평문 저장(또는 전송)	중요정보(패스워드, 개인정보 등) 저장(또는 전송)시 암호화 하지 않아 공격자에게 누출될 수 있는 보안약점
⑥	하드코딩된 패스워드	소스코드 내에 비밀번호를 하드코딩함에 따라 관리자 비밀번호가 노출되거나, 주기적 변경 등 수정(관리자 변경 등)이 용이하지 않는 보안약점
⑦	충분하지 않은 키 길이 사용	데이터의 기밀성, 무결성 보장을 위해 사용되는 키의 길이가 충분하지 않아 기밀정보 누출, 무결성이 깨지는 보안약점
⑧	적절하지 않은 난수 값 사용	예측 가능한 난수사용으로 공격자로 하여금 다음 숫자 등을 예상하여 시스템 공격이 가능한 보안약점
⑨	패스워드 평문 저장	기밀정보인 비밀번호를 암호화하지 않아 노출될 수 있는 보안약점
⑩	하드코딩된 암호화 키	소스코드 내에 암호화키를 하드코딩 하는 경우, 향후 노출될 수 있으며, 키 변경 등 수정이 용이하지 않는 보안약점
⑪	취약한 패스워드 허용	비밀번호 조합규칙(영문, 숫자, 특수문자 등) 및 길이가 충분하지 않아 노출될 수 있는 보안약점
⑫	사용자 하드디스크에 저장되는 쿠키를 통한 정보노출	쿠키(세션 ID, 사용자 권한정보 등 중요정보)를 사용자 하드디스크에 저장함으로써 개인정보 등 기밀정보가 노출될 수 있는 보안약점
⑬	보안속성 미적용으로 인한 쿠키 노출	쿠키에 보안속성을 적용하지 않을 경우, 쿠키에 저장된 중요 데이터가 공격자에 노출될 수 있는 보안약점
⑭	주석문 안에 포함된 패스워드 등 시스템 주요정보	소스코드내의 주석문에 비밀번호가 하드코딩되어 비밀번호가 노출될 수 있는 보안약점
⑮	솔트 없이 일방향 해쉬 함수 사용	공격자가 솔트없이 생성된 해쉬값을 얻게 된 경우, 미리 계산된 레인보우 테이블을 이용하여 원문을 찾을 수 있는 취약점
⑯	무결성 검사없는 코드 다운로드	원격으로부터 소스코드 또는 실행파일을 무결성 검사없이 다운로드 받고 이를 실행하는 경우 공격자가 악의적인 코드를 실행 할 수 있는 보안약점

3. 시간 및 상태 : 동시 또는 거의 동시 수행을 지원하는 병렬 시스템, 하나 이상의 프로세스가 동작되는 환경에서 시간 및 상태를 부적절하게 관리하여 발생할 수 있는 보안약점

번호	보안약점	설명
①	경쟁조건: 검사시점과 사용시점(TOCTOU)	멀티 프로세스 상에서 자원을 검사하는 시점과 사용하는 시점이 달라서 발생하는 보안약점
②	제어문을 사용하지 않는 재귀함수	적절한 제어문 사용이 없는 재귀함수에서 문헌재귀가 발생하는 보안약점

4. 에러처리 : 에러를 처리하지 않거나, 불충분하게 처리하여 에러정보에 중요정보(시스템 등)가 포함될 때 발생할 수 있는 보안약점

번호	보안약점	설명
①	오류메시지를 통한 정보노출	개발 시 활용을 위한 오류정보의 출력메시지를 배포될 버전의 SW에 포함시킬 때 발생하는 보안약점
②	오류상황 대응 부재	시스템에서 발생하는 오류상황을 처리하지 않아 프로그램 다운 등 의도하지 않은 상황이 발생 할 수 있는 보안약점
③	적절하지 않은 예외처리	예외에 대한 부적절한 처리로 인해 의도하지 않은 상황이 발생될 수 있는 보안약점

5. 코드오류 : 타입 변환 오류, 자원(메모리 등)의 부적절한 반환 등과 같이 개발자가 범할 수 있는 코딩오류로 인해 유발되는 보안약점

번호	보안약점	설명
①	널(Null) 포인터 역참조	Null로 설정된 변수의 주소 값을 참조했을 때 발생하는 보안약점
②	부적절한 자원 해제	사용된 자원을 적절히 해제 하지 않으면 자원의 누수 등이 발생 하고, 자원이 모자라 새로운 입력에 처리 못하게 되는 보안약점

6. 캡슐화 : 중요한 데이터 또는 기능성을 불충분하게 캡슐화 하였을 때
인가되지 않은 사용자에게 데이터 누출이 가능해지는 보안약점

번호	보안약점	설명
①	잘못된 세션에 의한 데이터 정보 노출	잘못된 세션에 의해 권한 없는 사용자에게 데이터 누출이 일어날 수 있는 보안약점
②	제거되지 않고 남은 디버그 코드	디버깅을 위해 작성된 코드를 통해 권한 없는 사용자 인증 우회 (또는 중요정보) 접근이 가능해지는 보안약점
③	시스템 데이터 정보노출	사용자가 볼 수 있는 오류 메시지나 스택 정보에 시스템 내부 데이터나 디버깅 관련 정보가 공개되는 보안약점
④	Public 메소드부터 반환된 Private 배열	private로 선언된 배열을 public으로 선언된 메소드를 통해 반환(return)하면, 그 배열의 레퍼런스가 외부에 공개되어 외부에서 배열의 수정될 수 있는 보안약점
⑤	Private 배열에 Public 데이터 할당	public으로 선언된 데이터 또는 메소드 인자가 private 선언된 배열에 저장되면, private 배열을 외부에서 접근할 수 있게 되는 보안약점

7. API 오용 : 의도된 사용에 반하는 방법으로 API를 사용하거나, 보안에 취약한 API를 사용하여 발생할 수 있는 보안약점

번호	보안약점	설명
①	DNS lookup에 의존한 보안결정	DNS는 공격자에 의해 DNS 스푸핑 공격 등이 가능하므로 보안결정을 DNS 이름에 의존할 경우, 보안결정 등이 노출되는 보안약점

제3절 웹 서버 보안 관리

항목	설명
OS 및 웹 서버 최신판치	보안취약점 정보 사이트(또는 OS 벤더사이트)를 주기적으로 방문하여 최신 취약점 정보와 패치에 관련된 보안대책 적용
웹 서버 전용 호스트	웹 서버 전용 호스트로 구성 ① 웹 서비스 운영에 필요한 최고한의 프로그램만 운영 ② 관리자만 시스템의 모든 권한을 사용할 수 있도록 설정
서버 접근 제어	관리목적의 웹 서버 접근은 콘솔 접근만 허용 ① 부득이 하게 원격서비스가 필요할 경우 관리자 PC의 IP만 접근이 가능하도록 설정
DMZ 영역에 위치	웹 서버는 DMZ 영역에 위치 ① 웹 서버를 방화벽으로 보호하며, 침투가 일어날 시 내부 네트워크 침입은 불가능하도록 구성
취약한 관리자 계정	관리자 계정은 강력한 패스워드 규칙을 사용 ① 패스워드의 경우 최소 8자 이상이어야 하며, 쉽게 유추할 수 없도록 설정 ※ 사전에 없는 단어 사용, 특수문자 · 영문자 · 숫자 포함 권고 ② 관리자 계정 패스워드의 주기적인(분기별) 변경
파일 접근 권한	일반 사용자 계정으로 관리자가 사용하는 파일들 혹은 시스템 주요 파일들을 변경할 수 없도록 보안설정
취약점 점검	정기적으로 취약점 점검 도구와 보안 점검항목을 사용하여 호스트 OS의 보안 취약점 점검(년 1회 이상 권고)

제4절 용어정의

■ XPath(XML Path Language)

W3C의 표준으로 확장 생성 언어 문서의 구조를 통해 경로 위에 지정한 구문을 사용하여 항목을 배치하고 처리하는 방법을 기술하는 언어이다. XML 표현보다 더 쉽고 약어로 되어 있으며, XSL 변환(XSLT)과 XML 지시자 언어(XPointer)에 쓰이는 언어이다. XPath는 XML 문서의 노드를 정의하고 위하여 경로식을 사용하며, 수학 함수와 기타 확장 가능한 표현들이 있다.

■ DMZ(Demilitarized zone, DMZ)

조직의 내부 네트워크와 외부 네트워크 사이에 위치한 서브넷이다. 내부 네트워크와 외부 네트워크가 DMZ로 연결할 수 있도록 허용 하면서도, DMZ 내의 컴퓨터는 오직 외부 네트워크에만 연결할 수 있도록 한다는 점이다. 즉 DMZ 안에 있는 호스트들은 내부 네트워크로 연결할 수 없다.

■ Mybatis 프레임워크

Java에서 DB를 편하게 핸들링 할 수 있게 해주는 프레임워크이다.

■ 세션 하이재킹(Session Hijacking)

다른 사람의 세션 상태를 훔치거나 도용하여 액세스하는 해킹 기법이며, 일반적으로 세션 ID 추측 및 세션 ID 쿠키 도용을 통해 공격이 이루어진다. 하이재킹으로 인한 직접적인 피해는 세션 상태에 어떤 정보가 저장되어 있느냐에 달려 있지만 그보다 더 위험한 것은 ID와 패스워드를 사용하는 인증 절차를 건너뛰어 서버와 사용자가 주고받는 모든 내용을 그대로 도청하거나 서버의 권한을 확보할 수도 있다는 점이다.

- HTTPS(Hypertext Transfer Protocol over Secure Socket Layer)

월드 와이드 웹 통신 프로토콜인 HTTP의 보안이 강화된 버전이다.

- 해쉬 함수

주어진 원문에서 고정된 길이의 의사난수를 생성하는 연산기법이며 생성된 값은 '해쉬 값'이라고 한다. MD5, SHA, SHA-1, SHA-256 등의 알고리즘이 있다.

- DES 알고리즘

DES(Data Encryption Standard)암호는 암호화 키와 복호화 키가 같은 대칭키 암호로 64비트의 암호화 키를 사용한다. 전수공격(Brute Force)에 약하다.

- AES(Advanced Encryption Standard)

미국 정부 표준으로 지정된 블록 암호 형식으로 이전의 DES를 대체하며, 미국 표준 기술 연구소(NIST)가 5년의 표준화 과정을 거쳐 2001년 11월 26일에 연방 정보처리표준(FIPS 197)으로 발표하였다.

- SHA(Secure Hash Algorithm)

해쉬 알고리즘의 일종으로 MD5의 취약성을 대신하여 사용한다. SHA, SHA-1, SHA-2(SHA-224, SHA-256, SHA-384, SHA-512) 등의 다양한 버전이 있으며, 암호 프로토콜인 TLS, SSL, PGP, SSH, IPSec 등에 사용된다.

- 화이트리스트(Whitelist)

블랙리스트(Black List)의 반대개념으로 신뢰할 수 있는 사이트나 IP주소 목록을 말하며, 즉 신뢰할 수 있는 글이나 문자열의 목록을 말한다.

제5절 참고문헌

- [1] 미래창조과학부 / 주요정보통신기발시설 취약점 분석·평가 기준 / 2013.
- [2] 안전행정부, 한국인터넷진흥원 / 정보시스템 개발·운영자를 위한 홈페이지 SW(웹) 개발보안 가이드 / 2012.
- [3] OWASP / OWASP Top Ten Project 2013 /
https://www.owasp.org/index.php/Top_10_2013-Top_10
- [4] OWASP / OWASP Top Ten Project 2010 /
https://www.owasp.org/index.php/Top_10_2010
- [5] OWASP / OWASP Guide Project /
https://www.owasp.org/index.php/Category:OWASP_Guide_Project

정보시스템 개발·운영자를 위한 홈페이지 취약점 진단·제거 가이드

2014년 1월 인쇄

2014년 1월 발행

발행처 : 한국인터넷진흥원

서울특별시 송파구 중대로 135

Tel: (02) 405-5154, 5394

인쇄처 : 강산

Tel: (02)2279-8494

(비매품)

- 본보고서는 미래창조과학부의 출연금으로 수행한 정보보호 인프라 확충사업의 결과입니다.
- 본보고서의 내용을 발표할 때에는 반드시 한국인터넷진흥원의 정보보호 인프라 확충사업의 결과임을 밝혀야 합니다.
- 본보고서의 판권은 한국인터넷진흥원이 소유하고 있으며, 한국인터넷진흥원의 허가 없이 무단전재 및 복사를 금합니다.